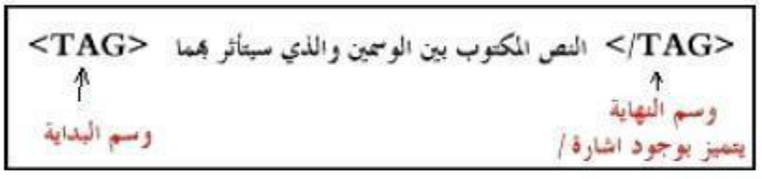
جامعة حماة
كلية الاقتصاد
التعليم المفتوح
برنامج : التسويق و التجارة

مهارات حاســـــوب (٢)

المقدمة

Html: إنها اللغة المستخدمة لإنشاء صفحات الإنترنت. (والكلمة إختصار لـــHyper Text Markup Language).
وهي ليست لغة برمجة بالمعنى والشكل المتعارف عليه للغات البرمجة الأخرى كلغة C . فهي مثلاً لا تحتوي على جمل التحكم والدوران، وعند الحاجة لاستخدام هذه الجمل يجب تضمين شيفرات من لغات أخرى كـــ Java, JavaScript, CGI . كذلك فهي لا تحتاج إلى مترجم خاص به Compiler . وهي غير مرتبطة بنظام تشغيل معين، لأنه يتم تفسيرها وتنفيذ تعليماتها مباشرة من قبل متصفح الإنترنت وبغض النظر عن النظام المستخدم. لذلك فهي لغة بسيطة جداً، وسهلة الفهم والتعلم ولا تحتاج لمعرفة مسبقة بلغات البرمجة والهيكلية المستخدمة فيها. بل ربما كل ما تحتاجه هو القليل من التفكير المنطقي وترتيب الأفكار.

تتكون مفردات لغة Html من شيفرات تسمى TAGS أي الوسوم. وهي تستخدم بشكل أزواج وتكتب بالصيغة التالية (من اليسار إلى اليمين) :-



فعلى سبيل المثال الوسم <B> يستخدم لكتابة الكلمات بخط أسود عريض Bold وذلك بالشكل التالي:

</B> Text <B>

وهناك بعض الوسوم الخاصة التي تستخدم بصورة مفردة مثل وسم نهاية السطر
 أو قد تستخدم بكلتا الحالتين مثل وسم الفقرة <P>.
وسوف نناقش هذه الوسوم وغيرها بالتفصيل في حينه إن شاء الله

كيف نبدأ...

لا يتطلب كتابة ملف HTML أية برامج خاصة فهي كما قلنا لغة لا تحتوي على برنامج مترجم. بل نحتاج فقط إلى برنامج لتحرير النصوص البسيطة ومعالجتها، وبرنامج المفكرة الموجود في Windows يفي بهذا الغرض. وكذلك إلى أحد متصفحات الإنترنت Netscape Navigator أو MS Internet Explorer لمعاينة الصفحات التي نقوم بتصميمها. وعليك فقط أن نقوم بحفظ النص المكتوب بملف يحمل الاسم الممتد html. أو htm.
والجدير ذكره أنه يوجد العديد من البرامج التي تستخدم لإنشاء صفحات Html. دون الحاجة لمعرفة هذه اللغة حيث يقوم المستخدم من خلالها بكتابة الصفحات وتصميمها بما تحويه من نصوص ورسومات وجداول ثم يقوم البرنامج بتخليق الوسوم المناسبة وتحويل هذه الصفحات من وراء الكواليس تلقائياً وحفظها بتنسيق html. أي أن دور المستخدم ينحصر في الكتابة والتصميم فقط، دون معرفته للشيفرة التي استخدمت. وبالتالي عدم قدرته على التحكم بأي وسم أو تعديل الشيفرة حسب الحاجة، إلا من خلال إعادته للتصميم الأساسي ثم إعادة التحويل والحفظ من قبل البرنامج. وهذه الطريقة على سهولتها وسرعتها نسبياً، إلا أني لا أنصح باستخدامها لمن يريد معرفة هذه اللغة والتمكن منها.

قبل أن نبدأ : حسناً، لديك محرر نصوص ممتاز لكتابة ملفات HTML ولديك متصفح إنترنت لمعاينتها، ولديك هذه الدروس التي ستنطلق معها إلى عالم تصميم صفحات الويب. هل هذا يكفي؟ برأيي المتواضع، لا.

تحتاج دائماً وأبداً إلى تطبيق ما تتعلمه بصورة عملية أكثر من مجرد الأمثلة المدرجة في الدروس. ما رأيك في أن تفكر بموضوع ما يستهويك وتحب أن تتعاطى به؟ وتخيل أنك ستقوم بإنشاء موقع ويب عنه بصورة واقعية. ومع تقدمك في الدروس قم بتطبيق ما فيها على صفحاتك. ستجد الكثير من المتعة في هذا، وستسر جداً عندما ترى صفحتك تتبلور أمام عينيك يوماً بعد يوم، والأهم من هذا كله هو أنك ستكتشف أي ثغرات في استيعابك لهذه الدروس (وعندها من المؤكد أنك ستقوم بتلافيها) وقد تكتشف كذلك ثغرات ارتكبها كاتب هذه الدروس.

والآن... لنبدأ

الأساسيات

أهلاً وسهلاً بك إلى الدرس الأول من دروس HTML. سوف أقوم في هذا الدرس بسرد الوسوم الأساسية لصفحة الويب ومناقشتها معك واحداً تلو الآخر. لنصل في النهاية إلى إنشاء صفحة ويب بسيطة.

لنأخذ الوسوم التالية:

وسم البداية	وسم النهاية
<HTML>	</HTML>
<HEAD>	</HEAD>
<TITLE>	</TITLE>
<BODY>	</BODY>

ماذا تلاحظ؟ أن كل منها يتألف من زوج من الوسوم أحدهما وسم البداية، والآخر وسم النهاية. ويتميز وسم النهاية بوجود الرمز / . تأمل الرسم التالي، فهو يعطي فكرة عن تركيب ملف Html

☜ إذن فملف Html يبدأ دائماً بالوسم <HTML> وينتهي بالوسم </HTML>. لا تنسى ذلك!

أما الوسم <HEAD> فيحدد بداية المقطع الذي يحتوي على المعلومات الخاصة بتعريف الصفحة. كالعنوان الظاهر على شريط عنوان المتصفح. وهذا العنوان بدوره يحتاج لأن يوضع بين الوسمين: <TITLE> ... </TITLE> وبالطبع يجب كتابة الوسم </HEAD> لكي ننهي هذا المقطع.

نأتي إلى الوسم <BODY> والذي يتم كتابة نصوص صفحة الويب ضمنه، بالإضافة إلى إدراج الصور والجداول وباقي محتويات الصفحة. وهو أيضاً يحتاج إلى وسم الإنهاء </BODY>

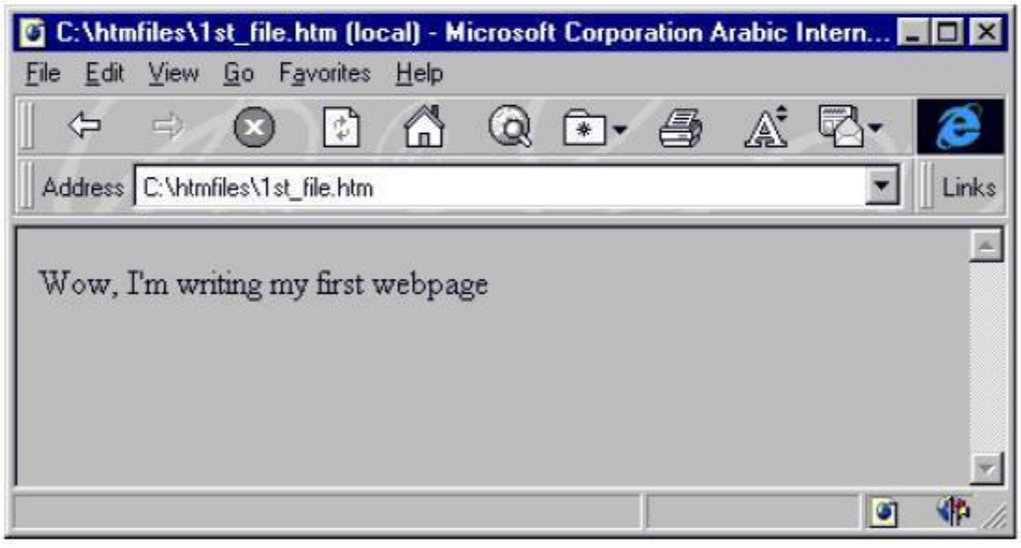
ما رأيك لو نبدأ بتطبيق هذه المعلومات بصورة عمليه؟ هيا... قم بفتح برنامج المفكرة واكتب ما يلي:

```
<HTML>
<HEAD>
<TITLE>
This is a test Webpage
</TITLE>
</HEAD>

<BODY>
Wow, I'm writing my first webpage
</BODY>
</HTML>
```

والآن قم بحفظ ما كتبته في ملف وبأي اسم تختاره. ولا تنسى أن الامتداد المستخدم في أسماء ملفات HTML هو *.htm* أو *.html*. مثلاً أنا اخترت الاسم 1st_file.htm ومن الأفضل أن تقوم بإنشاء مجلد مستقل على القرص الصلب لكي تحفظ به ملفاتك فهذا يسهل عليك عملية استرجاعها للعرض أو التحديث.

حان وقت العرض، لكي نشاهد نتيجة ما كتبناه. قم بتشغيل متصفح الإنترنت الذي تستخدمه. فإذا كان Netscape Navigator اختر الأمر File Open... من قائمة File. أما في MS Internet Explorer فاختر الأمر Open... من قائمة File. ثم حدد المسار الذي يوجد به الملف.

وذلك طبعاً حسب الافتراضات السابقة التي اتبعتها عند تخزين الملف. وهذا ما حصلت عليه:



هل حصلت على نفس النتيجة؟ إذن مبروك ☺ لقد قمت بإنشاء أول صفحة ويب خاصة بك.

وقبل أن نستمر أريد أن أنبهك إلى بعض الملاحظات عند كتابة صفحات الويب:

- لا يوجد فرق بين كتابة الوسوم بالأحرف الإنجليزية الكبيرة UPPERCASE أو الأحرف الصغيرة lowercase. لذلك تستطيع الكتابة بأي شكل منهما أو حتى الكتابة بكليهما.

- إن المتصفحات لا تأخذ بعين الاعتبار الفراغات الزائدة أو إشارات نهاية الفقرات (أي عندما تقوم بضغط مفتاح *Enter*) التي تجدها هذه المتصفحات في ملف Html. وبعبارة أخرى فإن باستطاعتك كتابة ملفك السابق بالشكل التالي:

```
<HTML><HEAD><TITLE> This is a test Webpage </TITLE></HEAD><BODY>
Wow, I'm writing my first webpage </BODY></HTML>
```

أو بالشكل التالي:

```
<HTML>
<HEAD>
<TITLE>
This
is a
test
Webpage
</TITLE>
</HEAD>
<BODY>
Wow,
I'm
writing
my
first
webpage
</BODY>
</HTML>
```

أو حتى بهذا الشكل:

```
<HTML> <HEAD> <TITLE>
This is a test Webpage
</TITLE>
</HEAD>
<BODY>
Wow, I'm writing my first webpage
</BODY>
</HTML>
```

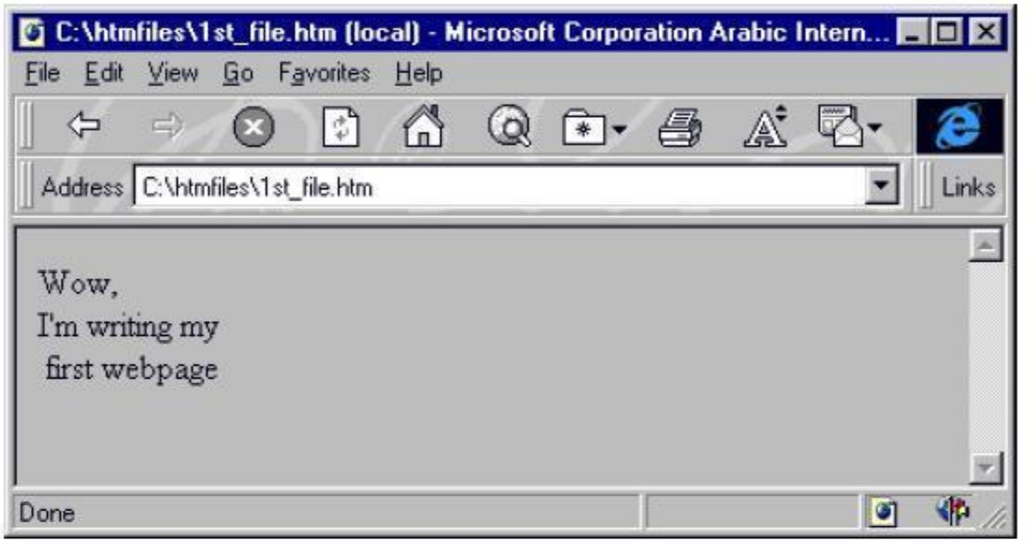
وفي كل الحالات ستحصل على نفس النتيجة.
لكن هذا لا يعني أن الفقرة المكونة مثلاً من عشرة أسطر ستمتد إلى عدة أمتار بعرض الشاشة. كلا بالطبع لأن المتصفح سيقوم بعمل التفاف تلقائي لها بحسب عرض الشاشة، مهما كان مقدار هذا العرض.
والآن قد تتساءل، إذن كيف يمكن التحكم بمقدار النص المكتوب في كل سطر وكيف يمكن تحديد نهاية الفقرة وبداية الفقرة التي تليها؟ سؤال وجيه!!! والإجابة عليه هي:

سوف نستخدم الوسم
 لتحديد النهاية للسطر. والبدء بسطر جديد (لاحظ أن هذا الوسم مفرد، أي ليس له وسم نهاية).

ونعود إلى المثال السابق، قم بتعديل الملف لكي يصبح بالشكل التالي

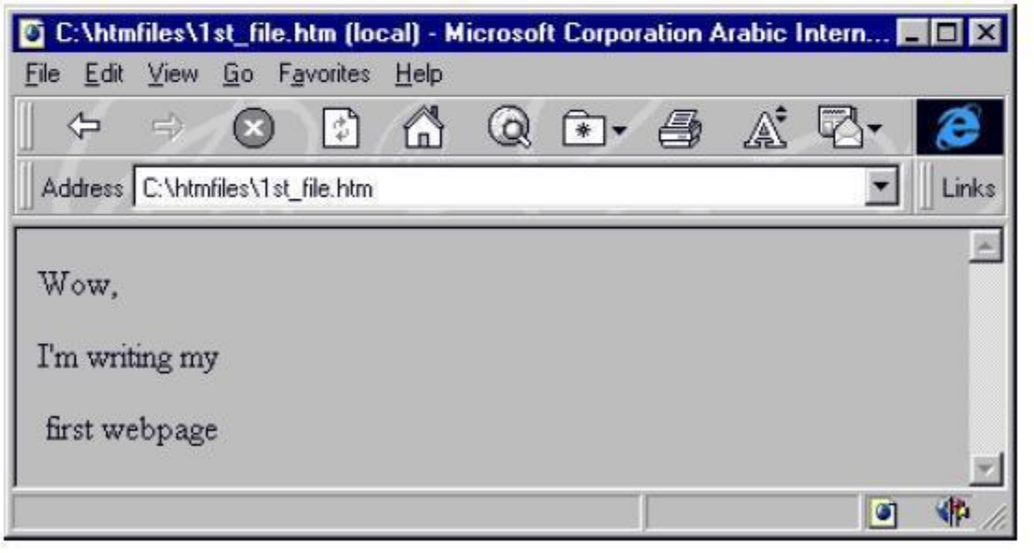
```
<HTML>
<HEAD>
<TITLE>
This is a test Webpage
</TITLE>
```

```
</HEAD>
<BODY>

Wow, <BR> I'm writing my <BR>
first webpage
</BODY>
</HTML>
```



وهناك أيضا الوسم <P> الذي يقوم تقريباً بنفس عمل الوسم السابق أي أنه ينهي السطر أو الفقرة ويبدأ بسطر جديد لكن مع إضافة سطر إضافي فارغ بين الفقرات.

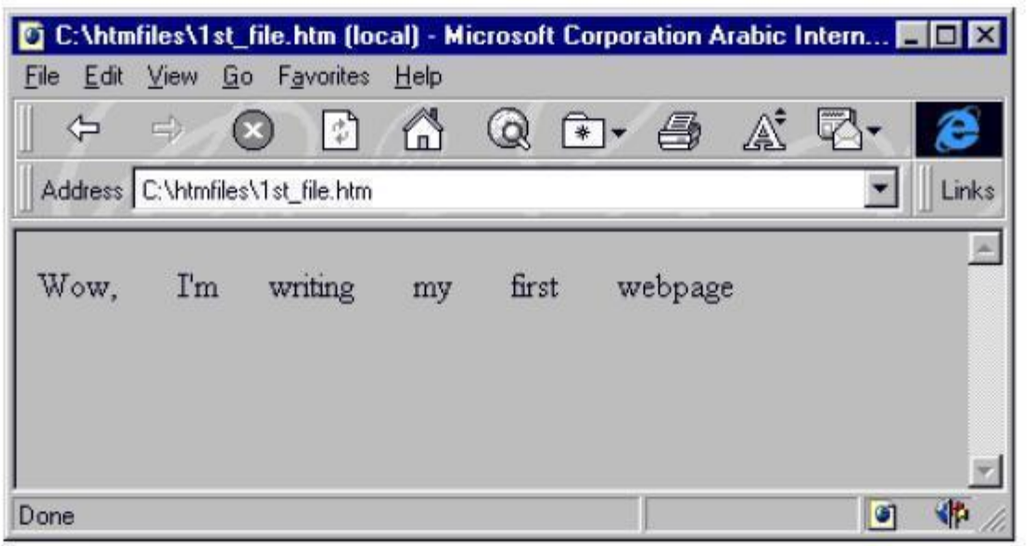
```
<HTML>
<HEAD>
<TITLE>
This is a test Webpage
</TITLE>
</HEAD>
<BODY>
Wow, <P> I'm writing my <P>
first webpage
</BODY>
</HTML>
```



أما الفراغات فتعتبر رموزاً خاصة لذلك لا نستطيع التحكم بها وبعددها إلا باستخدام الوسم   (والأحرف هي اختصار للعبارة Non Breakable Space). وإذا أردت إدخال عدة فراغات بين نص وآخر ما عليك إلا كتابة هذا الوسم بنفس عدد الفراغات المطلوب. كما يجب عليك التقيد بالأحرف الصغيرة هنا.

إذن لنعد إلى المفكرة ونكتب ملفنا بالشكل التالي:

```
<HTML>
<HEAD>
<TITLE>
This is a test Webpage
</TITLE>
</HEAD>
<BODY>
Wow,      
I'm      writing    
  my       first       webpage
</BODY>
</HTML>
```



وبمناسبة الحديث عن الرموز الخاصة فهناك العديد من هذه الرموز والتي يجب أن تكتب بصورة معينة وباستخدام الوسوم وليس مباشرة بصورتها العادية. خذ مثلا إشارتي *أكبر من وأصغر من وإشارة الاقتباس "*. كل هذه الإشارات تستخدم أصلاً مع الوسوم فهي محجوزه ضمن مفردات لغة HTML ومن الخطأ استخدامها بصورتها الصريحة لئلا يؤدي ذلك إلى حدوث مشاكل في طريقة عرض الصفحة. كذلك فإن هناك رموزاً غير موجودة أساساً على لوحة المفاتيح كرمز حقوق الطبع © ورمز العلامة المسجلة ® ونحتاج إلى هذه الطريقة(طريقة الوسوم) لكتابتها. وإليك جدول ببعض هذه الرموز ووسومها المكافئة. وألفت نظرك إلى أنها تكتب كما هي في الجدول وبدون إشارتي <>

الـكـود	الـــرمز
الـــرمـــوز	
‘	‘
’	’
‚	‚
“	“
”	”

„	„
†	†
‡	‡
‰	‰
‹	‹
›	›
♠	♠
♣	♣
♥	♥
♦	♦
‾	‾
←	←
↑	↑
→	→
↓	↓
™	™
&	&
<	<
>	>
"	"
×	×
÷	÷
–	–
—	—
¡	¡
¢	¢
£	£
¤	¤
¥	¥
€	€
¦	¦
§	§
	&#127;
¨	¨
©	©
ª	ª
«	«
¬	¬
®	®
¯	¯
°	°
±	±
²	²
³	³

´	´
µ	µ
¶	¶
·	·
¸	¸
¹	¹
º	º
»	»
¼	¼
½	½
¾	¾
¿	¿

Capital الحروف

À	À
Á	Á
Â	Â
Ã	Ã
Ä	Ä
Å	Å
Æ	Æ
Ç	Ç
È	È
É	É
Ê	Ê
Ë	Ë
Ì	Ì
Í	Í
Î	Î
Ï	Ï
Ð	&ET;
Ñ	Ñ
Ò	Ò
Ó	Ó
Ô	Ô
Õ	Õ
Ö	Ö
Ø	Ø
Ù	Ù
Ú	Ú
Û	Û
Ü	Ü
Ý	Ý
Þ	Þ

الحروف Small

ß	ß

à	à
á	á
â	â
ã	ã
ä	ä
å	å
æ	æ
ç	ç
è	è
é	é
ê	ê
ë	ë
ì	ì
í	í
î	î
ï	ï
ð	&et;
ñ	ñ
ò	ò
ó	ó
ô	ô
õ	õ
ö	ö
ø	&oslas;
ù	ù
ú	ú
û	û
ü	ü
ý	ý
þ	&torn;
ÿ	ÿ

الألوان – أضف لصفحتك بعض الحيوية

أهلاً وسهلاً بك إلى الدرس الثاني من دروس HTML. سوف نقوم في هذا الدرس بالتعرف على الخصائص التي يمكن إضافتها إلى الوسم <BODY> من أجل التحكم بالشكل العام للصفحة، وخصوصا فيما يتعلق بالألوان.

طبعاً أنت لا زلت تذكر الصفحة التي قمنا بكتابتها في الدرس الأول. صفحة بسيطة بخلفية رمادية وخط صغير نسبياً لونه أسود. وهذه هي الإعدادات الإفتراضية التي يعتمدها المتصفح عندما لا نقوم نحن بتحديد إعدادات أخرى. (ربما تقول: أهذه صفحة إنترنت! أين الألوان والرسومات والخطوط الجميلة والتنسيقات التي نراها في صفحات الإنترنت؟ معك حق لكن مهلاً فما زلنا في البداية).
سوف نستمر باستخدام صفحتنا هذه لتوضيح أمثلة هذا الدرس أيضاً، لكن لن أقوم بتكرار كتابة وسوم البداية طالما أن عملنا يتركز في الجزء المخصص لمحتويات الصفحة نفسها أي ضمن الوسمين <BODY> ... </BODY>.
إذن لنبدأ العمل!

نطلق كلمة خاصية (Attribute) على التعابير التي تضاف إلى الوسوم، من أجل تحديد الكيفية أو الشكل الذي تعمل بها هذه الوسوم. وبعبارة أخرى فإن الوسم يقوم بإخبار المتصفح عن العمل الذي يجب القيام به أما الخاصية فتحدد الكيفية التي سيتم بها أداء هذا العمل.

تأمل الشيفرة التالية:

```
<BODY BGCOLOR="FFFFFF">
...
</BODY>
```

لقد قمت بإضافة الخاصية BGCOLOR إلى الوسم <BODY> ، وهي تقوم بتحديد لون الخلفية للصفحة. أما FFFFFF فهي القيمة التي تمثل اللون المختار وهو هنا اللون الأبيض، (لاحظ أنها مكتوبه بين إشارتي " ") ولو أردت تمثيل اللون الأسود لكتبت الرمز 000000. أو الرمز CC6699 للون الأزرق الفاتح.....
فمن أين جاءت هذه القيم، وكيف؟... تابع القراءة وسوف تعرف

القليل عن الألوان...

تلاحظ أن القيم السابقة مكونة من ستة رموز، وهي مكتوبة بالصيغة التالية:-

هناك ثلاثة ألوان أساسية هي الأحمر والأخضر والأزرق، ولكل منها يوجد 256 درجة لونية ويعبر عن هذه الدرجات بالأرقام من 000 وحتى 255. ومن خلال مزج هذه الألوان بدرجاتها اللونية المختلفة نحصل على الألوان الأخرى.

* إن أي لون هو مزيج –وبنسبة معينة من الدرجات– من هذه الألوان الثلاثة *

فمثلا اللون الأسود مكون من الدرجة 000 من كل من اللون الأحمر والأخضر والأزرق. واللون الأبيض مكون من الدرجة 255 من هذه الألوان. أما اللون الأصفر فهو مكون من الدرجة 255 للون الأحمر، والدرجة 255 للون الأخضر، والدرجة 000 من اللون الأزرق... وهكذا بنفس الطريقة يتم تكوين باقي الألوان.

وبعملية حسابية بسيطة 256×256×256 ينتج لدينا أن عدد الألوان التي يمكن الحصول عليها بمزج الألوان الثلاثة السابقة هو 16777216 بالضبط.

حسنا، لكن من أي جاءت الرموز FFFFFF والتي عبرت عن اللون الأبيض بها. إنها ببساطة أرقام... مكتوبة بالنظام السداس عشري (نظام عددي أساسه الرقم 16 ويعبر عنه باستخدام الأرقام العادية من 0 إلى 9 والرموز A,B,C,D,E,F). فالرقم 255 بالنظام العشري العادي يكافئه الرقم FF بالنظام السداس عشري.
إذن فالرقم السداس عشري FF على اليسار يمثل الدرجة 255 للون الأحمر. والرقم FF في الوسط يمثل الدرجة 255 من اللون الأخضر. والرقم FF على اليمين يمثل الدرجة 255 من اللون الأزرق.
وعلى هذا المنوال يعبر عن اللون الأزرق الفاتح بالرقم السداس عشري: CC6699 أما اللون الأسود فرقمه هو 000000.

وهذا جدول ببعض الألوان ورموزها المكافئة بالنظام السداس عشري.

Aqua #00FFFF	Antiquewhite #FAEBD7	Aliceblue #F0F8FF
Beige #F5F5DC	Azure #F0FFFF	Aquamarine #7FFFD4
Blanchedalmond #FFEBCD	Black #000000	Bisque #FFE4C4
Brown #A52A2A	Blueviolet #8A2BE2	Blue #0000FF
Chartreuse #7FFF00	Cadetblue #5F9EA0	Burlywood #DEB887
Cornflowerblue #6495ED	Coral #FF7F50	Chocolate #D2691E
Cyan #00FFFF	Crimson #DC143C	Cornsilk #FFF8DC

Olivedrab #688E23	Olive #808000	Oldlace #FDF5E6
Orchid #DA70D6	Orangered #FF4500	Orange #FFA500
Paleturquoise #AFEEEE	Palegreen #98FB98	Palegoldenrod #EEE8AA
Peachpuff #FFDAB9	Papayawhip #FFEFD5	Palevioletred #D87093
Plum #DDA0DD	Pink #FFC0CB	Peru #CD853F
Red #FF0000	Purple #800080	Powderblue #B0E0E6
Saddlebrown #8B4513	Royalblue #4169E1	Rosybrown #BC8F8F
Seagreen #2E8B57	Sandybrown #F4A460	Salmon #FA8072
Silver #C0C0C0	Sienna #A0522D	Seashell #FFF5EE
Slategray #708090	Slateblue #6A5ACD	Skyblue #87CEEB
Steelblue #4682B4	Springgreen #00FF7F	Snow #FFFAFA
Thistle #D8BFD8	Teal #008080	Tan #D2B48C
Violet #EE82EE	Turquoise #40E0D0	Tomato #FF6347
Whitesmoke #F5F5F5	White #FFFFFF	Wheat #F5DEB3
	YellowGreen #9ACD32	Yellow #FFFF00

بيان بألوان الخلفيات

#00FF00	#00CC00	#009900	#006600	#003300	#000000
#00FF33	#00CC33	#009933	#006633	#003333	#000033
#00FF66	#00CC66	#009966	#006666	#003366	#000066
#00FF99	#00CC99	#009999	#006699	#003399	#000099
#00FFCC	#00CCCC	#0099CC	#0066CC	#0033CC	#0000CC
#00FFFF	#00CCFF	#0099FF	#0066FF	#0033FF	#0000FF
#33FF00	#33CC00	#339900	#336600	#333300	#330000
#33FF33	#33CC33	#339933	#336633	#333333	#330033
#33FF66	#33CC66	#339966	#336666	#333366	#330066
#33FF99	#33CC99	#339999	#336699	#333399	#330099
#33FFCC	#33CCCC	#3399CC	#3366CC	#3333CC	#3300CC
#33FFFF	#33CCFF	#3399FF	#3366FF	#3333FF	#3300FF

#66FF00	#66CC00	#669900	#666600	#663300	#660000
#66FFCC	#66CC33	#669933	#666633	#663333	#660033
#66FF66	#66CC66	#669966	#666666	#663366	#660066
#66FF99	#66CC99	#669999	#666699	#663399	#660099
#66FFCC	#66CCCC	#6699CC	#6666CC	#6633CC	#6600CC
#66FFFF	#66CCFF	#6699FF	#6666FF	#6633FF	#6600FF
#99FF00	#99CC00	#999900	#996699	#993300	#990000
#99FF33	#99CC33	#999933	#996633	#993333	#990033
#99FF66	#99CC66	#999966	#996666	#993366	#990066
#99FF99	#99CC99	#999999	#996699	#993399	#990099
#99FFCC	#99CCCC	#9999CC	#9966CC	#9933CC	#9900CC
#99FFFF	#99CCFF	#9999FF	#9966FF	#9933FF	#9900FF
#CCFF00	#CCCC00	#CC9900	#CC6600	#CC3300	#CC0000
#CCFF33	#CCCC33	#CC9933	#CC6633	#CC3333	#CC0033
#CCFF66	#CCCC66	#CC9966	#CC6666	#CC3366	#CC0066
#CCFF99	#CCCC99	#CC9999	#CC6699	#CC3399	#CC0099
#CCFFCC	#CCCCCC	#CC99CC	#CC66CC	#CC33CC	#CC00CC
#CCFFFF	#CCCCFF	#CC99FF	#CC66FF	#CC33FF	#CC00FF
#FFFF00	#FFCC00	#FF9900	#FF6600	#FF3300	#FF0000
#FFFF33	#FFCC33	#FF9933	#FF6633	#FF3333	#FF0033
#FFFF66	#FFCC66	#FF9966	#FF6666	#FF3366	#FF0066
#FFFF99	#FFCC99	#FF9999	#FF6699	#FF3399	#FF0099
#FFFFCC	#FFCCCC	#FF99CC	#FF66CC	#FF33CC	#FF00CC
#FFFFFF	#FFCCFF	#FF99FF	#FF66FF	#FF33FF	#FF00FF

أما كيف تعرف الرمز الخاص باللون الذي تريد اختياره، فيوجد برامج خاصة تستطيع من خلالها دمج الألوان الثلاثة بنسب مختلفة، ومن ثم يقوم البرنامج بتوليد الرمز السداس عشري المكافئ للون الناتج.

ملاحظة مهمة:
بعض المتصفحات لا تتعرف على رموز الألوان إلا بوضع إشارة # قبل هذه الرموز، لذلك من الأفضل استخدامها دائماً.

وبالنسبة لبعض الألوان الأساسية والدارجة، من الممكن استخدام أسماء هذه الألوان مباشرة بدلاً من الأرقام السداس عشرية. وهذا جدول يوضح هذه الألوان ومسمياتها:

White		Black	
Green		Red	
Purple		Marron	
Blue		Navy	

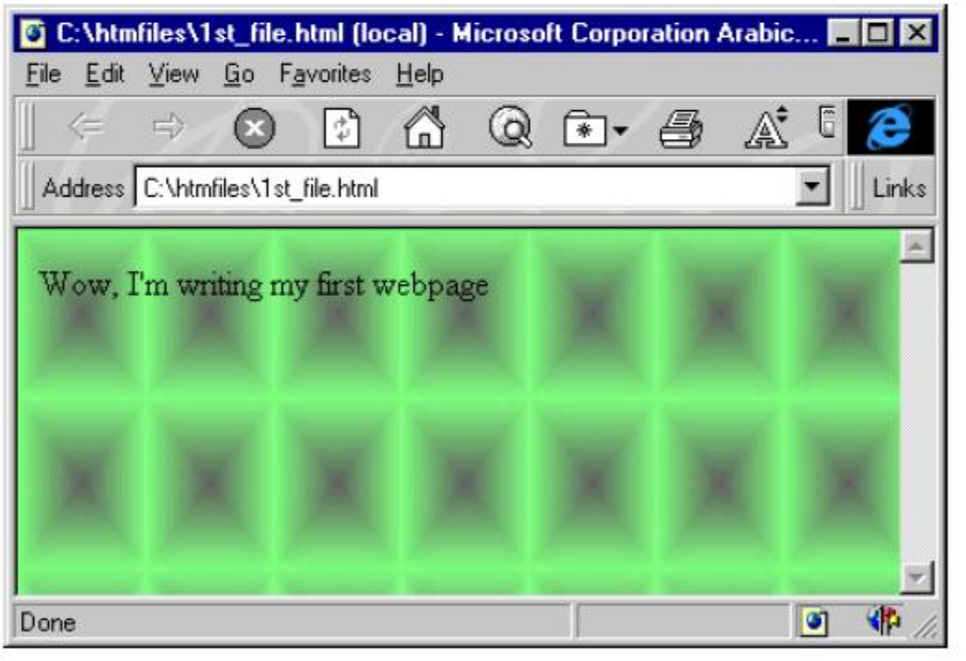
Lime		Teal	
Silver		Gray	
Aqua		Olive	
Yellow		Fuchsia	

ونعود إلى الوسوم و خصائصها ...

```
<BODY BGCOLOR="#FFFFFF" BACKGROUND="image.jpg">
...
</BODY>
```

تقوم الخاصية BACKGROUND بتحديد صورة كخلفية (ورق جدران) للصفحة وقد استخدمت الصورة التالية:

والمسماة image.jpg في صفحتي وكانت هذه النتيجة



تلاحظ أن المتصفح قد قام بتكرار عرض الصورة بطريقة التجانب وأنها أصبحت تغطي كل الشاشة. بحيث حجبت أيضاً اللون الأبيض الذي حددناه كلون الخلفية (من خلال الخاصية BGCOLOR) والحقيقة أن اللون يظهر فقط عندما

لا نقوم باستخدام صورة ما كخلفية. ومع ذلك يفضل تحديده إحتياطاً خاصة وأن بعض المتصفحات القديمة توصف بأنها متصفحات نصية Text-Based Browsers (أي ليس بإمكانها عرض الصور). أو ربما هناك بعض المستخدمين الذين قاموا بإلغاء خيار عرض الصور تلقائياً من متصفحاتهم. إذن لنعطهم على الأقل فرصة مشاهدة بعض الألوان إن لم يستطيعوا مشاهدة الصور.

إننا نستطيع استخدام الصور بأحجام مختلفة طولياً أو عرضياً كخلفيات للصفحة، والمتصفح نفسه هو الذي يقوم تلقائياً بعرضها في وضع التجانب مما يعطي الانطباع بأنها صورة كبيرة.

ولنكمل مع باقي الخصائص في وسم <BODY>: ربما لاحظت خلال استخدامك للإنترنت أن معظم الوصلات التشعبية (Links) التي تنقر عليها لتنقلك إلى صفحات أو مواقع أخرى على الشبكة هي دائماً مميزة باللون الأزرق، وأن الوصلات التي قمت بزيارتها فعلاً قد تحول لونها إلى القرمزي. حسناً، هذه هي الألوان الإفتراضية التي تعتمدها المتصفحات. لكن قد لا يعجبك ذلك وتريد تغيير هذا النظام. أو ببساطة ربما تريد استخدام لون أو صورة غامقة لخلفية الصفحة بما سيؤدي إلى اختفاء هذه الوصلات أو حتى اختفاء نص الصفحة نفسها. فما العمل؟
إليك هذه الخصائص التي تقوم بالتحكم في ألوان النصوص:

TEXT="#rrggbb"	تحديد لون النص الأساسي للصفحة
LINK="#rrggbb"	تحديد لون الوصلات التشعبية
VLINK="#rrggbb"	تحديد لون الوصلات التشعبية التي تمت زيارتها visited links
ALINK="#rrggbb"	تحديد لون الوصلة التشعبية الفعالة أي عندما يتم النقر عليها active links

والآن، دعنا نجمل الخصائص السابقة في عبارة واحدة. وسوف أكتب الرموز الخاصة بالألوان بنفس تلك الألوان التي تمثلها. وألفت نظرك إلى أنه لا أهمية للترتيب في كتابة هذه الخصائص داخل العبارة.

```
<BODY BACKGROUND="backimag.jpg"
BGCOLOR="#ffff00"
TEXT="#000066"
LINK="#00ff00"
VLINK="#ff0000"
ALINK="#999999">
```

حاول أن تحللها! هل استنتجت أنني قد حددت الصورة backimag.jpg كخلفية للصفحة؟ وأنني اخترت اللون الأصفر للخلفية (في حالة عدم عرض الصورة السابقة كخلفية)؟ وان النص سيظهر باللون الأزرق الغامق؟ أما الوصلات التشعبية فلونها أخضر، والوصلات التي تمت زيارتها ستظهر باللون الأحمر. أما تلك الوصلة الفعالة فستظهر باللون الرمادي في لحظة النقر عليها بالفأرة.

إذا كانت هذه هي استنتاجاتك... فمبروك، لقد نجحت. وكل ما أتمناه أن تكون قد قضيت وقتاً ملوناً وزاهياً مع هذا الدرس.

الخطوط

أهلاً وسهلاً بك إلى الدرس الثالث من دروس HTML. لا زلنا نناقش معاً أساسيات تنسيق صفحات الإنترنت والتحكم بخصائصها. وسوف نتابع ذلك في هذا الدرس من خلال التعرف على الوسوم الخاصة بالخطوط.
سوف تلاحظ في هذا الدرس والدروس اللاحقة أن هناك أكثر من طريقة لأداء نفس العمل، أو إعطاء نفس الخصائص لصفحات الإنترنت. وبالمقابل قد يبدو لك أن بعض الوسوم والخصائص متشابهة في تأثيرها، لكن بالقليل من التدقيق والتجربة ستكتشف أن لكل وسم خصوصيته.

ولنبدأ

راجع صفحتنا البسيطة التي عملنا فيها في الدرسين السابقين. إننا لم نقم بالتعامل مع الخطوط فيها ولا بأي شكل من الأشكال. أي أننا تركناها على إعداداتها الافتراضية.
وبالمناسبة فإن هذه الإعدادات هي خط عادي، نوعه Times New Roman وحجمه 3 (بمقياس متصفحات الإنترنت).

الوسم الأول الخاص بالخطوط هو <FONT> ... <FONT/>
وهو يقوم بالتحكم بالخطوط من حيث النوع واللون والحجم. أما الخصائص التي نستخدمها مع هذا الوسم والوسوم الأخرى للخطوط فهي كالتالي:

Face	تقوم هذه الخاصية بتحديد نوع الخط الذي نريده، وقد نقوم بتحديد أكثر من نوع معاً. وفي هذه الحالة إذا لم يتواجد الخط المحدد أولاً على جهاز الشخص الذي يتصفح الموقع يتم إعتماد الخط الثاني ... وهكذا
	
	`<FONT FACE="Traditional Arabic, Arabic Transparent, Simplified Arabic">`
	`... Text ...`
	`</FONT>`
	
	طبعاً لا تنس أن تتأكد من كتابة أسماء الخطوط بالصورة الصحيحة هجائياً.

Color	أما هذه الخاصية فتحدد لون الخط، وذلك بنفس مبادئ تحديد الألوان التي تحدثنا عنها في الدرس السابق
	
	`<FONT COLOR="#FF0000">`
	`... Text ...`
	`</FONT>`

Size	ولتحديد حجم الخط نستخدم هذه الخاصية. وفقط هناك سبعة أحجام لأي خط تستطيع المتصفحات التعرف عليها.
	ونقوم بتحديد الحجم المطلوب بأسلوبين: أولهما المباشر. حيث يتم كتابة رقم يتراوح ما بين 1- 7. أي أننا نختار الحجم الذي نريده مباشرة.
	
	`<FONT SIZE="4">`
	`... Text ...`
	`</FONT>`

وإليك نماذج بأحجام الخطوط

خط بحجم 1
خط بحجم 2
خط بحجم 3 (الخط الافتراضي)
خط بحجم 4
خط بحجم 5
خط بحجم 6
خط بحجم 7

أما الأسلوب الثاني فهو النسبي: حيث تكتب الأرقام من 1 إلى 6 مرفقة إما بإشارة + أو بإشارة −.

```
<FONT SIZE="+4">
... Text ...
</FONT>
```

وفي هذه الطريقة فإن الأرقام 1−6 تمثل درجات التكبير (+) أو التصغير (−) للخط وذلك نسبة إلى الحجم الافتراضي. فمثلا الرقم +4 يعني تكبير الخط أربع درجات عن الحجم الافتراضي وهو 3، أي أنه يصبح بالحجم 7. بالمقابل فأن الرقم −1 يعني تصغير الخط درجة واحدة أي يصبح بالحجم 2.
ولتوضيح هذا الأسلوب، إليك هذه النماذج:

خط بحجم −3
خط بحجم −2
خط بحجم −1
خط بحجم +0 (أو −0 وهو الافتراضي)
خط بحجم +1
خط بحجم +2
خط بحجم +3
خط بحجم +4
خط بحجم +5

لاحظ أنه حتى في الأسلوب النسبي لا نستطيع الحصول على أكثر من سبعة أحجام للخطوط.

حتى و إن حاولنا كتابة أرقام أكبر أو أصغر كما فعلت هنا بكتابة الحجم −3 أو +5.	

والآن أعرف ماذا تريد أن تسأل، ستقول لقد ثبت حجم الخط على حده الأدنى عند الدرجة -2 وعلى حده الأعلى عند الدرجة +4. إذن ما الفائدة من وجود الدرجات الأخرى الأقل من -2 والأكبر من +4؟
حسنا وأنا أجيبك بسؤال آخر: ماذا لو قمنا بتغيير الحجم الافتراضي للخط في كل الصفحة إلى 1 بدلاً من 3؟ (وسوف نقوم بذلك فعلاً بعد قليل)، ألا نحتاج في هذه الحالة إلى الدرجات من +1 إلى +6 لتمثيل الأحجام الأكبر منه؟ وإذا قمنا بتحديد 7 كحجم إفتراضي ألا نحتاج إلى الدرجات من -1 إلى -6 لتمثيل الأحجام الأصغر منه؟ إذن نحن نحتاج فعلاً إلى هذه الدرجات لكي نغطي جميع الإحتمالات الواردة.
أرجو أن يكون هذا الجواب قد أقنعك :-)

وهذه بعض الأمثلة لتوضح لك كيفية استخدام هذا الوسم، وسوف أرفق نتيجة كل مثال بعده مباشرة.

```
<FONT FACE="arial" SIZE="6" COLOR="#FF0000">
This font is Arial, Size is 6, Color is Red
</FONT>
```

This font is Arial, Size is 6, Color is Red

```
<FONT FACE="arial" SIZE="+3" COLOR="#FF0000">
This font is Arial, Size is +3, Color is Red
</FONT>
```

This font is Arial, Size is +3, Color is Red

```
<FONT FACE="Times New Roman" SIZE="5" COLOR="#0000FF">
This font is Times New Roman, Size is 5, Color is Blue
</FONT>
```

This font is Times New Roman, Size is 5, Color is Blue

```
<FONT FACE="courier" SIZE="2" COLOR="#800000">
This font is Courier, Size is 2, Color is Maroon
</FONT>
```

This font is Courier, Size is 2, Color is Maroon

```
<FONT FACE="Arial" SIZE="5" COLOR="#00FF00"> This </FONT>
<FONT FACE="Times New Roman" SIZE="7" COLOR="#FF00FF"> is </FONT>
```

```
<FONT FACE="Arial" SIZE="2" COLOR="#FF0000"> multi </FONT>
<FONT FACE="Impact" SIZE="4" COLOR="#000000"> colors, </FONT>
<FONT FACE="Courier" SIZE="2" COLOR="#0000FF"> multi </FONT>
<FONT FACE="Times New Roman" SIZE="3" COLOR="#008080"> faces, </FONT>
<FONT FACE="Courier" SIZE="6" COLOR="#FFFF00"> and </FONT>
<FONT FACE="Arial" SIZE="5" COLOR="#808080"> multi </FONT>
<FONT FACE="Impact" SIZE="2" COLOR="#800000"> sizes </FONT>
<FONT FACE="Times New Roman" SIZE="7" COLOR="#00FFFF"> text </FONT>
```

This is multi colors, multi faces, and multi sizes text

```
<FONT FACE="Impact" SIZE="6" COLOR="#000000">C </FONT>
<FONT FACE="Impact" SIZE="6" COLOR="#008080">O</FONT>
<FONT FACE="Impact" SIZE="6" COLOR="#FF0000">L</FONT>
<FONT FACE="Impact" SIZE="6" COLOR="#0000FF">O</FONT>
<FONT FACE="Impact" SIZE="6" COLOR="#800000">R</FONT>
<FONT FACE="Impact" SIZE="6" COLOR="#FF00FF">S</FONT>
```

COLORS

ننتقل الآن إلى الوسم الثاني من الوسوم الخاصة بالخطوط وهو <BASEFONT>.
وعمله هو تحديد نوع الخط وخصائصه بالنسبة للصفحة كلها . أي أنه يقوم بتعريف نوع الخط الأساسي الذي سيستخدم في الصفحة من بدايتها إلى نهايتها ويحدد لونه وحجمه.
هل لاحظت انه وسم مفرد ولا يحتوي على وسم للنهاية؟ بالطبع ما الحاجة إلى وسم النهاية طالما أنه يتعامل مع الصفحة ككل ومع الإعدادات الأساسية لها، وليس مع كلمة أو سطر أو فقرة بذاتها. لذلك فإن هذا الوسم يكتب عادة في أول الملف، ويفضل مباشرة بعد وسم <BODY>. أما الخصائص المستخدمة معه فهي نفس الخصائص سالفة الذكر مع <FONT> ، (نستطيع استخدام الخاصية Name معه بدلاً من Face). وبنفس الطريقة وبدون أي اختلافات. وإليك هذه الشيفرة كمثال:

```
<BASEFONT Name="Arial" COLOR="#FF0000" SIZE="5">
```

وبدراسة هذا المثال نستنتج أنه يقوم بتعديل الخط الافتراضي للصفحة بحيث يصبح نوعه Arial وحجمه 5 ولونه أحمر.
وبالتالي فإن كل النصوص المكتوبة في تلك الصفحة سيطبق عليها هذا النمط من الخط. ما لم نقم طبعا باستخدام الوسوم <Font> ... <Font/> لتعديلها والتحكم بمظهرها كما فعلنا في الأمثلة السابقة، فهي أكثر تحديداً وأكثر مرونة من الوسم <BASEFONT>

وبمناسبة الحديث عن الألوان وتغيير اللون الأساسي لنص الصفحة. ألا تذكر أننا في الدرس السابق تكلمنا عن الخاصية Text التي تكتب مع الوسم <Body> والتي استخدمناها لتحديد لون نص الصفحة... أنا لا زلت أذكر ذلك.
لا يوجد تعارض بين هذه الخاصية وخاصية Color في الوسم <BASEFONT> فأنت بكل بساطة تستطيع استخدام أي منهما في صفحتك. وإذا حدث واستخدمت كلاهما فإن اللون المحدد مع الوسم <BASEFONT> هو الذي سيطبقه المتصفح ويعتمده.

هناك وسوم خاصة تستخدم لتمييز العناوين Headings في صفحات الإنترنت وهي:

</Hn> ... <Hn>

وحرف n هو رقم بين 1-6 يمثل مستوى العنوان.

```
<H1> Heading 1 </H1>
<H2> Heading 2 </H2>
<H3> Heading 3 </H3>
<H4> Heading 4 </H4>
<H5> Heading 5 </H5>
<H6> Heading 6 </H6>
```

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

ونأتي الآن إلى التنسيقات والتأثيرات التي يمكن إضافتها إلى النصوص. وفيما يلي الوسوم الخاصة بها متبوعة بمثال ونتيجته:

* الخط الغامق (الأسود العريض)، ونستخدم له الوسوم التالية:

```
<B> ... </B>
<STRONG> ... </STRONG>
```

This is **Bold Text**	<B> Bold Text </B>
This is **Strong Text**	<STRONG> Strong Text </STRONG>

* الخط المائل

```
<I> ... </I>
<EM> ... </EM>
```

This is *Italic Text*	<I> Italic Text </I>
This is *Emphasized Text*	<EM> Emphasized Text </EM>

* الخط المسطر

```
<U> ... </U>
```

This is <u>Undelined Text</u>	<U> Undelined Text </U>

* الخط المرتفع

^{...}

This is $^{\text{Superscript Text}}$	^{Superscript Text}

* الخط المنخفض

_{...}

This is $_{\text{Subscript Text}}$	_{Subscript Text}

* خط كبير

<BIG> ... </BIG>

This is Big Text	<BIG> Big Text </BIG>

* خط صغير

<SMALL> ... </SMALL>

This is Small Text	<SMALL> Small Text </SMALL>

* نص يعترضه خط

<STRIKE> ... </STRIKE>
<S> ... </S>

This is ~~Striked Text~~	<STRIKE> Striked Text </SRTIKE>
This is ~~Striked Text~~ too	<S> Striked Text </S>

* نص الآلة الطابعة TeleType

<TT> ... </TT>

This is `TeleType Text`	<TT> TeleType Text </TT>

وهذا النص يعرف أيضاً بالنص موحَد المسافات Monospaced Text. ولتوضيح هذا المفهوم إليك المثال التالي:
إذا أخذنا الحرفين m,i وكتبنا كل منهما عشر مرات متتالية نلاحظ أن المساحة التي شغلها الحرف m هي أضعاف المساحة التي شغلها الحرف i

iiiiiiiiii
mmmmmmmmmm

أما عند استخدام الوسم <TT> ... </TT> فإن المساحة التي يشغلها كلا الحرفين تصبح موحدة

iiiiiiiiii
mmmmmmmmmm

وهذه أمثلة تجمع بين عدة تنسيقات معاً:

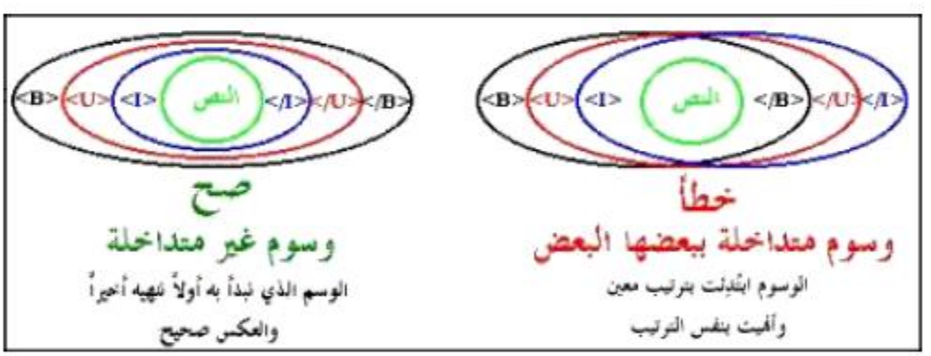
<B><I><U>
This is a Bold, Italic and Underlined Text
</U> </I> </B>

<u>This is a Bold, Italic and Underlined Text</u>

<FONT COLOR="#FF0000" SIZE="+3"><U><I>
This text is red, size +3, Italic, and Underlined
</I> </U> </FONT>

<u>*This text is red, size +3, Italic, and Underlined*</u>

وقد أردت من هذه الأمثلة توضيح مسائل معينة أولها: أن بإمكاننا استخدام عدة وسوم وتنسيقات معاً في نفس الوقت ولنفس المقطع من النص. (وذلك لجميع الوسوم وليس فقط لوسوم الخطوط). وكما ذكرت سابقاً، لا أهمية لترتيب هذه الوسوم ولا أيها ورد أولاً... **<u>لكن</u>**
عند استخدام الوسوم المتعددة في مقطع واحد يجب مراعاة عدم التداخل بينها!... كيف؟ أنظر إلى الرسم التالي:



فكتابة الوسوم السابقة بالطرق التالية هو خطأ:

<B><I><U>
This is a Bold, Italic and Underlined Text
</B> </I> </U>

<B><I><U>
This is a Bold, Italic and Underlined Text
</B> </U> </I>

أعرف أنك لم تصدقني وأنك قمت بتجربة هذه الوسوم وربما حصلت على نتيجة صحيحة. حسناً العبرة ليست في عبارة واحدة مكونة من وسمين أو ثلاثة تكتبها في ملف صغير بل في صفحة إنترنت كاملة قد تتألف من مئات أو حتى آلاف الوسوم مكتوبة في ملف خالٍ من الأخطاء المنطقية والتداخلات التي قد تسبب الإرباك للمتصفحات، وتؤدي إلى عدم عرض هذه الصفحة بالشكل المناسب والمطلوب.

لذلك فأهمية أن تتجنب وجود الوسوم المتداخلة في صفحتك هو بنفس الأهمية التي يجب أن توليها لكتابة هذه الوسوم بالصورة الصحيحة إملائياً. وإلا فالمتصفحات لا ترحم. وكثيرة هي المرات التي حصل فيها المصممون على صفحات منهارة بسبب نسيان حرف واحد أو إشارة مثل < أو > أو "

بإختصار شديد... وكقاعدة أساسية، الصفحة المصممة جيداً هي الصفحة ذات الوسوم الصحيحة وغير المتداخلة.

الفقرات والقوائم

في هذا الدرس سوف نناقش الوسوم الخاصة بالفقرات بشكل خاص وترتيب الصفحات وتنسيقها بشكل عام.
صحيح أن استخدامك للألوان والرسومات في الصفحة يضفي عليها نوعاً من الحيوية، وأن الخطوط تعطي صفحتك رونقاً وجمالاً. لكنك إن لم تهتم بترتيب صفحتك أو تقضي بعض الوقت في تنسيق هيكلها العام وتنظيم فقراتها وقوائمها، فإنه من الصعب عليك الحصول على صفحة ويب ناجحة. فالترتيب هو الخطوة الأولى لجذب اهتمام الزائر أو القارئ لصفحتك وتسهل عليه فهم الخطوط العريضة للصفحة.

لقد قمت في الدرس الأول بإيضاح بعض الوسوم الخاصة بالفقرات. ولا بأس من تذكيرك بها. فالوسم <P> يقوم بإنهاء الفقرة. والوسم
 ينهي السطر الحالي وينقل النص إلى سطر جديد. والوسم   يقوم بإضافة الفراغات، ويجب تكرار كتابته بنفس عدد الفراغات المطلوب.
ونتابع في هذا الدرس مع هذه الوسوم وغيرها.

لقد قلت إن الوسم <P> هو وسم مفرد لكنه يستخدم أيضاً كوسم مزدوج <P> ... </P> وفي هذه الحالة يمكننا من تحديد إتجاه الفقرة وإتجاه النص فيها حيث يستخدم معه الخصائص ALIGN, DIR.

فالخاصية ALIGN تحدد محاذاة الفقرة وهي تأخذ القيم Left, Right, Center وأوضحها بالأمثلة التالية:

```
<P Align="left"> This is a left-aligned paragraph </P>
```

This is left-aligned paragraph

```
<P Align="right"> This is right-aligned paragraph</P>
```

This is a right-aligned paragraph

```
<P Align="center"> This is a centered paragraph</P>
```

This is a centered paragraph

كذلك لتوسيط الفقرات أو الكائنات بشكل عام في الصفحة نستطيع استخدام الوسوم <CENTER> ... <CENTER/>

```
<CENTER> This is a centered text </CENTER>
```

This is a centered text

أما الخاصية DIR والتي نستخدمها أيضاً مع <P> فتقوم بتحديد إتجاه قراءة النص وتأخذ القيم

LTR	إتجاه النص من اليسار إلى اليمين (Left To Right)

RTL	إتجاه النص من اليمين إلى اليسار (Right To Left)

(تذكر هذه الخاصية جيداً فهي مهمة عند كتابة صفحات باللغة العربية)

ولتنسيق الفقرات أيضاً يوجد الوسوم
<BLOCKQUOTE> ... </BLOCKQUOTE> أي وسوم الفقرات المقتبسة. ووظيفتها تمييز الفقرة من خلال إدراج مسافة إضافية على الهامشين الأيمن والأيسر لها.

والحقيقة أنك تستطيع وضع عدة وسوم معاً إذا أردت إدراج هوامش أكبر. كما في المثال التالي:

```
<BLOCKQUOTE>
<BLOCKQUOTE>

النص هنا

</BLOCKQUOTE>
</BLOCKQUOTE>
```

وبالطبع ليس شرطاً أن تستخدم هذا الوسوم مع الفقرات المقتبسة فقط. فأنا مثلاً أضعها في بداية ونهاية كل صفحة من صفحات هذا الموقع. وبالتالي يظهر النص بعيداً قليلاً عن حاشية الصفحة فهذا أفضل من أن يكون ملاصقاً لها وأجمل.

والآن تأمل هذا الشكل وحاول أن تستنتج كيف قمت بإعداده...!؟

```
D   C   B   A
H   G   F   E
L   K   J   I
P   O   N   M
T   S   R   Q
```

ربما توصلت إلى أني استخدمت عدداً كبيراً من وسوم الفراغات   ونهاية السطر
. حسناً، إستنتاجك لا بأس به ولكنه ليس دقيقاً فأنا لم أستخدم أياً من هذه الوسوم هنا. بل كل ما فعلته بعد إعداد هذا الشكل هو وضعه ضمن:

<PRE> ... </PRE>

وهما اختصار لكلمة Preformated أي المنسق مسبقاً. وبالفعل فقد احتفظ هذا الشكل بالتنسيق المسبق الذي تم إعداده به. لكن تم تحويل الخط إلى خط موحد المسافات ولو لم أقم بوضعه ضمن هذه الوسوم لكانت النتيجة كالتالي:

A B C D E F G H I J K L M N O P Q R S T

لاحظ أن هذا الوسم يستخدم مع الفقرات التي لا نحتاج فيها إلى تنسيقات متعددة للخطوط أو الألوان. بل فقط مع الفقرات العادية موحدة الخط والتنسيقات.

القوائم

تحتوي لغة HTML على مجموعة من الوسوم الخاصة بتنظيم البيانات في قوائم وباستخدام عدة خيارات. وهناك نوعين من القوائم:
أولهما المتسلسلة Ordered Lists. واليك المثال التالي عليها

أسماء بعض المدن الفلسطينية

1. القدس
2. نابلس
3. رام الله
4. الخليل
5. جنين
6. طولكرم

وثانيهما القوائم غير المتسلسلة Unordered Lists وهذا مثال عليها

أسماء بعض الجامعات الفلسطينية

- جامعة النجاح
- جامعة القدس المفتوحة
- جامعة بيرزيت
- جامعة الخليل

عند التعامل مع القوائم بنوعيهما نحتاج إلى وسوم خاصة بتحديد بداية ونهاية القائمة ووسوم تحدد بنود هذه القائمة.
بالنسبة للقوائم المتسلسلة نستخدم الوسوم

</OL> ... <OL>

أما بالنسبة للقوائم غير المتسلسلة فنستخدم

</UL> ... <UL>

ولتعيين كل بند من بنود القائمة نستخدم الوسم <LI> وهو وسم مفرد يكتب في بداية السطر الخاص بكل بند List Item.

إذن عندما قمت بإنشاء القوائم السابقة استخدمت الشيفرة التالية:

```
<OL>
القدس<LI>
نابلس<LI>
```

```
الله رام<LI>
الخليل<LI>
جنين<LI>
طولكرم<LI>
</OL>



<UL>
النجاح جامعة<LI>
جامعة القدس المفتوحة<LI>
بيرزيت جامعة<LI>
جامعة الخليل<LI>
</UL>
```

والخاصية الوحيدة التي تستخدم مع هذه الوسوم هي TYPE ووظيفتها تحديد شكل الرمز الظاهر مع بنود القائمة، وعادة تستخدم مع وسوم بداية القوائم <UL> أو <OL> وبذلك نحدد رمزاً واحداً لكل القائمة. ولكن نستطيع استخدامها أيضاً مع وسم البنود <LI> لإعطاء تحكم أكبر في مظهر القائمة من خلال تحديد رمز مختلف لكل بند.

فعند وضعها ضمن تعريف القوائم المتسلسلة تأخذ القيم: A, a, I, i التي تغير رموز الترقيم من الأرقام العادية الإفتراضية (والتي رمزها 1) إلى ترقيم باستخدام الأحرف اللاتينية الكبيرة أو الصغيرة، أو باستخدام الأرقام الرومانية كما ترى في الجدول التالي:

<OL TYPE="i">	<OL TYPE="I">	<OL TYPE="a">	<OL TYPE="A">
.i	.I	.a	.A
.ii	.II	.b	.B
.iii	.III	.c	.C
.iv	.IV	.d	.D
.v	.V	.e	.E

والحديث عن هذه الخاصية يقودني إلى الحديث عن مسألة مهمة في لغة HTML وهي مسألة الوسوم والخصائص المحددة بمتصفح معين دون غيره أي التي تعمل مع أحد المتصفحات ولا تعمل مع غيره.

والسبب في ذلك أن هذه الخاصية تستخدم أيضاً مع القوائم غير المتسلسلة، لكن ليس بصورة مطلقة...كيف؟ أنت ترى أن الرمز الموجود عند كل بند في القائمة هو عبارة عن نقطة سوداء يطلق عليها اسم Disc وهي المعرفة ضمناً في خاصية TYPE. لكن هناك رموز أخرى يمكن إظهارها وهي المربع square, والدائرة المفرغة circle وتعرف بالشكل التالي:

```
<UL TYPE="square">
<UL TYPE="circle">
```

ولكن للأسف هذه الخاصية لا تعمل ولا يظهر تأثيرها إلا مع متصفح نيتسكيب وليس مع مايكروسوفت إكسبلورر الذي يتعامل فقط مع القيمة الإفتراضية للخاصية. (رجاءً لا يغضب مستخدمو إكسبلورر فهناك الكثير من الوسوم والخصائص التي لا يستطيع نيتسكيب عرضها أيضاً).
ولإتمام الحديث عن القوائم، أذكر لك أن هناك وسوماً أخرى تستخدم لإنشاء القوائم غير المتسلسلة، وبنفس الطريقة المستخدمة مع <UL>...</UL> وهذه الوسوم هي:

<DIR> ... </DIR>
<MENU> ... </MENU>

هناك نوع خاص من القوائم يدعى قوائم الشرح أو التعريفات Definition Lists وكما يدل الإسم تستخدم عندما نريد إدراج قائمة من المصطلحات يتبع كل واحد منها شرح أو تعليق.

HTML
Hyper Text Markup Language
WWW
World Wide Web
FTP
File Transfer Protocol
GIF
Graphical Interchange Format
JPG, JPEG
Joint Photographic Experts Group

ونحتاج لإنشاء هذه القوائم إلى ثلاثة وسوم:
الأول <DL> ... </DL> لتعريف بداية ونهاية القائمة.
والثاني <DT> ويوضع قبل كل مصطلح لتحديده، وهو وسم مفرد.
أما الثالث فهو <DD> وهو وسم الشرح أو التعليق وهو أيضا مفرد. ولنقم الآن بكتابة شيفرة القائمة السابقة

```
<DL>
<DT>HTML <DD>Hyper Text Markup Language
<DT>WWW <DD>World Wide Web
<DT>FTP <DD>File Transport Protocol
<DT>GIF <DD>Graphical Interchange Format
<DT>JPG, JPEG <DD>Joint Photographic Experts Group
</DL>
```

الصور والرسومات

في هذا الدرس سوف أقوم بالحديث عن الصور والرسومات وما يتعلق بالتعامل معها، بالإضافة إلى التعريف بأنواع الملفات الرسومية الدارجة في الإنترنت.

لقد اقتصر حديثنا عن الصور حتى الآن على إضافة خلفيات للصفحات، وكان ذلك في الدرس الثاني أما إدراج الصور ضمن الصفحات نفسها فله حكاية أخرى، أبدأ بروايتها لك الآن.

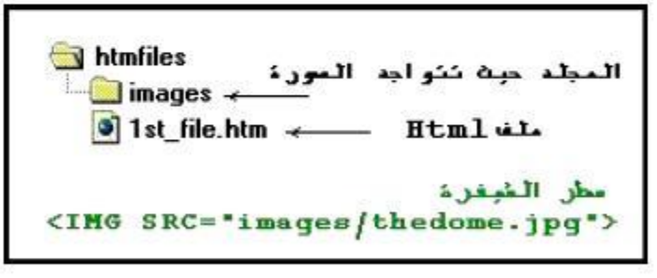
إن الوسم الرئيسي المستخدم لتعريف صورة ما داخل الصفحة هو <IMG> وهو وسم مفرد. لكن هل يكفي هذا لإدراج صورة؟ كلا، بالطبع يجب أن نحدد الصورة التي نريدها. لذلك نضيف الخاصية له SRC لتحديد موقع واسم الصورة.

الصورة التالية إسمها thedome.jpg وعندما قمت بإدراجها. كانت الشيفرة الخاصة بذلك هي

```
<IMG SRC="thedome.jpg">
```

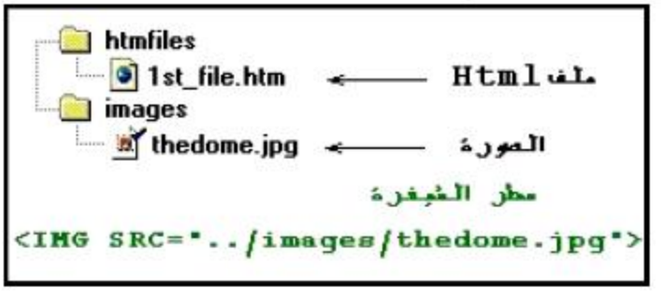
والصيغة هذه تفترض أن الصورة موجودة في ***نفس الدليل الفرعي أو المجلد*** حيث يتواجد ملف HTML الذي أعمل عليه، وقمت باستدعاء الصورة من خلاله. لكن ماذا لو كانت الصورة في مجلد فرعي آخر؟ حسنا سوف اناقش معك حالتين لهذه المسألة.

الحالة الأولى أن تكون الصورة موجودة في مجلد متفرع عن المجلد الموجود به ملف HTML حسب الشكل التالي:



نقوم في هذه الحالة بكتابة إسم هذا المجلد تتبعه إشارة / ثم اسم الصورة.

الحالة الثانية: أن يكون ملف HTML موجوداً في مجلد ما وتكون الصورة موجودة في مجلد آخر بنفس المستوى. أي أنهما مجلدين متجاورين وليسا متفرعين أحدهما عن الآخر.



وفي هذه الحالة نكتب .. (نقطتين) لتوجيه المتصفح للخروج من المجلد الفرعي الحالي (حيث يوجد ملف HTML) ومن ثم الدخول إلى المجلد images حيث توجد الصورة.

وبشكل عام، مهما كانت مواقع تواجد الملفات فإن عملية تحديد مواقعها والوصول إليها لا تخرج عن نطاق هذا النمط من الشيفرة. أي كتابة النقطتين للخروج من مجلد فرعي، وكتابة اسم المجلد الذي يجب الدخول إليه.

إن الأبعاد الأساسية لهذه الصورة هي 200×145 بيكسل Pixel وكما تلاحظ تم إدراج الصورة مع المحافظة على هذه الأبعاد. ومع ذلك فنحن نستطيع التحكم أيضاً بها وإظهار الصورة بالحجم الذي نريده من خلال هذا الوسم. كيف؟ بإضافة الخصائص HEIGHT, WIDTH متبوعة بأرقام تمثل الإرتفاع والعرض المطلوبين.

```
<IMG SRC="thedome.jpg" HEIGHT="70" WIDTH="120">
```

```
<IMG SRC="thedome.jpg" HEIGHT="300" WIDTH="500">
```

الخاصية التالية التي تستخدم مع <IMG> هي ALT وفيها نحدد نصاً بديلاً يظهر مكان الصورة. وهذا النص يلاحظ خصوصاً عندما يكون خيار "إظهار الصور تلقائياً" غير فعال في المتصفح. كما تستطيع ملاحظته في الفترة التي تسبق تحميل الصور وخاصة في المواقع بطيئة التحميل.

```
<IMG SRC="thedome.jpg" ALT="The Dome Of The Rock">
```

عندما نقوم بإدراج صورة ضمن فقرة فإن موقع ظهورها يتحدد بالطبع حسب ترتيب ورودها في الفقرة، مثلها مثل أي كلمة أو عبارة أخرى. ونستخدم الخاصية ALIGN لتحديد محاذاة الصورة مع النص المرافق لها أو لنقل بعبارة أخرى: ***تحديد موقع النص الذي يليها بالنسبة لها*** وهي تأخذ القيم: MIDDLE, LEFT, ,BOTTOM, TOP RIGHT وأوضح لك تأثير كل قيمة كما يلي:

في الحالة العادية (مثل هذه) وعندما لا نقوم بتحديد أي محاذاة فإن النص الذي يلي الصورة يظهر بمحاذاة الحافة السفلى لها. وهذه هي الحالة الإفتراضية لظهور الصور والتي تمثلها القيمة BOTTOM

```
<IMG SRC="image.jpg" ALIGN="BOTTOM">
```

TOP

وعند تحديد هذه القيمة فإن السطر الأول من النص الذي يلي الصورة يقع بمحاذاة الحافة العليا لها. أما باقي النص فيمتد أسفلها.

```
<IMG SRC="image.jpg" ALIGN="TOP">
```

MIDDLE

أما عند تحديد هذه القيمة فإن السطر الأول من النص يقع بمحاذاة منتصف الصورة. كذلك فإن باقي النص يمتد أسفلها.

```
<IMG SRC="image.jpg" ALIGN="MIDDLE">
```

LEFT

هذه القيمة تؤدي إلى محاذاة الصورة إلى أقصى اليسار. مع التفاف النص الذي يليها على الجهة اليمنى ولعدة أسطر حسب ارتفاع الصورة.

```
<IMG SRC="image.jpg" ALIGN="LEFT">
```

RIGHT

أما هذه القيمة فتؤدي إلى محاذاة الصورة إلى أقصى اليمين. مع التفاف النص الذي يليها على الجهة اليسرى ولعدة أسطر حسب ارتفاع الصورة.

```
<IMG SRC="image.jpg" ALIGN="RIGHT">
```

والآن بعد أن قمنا بتحديد محاذاة الصورة نحتاج إلى تحديد المسافة الفاصلة بينها وبين النص الذي يجاورها. ونستخدم لذلك الخصائص التالية:

VSPACE: لتحديد المسافة العمودية الفاصلة بين النص والحافتين العليا والسفلى للصورة.

HSPACE: لتحديد المسافة الأفقية الفاصلة بين النص والحافتين اليمنى واليسرى للصورة.

مثال:

```
<IMG SRC="image.jpg" ALIGN="RIGHT" VSPACE="20" HSPACE="20">
```

النتيجة: هذه الشيفرة ستدرج الصورة المسماه image.jpg مع محاذاتها ليمين الصفحة وإضافة مسافة فارغة مقدارها 20 بيكسل على الجهات الأربعة. (قارن بين هذا الإطار والإطار السابق الذي وضحت فيه خاصية ALIGN مع القيمة RIGHT. ولاحظ المسافة بين الصورة والنص المرافق لها.)

الخاصية الأخيرة والتي تستخدم مع الوسم <IMG> هي BORDER ووظيفتها إضافة إطار حول الصور والتحكم بسُمكِه. وهذه الخاصية تستخدم بشكل خاص عند تعيين صورة ما كوصلة تشعبية. ويتم التحكم بالسُمك من خلال إسناد رقم يمثل السُمك بالبيكسل. والقيمة الإفتراضية له هي 0 أي لا يوجد إطار حول الصورة.
مثلاً لإضافة إطار سُمكه 5 بيكسل نكتب الشيفرة التالية:

```
<IMG SRC="image.jpg" BORDER="5">
```

والآن حان الوقت لكي نناقش معاً بعض الأمور التي تتعلق بالصور والرسومات بشكل عام.

* هل حاولت أن تتعرف على أنواع الملفات الرسومية التي تقوم بتحميلها خلال تصفحك لمواقع الإنترنت؟
يزخر عالم الكمبيوتر بالعشرات من أنواع الملفات الرسومية وتنسيقات الصور. وكل منها يختلف عن غيره من عدة نواح، أذكر لك منها: الدقة، وعدد الألوان التي يستوعبها، والحجم التخزيني للملف. لكن هناك نوعين فقط من هذه الملفات يتم تداولهما حالياً في الإنترنت وهما:

JPEG ,JPG
إختصار لـِ Joint Photographic Experts Group. ويدعم هذا التنسيق صوراً بعيار 24 بت (أي 16.7 مليون لون). وميزة هذا التنسيق تتمثل في إمكانية ضغط الصور بنسب مختلفة عند تخزينها وبالتالي الحصول على صور صغيرة الحجم نسبياً.(أعني هنا حجم التخزين بالكيلوبايتات وليس أبعاد الصورة). لكن بالمقابل كلما إزدادت نسبة الضغط وصغر حجم الملف كان ذلك على حساب الجودة والوضوح.

GIF
إختصار لـِ Graphical Interchange Format وأقصى عدد للألوان في هذا التنسيق هو 265 لون. ومع ذلك فإن أحجام الصور المخزنة به كبير نسبياً مقارنة بتنسيق JPG. لكن هناك مزايا رائعة ينفرد بها تنسيق GIF مما يستدعي استخدامه في صفحات الويب، أولها القدرة على تخزين صور بخلفيات شفافة Transparent Images وثانيها الصور المتحركة Animated Gifs
والآن قد تسأل، أي من هذين التنسيقين أستخدم في صفحاتي؟! لا يوجد جواب قطعي لهذا السؤال لكن إليك هاتين المعادلتين:

JPG= الصور الحقيقية ذات العدد الكبير من الألوان، وذات الأبعاد الكبيرة
GIF= الصور قليلة الألوان وصغيرة الأبعاد مثل الأزرار.

---

* ما هي درجة إستبانة شاشتك Resolution? إذا كنت لا تعرف الجواب قم بفتح **لوحة التحكم** (control panel) في ويندوز وإختر أيقونة **العرض**(display) ثم اختر التبويب **إعدادات** (settings) وهناك سوف تشاهد **مساحة سطح المكتب** (screen resolution) الذي يدل على درجة إستبانة الشاشة، وعلى الأغلب ستكون 640×480 أو 800×600، وهناك درجات أعلى تعتمد على قدرة محول العرض. كذلك سوف تشاهد **لوح الألوان** (color quality) الذي يدل على عدد الألوان التي يمكن عرضها بالإعدادات الحالية للشاشة.
أما في ويندوز 3.11 أو 3.1 فاختر أيقونة برنامج إعداد Windows من لوحة التحكم فتظهر لك قائمة تجد بضمنها نوع وإستبانة الشاشة.

هذا الحديث يقودني إلى وحدة البيكسل Pixel وهي اختصار لـ Picture Element. إذا كانت شاشتك بإستبانة 640×480 فهذا يعني أنها مقسمة(نظرياً) إلى شبكة من 640 عمود و480 سطر. *وبمنتهى البساطة، إن كل خلية من هذه الشبكة تمثل بيكسل* وبالطبع كلما زادت الإستبانة كلما صغر حجم وحدة البيكسل.

---

* هل سبق لك وأن سمعت بمصطلح Thumbnail ضمن مصطلحات الإنترنت؟ حسناً، لا تلتفت إلى الترجمة الحرفية لهذه الكلمة، والتي تعني "ظفر الإبهام". فالمقصود حقيقة بها هي تلك الصورة الصغيرة جداً التي تقوم بالنقر عليها فتؤدي إلى عرض صورة بحجم أكبر. لذلك قد يكون المصطلح الأنسب لوصفها هو "العيّنة".
(وإذا كنت قد زرت أحد المواقع الإخبارية لرأيت كيف يتم عرض عينات وصور مصغرة للقطات الأحداث وعند النقر على العينة تظهر الصورة الأصلية. إذن أنت لست مجبراً على الإنتظار لوقت طويل لحين ظهور صورة ذات حجم كبير للقطة لست معنياً بها).

ومن الواضح أن استخدام العينات مفيد وعملي جداً وأن وضعها في المواقع التي تحتوي على العديد من الصور يؤدي إلى تقليل الزمن اللازم لتحميل الصفحات وتجنب ضياع الوقت بانتظار ظهور الصور الأصلية كبيرة الحجم. لأنها تعطي الزائر الحرية في النقر عليها إذا رغب في رؤية الأصل أو تجاهلها. أما كيف يتم عمل هذه العينات؟ فذلك باستخدام أحد برامج معالجة الرسوم كبرنامج Paint Shop Pro. بتصغير أبعاد الصور الأصلية إلى النسبة المطلوبة.

أعرف ماذا ستسأل الآن، ستقول ألم نتعلم قبل قليل كيفية عرض الصور مع التحكم بأبعادها؟ ألا يؤدي استخدام الخصائص WIDTH, HEIGHT إلى التحكم بحجم الصور وعرضها بنسب مصغرة حسب ما هو مطلوب؟
إن استخدامك لهذه الخصائص يؤدي إلى إظهار الصورة بحيث تبدو مصغرة، لكنك فعلياً قمت بإجبار متصفح الزائر على تحميل الصورة بالحجم والأبعاد الأصلية ثم عرضها بالحجم المصغر أي أنك في النهاية لم تحقق الغاية من وجود هذه العينات.

الوصلات التشعبية

Links... أو الوصلات التشعبية هي روح الإنترنت. وإذا كانت الإنترنت بمجملها هي شبكة العنكبوت فإن هذه الوصلات هي الخيوط التي تشكل هذه الشبكة وتؤلف حلقات الوصل بين الملايين من مواقعها. تنقر على وصلة ما فتنقلك إلى صفحة أخرى في نفس الموقع... وتنقر على وصلة أخرى لتنقلك كلياً إلى أحد المواقع في الجانب الآخر من العالم... وصلة تجعلك تحمّل ملفاً وأخرى تجعلك تشغّل مقطعا موسيقياً وثالثة تعرض لك صورة...
حسناً، من المؤكد أنك استنتجت الآن من هذه المقدمة أنك بصدد تعلم كيفية إدراج الوصلات التشعبية في صفحاتك... لقد صدق استنتاجك لذلك هيا إلى العمل...

هناك عدة خيارات للوصلات التشعبية، منها أن تكون الوصلة لموقع آخر، أو أن تكون لصفحة أخرى داخل الموقع نفسه، ومنها أن تكون لمكان آخر في نفس الصفحة (إلى أعلى أو أسفل على سبيل المثال) أو أن تكون وصلة لعنوان بريد إلكتروني E-mail وفي جميع الحالات فإن المبدأ واحد لكن تختلف بعض التفاصيل. وسوف أناقش معك كل حالة على حدة وبالتفصيل.

نستخدم الوسوم <A> ... </A> كوسوم أساسية لإدراج الوصلات التشعبية، وهي اختصار لكلمة Anchor. وهي لا تعمل لوحدها بل تتطلب إضافة خصائص معينة أولها وأهمها الخاصية **HREF** التي نحدد من خلالها الموقع الذي نريد الدلالة عليه، ويجب أن يكتب عنوان الموقع كاملاً.

<u>الحالة الأولى:</u> إدراج وصلة تشعبية تشير إلى موقع خارجي.
لنقم بإدراج وصلة تشعبية إلى أحد المواقع العربية الرائدة والرائعة، وهو موقع شركة صخر. وعنوانه http://www.sakhr.com في هذه الحالة يتم كتابة الشيفرة بالشكل التالي:

```
<A HREF="http://www.sakhr.com"> </A>
```

لكن بقي شيء واحد وهو العبارة أو الكلمة التي سيتم النقر عليها لتشغيل الوصلة، وهذه يجب أن توضع بين الوسمين <A> ... </A>. أي لكي تكتمل الوصلة السابقة يجب أن نكتب معها أي عبارة نريدها، لكي ينقر عليها الزائر فتنقله إلى العنوان المطلوب. ما رأيك بعبارة: Go To SAKHR والتي تصبح الشيفرة معها بالشكل التالي:

```
<A HREF="http://www.sakhr.com">Go To SAKHR</A>
```

وتظهر الوصلة كما يلي:

Go To SAKHR

لم تعجبك؟ ليس ذلك مشكلة فأنت تستطيع كتابة أي شيء تريده كعنوان للوصلة التي تريدها. ما رأيك لو جعلنا كلمة SAKHR هي فقط العنوان لهذه الوصلة.

```
Go To <A HREF="http://www.sakhr.com">SAKHR</A>
```

Go To SAKHR

بل إنك تستطيع إدراج صورة أو (زر) كبديل عن الكلمات –كما تشاهد في الكثير من المواقع– وكل ما عليك فعله في هذه الحالة هو كتابة الوسم الخاص بإدراج الصورة بين الوسمين <A> ... </A> بالشكل التالي:

```
<A HREF="http://www.sakhr.com"><IMG SRC="sakhrlgo.gif"></A>
```

والذي يؤدي إلى ظهور الصورة التالية كوصلة تشعبية لموقع صخر

وبشكل عام فإن أي شيء يوضع بين الوسمين <A> ... </A> سوف يكون الوسيلة أو العنوان الذي ينقلنا إلى الموقع المشار إليه في الوصلة التشعبية، سواءً كان هذا الشيء نصاً أو صورة أو كلاهما معاً.

والآن هل تلاحظ الإطار الظاهر حول الصورة؟ وهل تذكر متى قمنا بالحديث عن هذا النوع من الإطارات؟ نعم، في الدرس السابق. عند إدراج صورة كوصلة تشعبية يظهر حولها إطار سمكه 2 بيكسل وهذه هي الحالة الإفتراضية. وبالطبع نستطيع إزالته بكتابة الخاصية BORDER="0" ضمن وسم الصورة.

```
<A HREF="http://www.sakhr.com"><IMG SRC="sakhrlgo.gif" BORDER="0"></A>
```

أو حتى تكبيره بكتابة السمك المطلوب مع هذه الخاصية.

```
<A HREF="http://www.sakhr.com"><IMG SRC="sakhrlgo.gif" BORDER="6"></A>
```

ننتقل الآن إلى الحالة الثانية، وهي أن تشير الوصلة التشعبية إلى ملف موجود في نفس الموقع (أي ملف محلي) سواءً كان ملف HTML أو صورة أو غير ذلك. وفي هذه الحالة فإن ما يكتب مع الخاصية HREF هو اسم هذا الملف المطلوب الوصول إليه.

لنقم بإنشاء وصلة تشعبية تقودنا إلى الصفحة الرئيسية لهذا الموقع ولنفترض أن الملف الذي يحتويها اسمه index.html والشيفرة الخاصة بذلك هي:

```
<A HREF="index.html">Main Page</A>
```

Main Page

وأذكرك بأنك تستطيع إدراج صورة هنا أيضاً كعنوان للوصلة التشعبية وبنفس التفاصيل التي شرحتها في الحالة السابقة.

هيا ندرج صورة مصغرة كعنوان لوصلة تشعبية للصورة الأصلية.

```
<A HREF="nablus1.jpg"><IMG SRC="nablus_1.jpg" BORDER="0"></A>
```

في هذا المثال قمت بتحديد الصورة المصغرة المسماه nablus_1.jpg كوصلة تشعبية تصلنا إلى الصورة الأصلية المسماه nablus1.jpg

لكن إنتبه ✋ إذا كان الملف المطلوب والذي تريد الإشارة إليه موجوداً في مجلد مختلف عن المجلد الذي يوجد به الملف الحالي، فيجب عليك تحديد المسار الكامل لهذا الملف وذلك بنفس الطريقة التي ناقشناها في الدرس السابق عندما قمنا بإدراج الصور.

الحالة الثالثة هي أن نقوم بالإشارة إلى مكان آخر داخل نفس الصفحة، إلى أولها مثلاً أو إلى آخرها أو أي مكان آخر نريده...
طبعاً مهما بلغت درجة الذكاء والألمعية التي يتصف بها الكمبيوتر ومتصفح الإنترنت، فهما لا يستطيعان معرفة ما يدور بفكرك وبالتالي لا يستطيعان معرفة المكان الموجود في نفس الصفحة والذي تريد نقل زائرك إليه من خلال الوصلة التشعبية. لذلك يجب أن تقوم أنت بتحديده.

والمبدأ هنا هو أن تقوم بتعريف أو تسمية هذا المكان بإسم معين سوف تقوم لاحقا باستخدامه في الوصلة التشعبية. وفي هذه الحالة يتحتم عليك استخدام الخاصية الثانية للوسم <A> وهي NAME

لنقم معاً بإدراج وصلة تشعبية تقوم بنقل الزائر من مكان وجود الوصلة إلى فقرة أخرى
أول ما يجب فعله هو الذهاب إلى هذه الفقرة واختيار أول كلمة فيها ثم وضعها داخل الوسوم <A> ... </A>

```
<A>LINKS</A>
```

والآن حان الوقت لاستخدام الخاصية NAME فالخطوة الثانية هي تعريف هذه الكلمة بأي اسم نريده (المهم أن نبقى متذكرين له). سوف أقوم بإعطاء الاسم attrib1

```
<A NAME="attrib1">LINKS</A>
```

لقد أصبحت هذه الفقرة جاهزة لكي نقوم بإدراج وصلات تشعبية إليها من أي مكان في الملف، بل ومن أي ملف آخر ... وأكثر من ذلك أنه إذا أراد أحد ما في أحد المواقع الأخرى أن يضع وصلة تشعبية لها من موقعه فإن باستطاعته ذلك شرط أن يعرف الإسم الذي عرّفناها به وهذا ليس صعباً بالطبع.

الخطوة الثالثة هي إدراج الوصلة التشعبية لهذه الفقرة.
ويلزمنا هنا معرفة اسم الملف الذي توجد به هذه الفقرة (أي الملف الذي نعمل به) واسمه htutor06.html لأنه سيشكل المدخل الأساسي للوصول إلى الفقرة المحددة. وتكون شيفرة الوصول إلى هذه الفقرة هي كالتالي:

```
<A HREF="htutor06.html#attrib1">3rd Paragraph</A>
```

3rd Paragraph

لاحظ أننا لم نكتف بذكر اسم الفقرة لوحدها بل يجب أن تقرن باسم الملف الأب الذي يتضمنها من خلال إشارة #

أما الحالة الأخيرة والتي نقوم فيها بإدراج وصلة تشعبية لعنوان بريد إلكتروني، يؤدي النقر عليها إلى إطلاق برنامج البريد الإلكتروني للزائر بشكل تلقائي. فالإختلاف الوحيد الذي يطرأ هنا هو كتابة كلمة MAILTO بعد خاصية HREF لكي تدل على أن العنوان الذي يلي هو عنوان EMAIL وليس أي عنوان آخر

```
<A HREF="MAILTO:yahya@palnet.com"> Email Me </A>
```

Email Me

والآن بعد أن انتهيت من سرد أساسيات استخدام الوصلات التشعبية وإدراجها في صفحات الويب بقي هناك بعض التوضيحات والملاحظات التي أجد أن من المهم ذكرها لك.
عندما قمنا بالتوصيل إلى عنوان خارجي، سواء كان لموقع ويب أو عنوان Email لاحظنا أننا استخدمنا كلمات مفتاحية ميزت طبيعة هذا العنوان، وأعطت المتصفح فكرة عن طبيعة التعامل مع هذا العنوان وطريقة الاتصال به. فعندما أردنا التشعب إلى موقع الويب كتبنا كلمة HTTP والتي تدل على نوع البروتوكول المستخدم في الاتصال بهذا الموقع، وهو بروتوكول نقل النصوص المتشعبة HyperText Transfer Protocol وعندما كتبنا عنوان Email استخدمنا كلمة MAILTO قبل هذا العنوان. وبالتالي قمنا بالإيعاز للمتصفح بفتح برنامج البريد الإلكتروني الافتراضي وتجهيزه لإرسال رسالة إلى العنوان المدرج. وحتماً لقد صادفت مثل هذه الحالة كثيراً خلال تجولك في مواقع الويب.

لكن هنا مجالات أخرى لاستخدام الإنترنت ولكل منها بروتوكوله الخاص. فمثلاً هناك الآلاف من المزودات المنتشرة عبر الإنترنت والتي تحتوي على أعداد هائلة لا تحصى من الملفات والبرامج الجاهزة للتحميل ويتم الوصول إليها عبر بروتوكول خاص لنقل الملفات يدعى FTP (File Transfer Protocol). ومن هذه المجالات أيضاً والتي لا تقل أهمية عن الويب أو البريد الإلكتروني المجموعات الإخبارية Groups News أو مجموعات النقاش التي تختص كل منها بمناقشة موضوع معين. وهذه تعمل من خلال بروتوكول NNTP (Network News Transfer Protocol).

إن تعدد مجالات إستخدام الإنترنت وتعدد البروتوكولات فيها لا يعني أنك تحتاج لأن يكتظ سطح مكتبك بالعديد من البرامج للتعامل معها. فمعظم المتصفحات التي نستخدمها تحتوي على برامج خاصة تدعم هذه الخدمات.
فمثلاً عند النقر على عنوان مزود FTP يتم الدخول إليه مثله مثل أي موقع ويب عادي وتظهر قائمة المجلدات والملفات فيه بشكل مشابه للمستكشف في Windows. أما النقر على عنوان إحدى المجموعات الإخبارية فيؤدي إلى سلوك مشابه للنقر على عناوين البريد الإلكتروني، أي إطلاق برنامج تصفح خاص بالمجموعات الإخبارية يكون مدمجاً ضمن حزمة المتصفح الأصلي.

والآن... أعتقد أنه ليس من الصعب عليك استنتاج الكيفية التي نضيف بها وصلات تشعبية لمزود FTP.
إليك هذا العنوان لأحد المزودات التي يحتوي على الكثير من البرامج المجانية أو المشتركة

ftp://ftp.simtel.net/pub/simtelnet/win95/

وكل ما عليك فعله هو كتابة الشيفرة التالية:

```
<A HREF="ftp://ftp.simtel.net/pub/simtelnet/win95/">Simtel FTP Server</A>
```

Simtel FTP Server

أما بالنسبة للمجموعات الإخبارية فتكتب الوصلات التشعبية لها باستخدام الكلمة المفتاحية NEWS. فعلى سبيل المثال، لوضع وصلة تشعبية لمجموعة النقاش alt.html الخاصة بمناقشة لغة HTML نكتب الشيفرة التالية:

```
<A HREF="news:alt.html">Alt.Html</A>
```

Alt.Html

الجداول

هذا الدرس سيكون الأول من درسين حول الجداول. وقد فضلت تجزئتها إلى قسمين وذلك لأهمية هذا الموضوع وتعدد خصائص الجداول واحتمالات استخدامها في صفحات الويب.

تعد الجداول من أقوى الأدوات التي تتضمنها لغة HTML وأكاد أجزم بأنه لا يوجد موقع في الإنترنت إلا ويستخدمها بصورة أو بأخرى. والحقيقة أن وضع الجداول في صفحات الويب لا يقتصر على تلك القوائم من البيانات التي نحتاج لترتيبها في صفوف وأعمدة، بل يتعدى ذلك إلى استخدامها في تصميم الصفحات نفسها وتنظيمها، والتحكم بمظهرها بصورة قوية وفعالة لا يمكن أداؤها مهما استخدمنا من وسوم خاصة بتنسيق الصفحات.

إن التعامل مع الجداول وإدراجها في صفحات الويب سهل جداً مثله مثل أي أداة من أدوات HTML لكنه قد يبدو لك مربكاً بعض الشيء وخاصة في البداية، وذلك لتعدد الخصائص التي تستعمل معها وتعدد الأوجه التي نستطيع التصرف بها. لكن لا تقلق فكل شيء يبدو صعباً في بدايته ولكن سرعان ما يصبح سهلاً.

هل أنت مستعد؟ إذن هيّا بنا...

بداية، إليك هذا الوصف البسيط للوسوم الأساسية الخاصة بالجداول

<TABLE>...</TABLE>	وسوم تعريف الجدول
<TR>...</TR>	Table Row وسوم تعريف الصف في الجدول
<TD> *Cell Data* </TD>	Table Data وسوم تعريف الخلايا في الصف وتعريف محتويات كل خلية

والآن لنتكلم بصورة أكثر دقة وتفصيلاً:

هذه هي الوسوم التي نبدأ بها لإدراج جدول مكون من خلية واحدة أو من مليون خلية... الأمر سيان

</TABLE> ... <TABLE>

والآن بعد إدراج هذين الوسمين، هناك سؤالين ينبغي الإجابة عليهما. الأول: كم عدد الصفوف التي نريدها في الجدول؟ ثلاثة، أربعة، مائة؟ لا بأس، قم بإضافة الوسوم

</TR> ... <TR>

بنفس عدد الصفوف التي تريدها. ولنفترض هنا أنها ثلاثة.

```
<TABLE>

<TR>
</TR>

<TR>
</TR>

<TR>
</TR>

</TABLE>
```

والسؤال الثاني هو، كم عدد الخلايا (أو الأعمدة) التي نريدها في كل صف؟
وهنا نضيف الوسوم

</TD> ... <TD>

بنفس عدد الخلايا المطلوب. ومن البديهي أن نكتبها بين الوسوم <TR> ... </TR> طالما أن الخلايا هي جزء من الصفوف. وهنا سأفترض أننا نريد خليتين في كل صف، وبذلك يجب تكرار كتابتها مرتين لكل صف.
وأذكرك أن النص الذي نريد إدراجه في الخلية يكتب ضمن هذين الوسمين.

```
<TABLE>

<TR>

<TD> Data </TD>
<TD> Data </TD>

</TR>

<TR>

<TD> Data </TD>
<TD> Data </TD>

</TR>

<TR>

<TD> Data </TD>
<TD> Data </TD>

</TR>

</TABLE>
```

هل اتضحت لك الصورة الآن. أنظر إلى نتيجة العمل التي حصلنا عليها.

Data	Data
Data	Data
Data	Data

هناك شيء ما ينقص. بالطبع ... الحدود. انتظر قليلاً وستعرف الخاصية التي تقوم بإضافة الحدود للجدول وغيرها من الخصائص الأخرى. لأنني قبل أن أستمر أود أن ألفت نظرك لمسألة معينة في الجداول. وهي أن طريقة التعامل معها تتم على ثلاثة مستويات:

- مستوى الجدول ككل
- مستوى الصفوف ككل أو كل واحد على حده
- مستوى الخلايا ككل أو كل واحدة على حده.

ولكل من هذه المستويات خصائصه التي ينفرد بها كما أن هناك خصائص مشتركة تستخدم مع كل الوسوم.

نبدأ بمناقشة الخصائص التي تستخدم مع الوسوم <TABLE> ... </TABLE> وسأقوم أولاً بسردها لك، ومن ثم إدراج بعض الأمثلة التي توضحها.

BORDER	تقوم هذه الخاصية بإضافة حدود للجدول وتحديد سماكتها، والقيمة الإفتراضية لها هي صفر أي لا حدود <TABLE BORDER="5"> <TABLE BORDER="0">
WIDTH	نستخدم هذه الخاصية لتحديد عرض الجدول ككل. وهناك أسلوبين لتحديد العرض: المطلق أي بكتابة الرقم الذي يمثل العرض بصورة مباشرة. والنسبي أي كتابة رقم نسبي مئوي يحدد عرض الجدول حسب عرض نافذة المتصفح. (أي أن عرض الجدول سيختلف باختلاف عرض نافذة المتصفح). <TABLE WIDTH="600"> <TABLE WIDTH="80%">
HEIGHT	لتحديد ارتفاع الجدول، ويكون تحديد هذا الإرتفاع من خلال قيمة مطلقة تحدد الإرتفاع بالبيكسل. أو قيمة نسبية تحدد ارتفاع الجدول بالنسبة لإرتفاع صفحة المتصفح <TABLE HEIGHT="500"> <TABLE HEIGHT="100%">
CELLSPACING	لتحديد المسافة بين كل خلية من خلايا الجدول <TABLE CELLSPACING="10">
CELLPADDING	لتحديد المسافة الفاصلة بين الحدود وبداية النص في كل خلية. أو لنقل: تحديد حجم الهوامش لخلايا الجدول. <TABLE CELLPADDING="10">

ALIGN	لتحديد محاذاة الجدول أفقياً على الصفحة يميناً أو يساراً. وهو يأخذ القيم left ,right <TABLE ALIGN="Left"> <TABLE ALIGN="Right">
BGCOLOR	ويستخدم لتحديد لون الخلفية للجدول <TABLE BGCOLOR="#00FFFF">

هذه هي الخصائص المستعملة مع الجدول. وسأقوم الآن بتطبيقها على المثال الوارد في بداية هذا الدرس وسأكتفي بكتابة وسم البداية أما باقي الوسوم فهي نفسها:

<TABLE BORDER="5">

Data	Data
Data	Data
Data	Data

<TABLE BORDER="5" CELLPADDING="5">

Data	Data
Data	Data
Data	Data

<TABLE BORDER="5" CELLPADDING="5" CELLSPACING="10">

Data	Data
Data	Data
Data	Data

```
<TABLE BORDER="5" CELLPADDING="5" CELLSPACING="10"
BGCOLOR="#FFFF00">
```

Data	Data
Data	Data
Data	Data

```
<TABLE BORDER="5" CELLPADDING="5" CELLSPACING="10"
BGCOLOR="#FFFF00" HEIGHT="300">
```

Data	Data
Data	Data
Data	Data

```
<TABLE BORDER="5" CELLPADDING="5" CELLSPACING="10"
BGCOLOR="#FFFF00" HEIGHT="300" WIDTH="75%">
```

Data	Data
Data	Data
Data	Data

ونتكلم الآن عن الخصائص المستخدمة مع وسوم الصف <TR> ... </TR> ولا بأس من تذكيرك أن عدد الصفوف في الجدول يتحدد بعدد هذه الوسوم. أما أهم الخصائص التي تضاف لهذا الوسم فهي:

ALIGN	لتحديد محاذاة النص أفقياً داخل الخلايا التي يتكون منها الصف، والقيم المحتملة لها هي Right, Left, Center والقيمة الإفتراضية هي Center
VALIGN	لتحديد المحاذاة العمودية للنص داخل خلايا الصف، وذلك إما للأعلى أو للأسفل أو في المنتصف أو على امتداد الخط الأساسي للخلية. وقيمها على التوالي هي: Baseline ,Top, Bottom, Middle
BGCOLOR	لتحديد لون الخلفية للخلايا التي يتكون منها الصف. وهنا يتم تجاهل لون الخلفية المحدد ضمن وسم <TABLE> ويتم تطبيق اللون المحدد هنا.

ونعود الآن إلى جدولنا السابق لنطبق عليه هذه الخصائص من خلال الأمثلة التالية:

```
<TABLE BORDER="5" HEIGHT="300">
<TR ALIGN="Left">
<TD> Data </TD>
<TD> Data </TD>
</TR>

<TR ALIGN="Right">
<TD> Data </TD>
<TD> Data </TD>
</TR>

<TR ALIGN="Center">
<TD> Data </TD>
<TD> Data </TD>
</TR>
</TABLE>
```

Data	Data
Data	Data
Data	Data

```
<TABLE BORDER="5" HEIGHT="300">
<TR VALIGN="Top">
<TD> Data </TD>
<TD> Data </TD>
</TR>

<TR VALIGN="Bottom">
<TD> Data </TD>
<TD> Data </TD>
</TR>

<TR VALIGN="Baseline">
<TD> Data </TD>
<TD> Data </TD>
</TR>
</TABLE>
```

Data	Data
Data	Data
Data	Data

```
<TABLE BORDER="5" HEIGHT="300" BGCOLOR="#FFFFFF">
<TR BGCOLOR="#808080">
<TD> Data </TD>
<TD> Data </TD>
</TR>

<TR BGCOLOR="#C0C0C0">
<TD> Data </TD>
<TD> Data </TD>
</TR>

<TR>
<TD> Data </TD>
<TD> Data </TD>
</TR>
</TABLE>
```

Data	Data
Data	Data
Data	Data

```
<TABLE BORDER="0" HEIGHT="100%" WIDTH="100%">
<TR BGCOLOR="#808080">
<TD> Data </TD>
<TD> Data </TD>
</TR>

<TR BGCOLOR="#C0C0C0">
<TD> Data </TD>
<TD> Data </TD>
</TR>

<TR BGCOLOR="#FFFFFF">
<TD> Data </TD>
<TD> Data </TD>
</TR>
</TABLE>
```

Data	Data
Data	Data
Data	Data

نتابع معاً في هذا الدرس الحديث عن الجداول. وأنا أفترض أنك قد أنهيت الدرس السابق بنجاح، وأن لديك الآن فكرة جيدة جداً عن الجداول وكيفية إنشائها والتعامل مع خصائصها ومع الصفوف وخصائصها. ونكمل الآن من حيث توقفنا، أي مع خصائص الخلايا.

هل تذكر ما قلناه عن عدد الخلايا في الصف الواحد؟ إن عدد الخلايا المطلوب يتحدد من خلال كتابة الوسوم <TD> ... </TD> مرات بنفس العدد المطلوب. ومن الممكن أن تحتوي الخلية على أي عنصر من عناصر لغة HTML : نصوص، رسوم، قوائم، وصلات تشعبية، بل وحتى جداول. (نعم، تستطيع إدراج جدول داخل جدول آخر)

لنسترجع معا المثال الذي قمنا بالتدرب عليه في الدرس السابق، فسوف نكمل هذا الدرس معه. وهو جدول صغير مكون من ثلاثة صفوف وعمودين (أي خليتين في كل صف).

```
<TABLE>

<TR>

<TD> Data </TD>
<TD> Data </TD>

</TR>

<TR>

<TD> Data </TD>
<TD> Data </TD>

</TR>

<TR>

<TD> Data </TD>
<TD> Data </TD>

</TR>

</TABLE>
```

أما الخصائص المستخدمة مع الخلايا، فهذا جدول بها:

ALIGN	تحدد محاذاة النص الموجود في الخلية أفقياً، والقيم المستخدمة هي Left, Center, Right
VALIGN	تحدد المحاذاة العمودية للنص، وهو يأخذ القيم Top, Middle, Bottom, Baseline

WIDTH	تحدد عرض الخلية، وذلك بكتابة القيمة المباشرة للعرض المطلوب بالبيكسل، أو بكتابة رقم يمثل النسبة المئوية. ويكفي تحديد العرض للخلايا في أحد الصفوف لكي يتم تطبيقه على كل الخلايا في كل الصفوف.
HEIGHT	تحدد الإرتفاع المطلوب للخلية في الصف، وذلك بالطرق المباشرة أو النسبية. وقيامك بتحديد ارتفاع إحدى الخلايا في الصف يؤدي إلى تطبيقه على كل الخلايا فيه.
BGCOLOR	تحدد لون خلفية الخلية
COLSPAN	يقوم بدمج الخلية الحالية مع العدد المطلوب من الخلايا التي تليها أفقياً <TD COLSPAN="n"> حيث n هو عدد الخلايا التي سيتم دمجها
ROWSPAN	يقوم بدمج الخلية الحالية مع العدد المطلوب من الخلايا التي تليها عمودياً (أي أسفلها). <TD ROWSPAN="n"> وبالطبع n هو عدد الخلايا التي سيتم دمجها

وقبل أن نستمر، يبدو لي أن هناك بعض الملاحظات المهمة التي ينبغي ذكرها:

- كما تلاحظ هناك خصائص تتكرر مع جميع الوسوم. خذ مثلاً الخاصية BGCOLOR. كيف يتم التعامل معها إذا كررت مع جميع الوسوم؟ بكل بساطة يتم تطبيق اللون المحدد مع وسم الخلية، فإذا لم يكن محدداً يطبق اللون المحدد مع وسم الصف، فإذا لم يوجد يطبق اللون المحدد مع وسم الجدول. وإذا لم يكن هذا محدداً بدوره يتم إعتماد لون خلفية الصفحة المحدد في الوسم <BODY>.
- الملاحظة الثانية تتعلق بالخصائص WIDTH, HEIGHT. يختلف أسلوب التعامل مع هذه الخصائص من متصفح لآخر، بل وتختلف أيضاً طريقة تفسير القيم المحددة معها وخصوصاً فيما يتعلق بالنسب المئوية. (راجع الموضوع: الوسوم الخاصة والمتصفحات).
وبدون الخوض في تفاصيل هذه الإختلافات التي لن تؤدي إلا إلى المزيد من الإشكالات لديك... وبعد التجربة يبدو أن أفضل طريقة للتعامل مع هذه الخصائص هي قيامك بتحديد العرض (وكذلك الإرتفاع إذا أردت ذلك) للجدول ككل من خلال الوسم <TABLE>. ثم استخدام هذه الخصائص في وسوم الخلايا وتحديد العرض المطلوب لكل خلية على حده في الصف الأول، والارتفاع المطلوب لكل صف في الجدول.
وهذه برأيي أفضل طريقة تضمن بها أفضل مشاهدة للجدول لجميع زوار موقعك.
- إذا أردت أن تحتوي بعض الصفوف في الجدول على عدد من الخلايا أقل من باقي الصفوف، فلا يكفي أن تقوم بحذف وسوم الخلايا منها. (كما ترى في الشيفرة التالية:)

```
<TABLE BORDER="5">
    <TR>
```

```
            <TD> Data </TD>

        </TR>

        <TR>

            <TD> Data </TD>
            <TD> Data </TD>

        </TR>

        <TR>

            <TD> Data </TD>

        </TR>

    </TABLE>
```

لأن هذا ما ستحصل عليه:

	Data	
Data	Data	
	Data	

لقد بقي مكان الخلايا المحذوفة محجوزاً كما لو أنها لم تحذف. أما الخلايا الباقية فظلت محتفظة بنفس خصائصها، أي أننا لم نستفد من عملية الحذف. والحقيقة أن الطريقة المثلى لذلك هي أن تقوم **بدمج** الخلايا معاً وذلك باستخدام الخصائص COLSPAN, ROWSPAN.

إذن لنقم بإعادة كتابة شيفرة الجدول مع استخدام هذه الخصائص:

```
<TABLE BORDER="5">

<TR>

<TD COLSPAN="2"> Data </TD>

</TR>

<TR>

<TD> Data </TD>
<TD> Data </TD>
```

```
</TR>

<TR>

<TD COLSPAN="2"> Data </TD>

</TR>

</TABLE>
```

	Data
Data	Data
	Data

لاحظ أن العدد 2 هو عدد الخلايا التي قمنا بدمجها. ولاحظ أيضاً انني لم أقم بإعادة وسوم الخلايا المحذوفة لأننا أصلاً لا نحتاج لها بعد أن قمنا بالدمج. **وكقاعدة أساسية: كل خلية يتم دمجها يجب بالمقابل حذف وسوم التعريف الخاصة بها. ما عدا تعريف الخلية الأساسية بالطبع.**

مثال آخر: لنقم بدمج الخلايا الموجودة في العمود الأول

```
<TABLE BORDER="5">

<TR>

<TD ROWSPAN="3"> Data </TD>
<TD> Data </TD>

</TR>

<TR>

<TD> Data </TD>

</TR>

<TR>

<TD> Data </TD>

</TR>

</TABLE>
```

ومرة أخرى بعد تعريف خاصية الدمج العمودي، قمت بحذف تعريف الخلايا المدموجة من الصف الثاني والثالث. وهذا هو الجدول الناتج.

Data	Data
Data	

Data	

هناك نوع خاص من الخلايا التي يتم تعريفها باستخدام الوسوم <TH> ... </TH> وهي اختصار Table Header أي ترويسة الجدول.
والفرق الوحيد بينها وبين <TD> ... </TD> هو أن النص الذي تحتويه يظهر بخط أسود عريض ومحاذاته في منتصف الخلية بصورة إفتراضية. (ليس بالشيء المهم، كما أعتقد)، خاصة وأن الخصائص المستخدمة معها هي نفس خصائص <TD> وبنفس التفاصيل التي ذكرت.

الوسوم الأخيرة المستخدمة في الجداول هي <CAPTION> ... </CAPTION> وهي تختص بإضافة عنوان رئيسي للجدول ككل. لذلك فهي عندما تكتب يتم وضعها مباشرة بعد الوسم <TABLE> وبصورة مستقلة وليس ضمن وسوم الصفوف أو الخلايا.

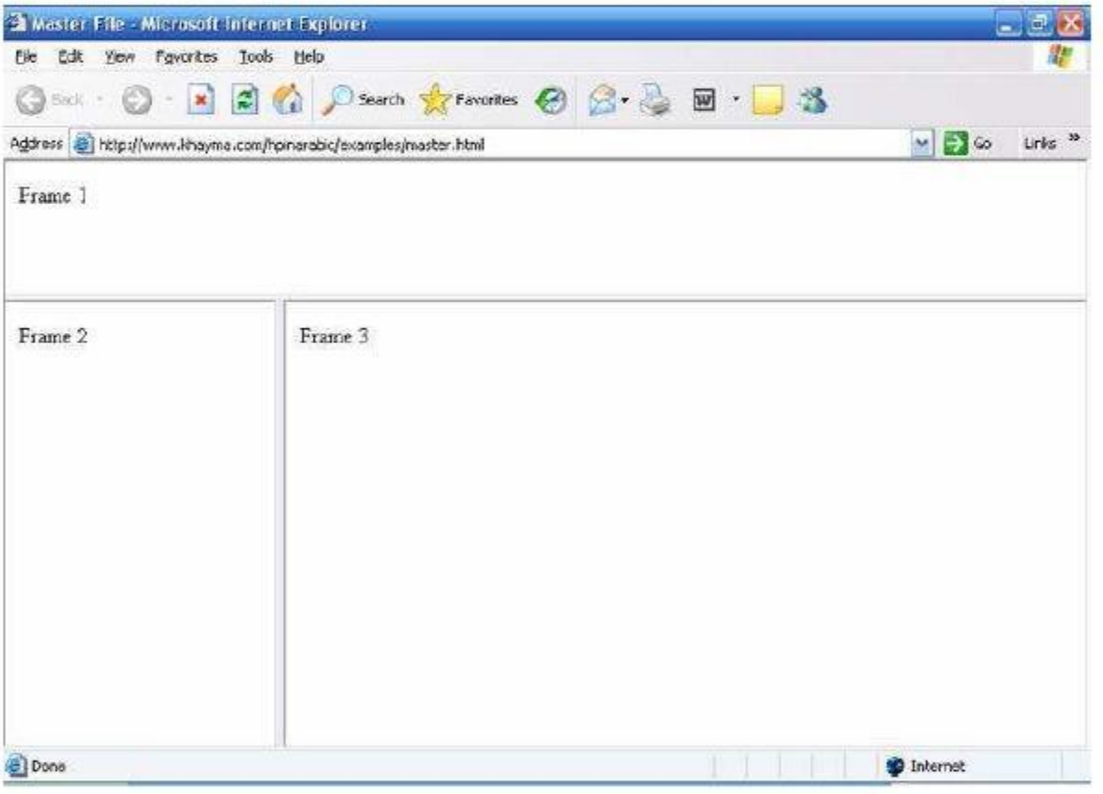
```
<TABLE BORDER="5">
   <CAPTION> Table Caption </CAPTION>

<TR>

<TD> Data </TD>
<TD> Data </TD>

</TR>

<TR>

<TD> Data </TD>
<TD> Data </TD>

</TR>

<TR>

<TD> Data </TD>
<TD> Data </TD>

</TR>

</TABLE>
```

Table Caption	
Data	Data
Data	Data
Data	Data

الإطارات

في هذا الدرس والدرسين التاليين سوف نقوم بالتعرف على الإطارات Frames وطريقة عرض صفحات الويب باستخدامها...

فهل تعرف ما هي الإطارات؟ حسنا، سأوضحها لك... هل سبق لك وأن زرت إحدى الصفحات لتشاهد أنها مقسمة إلى عدة أقسام بحيث يظهر في كل منها صفحة مستقلة، وتبدو بصورة منفصلة عن الأقسام الأخرى. وربما تكون قد قمت بالنقر على إحدى الوصلات التشعبية الموجودة في أحد الأقسام لتظهر الصفحة المتعلقة بها في القسم الآخر.
إذا كنت قد شاهدت مثل هذه الصفحات فهذا يعني أن الإطارات مألوفة لديك وإلا فشاهد مثالاً على صفحة ذات إطارات



كما تشاهد، فإن الصفحة مكونة من ثلاثة أقسام: علوي وأيسر وأيمن... والحقيقة أن كل قسم منها هو عبارة عن ملف Html كامل ومستقل بحد ذاته. وهي مجرد صفحات عادية لا تختلف أبداً عن تلك التي تعلمت إنشاءها في الدروس السابقة، ولا علاقة لكل منها بالصفحات الأخرى من حيث التركيب والتعريف.
أما كيف تم جمعها معا لتظهر بالشكل الذي تشاهده؟ فهنا بيت القصيد. فبالإضافة إلى الصفحات والملفات الإعتيادية يوجد دائماً ملف أساسي يتم إنشاؤه خصيصا لتعريف صفحة الإطارات وتجميعها وتحديد خصائصها. أي أن المعادلة تتلخص بـــ:

مكونات صفحة الإطارات = عدد ملفات الصفحة نفسها + صفحة الملف الأساسي الذي يجمعها.

أي أنني في المثال السابق إحتجت فعلياً إلى أربعة ملفات لتكوين الصفحة.

وقبل أن نبدأ... لنقم بالتحضير للأمثلة التي سترد في هذا الدرس. لذلك قم بإنشاء ثلاثة أو أربعة ملفات بسيطة لكي تستخدمها في تطبيق الأمثلة أو استخدم ملفاتك القديمة التي قمت بالتدرب عليها في الدروس السابقة. أنا قمت بإنشاء ملفات على النمط التالي (وهي التي استخدمتها في المثال) وأسميتها frame1.html, frame2.html, frame3.html

```
<HTML>
<HEAD>
<TITLE>Frame1</TITLE>
</HEAD>
<BODY>
Frame 1
</BODY>
</HTML>
```

ونبدأ الآن بتعريف الملف الرئيسي الذي سيضم كافة الإطارات والملفات. وهو بالمناسبة ملف ذو حالة خاصة حيث نقوم باستخدام الوسوم

<FRAMESET> ... </FRAMESET>

بدلاً من الوسوم <BODY> ... </BODY>

((إذن الملف الرئيسي للإطارات لا يتضمن تعريفا باستخدام BODY))

```
<HTML>
<HEAD>
<TITLE>Master File</TITLE>
</HEAD>

<FRAMESET>
</FRAMESET>

</HTML>
```

نأتي الآن إلى الخصائص: والخاصية الأولى التي تستخدم مع هذه الوسوم هي COLS وهي تعرّف عدد وأحجام الإطارات العمودية للصفحة. وتُحدد الأحجام بطريقتين (هل عرفتهما؟) نعم... إنهما الطريقة المباشرة والطريقة النسبية...أو كلاهما معا.

والآن إليك هذه الأمثلة التي توضح مفهوم الأعمدة... **لكن إنتبه! فهذه الشيفرة غير مكتملة وكتابتها بهذا الشكل فقط لن يؤدي إلى أي نتيجة ولا إلى ظهور أي إطارات** حيث ينقصها وسوم أخرى خاصة بمصدر الملفات الظاهرة داخل الإطارات، وسأقوم بسرد باقي الخصائص المهمة لاحقاً.

يحدد إطارين عموديين حجم كل منهما 50% من حجم الشاشة	<FRAMESET COLS="50%,50%"> </FRAMESET>
يحدد ثلاثة إطارات أحجامها 20% و 50% و 30% على التوالي من حجم الشاشة	<FRAMESET COLS="20%,50%,30%"> </FRAMESET>
يحدد ثلاثة إطارات عمودية الأول حجمه 200 بيكسل، والثاني 300 بيكسل،	<FRAMESET COLS="200,300,*"> </FRAMESET>

أما الثالث * أي انه غير محدد بحجم معين ولكنه سيكون بالحجم المتبقي من الشاشة (طالما أننا لا نعرف استبانة الشاشة التي يستخدمها زائر الموقع)	
يحدد أربعة إطارات حجم الأول هو 200 بيكسل، والثالث 15% من حجم الشاشة، والرابع 20% من حجم الشاشة أما الثاني فسيكون حجمه بما تبقى من الشاشة.	<FRAMESET COLS="200,*,15%,20%"> </FRAMESET>
يحدد ثلاثة إطارات الأول حجمه 150 بيكسل.... أما المساحة المتبقية فتقسم على أساس أن الإطار الثالث حجمه هو ضعفي (*2) حجم الإطار الثاني (*)	<FRAMESET COLS="150,*,2*"> </FRAMESET>

أما الخاصية الثانية فهي ROWS وأعتقد أنك استنجت طبيعة عملها. نعم هي تحدد عدد وحجم الإطارات الأفقية (الصفوف) داخل الصفحة. وذلك بنفس الأسلوب المتبع مع الأعمدة، أي إما باستخدام الطريقة النسبية أو المطلقة. وسأقوم بسرد بعض الأمثلة لتوضيحها (وأذكرك ثانية أن هذه الأمثلة غير مكتملة):

يحدد إطارين أفقيين ارتفاع كل منهما 50% من ارتفاع الشاشة	<FRAMESET ROWS="50%,50%"> </FRAMESET>
يحدد ثلاثة إطارات أفقية ارتفاعاتها 20% و 50% و 30% على التوالي من ارتفاع الشاشة	<FRAMESET ROWS="20%,50%,30%"> </FRAMESET>
يحدد ثلاثة إطارات أفقية الأول ارتفاعه 50 بيكسل، والثاني 120 بيكسل، والثالث سيكون بالإرتفاع المتبقي من الشاشة	<FRAMESET ROWS="50,120,*"> </FRAMESET>
يحدد أربعة إطارات أفقية ارتفاع الأول هو 50 بيكسل، والثالث 15% من ارتفاع الشاشة، والرابع 20% من ارتفاع الشاشة أما الثاني فسيكون ارتفاعه بما تبقى من ارتفاع الشاشة.	<FRAMESET ROWS="50,*,15%,20%"> </FRAMESET>
يحدد إطارين الثاني ارتفاعه ضعفي ارتفاع الأول	<FRAMESET COLS="*,2*"> </FRAMESET>

لم ننته بعد من ذكر كل الخصائص المتعلقة بالوسوم <FRAMESET> فلا زال هناك الكثير. ولكن من الضروري أن نقوم الآن بالإنتقال إلى وسم آخر للإطارات لأنه مرتبط إرتباطاً وثيقاً بالوسوم السابقة وخصائصها المذكورة أعلاه، وهي <FRAME> فما هو عمل هذا الوسم؟

حسنا، كل ما قمنا به حتى الآن هو تعريف مجموعة من الإطارات وخصائصها (فقط تعريف الإطارات) لكن لم نحدد ماهية هذه الإطارات ولا محتوياتها ولا مصادرها. تماما كما نقوم بتعريف صفحات الويب الإعتيادية وخصائصها في الوسم <BODY> دون أن يعني ذلك تحديد محتويات هذه الصفحات. فإذا أردنا فيما بعد إدراج صورة مثلاً نستخدم الوسم الخاص بذلك وهو <"IMG SRC="imagname.ext>

وفي حالة الإطارات فإننا نستخدم الوسم <FRAME> وهو وسم مفرد أي ليس له وسم نهاية تماماً مثل <IMG>. وفيه نقوم بتحديد مصدر وخصائص كل ملف نريد إظهاره داخل أحد الإطارات. ويتم استخدام هذا الوسم مرات بنفس

عدد الإطارات المذكورة داخل <FRAMESET>. وسوف أقوم مباشرة باستخدام الخاصية SRC لتحديد مصدر الملف.

دعنا نقوم الآن بإتمام الشيفرة لبعض الأمثلة المذكورة أعلاه. ونبدأ بالمثال الأول:

```
<FRAMESET COLS="50%,50%">
   <FRAME SRC="frame1.html">
   <FRAME SRC="frame2.html">
</FRAMESET>
```

الآن ... والآن فقط أصبح لديك صفحة إطارات محترمة.

مثال آخر:

```
<FRAMESET COLS="200,400,*">
   <FRAME SRC="frame1.html">
   <FRAME SRC="frame2.html">
   <FRAME SRC="frame3.html">
</FRAMESET>
```

مثال ثالث:

```
<FRAMESET ROWS="50,*,15%,20%">
   <FRAME SRC="frame1.html">
   <FRAME SRC="frame2.html">
   <FRAME SRC="frame3.html">
   <FRAME SRC="frame4.html">
</FRAMESET>
```

ورابع:

```
<FRAMESET COLS="*,2*">
   <FRAME SRC="frame1.html">
   <FRAME SRC="frame2.html">
</FRAMESET>
```

وبالإضافة إلى ما ذكر، نستطيع إدراج صورة مباشرةً داخل الإطار وباستخدام <FRAME SRC> تماماً كما ندرجها باستخدام <SRC IMG> وإليك هذا المثال:

```
<FRAMESET COLS="50%,50%">
   <FRAME SRC="frame1.html">
   <FRAME SRC="thedome.jpg">
</FRAMESET>
```

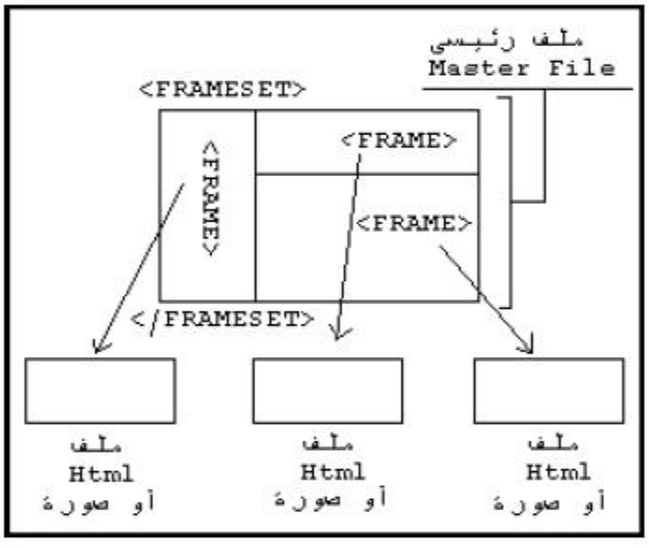
والآن لماذا لا نقم معاً بمراجعة الأفكار الأساسية السابقة الذكر وتلخيصها؟ وهذه هي:-

- لإدراج صفحة إطارات نحتاج إلى ملف رئيسي يعتبر بمثابة الوعاء الذي سيضم هذه

الإطارات.

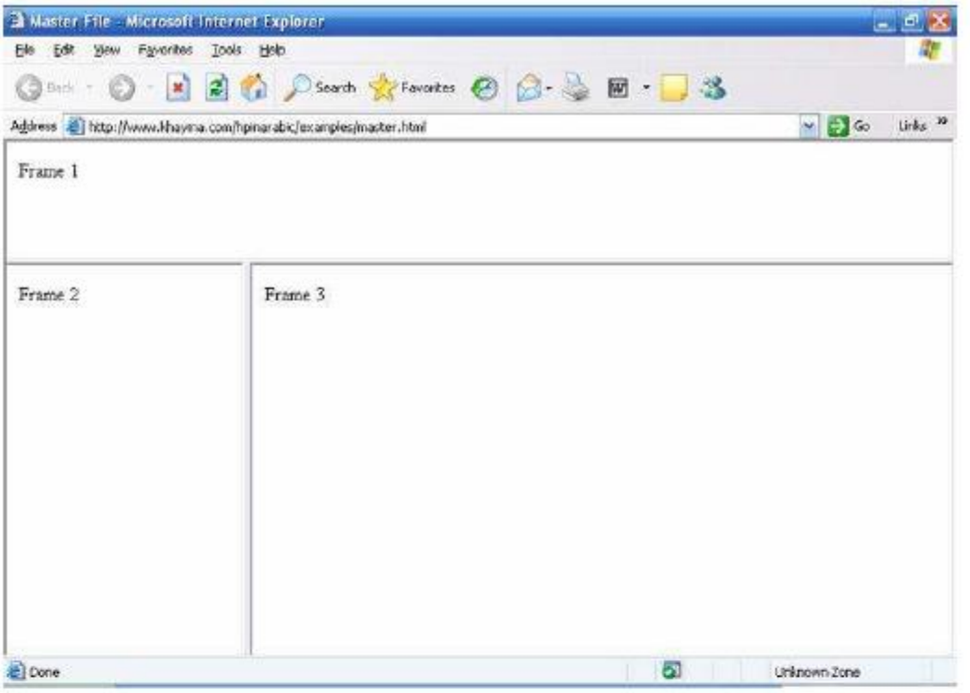
- الملف الرئيسي هو ملف HTML إعتيادي غير أننا نكتب الوسوم <FRAMESET> ...</FRAMESET> بدلاً من <BODY> ...</BODY>. وبالتالي فهو يحتوي على الخصائص التي نريدها للإطارات وتعريفاتها.
- نستخدم الخصائص COLS, ROWS لتحديد عدد الإطارات (صفوفاً كانت أو أعمدة) وأحجامها.
- الملفات الفرعية التي تظهر ضمن الإطارات هي ملفات عادية كالتي قمنا بإنشائها في الدروس السابقة أو صوراً. ولا تحتوي على أي تنسيق أو وسوم خاصة.
- نستخدم الوسم <FRAME> داخل الملف الرئيسي لمناداة الملفات الفرعية داخل الإطارات، وذلك مع الخاصية SRC. بالإضافة إلى استخدامه لتحديد باقي الخصائص .

كما نستطيع تمثيل هيكلية الإطارات من خلال الشكل التالي:

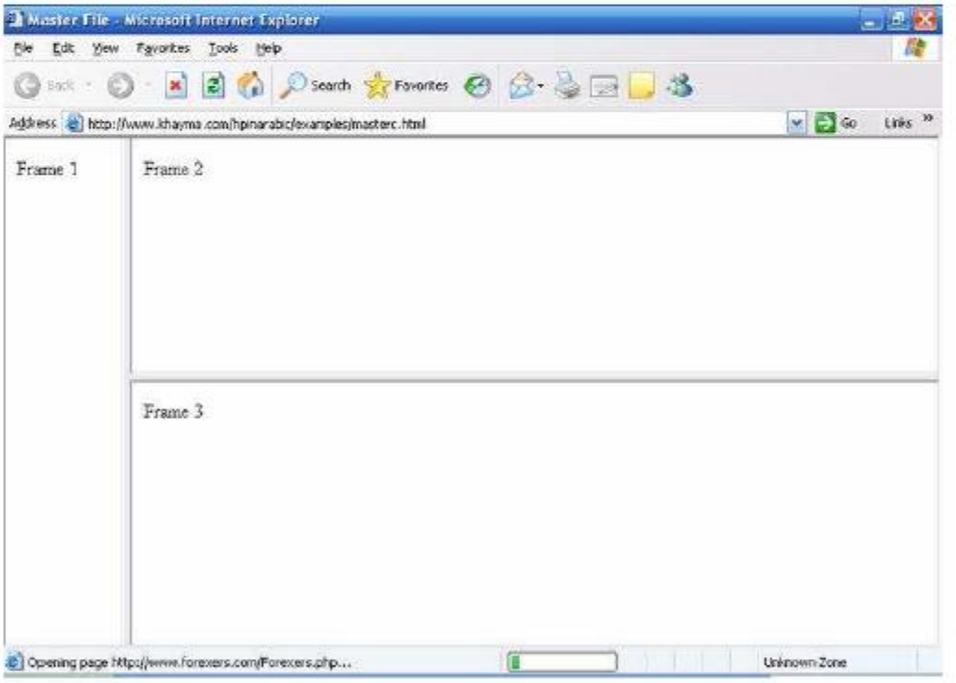


هل تأكدت من فهمك لهذه النقاط؟ لنتابع إذن ...
حتى الآن قمنا بتقسيم الصفحة إما لإطارات أفقية أو لإطارات عمودية. لكن نحتاج لمعرفة كيفية إدراج كلاهما في الصفحة. كما في الأمثلة التالية:

- صفحة مكونة من صفين، الثاني منهما مقسم بدوره إلى عمودين



- صفحة مكونة من عمودين، الثاني منهما مقسم بدوره إلى صفين



لنبدأ بالمثال الأول:
بما أن الصفحة تحتوي على صفين نقوم بتعريفهما أولاً حسب الإرتفاعات المرغوب بها:

```
<FRAMESET ROWS="100,*">
   <FRAME SRC="frame1.html">
   <FRAME SRC="frame2.html">
</FRAMESET>
```

لكن الصف الثاني مقسم إلى عمودين وهنا يعتبر بمفهوم لغة HTML وكأنه صفحة إطارات جديدة لذلك لا نحتاج لتعريفه كصف وبدلاً من ذلك نعاود استخدام تعريف الصفحات! أي <FRAMESET> مرة أخرى.

```
<FRAMESET ROWS="100,*">
  <FRAME SRC="frame1.html">

  <FRAMESET>
  </FRAMESET>

</FRAMESET>
```

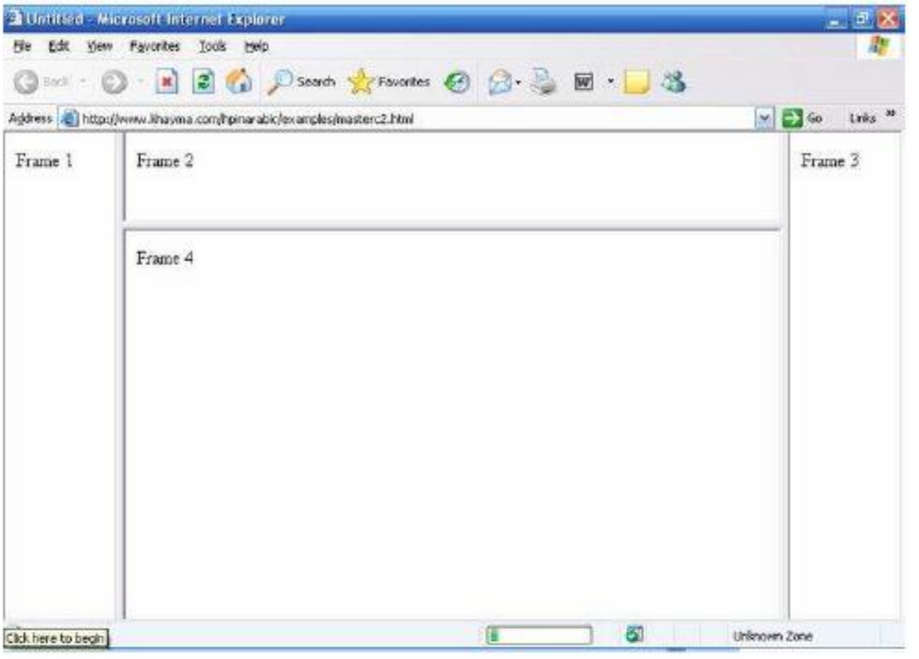
وبما أن الصف الثاني (أو لنقل الإطار الثاني) مقسم إلى عمودين، إذن بقي علينا إضافة تعريف لهذه الأعمدة. وبذلك تكون الشيفرة النهائية كالتالي:

```
<FRAMESET ROWS="100,*">
  <FRAME SRC="frame1.html">

  <FRAMESET COLS="200,*">
    <FRAME SRC="frame2.html">
    <FRAME SRC="frame3.html">
  </FRAMESET>

</FRAMESET>
```

الشيفرة الخاصة بالمثال الثانيهمة فهذه هي

```
<FRAMESET COLS="100,*">
  <FRAME SRC="frame1.html">

  <FRAMESET ROWS="200,*">
    <FRAME SRC="frame2.html">
    <FRAME SRC="frame3.html">
  </FRAMESET>

</FRAMESET>
```

لنقم الآن بإدراج مثال آخر وتحليله:



يوجد لدينا ثلاثة أعمدة، أليس كذلك؟ إذن لنقم بتعريف صفحة إطارات ذات ثلاثة أعمدة (طبعاً لا يوجد أهمية للأحجام المذكورة، فأنا اخترتها حسب رغبتي وتستطيع أنت اختيار الأحجام التي تريدها).

```
<FRAMESET COLS="100,*,100">
   <FRAME SRC="frame1.html">
   <FRAME SRC="frame2.html">
   <FRAME SRC="frame3.html">
</FRAMESET>
```

العمود الأوسط من هذه الصفحة مقسم إلى صفين، إذن نستبدل تعريفه بتعريف آخر لصفحة إطارات مكونة من صفين (وهذا هو التعريف بصورة مستقلة)

```
<FRAMESET ROWS="80,*">
   <FRAME SRC="frame2.html">
   <FRAME SRC="frame4.html">
</FRAMESET>
```

وبعد دمج الشيفرتين السابقتين معاً نحصل على هذه الشيفرة النهائية:

```
<FRAMESET COLS="100,*,100">
   <FRAME SRC="frame1.html">

   <FRAMESET ROWS="80,*">
      <FRAME SRC="frame2.html">
      <FRAME SRC="frame4.html">
   </FRAMESET>

   <FRAME SRC="frame3.html">
</FRAMESET>
```

وصلنا الآن إلى نهاية وقد قمنا في هذا الدرس بمناقشة أساسيات إدراج الإطارات، ولكن بقي هناك الكثير ليقال في هذا المجال. وهو ما سنكمله في الدرسين القادمين

لا زلنا نناقش موضوع الإطارات . لقد تعرفت في الدرس السابق على الأساسيات في هذا الموضوع، وتعلمت كيفية إنشاء صفحة ويب باستخدام مبدأ الإطارات. ونتابع معاً التعرف على باقي الخصائص والتحديدات المتعلقة بها.

بداية، أجد من الضروري أن أذكرك بالوسمين الأساسيين للإطارات واللذين ندرجهما في الملف الأساسي، وهما <FRAMESET> والذي يوضع بدلاً من <BODY> ويشكّل الوسم الأساسي لتحديد مجموعة الإطارات في الصفحة وخصائص هذه المجموعة ككل. و <FRAME> الذي يوضع داخل نطاق الوسم السابق ويعرّف الملف المصدر لكل إطار ويحدد خصائصه. ومن المهم جداً أن نميز بين الخصائص المتعلقة بكل وسم وأن لا نخلط بينهما.

وعند الحديث عن الإطارات سوف نواجه تلك المشكلة العتيدة التي تؤرق دائماً مصممي صفحات الويب، ألا وهي مسألة توافق الوسوم والخصائص مع المتصفحات المختلفة (والتي قمت بتوضيحها في الدرس السادس عشر).

فمثلاً لدينا أربع خصائص للوسم <FRAMESET> ، لكن واحدة منها فقط تعمل مع كلا المتصفحين الرئيسيين Netscape, MS Explorer. وكما إعتدنا سوف لن أركز على هذه الخصائص ، لكن على الأقل سأكتفي بذكرها وذكر عملها وأترك لك حرية تجربتها إن أردت.

أولى هذه الخصائص هي FRAMEBORDER وهي تقوم بتحديد ظهور أو عدم ظهور الحدود حول الإطارات وتأخذ القيم 1 للظهور، و0 لعدم الظهور. وهذا مثال عليها:

```
<FRAMESET ROWS="50,*,15%,20%" FRAMEBORDER="0">
  <FRAME SRC="frame1.html">
  <FRAME SRC="frame2.html">
  <FRAME SRC="frame3.html">
  <FRAME SRC="frame4.html">
</FRAMESET>
```

أما الخصائص الثلاث الأخرى فهي:

- BORDER: تحدد سمك الحدود الظاهرة حول الإطارات وهي تأخذ قيماً بالبيكسل. (BORDER="n") وتعمل فقط مع Netscape
- BORDERCOLOR: لإضافة لون للحدود (BORDERCOLOR="rrggbb") وتعمل مع Netscape أيضاً.
- FRAMESPACING: لتحديد مسافات فارغة إضافية حول الإطارات وتأخذ قيماً بالبيكسل (FRAMESPACING="n") وهي تعمل مع MS Explorer.

أما الخصائص المستخدمة مع الوسم <FRAME> فهي كالتالي:

- MARGINHEIGHT: تحدد مقدار المسافة الفارغة المتروكة للهوامش العلوية والسفلية للإطار (بالبيكسل).

  MARGINHEIGHT="n"

- MARGINWIDTH: تحدد مقدار المسافة الفارغة المتروكة للهوامش اليمنى واليسرى للإطار (بالبيكسل).

```
MARGINWIDTH="n"
```

- SCROLLING: تحدد إمكانية ظهور أو عدم ظهور أشرطة التصفح الأفقية والعمودية على جوانب أو أسفل الإطار. وتأخذ القيم yes للظهور. no لعدم الظهور. و auto التي تحدد ظهور الأشرطة أو عدمه تلقائياً بحسب الحاجة إليها. تماماً كما يحدث في معظم تطبيقات Windows

```
SCROLLING="yes"
SCROLLING="no"
SCROLLING="auto"
```

- NORESIZE عند إضافة هذه الخاصية يتم منع عملية التحكم بحجم الإطار بالتصغير أو التكبير من خلال السحب والإفلات. وهي لا تأخذ أي قيم.

ولتوضيح مبدأ عمل هذه الخصائص إليك هذه الأمثلة، (سأقوم باستخدام الملف الرئيسي الذي أدرجت فيه الصورة في الدرس السابق، فهو أفضل مثال لتوضيحها). وقم بتفحصه والتدقيق في تفاصيله لكي تقارنها بما سينتج عن الأمثلة التالية، كذلك حاول القيام بتكبير الإطار أو تصغيره بوضع المؤشر على الحد ثم استخدام السحب والإفلات بالإتجاه المطلوب. وذلك لكي تستطيع تمييز عمل الخاصية NORESIZE

```
<FRAMESET COLS="50%,50%">
<FRAME SRC="thedome.jpg" MARGINHEIGHT="40">
<FRAME SRC="frame2.html">
</FRAMESET>
```

```
<FRAMESET COLS="50%,50%">
<FRAME SRC="thedome.jpg" MARGINHEIGHT="40" MARGINWIDTH="30">
<FRAME SRC="frame2.html">
</FRAMESET>
```

```
<FRAMESET COLS="50%,50%">
<FRAME SRC="thedome.jpg" MARGINHEIGHT="40" MARGINWIDTH="30" SCROLLING="yes">
<FRAME SRC="frame2.html">
</FRAMESET>
```

```
<FRAMESET COLS="50%,50%">
<FRAME SRC="thedome.jpg" MARGINHEIGHT="40" MARGINWIDTH="30" SCROLLING="yes" NORESIZE>
<FRAME SRC="frame2.html">
</FRAMESET>
```

كذلك هناك الخصائص FRAMEBORDER, FRAMESPACING, BORDER, BORDERCOLOR التي تستخدم مع هذا الوسم وبنفس التفاصيل التي ذكرت مع <FRAMESET>. لكنها بالطبع تحدد خصائص الإطار وحده وليس مجموعة الإطارات ككل في الصفحة. وهي تعمل على متصفحات معينة دون غيرها.

بقي لدينا الخاصية NAME والتي تعتبر أهم خاصية لهذا الوسم، فهي التي تحدد طريقة تنسيق العمل بين الإطارات والصفحات وأسلوب عرضها لذلك فقد فضلت أن أفرد لها موضوعاً خاصاً وبصورة مستقلة عن باقي الخصائص، وذلك في الدرس التالي إن شاء الله.

هناك وسم ثالث يتعلق بالإطارات، ويتم إدراجه داخل الملف الرئيسي وعادة في النهاية وهو:

<NOFRAMES> ... </NOFRAMES>

يستخدم هذا الوسم لتوفير بديل معين عن صفحة الإطارات في حالة قيام أحد الزوار بدخول الموقع مستخدماً متصفحاً لا يدعم الإطارات. (بالمناسبة فإن هناك متصفحات لا يمكن لها أن تعرض الإطارات مثل الإصدارات القديمة لـ Netscape, MS Explorer) لكنك حتماً تستطيع مشاهدتها وذلك لأن الإصدارات المعربة من هذه المتصفحات هي إصدارات حديثة نسبياً وتدعم الإطارات.

فإذا أردت أن تمنح زوار موقعك الذين لا يستخدمون متصفحاً حديثاً فرصة مشاهدة موقعك، فكل ما عليك فعله هو إدراج هذا الوسم في نهاية الملف الرئيسي للإطارات والبدء بكتابة صفحتك كما لو كانت صفحة ويب عادية.

```
<HTML>
<HEAD>
<TITLE>Main File</TITLE>
</HEAD>
<FRAMESET ROWS="50,*,15%,20%" FRAMEBORDER="0">
   <FRAME SRC="frame1.html">
   <FRAME SRC="frame2.html">
   <FRAME SRC="frame3.html">
   <FRAME SRC="frame4.html">
</FRAMESET>

<NOFRAMES>
<BODY>

أكتب صفحتك
بالصورة
الإعتيادية هنا

</BODY>
</NOFRAMES>

</HTML>
```

أما إذا كنت مصراً على إطاراتك ولا تريد إنشاء نسخة أخرى للموقع بدونها، فلم لا تكتب ملاحظة بسيطة ضمن هذا الوسم تخبر فيها زائرك بأن الموقع يحتوي على هذه الإطارات وأنه يحتاج إلى متصفح مناسب (على الأقل لكي تخفف عنه الصدمة)!!

وماذا بعد...؟ لم يبق أي شيء ليذكر في هذا الدرس فقد قمنا بمناقشة جميع خصائص الإطارات عدا الخاصية NAME. ما رأيك لو قمنا بالتدرب على إنشاء نسخة من هذا الموقع باستخدام الإطارات في الدرس التالي؟

النماذج

سيكون هذا الدرس الأول من درسين سنتحدث فيهما عن النماذج وكيفية تضمينها في صفحات الويب.

مع أن النماذج تعتبر من المواضيع المتقدمة (وغير السهلة) نوعاً ما في لغة HTML إلا أن معظم مواقع الويب تكاد لا تخلو من وجودها، وذلك لعدة أسباب لعل من أهمها إيجاد إمكانية للتفاعل بين الموقع وصاحبه من جهة والزوار من جهة أخرى...أحيانا قد تحتاج كمصمم لموقع ويب أن تعرف آراء زوار موقعك في مسائل معينة وقد تكتفي برسائل البريد الإلكتروني التي يرسلوها لك، لكن عندما تريد معرفة أشياء محددة بذاتها فإن النماذج هي الخيار الأفضل لك. بالإضافة إلى إمكانية تنظيم البيانات المدخلة من خلالها وسهولة وسرعة استخدامها من قبل زوار الموقع. ومن أبرز الأمثلة على النماذج في مواقع الويب هي دفاتر الزوار وصفحات البحث عن الكلمات أو العبارات داخل المواقع.

لا تكمن صعوبة التعامل مع النماذج في كونها معقدة بحد ذاتها، كلا ... فهي إحدى العناصر التي تدعمها لغة HTML وهي مجرد وسوم عادية مثلها مثل الوسوم التي تعاملنا معها في جميع الدروس السابقة. وبإمكانك إنشاء النماذج في موقعك بنفس السهولة التي تدرج فيها جدولاً أو إطاراً (هذا بالطبع إذا كنت تعتقد أن الجداول والإطارات سهلة) لكن التداخل بينها (وأعني النماذج) وبين لغات البرمجة المتقدمة في الويب مثل JavaScript, CGI هي ما يجعلها تختلف عن سابقيها من الوسوم أو العناصر الأخرى. خاصة إذا احتجت إلى بعض المقاطع البرمجية من هذه اللغات ضمن نماذجك. أما إذا اكتفيت بالإمكانات المتواضعة التي توفرها HTML بالنسبة للنماذج. فما من مشكلة... لأنه سيكون بإمكانك التعامل معها بكل بساطة. وفي هذا الدرس لن نتطرق بالطبع إلى أي من اللغات سوى HTML.

ولنفترض الشكل التالي لدفتر زوار

كم شكلاً من أشكال إدخال البيانات يوجد في هذا الدفتر؟ الحقيقة أنه يوجد ستة أشكال هي كالتالي:

وهي الأشكال الموجودة في الدفتر فقط. وأود أن ألفت نظرك إلى وجود أشكال أخرى سوف يتم التعامل معها من خلال هذا الدرس.

والآن إلى العمل

مع أن الأشكال السابقة تختلف عن بعضها البعض من حيث المبدأ والمظهر (وطريقة التعريف أيضاً) إلا أنها يجب أن تدرج جميعاً ضمن وسمين أساسيين للنماذج هما:

</FORM> ... <FORM>

وكما جرت العادة نحتاج لتحديد بعض الخصائص التي تتعلق بطبيعة هذا النموذج. ولدينا هنا ثلاث خصائص:

ACTION

تحدد العنوان الذي سيتم إرسال بيانات النموذج إليه لتتم معالجتها بالصورة المطلوبة. وعادة يكون هذا عنواناً لبريد إلكتروني Email سوف يتم إرسال بيانات النموذج إليه. أو قد يكون عنواناً لبرنامج CGI موجود على الكمبيوتر الخادم Server الذي تتواجد عليه صفحة الويب، حيث يستقبل هذه البيانات ويعالجها حسب التعليمات الموجودة فيه كأن يضيفها مثلاً إلى إحدى الصفحات (كما يحدث عادة في دفاتر الزوار) أو يتحقق من صحة بعض الحقول المدخلة ومطابقتها لمعايير معينة، أو أن يقوم بالبحث عن كلمة أو عبارة ضمن صفحات الموقع كما في نماذج البحث الموجودة في مواقع الويب.

```
<FORM ACTION="mailto:someone@domain.com"> ... </FORM>
<FORM ACTION="name_and_address_of_CGI_script"> ... </FORM>
```

METHOD

تحدد الطريقة التي سيتم بها التعامل مع العنوان المحدد في الخاصية السابقة ACTION. وهناك قيمتين لهذه الخاصية هما: GET التي تستخدم في حالة كون عملية المعالجة داخلية أي تتم داخل الخادم Server نفسه. ففي مثالنا السابق عندما نستخدم نموذج البحث عن كلمة في الموقع، فإن عملية المعالجة (أي البحث) تجري مباشرة في الموقع. والقيمة الثانية هي Post وتستخدم عندما تكون عملية المعالجة خارجية كأن يتم إرسال البيانات إلى عنوان بريد إلكتروني.

```
<FORM ACTION="mailto:someone@domain.com" METHOD="post"> ... </FORM>
<FORM ACTION="name_and_address_of_CGI_script" METHOD="get"> ... </FORM>
```

ENCTYPE

هذه الخاصية تحدد طريقة الترميز التي سيتم إرسال البيانات وفقاً لها. وهي تأخذ القيمتين التاليتين: (يجب أن تكتب هذه القيم كما هي نصاً وحرفاً)

- application/x-www-form-urlencoded
- text/plain

وبدون الخوض في الأسباب التقنية التي أدت إلى إيجاد هذين النوعين من طرق الترميز أو في أمور برمجية بعيدة عن موضوعنا، فإن الدافع لإستخدام أي من القيمتين هو طبيعة عملية المعالجة التي ستجرى على البيانات أو طبيعة برنامج

البريد الإلكتروني الذي ستستقبل هذه البيانات من خلاله (إذا كان يدعم MIME أم لا، وهي إختصار للعبارة Multi-purpose Internet Mail Extentions وهي من المعايير السائدة في الإنترنت والتي تتعلق بنقل جميع أنواع البيانات من صوت وصورة وليس فقط النصوص من خلال البريد الإلكتروني). وما يعنينا هنا هو الفرق بين الطريقتين من حيث طريقة إرسال واستقبال البيانات. فعند استخدام text/plain ستصل البيانات بالشكل التالي:

```
NAME=Yahya Al-Sharif
Address=Nablus , Palestine
Email=yahya@palnet.com
```

(الكلمات Name, Email ,Address هي أسماء الحقول في النموذج ونقوم نحن بتعريفها أثناء عملية تصميم النموذج أما النصوص الظاهرة بعد إشارة المساواة فهي البيانات المدخلة، وسوف نتحدث عن تعريف أسماء الحقول بعد قليل)

أما عند استخدام application/x-www-form-urlencoded فستصل البيانات بالشكل:

```
NAME=Yahya+Al-Sharif&Address=Nablus+,+Palestine&Email=yahya@palnet.com
```

ولك أن تخيل مبلغ الصعوبة في تحليلها إذا احتوت على عشرات الحقول. لذلك تتوفر برامج خاصة تعرف بـ Formaters تقوم بإعادة ترتيب البيانات المرسلة من خلال النماذج بشكل مفهوم بحيث تصبح كما لو كانت مرسلة بترميز text/plain لكن لا تعتقد أن الطريقة الأولى هي الأفضل دائماً فذلك يعتمد كما قلنا على طريقة المعالجة والنقل بالبريد. لذلك لا ضير من أن تجرب الطريقتين لتعرف أيهما أنسب لك.

إذن خلاصة القول: قد تكون أفضل صيغة لتعريف النموذج في حالة أردت استقبال البيانات من موقعك إلى عنوان بريدك الإلكتروني هي:

```
<FORM ACTION="mailto:email@domain.com" METHOD="post" ENCTYPE="text/plain">
...
</FORM>
```

وبهذا نكون قد إنتهينا من عملية تعريف النموذج وخصائصه، لكن انتظر فما زلنا في بداية الطريق.

نبدأ الآن في عملية تعريف أشكال البيانات في النموذج. ونستخدم الوسم <INPUT> لتعريفها والحقيقة أن هذه الأشكال هي مجرد خصائص أو بالأحرى قيم لخصائص تابعة لهذا الوسم. كيف؟ ... لنأخذ مثالاً على ذلك لأوضح لك هذا المفهوم

ملاحظة: إذا كنت تستخدم Sindbad 3.x فسيبدو الحقل والنص المجاور له بصورة معكوسة، وهي مشكلة ناتجة عن برنامج Netscape الذي يعمل من خلاله

Please enter your address: Nablus, Palestine

حسناً، لقد استخدمت الوسم <INPUT> لتعريف هذا الشكل (هذه إتفقنا عليها مسبقاً) ومن ثم قمت بإضافة الخاصية TYPE لهذا الوسم لتحديد نوع الشكل الذي أريده وأعطيتها القيمة TEXT أي

```
<FORM ...>
<INPUT TYPE="text">
</FORM>
```

لينتج لدينا هذا الشكل:

فقرة معترضة:

إليك جميع الأشكال (القيم) المستخدمة مع الخاصية TYPE وسوف أتركها الآن بدون تعليق لحين مناقشها لاحقاً بشكل مفصل. مع ملاحظة أن هناك شكلين آخرين ندرجهما بالوسوم <SELECT>, <TEXTAREA>

<INPUT TYPE="text">	
<INPUT TYPE="password">	
<INPUT TYPE="hidden">	
<INPUT TYPE="radio">	
<INPUT TYPE="checkbox">	
<INPUT TYPE="submit">	Submit Query
<INPUT TYPE="reset">	
<INPUT TYPE="button">	

أرجو أن أكون قد وضحت لك الآن وظيفة الخاصية TYPE وجميع القيم المستخدمة معها
ونعود الآن إلى مثالنا.. الخاصية الثانية المستخدمة مع <INPUT> هي NAME وتستخدم لتسمية حقل البيانات حيث قمت بإعطاء الإسم address لهذا الحقل في المثال. (لك كل الحرية في إعطاء الإسم الذي تريده للحقل). والحقيقة أن هذا الإسم يعرّف الحقل في داخل النموذج نفسه، بحيث يمكن استخدامه فيما بعد للحاجات البرمجية وضرورات المعالجة إن وجدت من قبل البرامج التي قد تضيفها كمصمم للموقع. وحتى عندما تريد أن يُرسل النموذج إليك بالبريد فإن حقوله تعرّف بالاسم الذي أدرجته لها من خلال هذه الخاصية. (لاحظ ما قلتُه سابقاً عن تعريف أسماء الحقول عندما تحدثنا عن الترميز والطرق التي تصل بها محتويات النموذج). وكما ترى لا يوجد (حتى الآن) ما يدل على أن هذا الحقل يختص بإدخال العنوان.

```
<FORM ...>
<INPUT TYPE="text" NAME="address">
</FORM>
```

أما العبارة enter your address Please : فهي مجرد عبارة توضيحية أضفتها ليعرف الزائر ما الذي يجب عليه كتابته وتستطيع صياغة هذه العبارة كما تريد. ففي كل الأحوال ليس لها علاقة بجوهر النموذج نفسه بعكس الخاصية NAME.

```
<FORM ...>
Please enter your address : <INPUT TYPE="text" NAME="address">
</FORM>
```

Please enter your address :

أما بالنسبة للعبارة الظاهرة داخل الحقل Nablus, Palestine (أو أي عبارة أخرى تريدها) وهي بمثابة القيمة الإفتراضية التي تريدها للحقل، فبالإمكان إظهارها من خلال الخاصية VALUE. وهذه الخاصية تستخدم في الحالات التي نتوقع فيها كتابة قيمة دارجة أو متكررة من قبل معظم الزوار وللتسهيل عليهم يتم تعيينها كقيمة إفتراضية وبالطبع مع توفر إمكانية حذفها وكتابة ما يريدنه بدلاً منها.

```
<FORM ...>
Please enter your address : <INPUT TYPE="text" NAME="address" VALUE="Nablus, Palestine">
</FORM>
```

Please enter your address : Nablus, Palest

قد نحتاج أحياناً إلى تحديد حجم الحقل ولذلك نستخدم الخاصية SIZE مع الرقم الذي نريده كحجم للحقل، لنجرب الرقم 40

```
<FORM ...>
Please enter your address : <INPUT TYPE="text" NAME="address" VALUE="Nablus, Palestine" SIZE="40">
</FORM>
```

Please enter your address : Nablus, Palestine

أو لنجرب الرقم 10 أيضاً

Please enter your address : Nablus, Pal

لا يوجد للخاصية SIZE أي صفة تحكمية بالنسبة لحجم المدخلات التي يمكن للزائر أن يكتبها داخل الحقل. وبعبارة أخرى: صحيح أننا حددنا حجم الحقل لكن ذلك يسري فقط على مظهره على الشاشة. ولا يوجد ما يمنع الزائر من الكتابة بحيث يتجاوز النص حجم الحقل المحدد. وهنا يأتي دور الخاصية MAXLENGTH لتتحكم بالحد الأقصى للنص المدخل.

```
<FORM ...>
Please enter your address : <INPUT TYPE="text" NAME="address" VALUE="Nablus, Palestine"
SIZE="40" MAXLENGTH="30">
</FORM>
```

Please enter your address : Nablus, Palestine

حاول الكتابة في هذا الحقل لأكثر من 30 حرفاً وأنظر ماذا سيحدث؟

إنتهينا الآن من خصائص الوسم INPUT فما رأيك بإجمالها مرة أخرى؟ حسناً، هذه هي:

- TYPE: لتحديد نوع (شكل) حقل البيانات.

- NAME: لتعيين اسم لحقل البيانات.
- VALUE: لتعيين قيمة إفتراضية (مبدئية) لحقل البيانات.
- SIZE: لتحديد حجم حقل البيانات.
- MAXLENGTH: لتعيين الحد الأقصى لعدد الحروف المدخلة في الحقل.

النوع الثاني من الحقول المستخدمة في النماذج هو حقل password وهو يشبه الحقل text من حيث الخصائص تماماً غير أن مدخلاته تظهر على شكل ****** مهما كانت، وهو الفرق الوحيد بينهما. وربما تكون قد استنتجت الآن أن هذا النوع من الحقول يستخدم عندما يوجد حاجة لإدخال كلمة سر من قبل الزائر في النموذج

```
<FORM ...>
Please enter your name :
<INPUT TYPE="text" NAME="the name" VALUE="" SIZE="40" MAXLENGTH="30">
Please enter your passwod :
<INPUT TYPE="password" NAME="the password" VALUE="" SIZE="40" MAXLENGTH="30">
</FORM>
```

Please enter your name :

: Please enter your password

لاحظ أنني لم أرغب في كتابة قيم إفتراضية VALUES للحقول، ولذلك تركتها فارغة وأستطيع أيضاً أن ألغيها نهائياً من الشيفرة. وأنا في هذا المثال أردت أن أوضح لك عدم أهمية كتابة قيمة إفتراضية للحقول في بعض الحالات.

نأتي الآن إلى النوع الثالث من أنواع الحقول وهو hidden أي الحقل المخفي. وكما نستنتج من اسمه فهو لن يظهر ضمن النموذج. وهذا مثال:

```
<FORM ...>
Please enter your name :
<INPUT TYPE="text" NAME="the name" VALUE="" SIZE="40" MAXLENGTH="30">

<INPUT TYPE="hidden" NAME="my forms" VALUE="form1">

Please enter your passwod :
<INPUT TYPE="password" NAME="the password" VALUE="" SIZE="40" MAXLENGTH="30">
</FORM>
```

Please enter your name :

: Please enter your passwod

لاحظ هنا أن وجود هذا الحقل مثل عدمه بالنسبة لمظهر النموذج، وأن الزائر لن يتعامل معه بل وربما لن يعرف أن هناك حقلاً مخفياً. والسؤال هنا: ما الفائدة من وجود شيء مخفي لا إمكانية لاستخدامه؟ ولكي أجيب على هذا السؤال دعني أطرح لك مثالاً أو حالة قد تواجهك كمصمم صفحات ويب...
لنفرض أن لديك ثلاث صفحات تتضمن كل منها نموذجاً ما وأن هذه النماذج متشابهة. وتحتوي على نفس الحقول. وعندما تصلك البيانات كيف ستستطيع تمييز أي من هذه النماذج استخدم لإرسال البيانات؟ بإمكانك إضافة هذا الحقل (الوهمي) وإسناد أي اسم وأي قيمة له في كل نموذج.

في النموذج الأول ...

```
<INPUT TYPE="hidden" NAME="my forms" VALUE="form1">
```

في النموذج الثاني ...

```
<INPUT TYPE="hidden" NAME="my forms" VALUE=" form2">
```

في النموذج الثالث ...

```
<INPUT TYPE="hidden" NAME="my forms" VALUE=" form3">
```

وبذلك عندما تصلك البيانات المرسلة من قبل أي زائر استخدم أي من النماذج الثلاثة سيصلك أيضا حقل إضافي قمت أنت نفسك بتعبئته سلفاً عندما صممت النموذج وذلك بأحد الأشكال التالية:

my forms=form1 أو
my forms=form2 أو
my forms=form3

إذن نستطيع القول أن الحقل المخفي هو لاستخدام المصمم وليس الزائر، وأن قيمته تدخل مباشرة عند التصميم.
ويستخدم بهدف تعريف قيم ما سيتم إرسالها جنباً إلى جنب ضمن بيانات النموذج التي قام الزائر بتعبئتها

ملاحظة مهمة بالنسبة للنماذج بشكل عام. من أجل إظهار النموذج بصورة مرتبة ومنسقة والتحكم بموقع الحقول فيه فمن الأفضل دائماً وضعه داخل جدول مع جعل الجدول بلا حدود.

```
<FORM ...>
<TABLE BORDER="0">
<TR>
<TD>Please enter your name : </TD>
<TD>
<INPUT TYPE="text" NAME="the name" VALUE="" SIZE="40" MAXLENGTH="30">
</TD>
</TR>

<TR>
<TD>Please enter your password :</TD>
<TD>
<INPUT TYPE="password" NAME="the password" VALUE="" SIZE="40" MAXLENGTH="30">
</TD>
</TR>
</TABLE>
</FORM>
```

وكما ترى تحتاج إلى القليل من العمل الإضافي لكنك بالمقابل ستحصل على النتيجة التالية

Please enter your name :

Please enter your password :

هكذا أفضل... أليس كذلك؟

وهو الدرس الثاني من درسين حول النماذج. لقد قمنا في الدرس السابق بمناقشة الوسوم الأساسية للنماذج وتعلمنا كيفية إدراج النماذج في صفحات الويب. كما قمنا بمناقشة بعض أشكال إدخال البيانات في النموذج وهي Text, Password, Hidden هل تذكر كيف نقوم بتعريفها؟ راجع الدرس السابق إن أردت المزيد من التوضيح.

سوف نتابع الآن مع مجموعة من الأشكال الخاصة بالاختيار من متعدد وهي بالمناسبة ثلاثة أنواع: Radio, Checkbox وقائمة الإختيار

نبدأ مع الشكل المسمى RADIO. ومن مسوغات استخدام هذا الشكل أن السؤال المطروح ينبغي أن يكون له إجابة واحدة فقط، أو بعبارة أخرى على الزائر أن يختار إجابة واحدة فقط.

وكمثال، لنفرض أنني أريد أن أسأل الزائر عن المتصفح الذي يستخدمه (كما هو موجود في دفتر الزوار على شكل قائمة إختيار) لكن بدلا من أن يكون على شكل قائمة إختيار أريده أن يكون على شكل RADIO وذلك بالشكل التالي:(أود أن أذكرك أن عناصر النموذج تظهر بشكل معكوس إذا كنت تستخدم Sindbad 3.0)

- Sindbad 3.0
- Sindbad 4.0
- Ms Explorer 3.0
- Ms Explorer 4.0

فكيف ننشيء مثل هذه القائمة؟ ... حسناً، لنبدأ من الصفر ونعرّف نموذجاً

```
<FORM>

</FORM>
```

ثم لنقم بتعريف هذا الشكل، هل تذكر الوسم الخاص بذلك؟ إنه <INPUT>

```
<FORM>
<INPUT TYPE="radio">
</FORM>
```

لكن بما أن هناك أربعة مدخلات، إذن نحتاج إلى أربعة وسوم

```
<FORM>
<INPUT TYPE="radio"> <BR>
<INPUT TYPE="radio"> <BR>
<INPUT TYPE="radio"> <BR>
<INPUT TYPE="radio"> <BR>
</FORM>
```

نحتاج الآن إلى تسمية هذه المدخلات، أي أننا سنستخدم الخاصية NAME معها. أما الاسم المعطى بحد ذاته فمن الأفضل أن يكون مرتبطاً نوعاً ما بموضوع السؤال، ليس لأن هذا ضروري للنموذج بل هو ضروري لك كشخص سيقوم باستقبال البيانات المرسلة من خلال النموذج، وبالتالي من الأفضل أن يوجد عنوان معبّر للبيانات بغرض التمييز. وبما أننا هنا نتحدث عن المتصفحات فليكن هذا الاسم هو browser

```
<FORM>
<INPUT TYPE="radio" NAME="browser"> <BR>
<INPUT TYPE="radio" NAME="browser"> <BR>
<INPUT TYPE="radio" NAME="browser"> <BR>
<INPUT TYPE="radio" NAME="browser"> <BR>
</FORM>
```

وكما تلاحظ من النتيجة أن هذه التسمية هي ضمنية فقط ولا تؤثر على شكل النموذج (راجع الدرس السابق) لكن أي إختيار سيقوم به الزائر من هذه الأربعة خيارات سوف يصلك تحت الاسم browser.

الخطوة التالية هي إعطاء قيمة لكل مدخلة في هذه القائمة وذلك حسب ما نراه مناسباً، إذن حان الوقت لاستخدام الخاصية VALUE:

```
<FORM>
<INPUT TYPE="radio" NAME="browser" VALUE="Sind3"> <BR>
<INPUT TYPE="radio" NAME="browser" VALUE="Sind4"> <BR>
<INPUT TYPE="radio" NAME="browser" VALUE="Msie3"> <BR>
<INPUT TYPE="radio" NAME="browser" VALUE="Msie4"> <BR>
</FORM>
```

وهنا أيضا نلاحظ أن لا تغيّر في شكل النموذج ظاهرياً مع إضافة هذه الخاصية. لكن مع ذلك فقد قمنا حتى الآن بتسمية الحقول وإعطاء كل حقل قيمة محددة. وفعلياً لقد إنتهينا من هذا النموذج. لكن بالطبع نحن لا نتوقع أن يكون الزائر عالماً بالغيب لكي يخمن أي من هذه الحقول تختص بكل قيمة. لذلك بقي علينا تعريف كل حقل باسم صريح يوضح محتواه.

```
<FORM>
<INPUT TYPE="radio" NAME="browser" VALUE="Sind3"> Sindbad 3.0 <BR>
<INPUT TYPE="radio" NAME="browser" VALUE="Sind4"> Sindbad 4.0<BR>
```

```
<INPUT TYPE="radio" NAME="browser" VALUE="Msie3"> MS Explorer 3.0<BR>
<INPUT TYPE="radio" NAME="browser" VALUE="Msie4"> MS Explorer 4.0<BR>
</FORM>
```

- Sindbad 3.0
- Sindbad 4.0
- MS Explorer 3.0
- MS Explorer 4.0

وهناك خاصية تتعلق بهذا النوع من الحقول، وهي أنك إذا أردت أن يظهر أحدها وقد تم اختياره بشكل تلقائي فعليك بإضافة الخاصية CHECKED إليه، مع ترك كل الحرية للزائر في اختيار ما يريده فيما بعد.

```
<FORM>
<INPUT TYPE="radio" NAME="browser" VALUE="Sind3"> Sindbad 3.0 <BR>
<INPUT TYPE="radio" NAME="browser" VALUE="Sind4" CHECKED>
Sindbad 4.0<BR>
<INPUT TYPE="radio" NAME="browser" VALUE="Msie3"> MS Explorer 3.0<BR>
<INPUT TYPE="radio" NAME="browser" VALUE="Msie4"> MS Explorer 4.0<BR>
</FORM>
```

- Sindbad 3.0
- Sindbad 4.0
- MS Explorer 3.0
- MS Explorer 4.0

وأخيراً... أود أن أوضح لك الصورة التي يصلك بها النموذج عند اختيار أحد حقوله (ولنفترض أنه الخيار الثالث). وهي بالشكل التالي:

browser=Msie3

نأتي الآن إلى الشكل الثاني من أشكال الإختيار من متعدد والذي يدعى CHECKBOX. ظاهرياً لا يختلف هذا الشكل عن الشكل الذي سبقه، لكن عملياً هناك اختلافات جذرية من حيث المفهوم والتعريف. وأنا أفضّل أن نبقى على استخدامنا للمثال السابق حتى يسهل علينا تمييز الفروق.

- Sindbad 3.0
- Sindbad 4.0
- MS Explorer 3.0
- MS Explorer 4.0

قبل أن نستمر لو قمتم بالنقر على أكثر من حقل في القائمة السابقة فإن باستطاعتك اختيار أكثر من حقل في نفس الوقت! وهذا هو الفرق الأول بين CHECKBOX و RADIO ففي RADIO يمكن اختيار حقل واحد فقط ليس أكثر.
لنقم الآن بتعريف هذه الحقول، وتسميتها بشكل مباشر ومن ثم سنعلّق عليها:

```
<FORM>
<INPUT TYPE="checkbox" NAME="Sind3" VALUE="Yes"> Sindbad 3.0 <BR>
<INPUT TYPE="checkbox" NAME="Sind4" VALUE="Yes"> Sindbad 4.0 <BR>
<INPUT TYPE="checkbox" NAME="Msie3" VALUE="Yes"> MS Explorer 3.0 <BR>
<INPUT TYPE="checkbox" NAME="Msie4" VALUE="Yes"> MS Explorer 4.0 <BR>
</FORM>
```

ماذا تلاحظ؟ أولاً لقد أسندنا القيمة checkbox للخاصية TYPE. ثم أعطينا لكل حقل في القائمة إسماً مميزاً في الخاصية NAME يختلف عن باقي الحقول. أما الخاصية VALUE فأعطيناها قيمة موحدة لجميع الحقول. وبالطبع قمنا في النهاية بكتابة الأسماء التعريفية لكل حقل.

في RADIO نستطيع اختيار حقل واحد فقط أما في CHECKBOX فنختار أكثر من حقل، لذلك يستخدم عادة في الحالات التي يحتمل أن نحصل فيها على عدة أجوبة لنفس السؤال.

في RADIO تكون أسماء الحقول موحدة والقيم مختلفة، أما في CHECKBOX فتكون الأسماء مختلفة والقيم موحدة

كيف ستصل البيانات؟ حسناً لنفرض أنه تم اختيار الحقلين الثاني والرابع فسوف تصلك النتيجة بالشكل التالي:

Sind4=Yes
Msie4=Yes

كما نستطيع أيضاً تعليم بعض الحقول بصورة تلقائية كما فعلنا مع RADIO باستخدام نفس الخاصية CHECKED

```
<FORM>
<INPUT TYPE="checkbox" NAME="Sind3" VALUE="Yes" CHECKED>
Sindbad 3.0 <BR>
<INPUT TYPE="checkbox" NAME="Sind4" VALUE="Yes"> Sindbad 4.0 <BR>
<INPUT TYPE="checkbox" NAME="Msie3" VALUE="Yes" CHECKED>
MS Explorer 3.0 <BR>
<INPUT TYPE="checkbox" NAME="Msie4" VALUE="Yes"> MS Explorer 4.0 <BR>
</FORM>
```

- [x] Sindbad 3.0
- [ ] Sindbad 4.0
- [x] MS Explorer 3.0
- [ ] MS Explorer 4.0

النوع الثالث من أشكال الإختيار من متعدد هو قوائم الاختيار، وهذا النوع سوف يقودنا إلى وسوم جديدة من وسوم التعريف والتي ستستخدم بدلاً من <INPUT> وهي

```
<SELECT>
  <OPTION>
  <OPTION>
  <OPTION>
```

```
   .....
   .....
</SELECT>
```

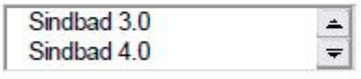
بحيث أن <SELECT> ... <SELECT/> تحددان بداية ونهاية القائمة، والوسم <OPTION> الذي يوضع دائما بينهما يستخدم لتحديد كل عنصر من عناصر القائمة. لنعد إلى مثالنا السابق لنرى كيف يمكن وضع الخيارات في قائمة

```
<FORM>
<SELECT>
   <OPTION> Sindbad 3.0
   <OPTION> Sindbad 4.0
   <OPTION> MS Explorer 3.0
   <OPTION> MS Explorer 4.0
</SELECT>
</FORM>
```

وبذلك تكون النتيجة هي:

وكالمعتاد لا يخلو الأمر من وجود خصائص تحدد طريقة عمل هذه الوسوم. وهناك خصائص مشتركة عرفناها في الأشكال السابقة سيتم استخدامها هنا أيضاً كما يوجد خصائص جديدة تتعلق فقط بهذا الشكل من حقول البيانات. فبالنسبة لـ <SELECT> يوجد الخاصية NAME وهي كما تعلم تحدد اسم القائمة. كما توجد الخاصية SIZE التي تحدد حجم (أو بالأحرى) ارتفاع القائمة، وبالتالي عدد البيانات الظاهرة فيها. وهي تأخذ أي قيمة عددية صحيحة.

```
<FORM>
<SELECT NAME="browser" SIZE="2">
   <OPTION> Sindbad 3.0
   <OPTION> Sindbad 4.0
   <OPTION> MS Explorer 3.0
   <OPTION> MS Explorer 4.0
</SELECT>
</FORM>
```



وطالما بالإمكان عرض القائمة بأي ارتفاع نريد، وقد يصل إلى حد عرض جميع بيانات القائمة معاً، فإن هناك إمكانية أيضاً لجعل اختيار البيانات من هذه القائمة متعدداً وليس فقط قيمة واحدة، كيف؟؟ بإضافة الخاصية MULTIPLE. لنقم الآن بعرض جميع القيم (لدينا أربعة قيم، إذن القيمة المكتوبة مع SIZE يجب أن تكون 4)، ومن ثم لنتح المجال أمام الزائر لاختيار أكثر من قيمة واحدة في القائمة.

```
<FORM>
<SELECT NAME="browser" SIZE="4" MULTIPLE>
   <OPTION> Sindbad 3.0
   <OPTION> Sindbad 4.0
   <OPTION> MS Explorer 3.0
   <OPTION> MS Explorer 4.0
</SELECT>
</FORM>
```

Sindbad 3.0
Sindbad 4.0
MS Explorer 3.0
MS Explorer 4.0

لاحظ أنه لأداء عدة اختيارات يجب أن تقوم بالضغط على المفتاح ctrl بصورة متواصلة أثناء عملية الإختيار.

أما الخصائص المستخدمة مع الوسم <OPTION> فهي VALUE والتي استخدمناها من قبل وسنستخدمها الآن لإعطاء قيمة لكل حقل بيانات في القائمة. وكذلك الخاصية SELECTED والتي نكتبها مع أي <OPTION> نريد أن يظهر وقد تم اختياره بصورة تلقائية.

```
<FORM>
<SELECT NAME="browser" SIZE="4" MULTIPLE>
   <OPTION VALUE="Sindbad 3.0"> Sindbad 3.0
   <OPTION VALUE="Sindbad 4.0" SELECTED> Sindbad 4.0
   <OPTION VALUE="MS Explorer 3.0"> MS Explorer 3.0
   <OPTION VALUE="MS Explorer 4.0"> MS Explorer 4.0
</SELECT>
</FORM>
```

Sindbad 3.0
Sindbad 4.0
MS Explorer 3.0
MS Explorer 4.0

الشكل التالي من أشكال حقول البيانات يدعى TEXTAREA

وهو المستخدم عادة لكتابة التعليقات الحرة في النموذج ويتم إدراجه بكتابة الوسوم

<TEXTAREA> ... </TEXTAREA>

هل تستطيع تخمين الخصائص المستخدمة معه؟ بالطبع لا بد من وجود الخاصية NAME لإعطاءه اسم التعريف. لكن لا وجود للخاصية VALUE ، وبالمقابل فإن أي نص يكتب بين الوسمين سيتم عرضه داخل الحقل بصورة تلقائية

```
<TEXTAREA NAME="comments">
Hello, please write your comments here :-)
</TEXTAREA>
```

كما توجد خصائص لتحديد مساحة هذا الحقل عرضاً وارتفاعاً، وهي COLS التي تحدد العرض و ROWS التي تحدد الإرتفاع

```
<TEXTAREA NAME="comments" COLS="30" ROWS="6">
</TEXTAREA>
```

أما الخاصية الأخيرة هنا فهي WRAP التي تحدد طريقة إلتفاف النص المكتوب داخل الحقل (لا تعمل هذه الخاصية مع MS Explorer 3.0) وهناك ثلاثة قيم تأخذها وهي على النحو التالي:

virtual : التي تعني أن النص سيلتف على عدة أسطر عند كتابته ولكنه سيصلك عند إرساله على شكل سطر واحد متتابع (حاول الكتابة داخل الحقول وأنظر كيفية تأثير هذه الخاصية على كل منها)

```
<TEXTAREA NAME="comments" COLS="30" ROWS="6" WRAP="virtual">
</TEXTAREA>
```

physical : تعني أن النص سيلتف على عدة أسطر وسيصلك أيضاً على هذا النحو عند إرساله

```
<TEXTAREA NAME="comments" COLS="30" ROWS="6" WRAP="physical">
</TEXTAREA>
```

off : تعني أن النص لن يلتف بصورة تلقائية على عدة أسطر لكنه على أية حال سيصلك بنفس الشكل الذي تم إدخاله به

```
<TEXTAREA NAME="comments" COLS="30" ROWS="6" WRAP="off">
</TEXTAREA>
```

حسناً، بافتراض أننا إنتهينا من كتابة الشيفرة الخاصة بالنموذج وننتظر من أي زائر للموقع أن يملأه، والسؤال هو كيف يمكن له أن يرسله فعلياً؟ نعود الآن إلى الوسم <INPUT> وهذه المرة مع النوع submit والتي ستقوم تلقائياً بإنشاء زر سيقوم عند النقر عليه بإرسال البيانات التي تم ملؤها في النموذج.

```
<INPUT TYPE="submit">
```

Submit Query

لاحظ أن Submit أو (Submit Query في Netscape) ظاهرة على الزر وهي العبارة الإفتراضية، فإذا أردت تغييرها فعليك باستخدام الخاصية VALUE لهذا الغرض

```
<INPUT TYPE="submit" VALUE="Press here to send the form">
```

Press here to send the form

في حالة كان زوار موقعك من النوعية المترددة من الناس والذين قد يغيرون آرائهم في آخر لحظة، يمكنك أن تتيح لهم إمكانية مسح ما كتبوه في النموذج وإلغاء الأمر، وذلك باستخدام reset كنوع TYPE للوسم <INPUT> بنفس طريقة التعريف والخصائص المستخدمة مع submit.

```
<INPUT TYPE="reset" VALUE="Forget about it">
```

Forget about it

والذي يقوم بإنشاء زر button> هو INPUTالشكل الأخير من أشكال البيانات في النماذج والمدرج مع الوسم < ضمن النموذج، وهو مرتبط بالنماذج التي تحتوي على نصوص برمجية (أو برامج مكتملة) من لغات متقدمة مثل JavaScript كونه يستخدم لتشغيل هذه البرامج وإطلاقها. وطبعاً هناك طرق معينة لربطها مع البرامج وليس هنا المجال لطرحها. لكن مبدئياً أقول إن طريقة الإدراج والتعريف هي ذاتها المستخدمة مع reset, submit.

```
<INPUT TYPE="button" VALUE="This is a sample button">
```

وأخيراً ... وصلنا الآن إلى نهاية حديثنا عن النماذج. فما رأيك؟ هل هو من المواضيع السهلة أم الصعبة؟ لا شيء سهل في بدايته. لذلك من الأفضل لك أن تحاول دائماً التدرب أولاً بأول على الوسوم المشروحة، بل والعودة إلى الدروس السابقة إذا اقتضى الأمر وخاصة إذا تداخلت بعضها مع الدروس الأحدث.

جافاســكريبت

مــقــدمـــــة

تستخدم جافاسكريبت في ملايين الصفحات على الإنترنت لتحسين التصميم، أو تصميم نماذج الإدخال، أو غير ذلك. تم تطوير جافاسكريبت بواسطة شركة نيتســكيب **Netscape** وهي واحدة من أشهر اللغات النصية على الإنترنت.

تعمل جافاسكريبت على معظم المتصفحات الرئيسية الموجودة اليوم مثل **Navigator** من شركة نيتسكيب و **Internet Explorer** من شركة مايكروسوفــت.

ما الذي يجب أن تعرفه الآن؟!

قبل الاستمرار، عليك الإلمام بالمفاهيم الأساسية في:

- صفحات الويب WWW، لغة النص المترابط HTML، وبعض الأساسيات في بناء صفحات الإنترنت.

مــا هـــــي جافاسكريبت؟

- صممت جافاسكريبت لإضافة التفاعلية لصفحات HTML.
- جافاسكريبت هي لغة برمجة نصية – لغة البرمجة النصية هي لغة خفيفة.
- جافاسكريبت هي عبارة عن أسطر قابلة للتنفيذ.
- جافاسكريبت يتم تضمينها –عادة- مباشرة في صفحات HTML.
- جافاسكريبت هي لغة مفسرة.
- يمكن لأي شخص استخدام جافاسكريبت بدون شراء ترخيص استخدام.
- جافاسكريبت مدعومة من غالبية متصفحات الإنترنت الشهيرة.

هل جــافا Java هي نفسهــا جافاسكريبت ؟

لا!

جافا و جافاسكريبت هما لغتين مختلفتين تماماً...

جافا (طورت بواسطة شركة صن مايكروسيستيمز) هي لغة برمجة قوية ومعقدة –وفي نفس التصنيف مع لغات برمجة مثل C و C++ .

ما الذي يمكن لــ جافاســكريبت أن تفــعله؟

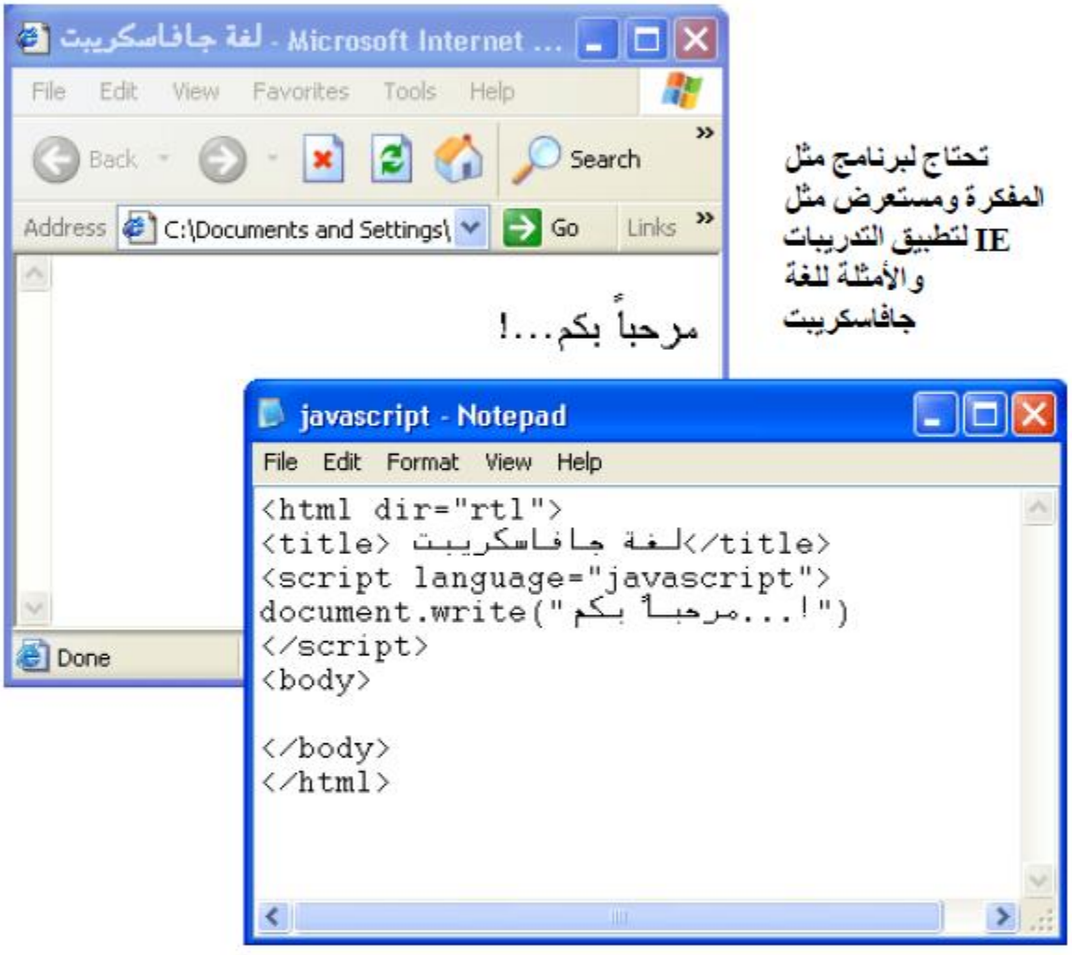
- **جافاسكريبت تعطي مؤلفي HTML أداة برمجية** – مؤلفي HTML ليسوا مبرمجين، ولكن جافاسكريبت هي لغة برمجة بسيطة التركيب! فبإمكان أي شخص أن يضع نصاً صغيراً لصفحات HTML الخاصة به.
- **يمكن لـ جافاسكريبت أن تضع نصاً متغيراً في صفحات HTML** – جملة جافاسكريبت مثل هذه document.write("<h1>"+name+"</h1>") يمكن أن تكتب نصاً متغيراً لصفحة HTML.
- **يمكن لـ جافاسكريبت أن تستجيب للأحداث** – يمكن لـ جافاسكريبت أن تنفذ أمراً عندما يحدث شيء ما، مثل عند انتهاء تحميل الصفحة أو عندما يضغط المستخدم على عنصر HTML.
- **جافاسكريبت يمكنها أن تتعرف على أو تكتب عناصر HTML** – يمكن لـ جافاسكريبت أن تكتب وتغير محتوى صفحات HTML.
- **يمكن أن تستخدم جافاسكريبت للتحقق من البيانات** – يمكن لـ جافاسكريبت أن تستخدم للتحقق من البيانات المدخلة في النماذج –في مواقع الإنترنت- قبل اعتمادها، وذلك يوفر الكثير من وقت المعالجة بالنسبة للخوادم Servers.

ما هي البرامـــج التي أحتاجهـــا؟

لا تحتاج –في الحقيقة –سوى إلى محرر نصوص! وذلك لكتابة النص البرمجي (الكود) الخاص بـHTMLومن ثم تضمين جمل جافاسكريبت فيها...

إذا كنت تصمم صفحاتك ببرنامج مثل FrontPage فبإمكانك وضع جمل جافاسكريبت ضمن محرر HTML في البرنامج...

إذا رغبت في تطبيق التمرينات في هذا الكتاب، استخدم برنامج المفكرة Note Pad المرفق بنظام التشغيل، واحفظ ملفات التدريبات بالامتداد file.htm أو file.html ، ويمكنك بعد ذلك استعراضها بمتصفح الإنترنت لديك، سواءً IE أو Navigator.



تحتاج لبرنامج مثل المفكرة ومستعرض مثل IE لتطبيق التدريبات والأمثلة للغة جافاسكريبت

كيف أضع نص جافاسكريبت داخل صفحـــة HTML؟

بالشكل التالي:

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```

لاحظ: علامات HTML باللون الأسود –نص جافاسكريبت باللون الأزرق –دالة جافاسكريبت باللون الأحمر

بعد حفظ الملف السابق باسم ex1.html مثلا وفتحه باستخدام متصفح إنترنت إكسبلورير ، سيظهر في الصفحة:

Hello World!

شرح المثال أعلاه:

لإدراج النص البرمجي في صفحة HTML ، نستخدم علامة <script>، ولتعريف لغة جافاسكريبت كلغة برمجة نصية نكتب:

```
<script type="text/javascript">
```

بعد ذلك يبدأ نص جافاسكريبت: أمر جافاسكريبت لكتابة النص على الصفحة هو document.write

```
document.write("Hello World!")
```

بعد ذلك يجب إقفال العلامة <script>

```
</script>
```

إنـــهاء الجمل بفاصلة منقوطـــة؟

مع لغات البرمجة التقليدية مثل C++ وجافا، كل جملة (تعليمة) برمجية تنتهي بفاصلة منقوطة " ; "..
معظم المبرمجين استمروا في هذه العادة عند الكتابة بلغة جافاسكريبت؛ لكن عموماً، كتابة الفاصلة المنقوطة أمر اختياري! على الرغم من ذلك؛ الفاصلة المنقوطة مطلوبة عند كتابة أكثر من جملة برمجية في سطر واحد.

كيفيـــة التعـــامل مع المتصفحات القديمـــة

المتصفحات التي لا تدعم جافاسكريبت، ستظهر النص البرمجي كمحتوى صفحة –بمعنى أنه سيظهر كما كتبته ضمن HTML - . لمنع ظهور النص البرمجي في المتصفحات التي لا تدعم جافاسكريبت؛ نستخدم علامة التعليقات في HTML كما يلي:

```
<script type="text/javascript">
<!--
    some statements
//-->
</script>
```

الشرطتين المائلتين للخلف (//) في نهاية سطر التعليقات هي رمز التعليقات في جافاسكريبت، وتمنع مترجم جافاسكريبت من ترجمة السطر البرمجي.
ملاحظة: لا يمكنك وضع // في بداية سطر التعليقات (--!><//) لأن المتصفحات القديمة سوف تعرض النص.

جافاسكريبت: ... أين أضع النـــص البرمجي؟

النصوص البرمجية في القسم body ستنفذ عندما يتم تحميل الصفحة... (يتم التنفيذ أولاً بأول).
النصوص البرمجية في القسم head ستنفذ عندما يتم استدعاؤها...

في القسم HEAD:
نضع النصوص البرمجية التي تحتوي على دوال functions في القسم head من الصفحة. بذلك نضمن أن النصوص البرمجية قد تم تحميلها قبل استدعاء الدوال.
في القسم BODY:
يتم تنفيذ النصوص البرمجية الموضوعة في القسم body.
النص البرمجي الخارجي:
كيفية الوصول للنص البرمجي الموجود في ملف آخر External File.

مكـــان وضـــع جافاسكريبت

النصوص البرمجية في الصفحة سيتم تنفيذها مباشرة أثناء تحميلها في المتصفح. وهذا ليس ما نتمناه دائماً؛ أحياناً نريد أن ينفذ النص عند تحميل الصفحة وأحياناً أخرى نريد أن يتم تنفيذ النص عندما يضغط المستخدم على عنصر ما.
النصوص البرمجية في القسم head : النصوص البرمجية التي تحتاج لاستدعاء أو التي تنفذ عند ما ينقر المستخدم على عنصر ما تكون في القسم head ، وبذلك تكون متأكداً من أن النص البرمجي سيتم تحميله قبل استخدامه.

```
<html>
<head>
<script type="text/javascript">
     some statements
</script>
</head>
```

النصوص البرمجية في القسم body: هنا يتم وضع النصوص التي تنفذ عند تحميل الصفحة (أولاً بأول) وهي التي ستكون محتوى الصفحة.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
     some statements
</script>
</body>
```

النصوص البرمجية في كلا القسمين head و body: يمكنك وضع عدد لا نهائي من النصوص البرمجية في القسمين head و body.

```
<html>
<head>
<script type="text/javascript">
     some statements
</script>
</head>
<body>
<script type="text/javascript">
     some statements
</script>
</body>
```

كيفية تنفيذ نص جافاسكريبت موجود في ملف خارجي

أحياناً تحتاج لتنفيذ نص جافاسكريبت ضمن أكثر من صفحة. بدون كتابة النص كل مرة في كل صفحة.
لتبسيط ذلك، يمكنك كتابة نص جافاسكريبت في ملف خارجي وذلك بحفظه بالامتداد js. (مثال: file.js)
كأن تكتب النص البرمجي التالي:

```
document.write("This script is external")
```

واحفظه بالاسم example.js .

ملاحظة: لا يمكن أن يحتوي النص البرمجي الخارجي على علامة <script>.

الآن ، يمكنك استدعاء النص البرمجي المحفوظ في الملف الخارجي من خلال صفة "scr" من أي مكان في صفحة HTML:

```
<html>
<head>
</head>
<body>
<script src="example.js"></script>
</body>
</html>
```

تذكر أن تضع النص البرمجي في المكان الصحيح الذي أردت أن تكتبه فيه.

١ـ أمر الطباعة :

الطريقة الأولى : طباعة جملة لا يقع عليها تأثير اي وسم من وسوم الـ **(html)** ...

```
document.write(" هنا الجملة المراد طباعتها");
```

الطريقة الثانية : طباعة جملة يقع عليها تأثير وسم ال **(html)** ...

```
document.write("<h1> هنا الجملة المراد طباعتها </h1>");
```

الطريقة الثالثة : طباعة جملة يقع عليها تأثير وسم ال **(htm)** و ال **(style)** ...

```
document.write("<h1 style=\"color : red\"> هنا الجملة المراد طباعتها </h1>");
```

الطريقة الرابعة : طباعة قيمة متغير ...

```
document.write( sum );
```

الطريقة الخامسة : طباعة قيمة متغير يسبقه جملة نصية **..**

```
document.write(": الناتج هو  " + sum );
```

الطريقة السادسة : طباعة قيمة متغير يسبقه ويعقبه جملة نصية ..

```
document.write(": الناتج هو  " + sum + " للعملية ");
```

ملاحظات مهمة :

- اي جملة نصية تريد ان تضعها بداخل كود الطباعة يجب ان تحصرها بين **(")** حتى لو احتوت على وسم ال **(html)**

- طباعة قيمة متغير .... ويوجد لها اكثر من حالة :

١ـ اذا كنت تريد فقط ان تطبع قيمة المتغير بمفرده ... تضعه بداخل أمر الطباعة من غير ان تحصره بـ **(").**
٢ـ اذا كنت تريد طباعة قيمة متغير تسبقه جملة نصية يجب ان تحصر الجملة النصية بين **(")** ومن ثم تذكر المتغير المراد طباعة قيمته ولكن يجب ان تضع قبله **(+ ")** ولتوضيح اكثر تضع اشارة **(+)** بالجهة التي بها النص....

مثلا ... جاء النص قبل المتغير نضع اشارة الـ **(+)** قبل المتغير

```
document.write(" : الناتج هو  " + sum );
```

مثلا ... جاء النص بعد المتغير نضع اشارة الـ **(+)** بعد المتغير

```
document.write( sum +" : الناتج هو  " );
```

مثلا ... جاء النص بعد المتغير وقبله نضع اشارة ال **(+)** بعد المتغير وقبله ..

يعني بإختصار ضع اشارة (+) بجانب المتغير المراد طباعته من الجهة التي بها الجملة النصية واذا لم يكن هناك جملة نصية اذا لا تضع اشارة (+)

سوف نذكر ما هو المتغير وما نقصد به لاحقا فلا تقلق فما عليك سوى معرفة طباعته اما ما هو ولأي غرض يستخدم سوف تعرف لاحقا ..

- تستطيع ان تستخدم (document.writeln) بدلا من (document.write) فبكتابة الأول يترك مسافة بين كل كلمة في جملة الطباعة اذا "جاز التعبير" بشكل عملي ..
- لطباعة جملة في سطرين اي " النزول الى سطر جديد " نضع بداخل جملة الطباعة (
) قبل الجملة التي تريد ان نجعلها في سطر جديد ..

مثل :

```
document.write(" Arab <br />Top ");
```

الطباعة سوف تكون بهذا الشكل :

Arab
Top

اي كل كلمة في سطر ...

٢- خروج نافذة للمستخدم نحدد نحن ما يكتب بها وهي خاصه في كائنات النوافذ Window Object.

كهذه النافذة :

وتسمى هذه النافذة بالجافا سكربت (alert) ويكتب كودها بهذه الطريقة ...

```
window.alert(" اهلا بك فى موقعنا ");
```

*** بعض الاوامر التي تستخدم داخل كود ال (alert):**

الامر	وصف الأمر	مثال ..
\n	سطر جديد	window.alert("hi \n all");
\t	لترك مسافة بين كل كلمة وكلمة بحيث تظهر كجدول	window.alert("hi \t all");
\r	لوضع كل كلمة بسطر ولكن بإختلاف ال position	window.alert("hi \r all");
\\	لكتابة رمز (\ )	window.alert("hi \\ all");
\"	لكتابة رمز (")	window.alert("hi \" all");
\'	لكتابة رمز (')	window.alert("hi \' all");

٣- خروج نافذة للمستخدم يحدد هو ما يكتب بها " اي نافذه قراءة من لوحة المفاتيح " وهي خاصه بكائنات النوافذ Window Object .

كهذه النافذة :

وتسمى هذه النافذة بالجافا سكربت **(prompt)** ويكتب كودها بهذه الطريقة ...

```
window.prompt ("من فضلك أدخل إسمك ","إسمك");
```

ملاحظة :
يأخذ كود الـ **(prompt)** بداخله قيمتين الاولى يكتب بها ما سوف يخرج للمستخدم في أعلى النافذة .. والقيمة الثانية يكتب بها قيمة ابتدائية بداخل صندوق الكتابة وهذه القيمة ليس لها اي قيمة فعلية فهي ما تلبث حتى ينتهي دورها بمجرد ان يكتب المستخدم مكانها اسمه او اي شيء يريده ... جرب ان تضع هذا الامر وإنظر ماذا يخرج لك والذي يعتبر ايضا من كائنات النوافذ **Window Object**

```
var con = confirm("هل تريد حفظ البيانات ؟");
```

وهي عبارة عن نافذة تخرج للمستخدم لسؤاله عن قبول أو رفض شيء معين أنت تحدده .

- **المتغيرات :**

نظرة سريعة على المتغيرات :
كما تعلم في أي لغة برمجة تعتبر المتغيرات ذو فائدة كبيرة فبواسطتها نستطيع التحكم ببرنامجنا بكل سهولة ...
ففي لغة الجافا سكربت تعرف المتغيرات بـ **(var)**.

مثال :

```
var name ;
```

ويجدر التنبيه هنا انه أذا اردنا أن يكون المتغير قيمة رقمية **(integer Or float)** " أعداد صحيحة أو أعداد ذات أرقام عشرية " فإننا نقوم بتحويلها بهذه الطريقة :

1- parseInt (أسم المتغير هنا) ;

وهذا سوف يحول العدد الى عدد صحيح **(integer)** أي يكون كتابة الكود كالتالي :

```
var number = 55 ;
parseInt (number) ;
```

2- parseFloat (أسم المتغير هنا) ;

وهذا سوف يحول العدد الى عدد عشري (float) أي يكون كتابة الكود كالتالي :

```
var number = 5.4 ;
parseFloat (number) ;
```

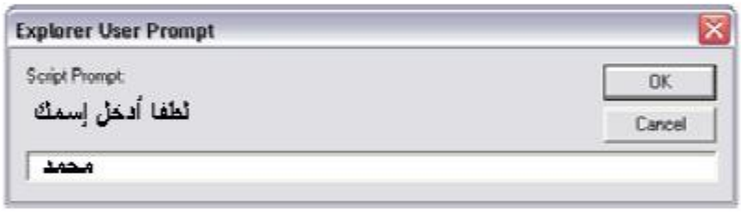
ملاحظة :

عليك الإلتزام بشكل الحرف أي الحرف الصغير والكبير (parseFloat) .
..وللحديث بقية سوف يذكر في وقته.

لنأخذ أمثلة على ما سبق ونطبقها ...

مثال ١ :

```
<html dir="rtl">
  <head><title>الجافا سكربت</title>
   <script>
      var name ;
      name = window.prompt ( "لطفا أدخل إسمك " , "إسمك" ) ;
      document.write( " أهلا بك " + name  ) ;
   </script>
  </head>
 <body> </body>

</html>
```



في هذا المثال قمنا بما يلي :
عرفنا متغير وجعلنا اسم المتغير هو **(name)** .

جعلنا هذا المتغير يساوي القيمة التي سوف يدخلها المستخدم من لوحة المفاتيح وذلك من خلال الأمر **(window.prompt)** . فبمجرد خروج النافذة للمستخدم ويكتب بها اسمه سوف يخزن الأسم المدخل بداخل المتغير **(name)** وهذا واضح من العبارة :

```
name = window.prompt ( "إسمك", " لطفا أدخل إسمك" ) ;
```

ثم قمنا بطباعة المتغير **(name)** وهذا من خلال الأمر :

```
document.write( " أهلا بك " + name  ) ;
```

وبالتالي سوف يطبع لنا ما خزن داخل المتغير وهو إسم المستخدم الذي أدخله . وبعد ذلك سوف يطبع الجملة النصية التي كتبناها وهي " أهلا بك " ... أي سوف يطبع لنا الجملة التالية على فرض اننا قمنا بإدخال الإسم " محمد " : **" أهلا بك محمد "**

مثال ٢ :

```
<html>
   <head><title>الجافا سكريبت</title>
     <script type = "text/javascript" >
        window.alert ("أهلا بك") ;
     </script>
   </head>
 <body></body>

</html>
```

وفي هذا المثال خروج نافذه للمستخدم وهي نافذة (alert) التي ذكرناها سابقا ..

- **العمليات الحسابية :**

أسم العملية الحسابية	الصيغة	خوارزمية الصيغة	مثال
عملية الجمع	+	x + y	x + y
عملية الطرح	-	y –x	x – y
عملية الضرب	*	x * y	X * y
عملية القسمة	/	x / y	x / y
عملية باقي القسمة	%	x % y	x % y

- **العمليات المنطقية :**

الصيغة	تمثيلها بالجافا	وصف الصيغة	مثال
=	==	x == y	قيمة x تساوي قيمة y
≠	=!	y =! x	قيمة x لا تساوي قيمة y
>	>	x > y	قيمة x أكبر من قيمة y
<	<	x < y	قيمة x أقل من قيمة y
>=	>=	x >= y	قيمة x أكبر أو تساوي قيمة y
<=	<=	x <= y	قيمة x أقل أو تساوي قيمة y

ملاحظة :
يوجد أولويات تعتمد في العمليات الحسابية البرمجية.

- **لنأخذ بعض العمليات الحسابية بشكل سريع :**

```
var number = 4 ;
number+ = 2 ;
```

المتغير **(number)** كان يحمل القيمة **(4)** ثم أضفنا له **(2)** فأصبح الناتج **(6)**.

ويمكن أن نكتب العمليات السابقة بطريقة برمجية أخرى وسوف يخرج لنا نفس الناتج :

```
var number = 4 ;
number= number+ 2 ;
```

وما ينطبق على عملية الجمع ينطبق على جميع العمليات الحسابية الأخرى ....

مثال :

```
var number = 4 ;
number * = number:
```

وهي عملية ضرب وضعنا بداخل المتغير **(number)** قيمة **(4)** ثم اجرينا عملية الضرب على هذا المتغير الذي يحتوي على **(4)** وهي ضربه بنفسه وبالتالي الناتج هنا يساوي **(16)**.

- **جمل الشرط :**

قد نحتاج أثناء كتابتنا للبرنامج أن نضع جملة شرط أو أكثر به فبناتج جملة الشرط نستطيع أن نتحقق من المطابقة أو عدمها.. إذا هي مقارنة بين قيمتين قد يتطابقوا وعندها يكون الجواب **(True)** أي صحيح أو قد لا يتطابقوا وعندها يكون الجواب **(False)** أي خاطئ .

ولتوضيح الصورة، كأن نتحقق من كلمة السر هل هي صحيحة أم لا إذا هنا سوف يكون لدينا كلمتين سوف نقارن بينهما وهما كلمة السر المسجلة لدينا وكلمة السر التي سوف يدخلها المستخدم .

- **لتتعرف إذا على جمل الشرط وأنواعها :**

١ـ الدالة الشرطية (if) :

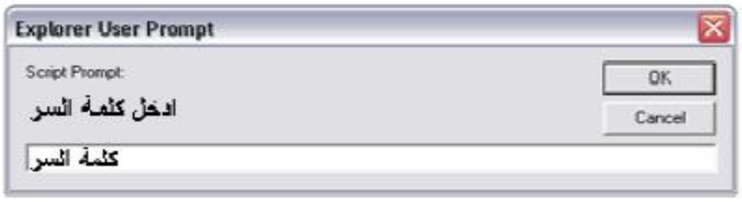
في هذه الدالة نقوم بالتأكد من الشرط فإذا تحقق يتم الدخول الى داخل الدالة وبعد الإنتهاء من أوامرها ينتقل الى تنفيذ الاوامر التي تقع خارجها وإذا لم يتحقق سوف يتم الانتقال مباشرة الى تنفيذ الاوامر التى تقع خارج الدالة .

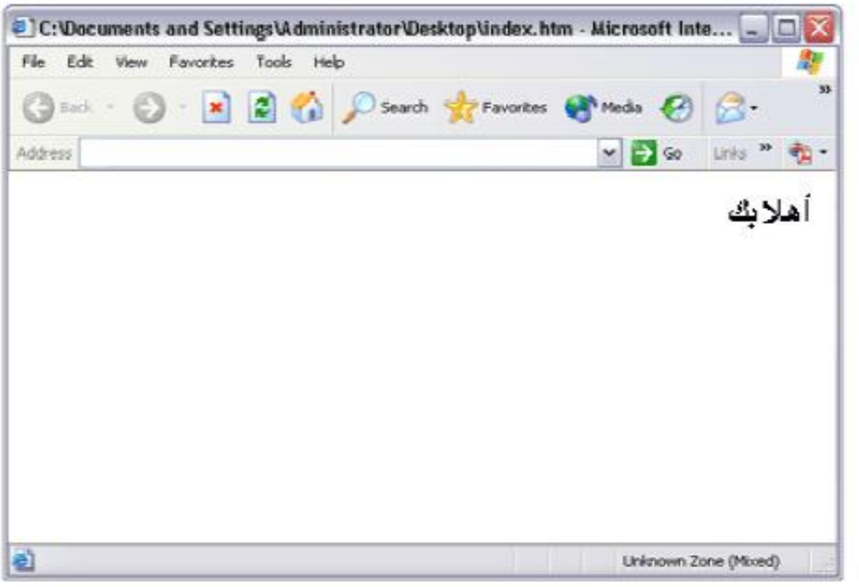
صيغتها:

```
If ( الشرط )
   {
        إذا تحقق ـ نفذ الامر الذي بداخل الدالة
   }
        أوامر خارج الدالة
```

لنأخذ مثالا كاملا ونرى كيف :

```
<html dir="rtl">
   <head><title>الجافا سكربت</title>
     <script type ="text/javascript">
       var pass_2="okman";
       var  pass_user;
       pass_user = window.prompt("ادخل كلمة السر","كلمة السر");
         if (pass_2 == pass_user)
            {
                document.write (" أهلا بك ") ;
            }
     </script>
   </head>
 <body></body>

</html>
```





في هذا المثال قمنا بتعريف متغير (pass_2) ووضعنا بداخله قيمة (okman) ثم قمنا بتعريف متغير (pass_user) ولم نضع بداخله قيمة على اساس ان المستخدم اي الزائر هو الذي سوف يضع القيمة وهنا سوف تخرج نافذة لتخبر المستخدم ان يدخل كلمة السر وعندها يضع المستخدم الكلمة وبالتالي سوف تخزن هذه القيمة بداخل المتغير (pass_user) وبعد ذلك ندخل بداخل جملة الشرط وهي الـ (if) وهنا سوف يقارن بين قيمة المتغير (pass_2) المخزنه مسبقا وبين قيمة المتغير (pass_user) الذي حدد قيمته المستخدم اذا جملة (if (pass_2==pass_user)) تعني هل المتغير (pass_2) المخزن مسبقا يساوي قيمة المتغير (pass_user) المدخلة من قبل المستخدم .. هنا اذا كانت الاجابه نعم عندها سوف يطبع له جملة أهلا بك واذا كانت لا اي غير متساويتين اي ان كلمة السر الذي ادخلها المستخدم ليست متطابقة مع كلمة السر المخزنه مسبقا ... عندها لن يطبع شيء .. وفي هذا المثال افترضنا انه ادخل كلمة السر صحيحه" اقصد هنا في صورة المثال" ..

٢- الدالة الشرطية (if / else) :

في هذه الدالة الشرطية نقوم بالتأكد من الشرط وفي هذه الحالة سيكون لنا حالتين :

الاولى :

تحقق الشرط وبالتالي الدخول الى داخل الدالة لتنفيذ الاوامر التى بداخلها ثم الانتقال الى تنفيذ الاوامر التى تقع خارجها .

الثانية :

عدم تحقق الشرط وبالتالي الانتقال الى جملة (else) وتنفيذ مابداخلها ثم الانتقال الى تنفيذ الاوامر التى تقع خارجها .

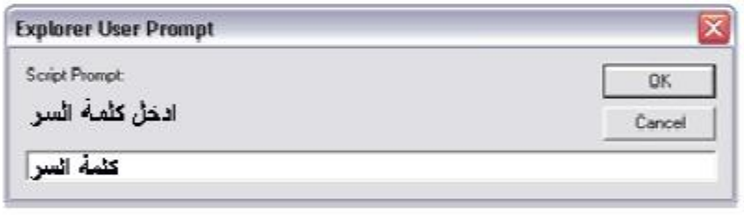
صيغتها :

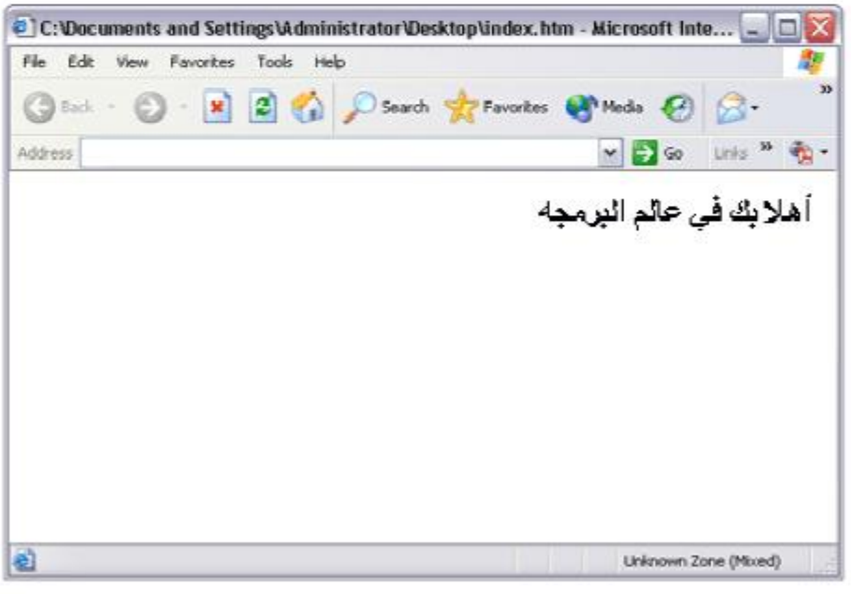
```
If ( الشرط )
    {
       إذا تحقق نفذ مابداخلها
    }
else
    {
     نفذ هذا الامر إذا لم يتحقق الشرط
    }
          تنفيذ الاوامر الخارجية
```

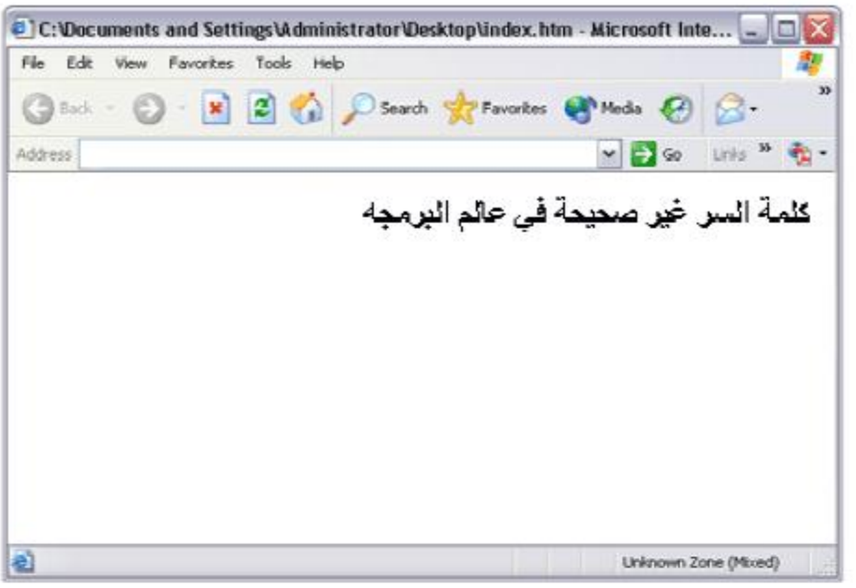
لنأخذ مثالا كاملا ونرى كيف ..؟

مثال :

```
<html dir="rtl">
  <head><title>الجافا سكربت</title>
    <script type ="text/javascript">
      var pass_2="okman";
      var  pass_user;
      pass_user = window.prompt("ادخل كلمة السر","كلمةالسر");
       if (pass_2 ==pass_user  )
             {
               document.write ("  أهلا بك  " ) ;
             }
       else
             {
               document.write ("  كلمة السر غير صحيحة  ") ;
             }
      document.write (" في عالم البرمجة ") ;
    </script>
  </head>
 <body></body>

</html>
```







في هذا المثال قمنا بتعريف متغير (pass_2) ووضعنا بداخله قيمة (okman) ثم قمنا بتعريف متغير (pass_user) ولم نضع بداخله قيمة على اساس ان المستخدم اي الزائر هو الذي سوف يضع القيمة و هنا سوف تخرج نافذة لتخبر المستخدم ان يدخل كلمة السر وعندها يضع المستخدم الكمله وبالتالي سوف تخزن هذه القيمة بداخل المتغير (pass_user) وبعد ذالك ندخل بداخل جملة الشرط و هي الـ (if) و هنا سوف يقارن بين قيمة المتغير (pass_2) المخزنه مسبقًا وبين قيمة المتغير (pass_user) الذي حدد قيمته المستخدم اذا جملة (if (pass_2==pass_user)) تعني هل المتغير (pass_2) المخزن مسبقًا تساوي قيمة المتغير (pass_user) المدخل من قبل المستخدم .. هنا اذا كانت الاجابه نعم عندها سوف يطبع له جملة أهلا بك واذا كانت لا اي غير متساوين اي ان كلمة السر الذي ادخلها المستخدم ليس مطابقة لكلمة السر المخزنه مسبقًا ... عندها سوف يطبع الجمله التي في (else) و هي كلمة السر غير صحيحه.. وبعد ان يطابق ويرى اذا كانت صحيح ام لا ليحدد اي جملة يطبع سوف يخرج من دالة الـ (if) الى تنفيذ الجملة الخارجية اي التي خارج الدالة والتي ليس لها علاقة بماذا كانت المطابقة لا ام نعم وبتالي طباعة .. عالم البرمجة .. اذا هنا إما ان تكون الكلمة المدخله صحيحه وعندها سوف يطبع جملة اهلا بك ثم يطبع جملة عالم البرمجه او ان تكون الكلمة المدخله خطأ وعندها يطبع كلمة السر غير صحيحه ثم يطبع عالم البرمجه .. وفي هذا المثال افترضنا انه ادخل كلمة السر صح " اقصد هنا في صورة المثال "..

- **معلومات مهمة عن الدالة (if) :**

١- قد يستخدم المبرمج شرط بداخل شرط وتسمى هذه الحالة (nested if) فلا يزعجك ذلك فلكل منا أسلوبه الخاص في وصف الامور كما أنه قد يتحتم علينا أوقات أن نستخدم شرط بداخل شرط .

مثال :

```
<html dir="rtl">
   <head><title>الجافا سكربت</title>
      <script type ="text/javascript">
         var name1 ="sami";
         var name2 ="mahmoud";
         if (name1 =="sami")
            {    if (name2 =="mahmoud")
                    document.write ("أهلا بك" ) ;
            }
         document.write (" في عالم البرمجة " ) ;
      </script>
    </head>
  <body></body>

</html>
```

في هذا المثال أضطررنا أن نستخدم شرط بداخل شرط وهذا الأسلوب يسمى (الأسلوب الهرمي اذا جاز التعبير) فهنا وضعنا شرطين بداخل بعض ... **لماذا** ؟
لنفرض أن لدينا عدة أشخاص جميعهم الإسم الأول سامي ولكن إسم الأب أي الإسم الثاني يختلف إذا هنا يجب أن نتحقق من الإسمين الأول والثاني إذا الخطوات التي نتبعها كما يلي :
نحضر جميع الأشخاص الذين إسمهم الأول (سامي) وهذا مافعلناه بالشرط الأول .

```
If ( name1 == " sami " );
```

فإذا كان الشرط قد تحقق أي أن أسم المستخدم (سامي) سوف يدخل الى الدالة ومن ثم سوف يجد شرطا آخر يسأله عن أسمه الثاني أي أسم الأب :

```
if ( name2 == " mahmoud ");
```

فإذا تحقق سوف ينفذ الامر الذي بداخل هذه الدالة . ومن ثم ينفذ الاوامر التي خارج الدالتين تماما .

لو لاحظت أنه في حالة عدم تحقق أي من الشرطين السابقتين سوف يخرج لتنفيذ ما بخارج الدالتين .

- وقد يستخدم شخص أيضا دالة (**if**) داخل دالة (**if**) داخل دالة (**if**) وهكذا فلا أريد أن أدخلك في متاهات وخوارزميات معقدة . فكما قلنا لكلا منا أسلوبه فقد يكون حل مسألة بسطر فيأتي شخص ويحلها بأربعة أسطر بطريقة معقدة... إذا جميعنا يستطيع أن يصل الى الحل ولكن بأساليب مختلفة وقد يتطلب الامر ان نستخدم جمل (**if**) بداخل بعضها.

- **الدالة الشرطية (switch) :**

عندما يكون لدينا عدة خيارات ونكون نريد أن نخرج بواحد منهم وهو الذي نريده أما ماهو الذي نريده من بين الخيارات والذي سوف نخرج به سوف يحدده المتغير الذي سوف ندخله والذي سوف يتفق مع واحدة من هذه الخيارات ويحققه .. أي هنا وكأننا نستخدم الجملة الشرطية (if) أو ماشابه ذلك اذا جاز العبير..
ويأتي أستخدام الدالة (switch) ليسهل العمل بشكل كبير وسوف نرى ذلك :

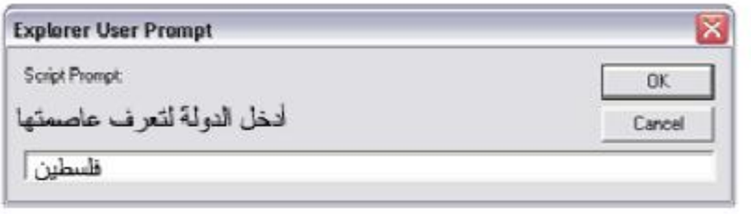
صيغتها :

```
switch ( المتغير ) { بداية

        case : الاحتمال الأول ;
               المطلوب لهذا الاحتمال

        case : الاحتمال الثاني ;
               المطلوب لهذا الاحتمال

        case : الاحتمال الثالث ;
               المطلوب لهذا الاحتمال
           .
           .
           .
        default :
                                  المطلوب لهذا الاحتمال .. وهو يطبق في حالة عدم تطابق المتغير مع اي case ذكر ..
                        نهاية " اي اغلاق " {
```

لنأخذ مثال ونحلله.. والذي من خلاله سوف نفهم ماقصدناه في عباراتنا السابقه.. فأنا أعرف جيدا عزيزي المتدرب أنه كلام مبهم وغير واضح لك الى الان.

مثال :

```
<html dir="rtl">
 <head><title>الجافا سكربت</title>
  <script type = "text/javascript">
   var country ;

  country=window.prompt( " أدخل الدولة لتعرف عاصمتها ", "0");

  switch ( country ) {

    case " فلسطين " :
        document.writeln("<h3> عاصمتها القدس الشريف</h3>") ;
            breack ;

    case " العراق " :
        document.writeln("<h3> عاصمتها بغداد </h3>") ;
            breack;

    case " السودان " :
        document.writeln("<h3> عاصمتها الخرطوم </h3>") ;
        breack ;

    case " الوطن العربي " :
        document.writeln("<h3> من المحيط الى الخليج </h3>") ;
         breack ;

    default :
         document.writeln("<h3>      الدولة التي أدخلتها ليست من ضمن الخيارات المتاحة </h3>") ;
                        }
     </script>
   </head>
  <body></body>
</html>
```



- في حالة ادخال داله ليست ضمن الدول المذكوره في ال(case) :

لنحلل المثال سويا فبه نعرف مابهم علينا :
في البداية عرفنا المتغير (country) وأسندناه الى جملة (window.prompet) لكي يدخله المستخدم بنفسه من لوحة المفاتيح أي انه سوف يدخل أسم دولة معينة ... وهنا سوف نأتي الى وضع المتغير في دالة (switch) كالتالي :

```
switch ( country )
```

ثم نضع الخيارات التي نريدها وسوف يقارن المتغير الذي أدخلنا قيمته بما هو موجود بجانب كلمة (case) كالشكل التالي :

```
case " السودان " :
```

وفي حالة مطابقته لأي واحدة من الخيارات التي وضعناها سوف ينفذ الجمل التي جاءت ضمن ال (case) الذي توافق معه وبالتالي طباعة الجملة والخروج من دالة (switch) ولهذا نضع بنهاية كل (case) أمر (break) ليتوقف , لاننا هنا نريد أن نخرج بقيمة واحدة وهي احدى الخيارات التي وضعناها " جرب أن لا تضع (break) وأنظر ماذا سيحدث فبالتجربة يحدد المفهوم " .
ولاحظ أننا وضعنا في النهاية الدالة وقبل اغلاقها (default) وهنا نعني أنه في حالة أن المستخدم أدخل قيمة للمتغير وكانت ليست ضمن الـ (case) سوف يخرج المستخدم بالجملة والامر الذي سوف نضعه بعد (default) ثم ننهي الدالة بوضع النهاية (}) فما رأيك عزيزي المتدرب أن نأخذ المثال وكأننا مستخدمون وننفذه الان على غير ما تم ادخاله في الصور السابقة من دولة فلسطين .

نفرض خرجت نافذة (prompt) وأدخلنا الدولة العراق مثلا فهنا سوف يأتي البرنامج ويأخذ كلمة العراق والتي هي اسندت الى المتغير ويضعها في دالة (switch)

```
switch ( country )
```

فيدخل بداخل دالة (switch) فيأتي الى الـ (case) الاول والتي هي :

```
case " فلسطين " ;
```

فيجد أن كلمة العراق لاتتطابق كلمة فلسطين إذا ليس هذا هو الخيار . فينتقل الى الـ (case) الآخر ...

فيجد أن المتغير به كلمة العراق يطابق هذا الـ (case) أي (العراق = العراق) فينفذ الجملة التي بداخل هذا الـ (case) التي هي :

```
document.writeLn( "<h3>  عاصمتها بغداد </h3> ") ;
```

وبالتالي طباعة الجملة التالية : " عاصمتها بغداد "

وبعد جملة الطباعة هذه يأتي أمر (break) أي توقف وبالتالي الخروج من دالة (switch) كاملتا دون المرور بالـ (case) التي تأتي بعدها .

- وأيضا لو فرضنا أن شخص أدخل أي دولة غير موجودة ضمن الخيارات لتكن الاردن أو السعودية أو الكويت أو الامارات ... الخ سوف يطبع هنا الجملة التي ضمن الامر (default) أي :

```
document.writeln( "<h3>  الدولة التي أدخلتها ليست من ضمن الخيارات المتاحة </h3> ") ;
```

▪ الذي أريد ان أوصله لك في النهاية هو شئ واحد وهي أننا ندخل المتغير في جملة (switch) للمقارنة مع الخيارات الموجودة بداخلها وعند مطابقة المتغير مع احدى الخيارات أي النتيجة (true) سوف ينفذ الجملة المتعلقة بهذه المطابقة مع العلم أن واحدة فقط من هذه الخيارات تعطي (true) والباقية (false) وعندما يكون كل الخيارات ليست مطابقه سوف ينفذ ما بداخل (default) اي ان جميع ال (case) غير مطابقة (false).. اذا ياتي عمل ال (default).

- **حلقات التكرار :**

ونعني بها تكرار جملة معينة عدة مرات فحلقة التكرار سوف تستمر مادام الشرط متحققا ويتم التحقق من الشرط في بداية الحلقة او نهايتها .

- **قد تتساءل مافائدة أستخدام حلقات التكرار ؟**

نجيب بذلك . . .

أفترض انك تريد طباعة جملة معينة ١٠ مرات فأنك سوف تحتاج الى حلقة التكرار فلو لم تستخدمها لإضطررت لكتابة كود الطباعة ١٠ مرات ولكن بحلقة التكرار لايأخذ منك سوى كتابة سطرين من الكود . وكذلك الحال لو أردت طباعة أسماء جميع من هم مسجلين لديك بقاعدة بيانات الموقع .

- **أنواع حلقات التكرار :**

١- حلقة التكرار (for) :

وفي هذه الحلقة يكون الشرط هو الذي يحدد طول الحلقة فما أن يصل الى نهايتها حتى يتوقف أي فشل الشرط ولكن مادام الشرط متحقق سوف يدخل الى داخل الحلقة وينفذ مابداخلها .

صيغتها :

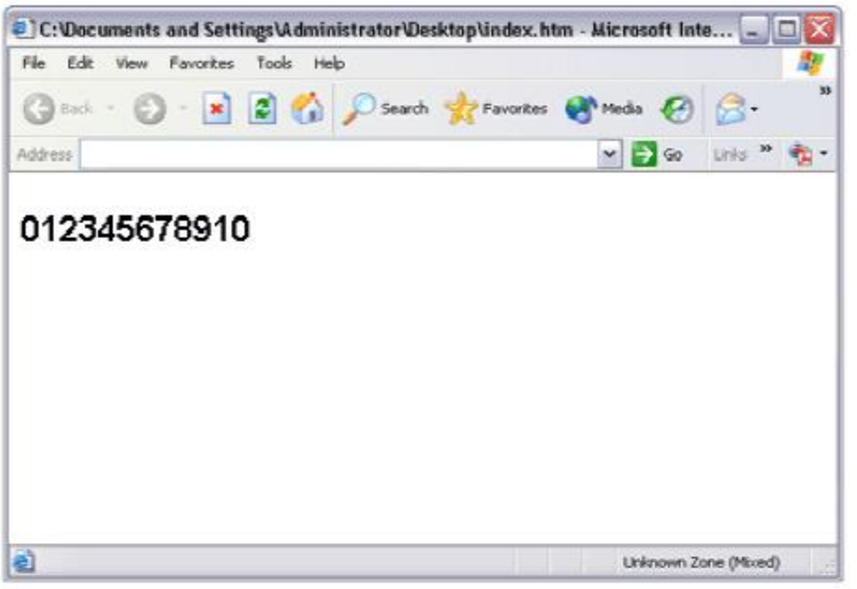
```
for ( بداية الحلقة ; شرط الحلقة ; مقدار الزيادة )
  {
        نفذ الامر
  }
```

شرط الحلقة : هو الذي سوف يحدد طول الحلقة (عددالتكرار)

مثال :

```
<html >
  <head><title>الجافا سكريبت</title>
   <script type = "text/javascript">

     for (var i = 0 ; i<= 10 ; i ++ )
         {
         document.write ( i ) ;
         }
   </script>
  </head>
 <body></body>
</html>
```

ملاحظة : انا عرفت المتغير i في داخل حلقة التكرار تستطيع تعريفه في الخارج كما في السابق ووضعه مباشرة ...



لنحلل هذا المثال سويا :
هنا أول شي عرفنا المتغير وهو **(i)** وجعلنا قيمته الأبتدائية
(i=0) أي أنه سوف يبدأ من **(0)** .
ثم حددنا الشرط والذي هو **(i < = 10)** أي أننا سوف ندخل الى داخل الحلقة وننفذ مابداخلها من أوامر مادامت قيمة **(i)** أقل أو تساوي **(10)** أي أنها لو وصلت الى **(11)** سوف يخرج من الحلقة ولن يدخلها إذا هنا حددنا نهاية **(i)** وهي **(10)** . أي أن التكرار سوف يكون ١٠ مرات ولاننا بدأنا من الصفر سوف يكون ١١ .
ثم حددنا مقدار زيادة المتغير **(i)** وحددناه بأن يزيد بـ **(1)** بكل مرة يدخل فيها الى الحلقة .

*** نأتي للتطبيق العملي للمثال " أي مايحدث بداخل البرنامج " :**
لنكتب الصيغة ونتتبعها :

```
for ( var i = 0 ; i < = 10 ; i ++ )
```

نبدأ بقيمة **(i)** كما ذكرنا وهي **(0)** .
ننتقل الى الخطوة التالية التى تليها وهي الشرط **(i < = 10)** وهنا يسأل هل **(i)** أقل أو تساوي **(10)** سوف تكون الاجابة نعم .
ثم ينتقل الى للخطوة التى تليها وهي **(i++)** أي يعني ذلك بعد الدخول الى حلقة التكرار والرجوع الى**(for)** زيد **(i)** بمقدار **(1)** .
دخلنا الى الحلقة الان سوف يطبع قيمة الـ **(i)** والتى هي **(0)** ثم يعود الى**(for)** ولكن بعد زيادة قيمة **(i)** بقيمة **(1)** لتصبح قيمتها **(1)**.
الان سوف يكون مافعلناه بالخطوات السابقة ولكن هذه المرة سوف تتغير قيمة الـ **(i)** الأبتدائية الى **(1)** بدلا من **(0)** .

- **للتوضيح أكثر نعيد الخطوات :**
  - الان سوف يسأل هل **(1)** أقل أو يساوي **(10)** سوف تكون الأجابة نعم فالـ **(1)** أصغر من **(10)** .
  - ثم ينتقل الى الخطوة التى تليها وهي **(i++)** أي يعني ذلك بعد الدخول الى حلقة التكرار وتنفذ مابداخلها والرجوع الى الـ **(for)** زيد قيمة **(i)** بمقدار **(1)** أي تصبح **(2)** .
  - وهنا سوف يعود الى الـ **(for)** وسوف تصبح قيمة ال **(i)** الأبتدائية "اذا جاز التعبير" وهي **(2)** ويسأل الشرط والخطوات التى ذكرناها سابقا ويستمر حتى يصل ال قيمة **(11)** وهنا سوف يحدث التالي :

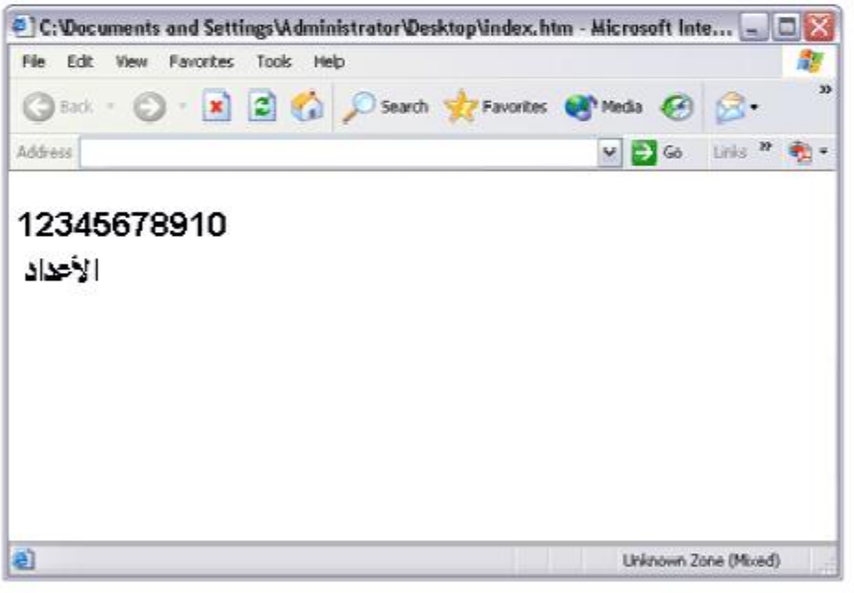
- سوف يجعل قيمة الـ (i) الأبتدائية (11) ثم ينتقل الى الخطوة التى تليها بالـ (for) وهي الشرط (i < = 10) سوف يسأل هل قيمة (i) والتى هي الان (11) أقل أو تساوي (10) سوف تكون الاجابة لا إذا سوف يخرج من الحلقة ولن ينفذ الجملة التى بداخلها .

- **يعني بإختصار . . .**

سوف يستمر بتنفيذ جملة الطباعة التى بداخل حلقة الـ (for) مادام الشرط متحققًا ولن يخرج منها إلا بعد فشل الشرط . فلو كان يوجد جملة تلي حلقة التكرار كالمثال الذي سوف أطرحه الان لن تنفذ الا بعد الخروج من حلقة التكرار .

مثال :

```
<html >
   <head><title>الجافا سكربت</title>
     <script type = "text/javascript">
         for (var i = 1 ; i <=10; i ++ )
             {
                document.write ( i ) ;
             }
         document.write (" <br /> الاعداد");
     </script>
   </head>
  <body></body>

</html>
```



ملاحظة : سوف احاول شرحه بطريقه اكثر تقريبا ...
هنا في هذا المثال سوف يبدأ قيمة الـ (i) وهي (1) وسوف يكون الشرط أن مادام قيمة الـ (i) أقل من أو يساوي (10) أطبع وعند فشل الشرط أخرج من حلقة التكرار وإذهب لتنفيذ الجملة التى تلي حلقة التكرار وهي طباعة جملة (الأعداد)

هنا يكون التنفيذ على النحو التالي : رسمة لتوضيح

```
i =1 → i ++ → i = 2
i =2 → i ++ → i = 3
i =3 → i ++ → i = 4
i =4 → i ++ → i = 5
i =5 → i ++ → i = 6
i =6 → i ++ → i = 7
i =7 → i ++ → i = 8
i =8 → i ++ → i = 9
i =9 → i ++ → i =10
```

```
i = ' 10 ' → i ++ → i =11
```

فنحن محددين أنه آخر قيمة تطبع هي (10) فبعد أن يزيد قيمة (i) سوف يفشل الشرط وبالتالي سوف يخرج من حلقة التكرار لان قيمة (i) اصبحت (11) اذا هي ليست اقل من (10) ثم ينتقل الى الجملة التى تلي حلقة التكرار والتى هي خارج حلقة التكرار وهي طباعة " الأعداد "

```
document.write ( " <br /> الاعداد");
```

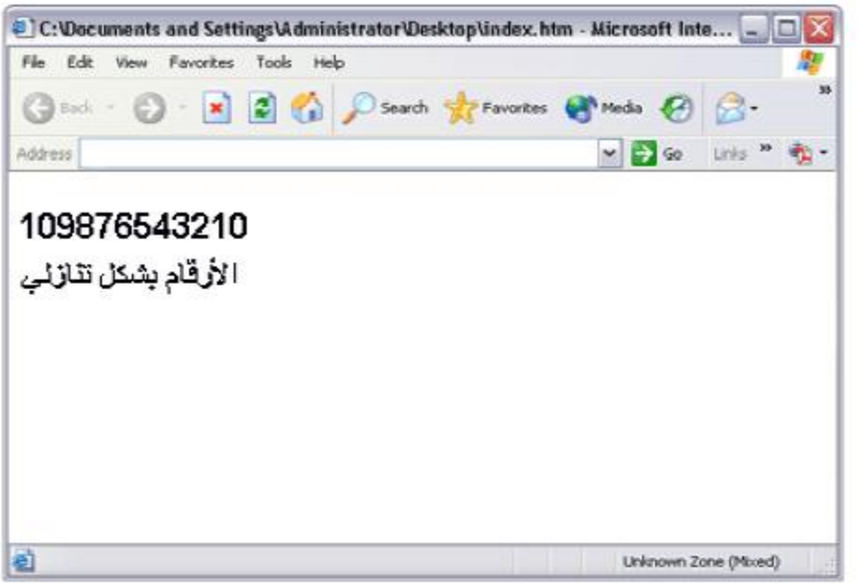
إذا النتيجة هي طباعة مايلي :

12345678910
الاعداد

لأطرح لك مثالا وأترك تحليله لك ...
لنفرض أننا نريد طباعة الارقام ولكن بالعكس أي من (10) الى (0)

تكون صيغته كالتالي :

```
<html >
    <head><title>الجافا سكربت</title>
      <script type = "text/javascript">
        for (var i =10 ;  i>=0 ; i--)
           {
              document.write( i ) ;
           }
        document.write ( " <br /> الأرقام بشكل تنازلي")
      </script>
    </head>
  <body></body>

</html>
```



النتيجة :

10987654321
الأرقام بشكل تنازلي

وهذا المثال يسمى طباعة ' **reverse** ' للقيم اي طباعة عكسية.

- **حلقة التكرار (While) :**

وفي هذه الطريقة سوف يستمر يتنفيذ مابداخل الحلقة مادام الشرط متحقق في كل مره تريد بها الدخول سوف يتحقق من الشرط أولا فإذا تحقق تقوم بالدخول الى داخل الحلقة وتنفيذ مابداخلها الى أن يفشل ونخرج من الحلقة مع العلم انه اذا فشل من البدايه اذا لن يدخل اطلاقا الى داخل حلقة التكرار ولكن اذا يوجد اوامر خارج حلقة التكرار اي بعد حلقة **while** سوف ينفذها .

صيغتها :

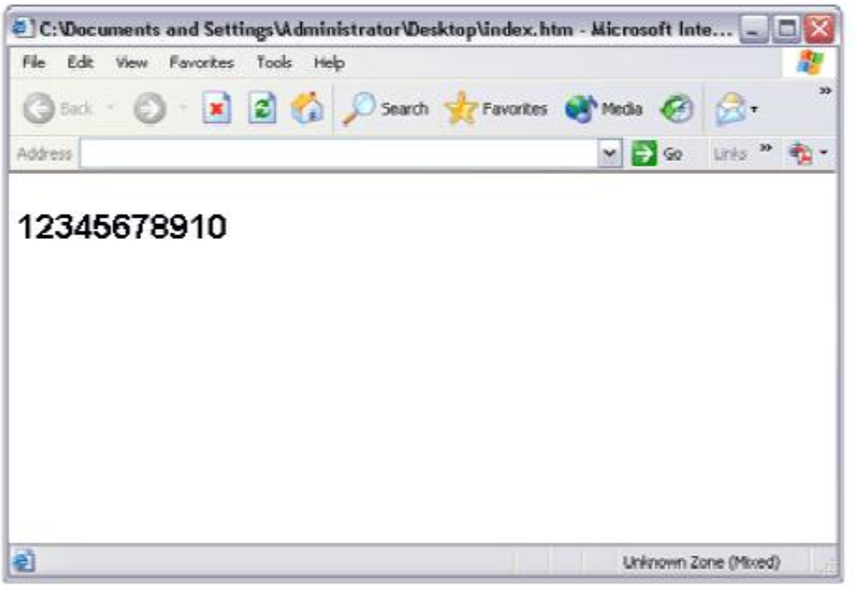
```
While ( نفذ مابداخل الحلقة مادام الشرط متحققا )
{
      نفذ الامر
  i ++   مقدار الزيادة
}
```

قبل الخوض بمثال يجب أن تحتوي جملة الـ (while) على مايلي :

١- متغير نضعه بالشرط لكي نتحقق من صحة الشرط .
٢- يجب وضع قيمة إبتدائية لهذا المتغير قبل جملة الـ (while) .
٣- يجب أن نذكر هذا المتغير ومقدار زيادته بداخل حلقة الشرط سواءا قبل تنفيذ الجملة التى بداخل حلقة التكرار أو بعدها .

مثال :

```
<html >
 <head><title>الجافا سكربت</title>
  <script type = "text/javascript">
   var   i = 1 ;
   while (i <= 10 )
    {
     document.write ( i ) ;
     i ++ ;
     }
  </script>
 </head>
<body></body>
</html>
```



لنحلل المثال سويا :
أول شئ عرفنا (i) كمتغير ووضعنا به القيمة الأبتدائية (1) قبل أن نضعه في الشرط (i <= 10) .
وبعد ذلك ذهبنا الى جملة (while) فسألنا الشرط هل قيمة (i) أقل أو تساوي (10) فكانت الاجابة نعم فالـ (1) أقل من (10) وبما أنه تحقق
الشرط فسوف ندخل بداخل الحلقة ...

```
While ( 1 <= 10 )
```

فدخلنا بداخلها ونفذنا الجملة التى بداخل حلقة التكرار التى هي طباعة قيمة (i) والتى هي (1) .

```
document.write ( i ) ;
```

ثم أنتقلنا الى الجملة التى تليها والتى هي (i++) وتعني زيد قيمة (i) بمقدار (1) فأصبحت قيمة (i) تساوي (2).

تم عاد مرة أخرى الى (while) فسأله الشرط هل قيمة (i) أقل أو تساوي (10) فكانت الاجابة بنعم فنحن عدنا للـ (while) بقيمة (i) بعد زيادتها بـ (1) أي (2) .

```
While ( 2 <= 10 )
```

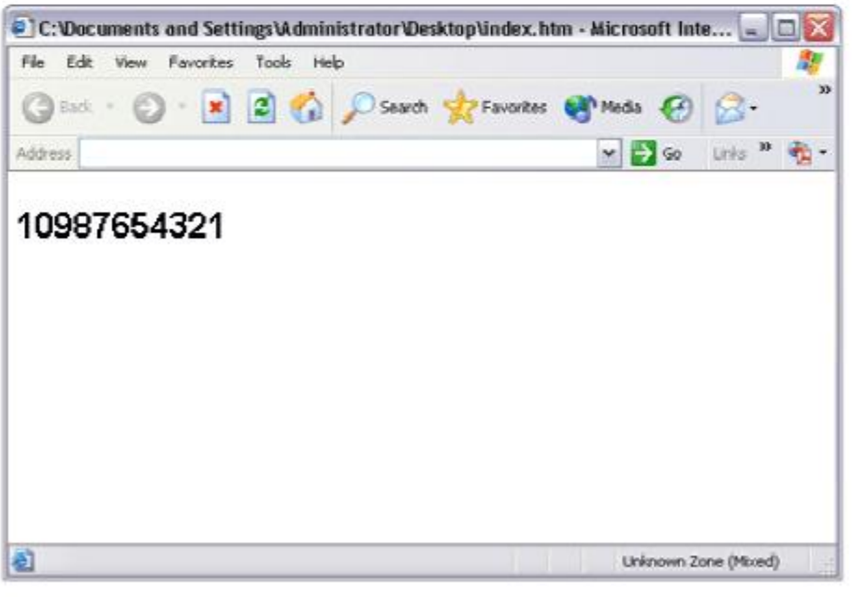
فبعد تحقق الشرط دخلنا الى داخل الحلقة فوجدنا أمر الطباعة قيمة (i) وهو (2) .
ثم أنتقلنا الى الجملة التى تليها والتى هي (i++) " أي زيد قيمة (i) بمقدار (1) وقيمة (i) عندنا الان (2) زيدها بواحد لتصبح (3) " .

ثم نعود الى الـ (while) مرة أخرى ومعنا قيمة الـ (i) الجديدة وهي (3) ويتحقق من الشرط مرة أخرى وهكذا نستمر 10 مرات الى أن تصل قيمة (i) الى (11) وهنا سوف يحدث التالي :

يأتي الجملة الـ (while) فسأله الشرط هل قيمة (i) التى هي (11) أقل أو تساوي (10) فيكون الجواب لا فيخرج من حلقة التكرار ولايدخل بداخلها . وإذا كان هناك جملة تلي حلقة التكرار سوف ينفذها بعد خروجه من حلقة التكرار .
(لاحظ جملة تلي حلقة التكرار وليس بداخله حلقة التكرار) .
مثال اخر (لنجعلك تقارن بين (while) و (for)) . .

مثال (1) :

```
<html >
    <head><title>الجافا سكربت</title>
  <script type = "text/javascript">
     var  i = 10 ;
    while (i>= 1 )
        {
          document.write ( i ) ;
           i -- ;
        }
  </script>
 </head>
<body></body>
</html>
```



- **حلقة التكرار (Do while) :**

وفي هذه الطريقة سوف يستمر بتنفيذ مابداخل الحلقة مادام الشرط متحقق و هنا سوف يدخل الى داخل الحلقة ومن ثم ينفذ الامر الذي بداخلها وبعد تنفيذها ينتقل ليتحقق من الشرط فإذا تحقق يعود مرة آخرى الى الحلقة وينفذ الامر مرة أخرى وإذا لم يتحقق يخرج من حلقة التكرار ولن يعود لها .

صيغتها :

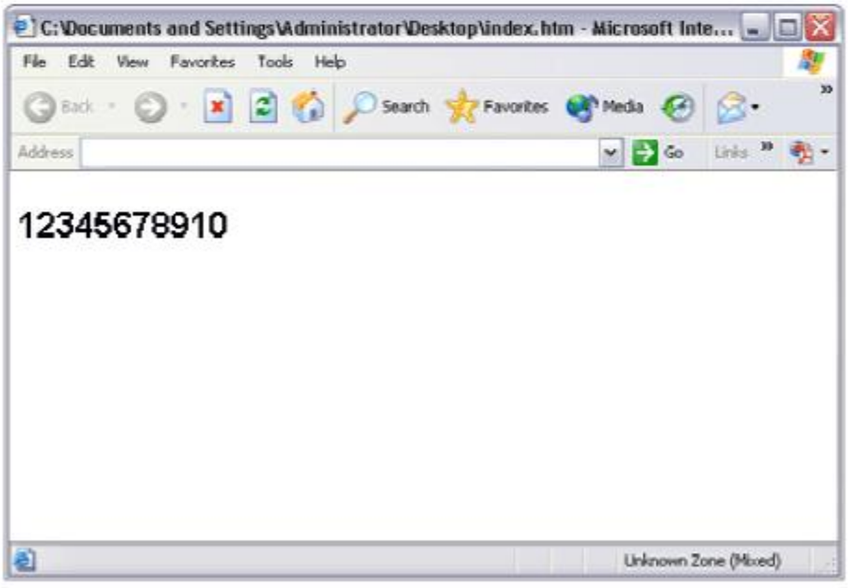
```
do {
        نفذ الامر
      مقدار الزيادة
   }
while (  ( عد الى داخل الحلقة مرة أخرى ونفذ مادام الشرط يتحقق
```

قبل الخوض في مثال يجب أن تحتوي جملة (do while) على مايلي :

١- متغير نضعه بالشرط لكي نتحقق من صحة الشرط .
٢- يجب وضع قيمة إبتدائية لهذا المتغير قبل جملة الـ (do while)
٣- يجب أن نذكر هذا المتغير ومقدار زيادته بداخل حلقة الشرط سواءا قبل تنفيذ الجملة التى بداخل حلقة التكرار أو بعدها .

مثال :

```
<html >
   <head><title>الجافا سكربت</title>
    <script type = "text/javascript">
        var  i = 1 ;
        do {
                document.write( i ) ;
                i ++ ;
            }
        while ( i <= 10  )
    </script>
   </head>
  <body></body>

</html>
```



لنحلل هذا المثال سويا :
وضعنا قيمة إبتدائية للـ (i) وهي (1) ثم دخلنا الى داخل (do while) ثم توجهنا الى الجملة التي تليها هي جملة الطباعة " أي طباعة قيمة (i) " إذا سوف يطبع (١) ثم وجدنا الامر (i++) اي زيادة قيمة (i) أي أصبحت قيمة (i) تساوي (2). ثم انتقلنا الى جملة الشرط التي حدد هل سوف نعود الى (do) أم لا . فوجدنا الشرط يسألنا هل قيمة (i) أي (2) أقل أو تساوي (10) فكان الجواب نعم إذا عدنا للـ (do) مرة أخرى ثم نفذنا مابداخلها فزدنا قيمة (i) بواحد فأصبحت (3) ثم انتقلنا الى الشرط وقد نحقق وكانت الاجابة بنعم أي أن الـ (3) أقل من (10) وهكذا استمرينا الى أن نصل لقيمة الـ (i) يساوي (11) وبالتالي عدم العودة الى (do) والخروج من حلقة التكرار .

قد تتساءل مالفرق بين الـ (while) والـ (do while) ؟
نحدد ذلك الفرق . . .

كما في (while) نتحقق من الشرط قبل الدخول الى الحلقة أي أننا لاننفذ أي شئ بداخلها مادام الشرط لم يتحقق وهذا أمر طبيعي لاننا لم ندخل الى الحلقة أصلا فكيف نعرف مابداخلها وننفذه .
أما بالـ (do while) كنا ندخل الى الحلقة وننفذ أمر ثم نفحص الشرط ولكن بعد أن نكون قد نفذنا هذا الامر ويجب التنبيه هنا " في حالة عدم تحقق الشرط لن نعود مرة أخرى الى (do) .
إذا الفرق هو أن بالـ (do while) ينفذ على الاقل أمر واحد في داخل حلقة التكرار حتى لو كان الشرط غير متحقق على العكس الـ (while) الذي لاينفذ أي امر مادام الشرط غير متحقق .

مثال توضيحي (لتوضح الفكرة بشكل عملي) :

```
var  i =5 ;
while ( i < 3 )
  {
     document.write ( i ) ;
      i ++ ;
  }
```

```
var  i =5 ;
do
  {
     document.write ( i ) ;
     i++ ;
  }  while ( i < 3 )
```

في مثال الـ (while) البرنامج لن يطبع شيئا وذلك لانه سألته جملة الشرط قبل الدخول هل الـ (5) أقل من (3) فكان الجواب لا . وهنا لن يدخل الى الحلقة وبالتالي عدم تنفيذ مابداخلها أي عدم طباعة قيمة (i)

في مثال الـ (do while) سوف يطبع البرنامج فقط مرة واحدة قيمة (i) ثم يتوقف فبعد طباعة قيمة (i) سوف يسأل عن الشرط وهو هل (6) أصغر من (3) فسوف تكون الاجابة لا وبالتالي عدم العودة الى جملة (do) أي عدم الدخول الى حلقة التكرار والاستمرار بالطباعة أي الخروج .

وفي هذا المثال نكون قد وضعنا الفرق بينهما ...

- **اذا نتيجة كود ال (while) أنه لن يطبع شيء ....**
- **ونتيجة ال do while هو طباعة فقط رقم (5) ....**

قد تتساءل عزيزي المتدرب مالفائدة من أخذنا الـ (while) والـ (do while) مادام أنهما يعملوا عمل الـ (for) ؟

عزيزي المتدرب نحن نذكر لك كل صيغ التكرار في لغة الجافا سكربت " والمستخدمة في جميع لغات البرمجه " ولك حرية الاختيار بإستعمال أي واحدة منها فقد تجد أن (for _ loop) هي أفضل لك ولكن عند غيرك تجد الـ (while) أو (do while) هي أفضل لـه وقد يتحتم عليك إستخدام أي واحدة منها مثل (do while) في كتابت برنامجك فهذا راجع لك وما مطلبك من برنامجك فكما قلنا لكلا منا أسلوبه وطريقته الخاصة بكتابة البرنامج .

- الامر (break) و (continue) :

نبدأ بالامر (break) :
قد نستخدمها في الـ (while) أو الـ (do – while) وقد نستخدمها بالـ (switch) .
أما لماذا نستخدمها ؟

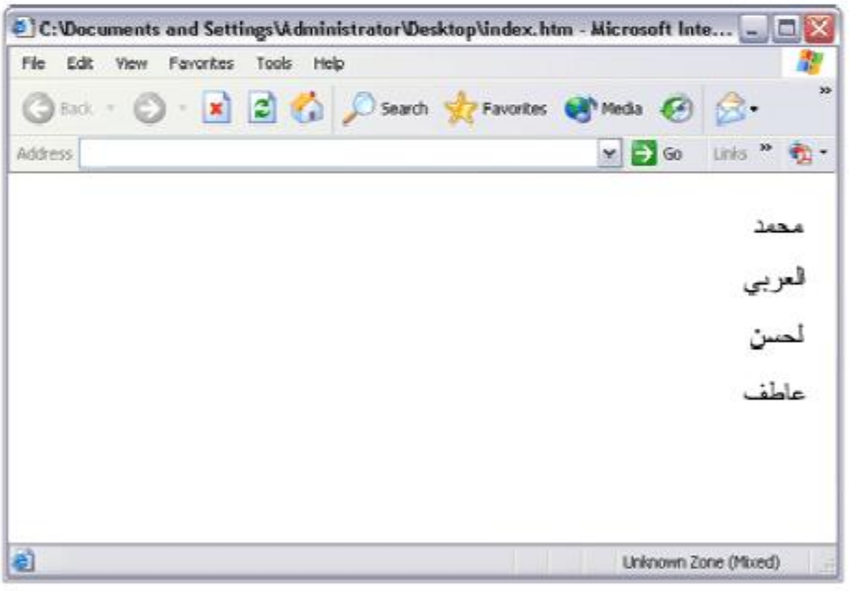
لنفرض أن لديك بيانات مخزنة في قواعد البيانات ولكن تريد فقط جزء من هذه البيانات أن تعرض وليس جميعا فهنا أستخدم (break) لكي أخرج مباشرة من داخل حلقة التكرار حتى لو لم أنتهي من الحلقة .

صيغتها :

```
break ;
```

لنأخذ مثالا :

```
<html dir="rtl">
    <head><title>الجافا سكربت</title>
        <script type = "text/javascript">
            var x = new Array("محمد","العربي","لحسن","عاطف","محمود","ثامر","خالد") ;
            for(var i=0 ; i <x.length ; i++)
                {
                    if(i == 4)
                    break ;
                    document.writeln(x[i] + "<br />") ;
                }
        </script>
    </head>
  <body></body>

</html>
```



لنحلل المثال سويا ..

عرفنا المتغير (x) والذي هو عبارة عن مصفوفة تحتوي بداخلها على (7) أسماء . ثم انشأنا حلقة التكرار (for) وبدأناها من (0) الى (i < x.lengh) أي أقل من العدد (7) .
وعزيزي المستخدم الامر الطبيعي والبديهي أن يطبع جميع الأسماء التي بالمصفوفة لكن هنا الذي سوف يحدث أنه سوف يطبع فقط أربعة أسماء . ويخرج خارج حلقة التكرار لماذا ...

لأننا وضعنا شرط داخل حلقة التكرار وهو (if i == 4) وبعده وضعنا (break) أي أننا حددنا إذا كانت قيمة (i) مساوية للعدد (4) يتوقف من حلقة التكرار وأيضا لاينفذ أمر (document.writeln) أي يخرج عند وصوله نقطة (break) ويخرج حتى لو لم تنتهي حلقة التكرار .
إذا هنا فقط سوف يطبع الأسماء الأربعة الاوائل والتي هم (**محمد _ العربي _ لحسن _ عاطف**) ولا تنسى اننا بدأنا من الصفر.

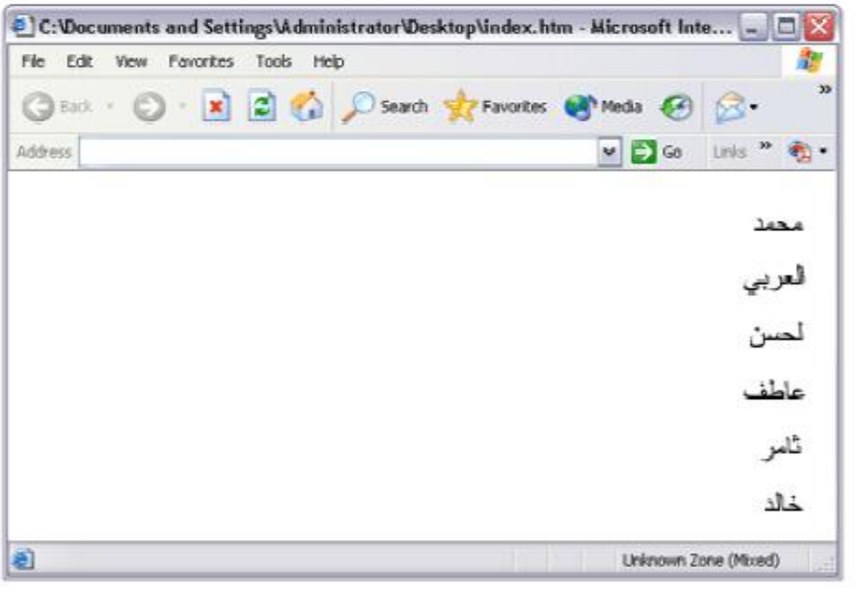
الامر (continue) :
هذا الامر عمله هو القفز عن قيمة معينة نحن نحددها أي يعمل (skip) ... لنفرض أن لدينا في قاعدة البيانات مثلا أو في مصفوفة سبعة أسماء ولكن الأسم رقم (4) لاتريد طباعته فقط هو... أما البقية نريد طباعتها إذا هنا الذي سوف يحل لنا هذا الامر هو (continue) .

صيغتها :

```
continue ;
```

لنأخذ مثالا :

```
<html dir="rtl">
    <head><title>الجافا سكريبت</title>
        <script type = "text/javascript">
              var x = new Array("محمد"," العربي"," لحسن","عاطف","محمود"," ثامر","خالد") ;
            for(var i=0 ; i <x.length ; i++)
                {
                    if(i == 4)
                    continue;
                   document.writeln(x[i] + "<br />") ;
                }
        </script>
    </head>
  <body></body>

</html>
```



لنحلل المثال سويا :

عرفنا المتغير (x) والذي هو عبارة عن مصفوفة تحتوي بداخلها على (7) أسماء . ثم انشأنا حلقة التكرار (for) وبدأناها من العدد (0) الى (i < x.lengh) أي أقل من العدد (7) .
وعزيزي المستخدم الامر الطبيعي والبديهي أن يطبع جميع الأسماء التي بالمصفوفة لكن هنا الذي سوف يحدث أنه سوف يطبع جميع الأسماء ماعدا أسم (محمود) . **لماذا ؟**
نحن هنا كتبنا شرط والذي وهو (if i == 4) وبعده كتبنا (continue) أي كأننا نقول هنا عندما تصل قيمة (i) للعدد (4) نفذ (continue) أي الموقع الرابع من المصفوفة ...تجاهله ولا تطبعه وانتقل الى الذي يليه اي ارجع وتجاوز عن امر الطباعة... فهنا سوف يطبع الأسماء الاربعة الاولى والتي هي (محمد _ العربي _ لحسن _ عاطف) أي أننا وصلنا الى الموقع الرابع من المصفوفة ثم تصبح قيمة (i) هي (4) الأسم (محمود) فينفذ الامر (continue) ويعمل (skip) أي تجاهل وقفز عن هذا الأسم ثم يستمر في طباعة بقية الأسماء .

ملاحظة :
قد يتطلب منك برنامج معين حسب وظيفته الى وضع for او while بداخل for او while اي nested loop كما في ال if تذكر

- **الدوال (function) :**

هنا سوف نتحدث عن كيفية إنشاء دالة أو أكثر خاصة بك بداخل برنامجك ولكن قد تتساءل ماهي الدوال سوف نجيبك بالتالي . .

عزيزي المتدرب الدالة هي من الاوامر والمتغيرات تكون بشكل مستقل بحيث نضعها داخل البرنامج ولا ينفذها البرنامج الا عندما يتم أستدعائها وهذا مايصطلح عليه بـ (call function) أما كيفية أستدعائها فيتم عن طريق كتابة أسم الدالة وأرسال المتغيرات لها " إن وجدت " .

بعد ماذكرناه قد تتساءل عزيزي المتدرب لماذا أستخدم هذه الدوال وماقيمتها بالنسبة لي أي ماهي أهميتها نجيبك بما يلي . . .

عزيزي المتدرب تستطيع أن لاتستخدم الدوال وتكتب برنامجك كالمعتاد ومن غير دوال ولكن تعرف ماذا سيحدث . . .

سوف تحدث سلبيات عدة منها :
أضاعة كثيرا من وقتك أضافة الى أن البرنامج سوف يحتوي على كود طويل للغاية فما بالك سوف تكون سرعة تنفيذ هذا الكود كذلك سوف يكون برنامجك صعب التطوير أعرف أنك قد أختلط عليك الامر لنطرح مثالا لتوضيح ما بهم لديك . . .

لنفرض أنك تريد أن تطلب من المستخدم أن يدخل رقمين والبرنامج الذي صممته يخرج ناتج جمعهما وتريد أن يكرر هذه العملية عشرات المرات فهنا تستطيع أستخدام الدوال فما عليك سوى أرسال الرقمين الذين يدخلهما في كل مرة الى الدالة. ولشدك للموضوع أكثر لا أريد أن أتكلم فقط على مثال الأرقام لأعطيك مثالا نستخدمه في المواقع ألست عندما تكون مشترك في موقع ما تدخل أسم المستخدم وكلمة السر للدخول الى الموقع وكذلك الحال لكثير من المشتركين في الموقع . إذا العملية تتكرر (أي فحص أسم المستخدم وكلمة المرور) عشرات المرات إذا لماذا لاأعمل دالة خاصة يدخل فيها المستخدم كلمة المرور وأسم المستخدم فتفحصها ونتأكد أنه مسجل أم لا.
عزيزي المستخدم قد لا يتوضح الامر لديك الى الان ولكن لاعليك فالعيب ليس منك فسوف تتوضح لك الامور بعد قليل إنشاء الله . . .

الصيغة العامة التى تكتب بهاالدوال (function) :
أي المتغيرات التى سوف تحدث عليها العمليات داخل الدالة

```
function   أسم الدالة وتختار أي أسم تريد غير محجوز   ( المتغيرات التى سوف ترسل للدالة )
  {   بداية الدالة
     العمليات التى تتم بداخل الدالة
     return   النتيجة المعادة ;
  }   نهاية الدالة
```

- **ملاحظة :**

يمكن أن تأتي صيغة كتابة الدالة بشكل آخر وهو كالتالي :

```
Function أسم الدالة وتختار أي أسم تريد ( المتغيرات التى سوف ترسل للدالة )
  {
     return   العمليات التى سوف تتم ومن ثم إرجاع النتيجة ;
  }
```

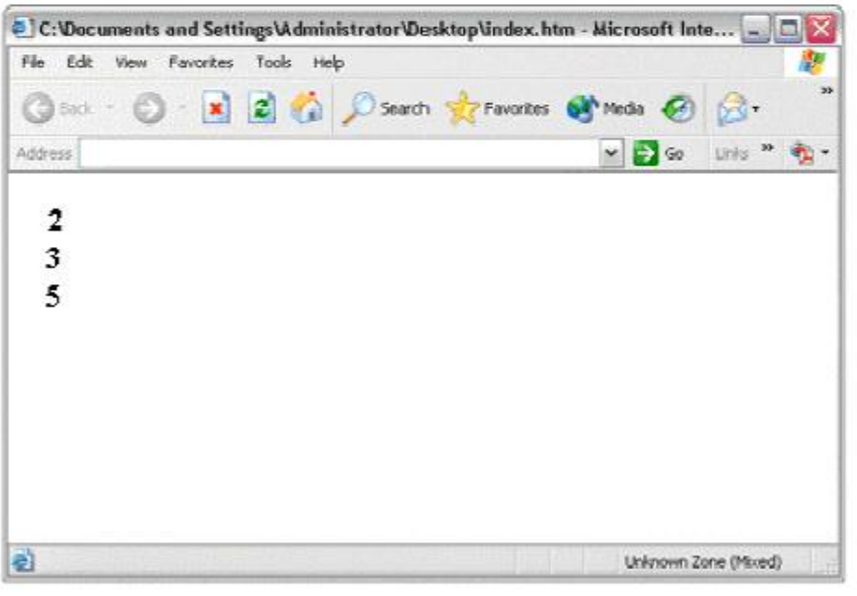
كما ترى لم نغير أي شئ فقط الذي غيرناه بدلا من أن نضع العمليات فوق الـ (return) ونخرج بناتج ونضعه بالـ (return) هنا وضعنا العملية كاملة بداخل (return) وبالتالي خروج النتيجة وإرجاعها مباشرة . (ولكن هذه العملية ليست دائما يمكن إستخدامها) لذلك يفضل إستخدام الصيغة الاولى وبشكل اكثر دقة حسب الحاجة .

عزيزي المتدرب قد يكون الامر لم يتوضح لديك تماما الى الان لاعليك هذا أمرا طبيعي فأنت هنا تتعلم !

إذا لنأخذ أمثلة تطبيقية فهي التى توضح لنا ما نريده.

المثال الأول :

```
<html >
    <head><title>الجافا سكريبت</title>
      <script type = "text/javascript">
          var  z ;
          var  x=2 ;
          var  y=3 ;
          document.write ( x ) ;
          document.write ( " <br> " + y ) ;
          z = summation ( x , y ) ;
          document.write ( " <br> " + z ) ;

       function   summation ( x , y )
          {
             var  sum = 0 ;
             sum= x + y ;
             return  sum ;
          }
      </script>
    </head>
  <body></body>

</html>
```



لنحلل المثال سويا . . .
قمنا هنا بتعريف ثلاثة متغيرات المتغير الاول وهو (x) ووضعنا بداخله القيمة الرقمية الصحيحة (2) والمتغير (y) ووضعنا بداخله القيمة الرقمية الصحيحة (3) والمتغير (z) الذي سوف نضع به القيمة التى ترجعها لنا الدالة (summation) أي ناتج الجمع (x+y) .

ثم قمنا بطباعة قيمة (x) من خلال الأمر (document.write (x)) وبالتالي طباعة رقم (2) فهذا الرقم الذي وضعناه بالـ (x) .
ثم كتبنا جملة الطباعة الخاصة بطباعة قيمة (y) وذلك من خلال الامر :

```
document.write (" < br > " + y)
```

ثم قمنا بوضع مايلي :

```
z = summation ( x , y ) ;
```

ماذا نعني بها . . .
نعني بها أن قيمة المتغير (z) تساوي القيمة التي سوف ترسلها الدالة (function) التي أسمها (summation) التي أرسلنا لها قيمتين وهما قيمة (x) وقيمة (y) لكي ترجع اي ترسل لنا الناتج لنضعه في (z). فنحن هنا نحصل على قيمة (z) من القيمة التي سترجعها الدالة.
إذا هنا نحن أستدعينا الدالة (summation) لكي ترجع لنا ناتج العملية ومن ثم نضعه في (z) إذا لنحصل على هذا الناتج.. نذهب الى الدالة المسماة (summation) لنذهب لها وسوف نجد بداخلها مايلي :

عرفنا متغير أسميناه (sum) وصفرناه " أي جعلنا قيمته صفرا " ووضعنا الصفر فقط لكي نحسب الرقمين بدقة وعدم وجود أي قيمة سابقة للـ (sum) فكما ترى عندما تفتح الالة الحاسبة ألا تضع لك صفرا ثم تضغط الرقم (2) ثم تضغط (+) ثم تضغط رقم (3) فيخرج لك الناتج (5) أي الصفر تحول الى (5) فهنا كأننا نقول أن قيمة الـ (sum) قبل العملية كانت صفرا وبعد العملية أصبحت (5). لنرجع لموضوعنا اذا ...
الان جعلنا (sum) تساوي قيمة الـ (x) المرسلة وجمعنا لها قيمة الـ (y) المرسلة أيضا وكما تعرف " قيمة (y) وقيمة (x) سوف تعرفهما الدالة مباشرة من خلال أرسالنا في البداية عند أستدعائها قيمة الـ (x) وقيمة (y) أي (2) و (3) "
.
وهنا سوف تتم عملية الجمع أي (2) + (3) ويكون الناتج (5) الذي سوف نضعه في المتغير (sum) إذا الان أصبحت قيمة (sum) هي (5) وهي ناتج العملية .

ثم كتبنا (return) ووضعنا بها متغير (sum) أتعرف عزيزي المتدرب ماذا تعني (return)؟ تعني أرجع . . إذا نحن سوف نرجع قيمة (sum) والتى هي (5) الى أين ؟؟؟
الى المكان (الام) الذي أستدعينا فيه الدالة وهو :

```
z = summation ( x , y ) ;
```

إذا هنا سوف يعيد للدالة (summation) قيمة (5) ثم وضعنا هذه القيمة بالمتغير (z) وذلك من خلال :

```
z = summation ( x , y ) ;
```

اي هنا سوف نرسل (sum) ليس كمتغير وانما كقيمة اذا سوف يرسل رقم (5) وهذا رقم خمسة ماذا يمثل..؟
انه يمثل ناتج الداله ...
(summation (x , y) ومنها سوف يتم اسناد القيمة (5) الى (z.) .

و بعد تنفيذ عملية استدعاء الداله يحق لنا الانتقال الى الخطوه التي تليها في الكود فلا يجوز الذهاب الى الخطوه التي تليها إلا بعد استدعائها وتنفيذها لترجع لنا الناتج ما نفذته وبعد ذلك ننتقل للذي يليها وهي بطباعة قيمة (z) من خلال الامر :

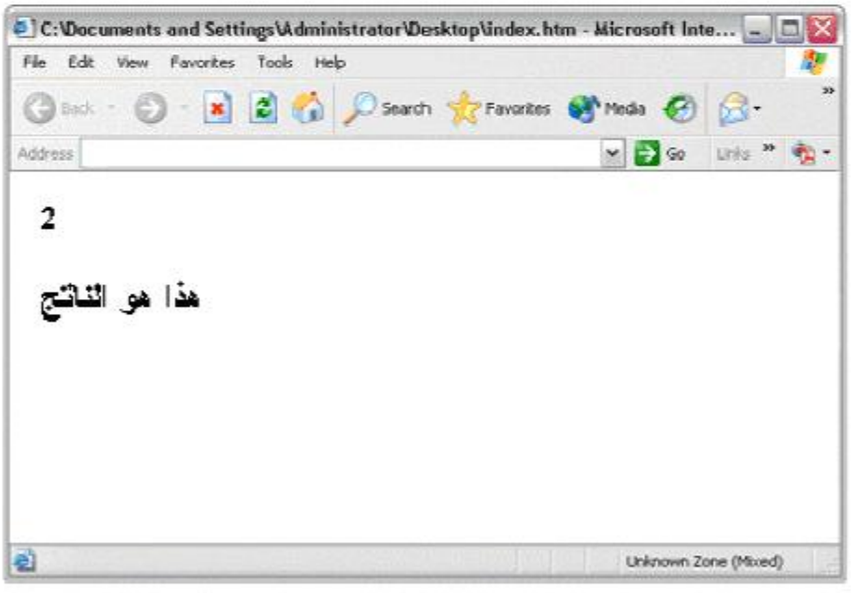
```
document.write ( " < br > " + z ) ;
```

فطبع لدينا القيمة (5) أي (2) + (3) = (5) و لاحظ عندما وصلنا الى (return) كنا قد أنتهينا تماما من العمل بداخل الدالة فلقد خرجنا منها وعدنا الى البرنامج " الذي بالاعلى " ولكن معنا قيمة (sum) .

ألم تتساءل عزيزي المتدرب هل يمكن للدالة (function) أن تأتي من غير أن نذكر بها (return) أي أنها لا ترجع أي شئ ؟؟؟.
نعم قد لاتحتوي الدالة (function) على (return) وبما أنها لا تحتوي على (return) إذا هي لن ترجع قيمة ففي هذه الحالة سوف نرى المثال التالي الذي يوضح لنا ...

مثال :

```
<html>
   <head><title >الجافا سكربت</title>
      <script type = "text/javascript">
         var  x = 4 , y = 2  ;
         subtract ( x , y ) ;
         document.write (" <h3>  هذا هو الناتج </h3> ") ;

        function   subtract ( x , y )
          {
             var  sub = 0 ;
             sub = x - y ;
             document.writeln ( sub ) ;
          }
      </script>
    </head>
  <body></body>

</html>
```



لنحلل المثال سويا :
هنا قمت بتعريف المتغيرات (x) و (y) ووضعت قيمة في كلا منهما ففي الـ (x) وضعت (4) وفي الـ (y) وضعت (2) ثم وضعت اسم الدالة (function) التى أريد أن أستدعيها " أي التى سوف تقوم بعملية الطرح بين الرقمين " وهنا أرسلت للدالة (x) و(y) المحتويان على رقمين أي كأنني أرسلت (4) التى هي (x) و (y) التى هي (2) إذا هنا بما أننا أستدعينا الدالة (function) إذا وجب علي أن أذهب الى هذه الدالة (function) وأقوم بالعمليات التى هي بداخلها ...
لنقوم بالذهاب لها ...

كتبنا أسم الدالة طبعا ووضعنا بين قوسين القيمتين المرسلة لها ثم دخلنا بداخل الداله ثم عرفنا متغير (sub) الذي سوف نضع به ناتج عملية الطرح وهي (x) ناقصا (y) ووضعنا الناتج في (sub) من خلال هذه المعادلة :

```
sub = x – y
```

ثم كتبنا أمر طباعة المتغير (sub) الذي هو ناتج العملية من خلال الامر :

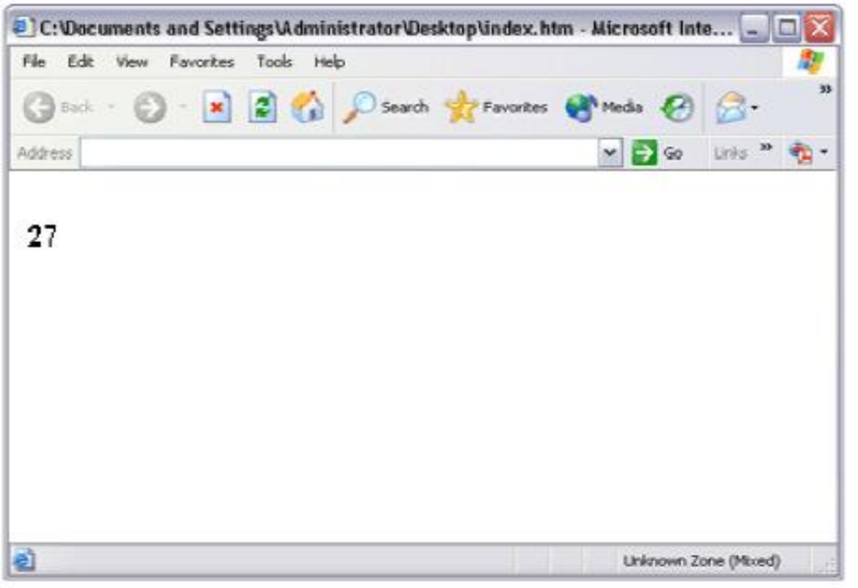
```
document.write (sub) ;
```

وبالتالي سوف يطبع رقم (2) ثم اغلقنا الداله . . .
ثم عدنا مرة أخرى الى مابعد أستدعاء الدالة " أي الى الاعلى " لكننا هنا عدنا ولايوجد بحوزتنا أي قيمة هو مجرد رجوع لأكمال البرنامج فنحن نرى أستدعاء الدالة فتذهب لها لتنفيذها ثم نعود لنكمل بقية البرنامج أي الاوامر التى تلي أستدعاء الدالة وهنا طبعا وضعنا أمر طباعة وهو طباعة الجملة " هذا هو الناتج" من خلال الامر :

```
document.write ( " <h3>  هذا هو الناتج </h3> " ) ;
```

لاحظ أننا طبعنا الناتج من خلال أمر الطباعة الموجود في الدالة ثم عدنا الى البرنامج فطبعنا الجملة النصية .
ولكن هل من الممكن أن يحتوي برنامجي المكتوب بالجافا سكربت على دوال فقط ... كأن دالة نستدعي دالة .. نعم ممكن ذلك لنأخذ مثال ونرى :

```
<html>
   <head><title >الجافا سكربت</title>
     <script type = "text/javascript">
       function   output ( )
         {
            var  x = 3 , t ;
             t = cube ( x ) ;
            document.writeln ( t ) ;
         }

       function cube ( y )
         {
            return  y*y*y ;
          }
     </script>
   </head>
  <body onload = " output ( ) " ></body>

</html>
```



وسوف أجعل تحليل هذا المثال لك ولكن سوف أوضح فقط بعض الامور التى تساعدك في تحليله... التى أعتقد أنها جعلت في فكرك بعض التساؤلات عندما رأيت كود البرنامج .

١- قد تتساءل لماذا لم نضع متغير بين القوسين عزيزي المتدرب هل هذه الدالة سوف تستقبل متغير " أو حتى قيمة من الخارج " الجواب لا إذا أدعها فارغة ولا أضع بها شيء فهي دالة لاتستقبل .

```
function  output ( )
```

٢- قد تتساءل لماذا كتبنا هنا (y) مع أننا عندما أستدعينا هذه الدالة من الدالة (output) كتبنا بهذا الشكل :

```
T = cube ( x ) ;
```

```
function  cube ( y )
```

أي وضعنا (x) وليس (y) عزيزي المتدرب فليس مهما أن تسمي المتغير الذي أرسلته بنفس أسم متغر الدالة المرسل لها فهذا ليس مهما طبعا ليس مهم تسمية المتغير في الدالة المرسل لها أي أقصد :
(function cube (y)) فهنا المهم أننا أرسلنا له (x) والتى قيمتها (3) .
لاتتزعج تستطيع أن تسمى بدلا من (y) (x) أي تصبح (function cube (x)) فقط ذكرتها من باب العلم بالشئ .

٣- هنا جعلناه يضرب رقم (3) بنفسه ثلاثة مرات وإرجاع النتيجة مباشرة . وهذا بدلا من تعريف متغير ونضع النتيجة بداخله ومن ثم نرسل هذا المتغير بـ (return) ونرجع القيمة قمنا بوضع العملية الحسابية مباشرة في ال return ... " لاحظ أننا ذكرنا سابقا اننا لم نضع في ال ( ) function output متغير بداخله وهذا لانه لن يستقبل متغير ولنقل بشكل اكثر دقه parameter اي اننا لن نمرر عليه قيم.. واذكرها هنا لك لكي لا تقول في نفسك ها نحن عدنا له بقيمة عملية الضرب ثلاث مرات اذا استقبل ! عزيز المتدرب هنا العملية تمت من خلال return وليس ارسال قيم لتمريرها ليتم عليها عملة معينه اي هنا تم استدعاء دالة لتقوم بعملية وترجع لنا قيمه .. تذكر عندما كنا نستدعي الداله من داخل برنامج من دون ان يكون بادخل دالة اخرى .

لاتتزعج تستطيع أن تكتبه كما تعلمنا سابقا أي :

```
return  y*y*y ;
```

```
var  z ;
z = y*y*y ;
return  z ;
```

فقط أحببت أن أريك طريقة ثانية من باب العلم بالشئ فقط لا غير وقد تلجأ له عند الحاجه .

٤- معلومة مهمة جدا ...
عندما تكتب برنامجك بالجافا سكربت ويكون يحتوي فقط على دوال كما في مثالنا هذا فيجب عليك أن تذكر في الـ (body) هذه العبارة :

```
< body onload = " output ( ) " >
```

- قد تتساءل لماذا وضعنا (onload) داخل الـ (body) ؟

الجواب لان برنامجنا عبارة عن دوال والدوال لاتعمل الا عند إستدعائها فأعتبر الـ (onload) هو إستدعائها " أي تشغيلها " أما لماذا ذكرنا أسم الدالة الاولى (output ( )) فهذا بسبب أن هذه الدالة هي الرئيسية فهي التى تستدعي الدالة الثانية . ولكن ليس دائما نضع (onload) مثلا في حالة ال (form) وسوف نتحدث عن هذا الموضوع لاحقا .

- **Math Object :**

وهي كائنات او طرق اذا جاز التعبير تقوم بالعمليات الحسابية .. وطريقة استدعائها او تطبيقها تكون على الصيغة التالية :

اول شيء ذكر اسم ال (Object) ثم نقطة (dot) ثم بين قوسين يوضع ما هو مراد حسابه (0) .. واسم ال (Object) هنا هو (Math) من عبارة " mathematical calculation " اي الطرق الحسابية " اذا جاز التعبير " اذا الصيغة العامه هنا هي :

```
Math. اسم العملية الحسابية ( الرقم او المتغير )
```

مثال :

```
Math.sqrt(9)
```

هنا كاننا نقول نريد الجذر التربيعي للعدد (9) اذا الجواب سوف يكون (3) وهذا ما سوف يخرجه لنا الصيغة السابقة

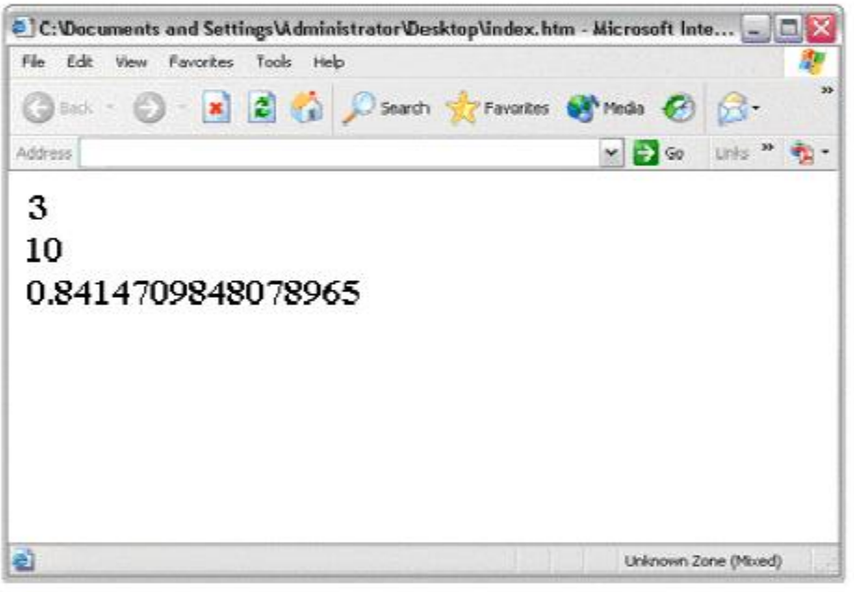
ملاحظة :

يجب هنا الالتزام بالحروف من كبيره وصغيره فركز حرف ال (M) ياتي بال (capital letter) اما البقية تأتي في (small letter) .... (Math)

الطريقة	الوصف	مثال
abs(x)	القيمة المطلقة لـ (x)	abs(7.2) = 7.2 abs(0.0) = 0.0 abs(-5.6) = 5.6
ceil(x)	التقريب لأكبر عدد حقيقي	ceil(9.2) = 10.0 ceil(-9.8) = -9.0
cos(x)	جيب التمام (جتا) لـ (x)	cos(0.0) = 1.0
exp (x)	طريقة الأس (e^x)	exp(1.0) = 2.71828 exp(2.0) = 7.38906
floor (x)	التقريب لأصغر عدد حقيقي	floor (9.2) = 9.0 floor (-9.8) = -10.0
log (x)	لوغارثم (x)	log (2.718282) = 1.0 log (7.389056) = 2.0
max (x,y)	أكبر قيمة من (x) و (y)	max (2.3,9.7) = 9.7 max (-2.3,-9.7) = -2.3
min (x,y)	أصغر قيمة من (x) و (y)	min (2.3,9.7) = 2.3 min (-2.3,-9.7) = -9.7
pow(x,y)	(x) مرفوع للأس (y) تعني (x^y)	pow(2.0,7.0) = 128.0 pow(9.0,0.5) = 3.0
round(x)	تقريب (x) لأقرب عدد حقيقي	round(9.7) = 10 round(9.25) = 9
sin(x)	جيب (جا) لـ (x)	sin(0.0) = 0.0
sqrt(x)	الجذر التربيعي لـ (x)	sqrt(900.0) = 30.0 sqrt(9.0) = 3.0
tan(x)	ظتا (x)	tan(0.0) = 0.0

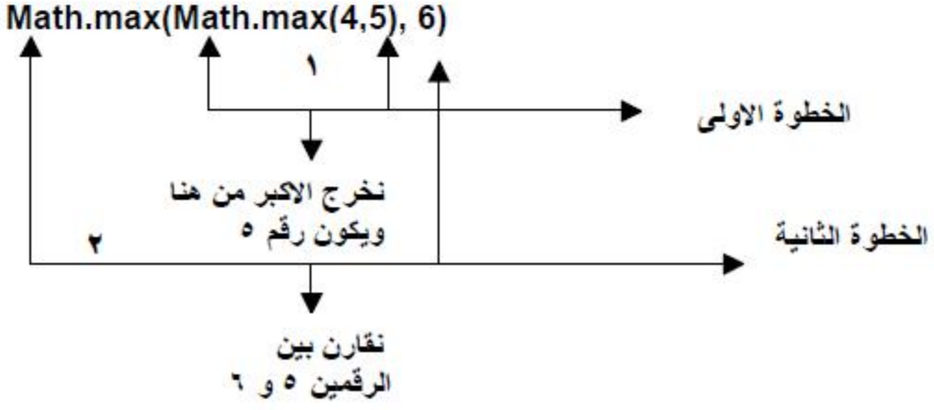
مثال :

```
<html>
   <head><title>الجافا سكربت</title>
      <script type = "text/javascript">

         document.write(Math.sqrt(9));
         document.write("<br />");
         document.write(Math.max(10,2));
         document.write("<br />");
         document.write(Math.sin(1.0));

      </script>
    </head>
  <body></body>

</html>
```



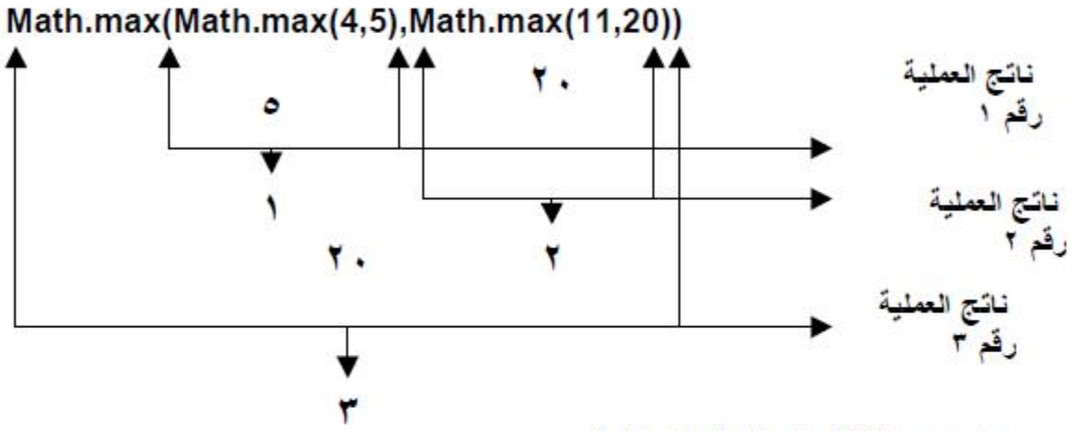
بشكل عام هكذا تكتب ال (Math Object) في البرنامج الجافا سكربت ..
فجملة الطباعة الاولى سوف تطبع الجذر التربيعي للعدد (9) ثم وضعت ال (
) بمفرده في جملة طباعة لكي لا يختلط عليك الامر وذلك لكي ننزل سطرا جديدا ثم تاتي جملة الطباعة الثالثة تطبع لنا الرقم الاكبر بين الرقمين الذين وضعناهما ثم وضعنا جملة الطباعة الثالثة والتي هي نزول سطر جديد " بداية سطر جديد " ثم وضعنا جملة الطباعة الرابعة والتي سوف تطبع لنا جيب الرقم واحد ...

ملاحظة :
لو اردت ان تعمل مقارنه بين ثلاثة ارقام والخروج بالاكبر وقتها تكتب هكذا :



هنا كاننا نقول اخرج الرقم الاكبر في ال (Math Object) الذي في الداخل اي بين الرقمين (4 و 5) ومن ثم طبق ال (Math Object) الذي في الخارج وهو سوف يكون بين الرقمين (5) الذي خرج من ال (Math Object) التي في الداخل وبين الرقم (6) الذي في ال (Math Object) في الخارج اي الرئيسية ..

- اذا اردنا ان نخرج الرقم الاكبر بين اربعة ارقام تكتب الصيغة هكذا :
- اي هنا ناخذ كل عملية على حدى وكانها لوحدها ...



- **تمارين إضافية على الدوال الرياضية :**

الثابت	الوصف	القيمة
Math.E	قيمة ثابتة	تقريبا تساوي 2.718
Math.LN2	اللوغاريثم الطبيعي لـ (2)	تقريبا تساوي 0.693
Math.LN10	اللوغاريثم لطبيعي لـ (10)	تقريبا تساوي 2.302
Math.LOG2E		تقريبا تساوي 1.442
Math.LOG10E		تقريبا تساوي 0.434
Math.PI		تقريبا تساوي 3.141592653589793
Math.SQRT1_2		تقريبا تساوي 0.707
Math.SQRT2	الجذر التربيعي لـ (2.0)	تقريبا تساوي 1.414

- **Methods of the String Object :**

الطريقة	الوصف
charAt(index)	يرجع الحرف المحدد بالـ (index)
charCodeAt(index)	يرجع شيفرة الحرف المحدد بالـ (index)
concat(string)	يدمج النص المحدد في الـ (string) المحدد قبل الامر (concat) مع النص المحدد في الـ (string) الذي بين القوسين اي الذي بعد (concat)
fromCharCode(value1,value2 ,....)	يحول الارقام المحددة بين الاقواس الى قيمتها الحرفية
indexOf(substring,index)	البحث عن النص المحددة في (substring) من بعد الموقع رقم (index) من النص المحدد في الـ (string)المحدد قبل الأمر. ويرجع أول موقع يحتوي على أول حرف من النص المحدد في (substring) من الـ (string)
lastIndexOf(substring, index)	البحث عن النص المحددة في (substring) من قبل الموقع رقم (index) من النص المحدد في الـ

	(string)المحدد قبل الأمر. ويرجع أول موقع يحتوي على أول حرف من النص المحدد في (substring) من الـ (string)
slice(start,end)	يرجع النص من الموقع (start) الى (end) من الـ (string) المحدد قبل الأمر
split(string)	تقسيم النص المحدد في (string) الى مجموعة كلمات نحن نحدد كيفيتها .
substr(start,length)	يرجع النص المحدد من الموقع (start) وعدد أحرفه (length) من الـ (string) المحدد قبل الأمر .
substring(start,end)	يرجع النص المحدد من الموقع (start) الى الموقع (end)
toLowerCase()	يحول الـ (string) المحدد قبل الأمر الى حروف انجليزية صغيرة .
toUpperCase()	يحول الـ (string) المحدد قبل الأمر الى حروف انجليزية كبيرة .
toString()	يرجع نفس الـ النص المحدد في الـ (string)
valueOf()	يرجع نفس الـ النص المحدد في الـ (string)
anchor(name)	يعمل عمل (<a> name)
blink()	يعمل عمل (<blink>...</blink>)
fixed()	يعمل عمل (<tt>...</tt>) أي الكتابة على شكل آلة طابعة
link(url)	يعمل عمل (<a>...)
strike()	يعمل عمل (< strick>...</strick>) أي يضع خط في وسط الكلام
sub()	يعمل عمل (_{...}) أي الكتابة تحت السطر
sup()	يعمل عمل (^{...}) أي الكتابة فوق السطر

مثال :

```
<html dir="rtl">
    <head><title>الجافا سكربت</title>
        <script type = "text/javascript">

            var  a ="samialr";
            document.write("<p> اطبع الحرف الموجود في الموقع صفر " + a.charAt(0) );

        </script>
    </head>
   <body></body>

</html>
```

▪ **معلومات تطبيقية :**

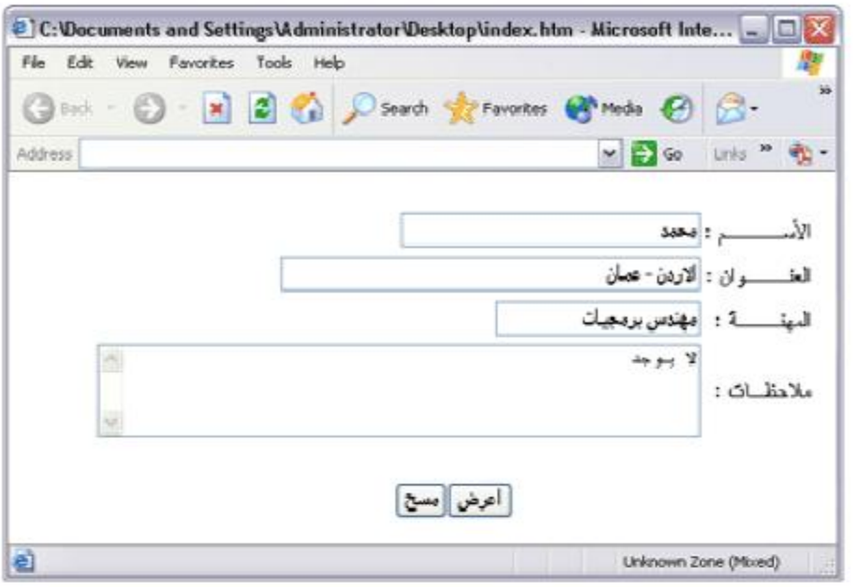
▪ نأتي هنا الى ذكر آخرموضوعين في الكتاب وهما كيفية التعامل بالجافا سكربت مع النماذج (forms) والشيء الآخر هو ظهور الكتابة ضمن شريط (status) لكي لا أطيل عليك عزيزي المستخدم فأنا أعلم أن الكتاب أثقل عليك من كثر مايحتوي على شرح ولهذا سوف أذكر مثال ونحلله مباشرة فكل شيء يفهم من خلال التطبيق ..

مثال :

وهو تعبئة نموذج من قبل المستخدم وبعد التعبئة يضغط على أيقونة موافق فيطبع بياناته في صفحة الأنترنت

```
<html dir="rtl">
    <head><title> java script </title>
      <script type="text/javascript">
        function information()
          {
            var name=info.name.value;
            var job=info.job.value;
            var add=info.address.value;
            var comm=info.command.value;
            document.writeln("<table cellspasing=\"0\"
            cellpadding=\"5\" border=\"1\" width=\"80%\"
            align=\"center\" dir=\"rtl\" bgcolor=\"#eaeaea\"
            bordercolor=\"#000000\">");
            document.writeln("<caption><h3>
            المعلومات التي سجلتها لدينا </h3></caption>");
            document.writeln("<tbody>");
            document.writeln("<tr><td width=\"30\"><b> الأســـم :
            </b></td><td>" + name + "</td></tr>");
            document.writeln("<tr><td width=\"30\"><b> المهنـــة
            :</b></td><td> " + job + "</td></tr>");
            document.writeln("<tr><td width=\"30\">
            <b>العنـوان :</b></td><td> " + add + "</td></tr>");
            document.writeln("<tr><td width=\"30\">
            <b>ملاحظات :</b></td><td> " + comm + "</td></tr>");
            document.writeln("</tbody>");
            document.writeln("</table>");
          }
      </script>
    </head>
```

```
<body >
  <form name="info" action="">
  <table cellspacing="0" cellpadding="5" border="0" width="80%" align="center">
    <tbody>
      <tr>
        <td>الأســـم : </td>
        <td><input type="text" name="name" value="" size="31"></td>
      </tr>
      <tr>
        <td>العنـوان :</td>
        <td> <input type="text" name="address" value=""  size="45"></td>
      </tr>
      <tr>
        <td>المهنـــة :</td>
        <td><input type="text" name="job" value="" size="20"></td>
      </tr>
      <tr>
        <td>ملاحظات :</td>
        <td><textarea name="command" rows="4" cols="50"></textarea></td>
      </tr>
      <tr>
        <td colspan="2" align="center">
          </br>
          <input type="button" name="show" value="اعرض"
          onclick="information()"><input type="reset" name="del" value="مسح" >
        </td>
      </tr>
    </tbody> </table></body></html>
```



عزيزي المتدرب لا تنزعج من طول الكود فالكود ليس بطوله بل بأوامره المستخدمه وتكنيكه ومن هنا لا اريد ان احلل الكود لانه لا يحتاج تحليل فلا عليك من رهبة كثرته وطوله ومن هنا سوف اشرح لك الامور التي بها اختصار الشرح بالكود ..

اولا هذا الكود ليس به امور كثيره او اوامر كثيرا بل هو بسيط الى ابعد الحدود لننظر كيف ذلك :

لقد عرفت المتغيرات بالشكل التالي :

```
var name=info.name.value ;
var job=info.job.value ;
var add=info.address.value ;
```

هنا سوف اتجاوز بشرحي قليلا عن النمط البرمجي البحت لكي تصل المعلومه اكثر اذا سمحت لي ..

هنا عرفنا المتغيرات بالشكل الذي نعرفه مثل (name) و (job) و (add)
ونذكر أخي المتدرب اننا كنا نعرف في الماضي... اي في المصفوفات متغير مثلا (a) ويكون هو عباره عن مصفوفه وكانت تكتب بشكل التالي :

```
var a = new Array( ) ;
```

ومنها كنا نعرف ان هذا المتغير هو عباره عن مصفوفه ..
وهنا نفس الشيء قلنا مثلا المتغير (name) هو عباره عن قيمه سوف يدخلها المستخدم في النموذج (form) كما كنا نقول ان هذا المتغير (a) هو عباره عن مصفوفه .. لهذا كتبنا المتغير (name) بهذا الشكل :

```
var name=info.name.value ;
```

اي هنا نقول ان المتغير (name) هو عباره عن قيمه سوف تكون مدخله من قبل المستخدم في حقل من حقول النموذج اي ال (form) والذي يحمل اسم (name) لكي تخزن بداخلها ..
اذا هنا نأتي للتساؤلك عزيزي المتدرب ... ماذا يعني الكود التالي :

```
info.name.value ;
```

لنفسرها سويا :

- (info) ونعني بها اسم النموذج الذي سوف يدخل به المستخدم القيمه والذي يكتب في ال (form)

```
< form name="info" action=" " >
```

فكما تعرف في لغة (html) النصيه يجب ان يحمل النموذج اسم اذا هنا يجب ان نضع نفس الاسم الذي حددناه ..للمتغير (name) وهذا لكي نخبر ان هذا المتغير متعلق بذلك النموذج ..

- (name) ونعني بها الاسم الذي سوف نضعه بالحقل الذي سوف يدخل به المستخدم القيمه ومن ثم تخزن في المتغير (name) والتي تكون في النموذج بهذا الشكل :

```
< input type="text"  name="name"  value="" size="3" >
```

وهذا لكي نميز ان الحقل هذا الذي في ال (input) والذي سوف يدخله المستخدم سوف يخزن في المتغير الذي اسمه (name) اي انه خاص به .

- (value) ونعني به ان هذه القيمه سوف يدخلها المستخدم لذلك هي غير معلومه الى الان ..

وبهذا تكون الصيغة للمتغير والقيمة التي سوف تخزن بها كتالي :

```
var name=info.name.value ;
```

إذا عرفنا لماذا نضع المتغير ونساويه بهذا الشكل كما في المصفوفه ولكن باختلاف الوظيفه وطريقة الكتابه .. وما ينطبق على المتغير (name) ينطبق على المتغير (job) و (add)

- قبل ان ننتقل الى بقية الكود لنتسأل هل المسميات هذه ثابته ويجب ذكرها دائما ..؟

الجواب : لا

فقد نأتي ونسمي المتغير ونكتبه بهذا الشكل. لنقل مثلا نريد ان نسمي متغر بإسم (a) فرضا. اذا نكتبه كتالي :

```
var a=myform.enter1.value ;
```

إذا بما اننا غيرنا المسميات يجب ان نثبت بها على النموذج اي ال (form) لكي يعرف ان هذا المتغير خاص به اذا التغيرات كتالي سوف تصبح ..

تعريف المتغير :

```
var a=myform.enter1.value ;
```

عند كتابتنا لكود النموذج نذهب و نكتب اسم ال (form) بهذا الاسم وهو الذي اخترناه من بداية برنامجنا " myform " كتالي :

```
< form name="myform" action=" " >
```

والحقل الذي سوف يدخل من خلاله المستخدم قيمة المتغير التي سوف تخزن به سوف يأخذ اسم " enter1 " كتالي :

```
< input type="text" name="enter1" value="" size="31" >
```

- **اذا نستنتج التالي :**

ان المسميات نحن من نحددها ولكن اذا حددناها منذ بداية البرنامج علينا الالتزام بها الى اخر البرنامج دون تغير ..

نأتي الان الى الشيء الاخر وهو الكود التالي وماذا نعني به :

```
<input type="button" name="show" value="اعرض" onclick="information()">
```

التساؤل هنا لماذا وضعنا في الحقل الخاص بال (button) هذا الكود :

```
onclick="information()"
```

عزيز المتدرب هنا بما انه حقل خاص بال (button) اذا يجب بعد ان يضغط عليه المستخدم ان ينفذ امر معين اذا هنا نضع اسم الداله اي ال function الذي كتبنا بداخله الكود لكي ينفذ الاوامر التي بداخله لهذا وضعنا كلمة (onclick) اي مجرد ما يضغط المستخدم على ال (button) نفذ الدالة اي ال (function) المسمى (information) وهو اسم الدالة التي سميناها في اول برنامجنا .. وبمجرد الضغط عليها سوف يقوم البرنامج بعمله وهو بناء جدول وبداخله القيم التي ادخلها المستخدم ..

وهكذا اكون قد حللته لك الى ابعد الحدود وكأنني شرحته وانا ارى انني شرحته فأندمجت في شرحه ولم اشعر بذلك..

اما لماذا لم اكتب شرح بقية الكود فلا يوجد جديد به فكله اوامر (document.writeln) اي اوامر طباعه وضع بداخلها كود ال (html) اما كيفية كتابة كود ال (html) بداخل كود الجافا سكربت ذكرناه في بداية كتابنا .. بقي الكود الذي في ال (body) اي في جسم الصفحة هذا عباره عن كود (html)..

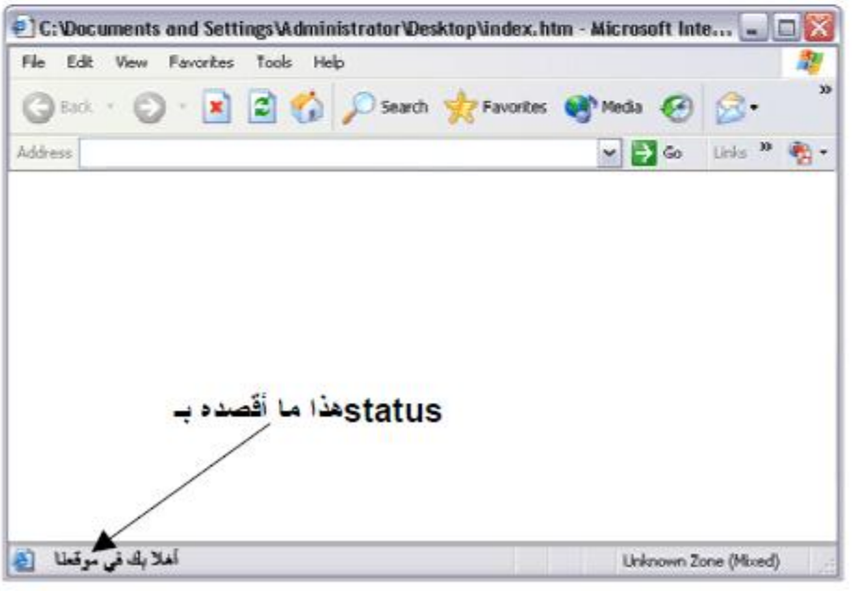
- اما لماذا هنا كتبنا كود طباعة الجدول في داخل كود الجافا سكربت وايضا كود في الجسم الصفحة ..؟

الجواب :

الاول اي كود انشاء الجدول في الجافا سكربت قصدنا به انه ينشئه في حالة تنفيذ الداله اي بعد ان يضغط المستخدم على ال (button) ويحتوي بداخله القيم التي سوف يطبعها ويظهرها للمستخدم اما الجدول الاخر الذي في ال (body) هو الجدول الخاص بالنموذج اي ال (form) الذي سوف يظهر للمستخدم عند دخوله للصفحة والذي سوف يملأ حقوله لتنتقل الى دالة الجافا سكربت وينفذها ..

ملاحظة :
قد تستخدم شروط معينه تلزم بها معبئ الفورم باستخدام احرف او رموز معينه على سبيل المثال ذكر رمز الـ (@) في تعبئة المستخدم خانة البريد الالكتروني للتأكد انه وضع بريد الكتروني وغيرها من الشروط التي أتركها لك لتقوم بها بنفسك لكي لا يكون كل شيء جاهز دائما ولكن سوف ألمح لك كيفية عملها ألا يمكنك عمل داله اي function تقوم بفحص خانة معينه مثلا تفحصها بجملة if بداخل function ويتم استدعاء ال function عند تعبئة المستخدم خانة معينه مثل خانة البريد مثلا .

- والمثال الآخر يتحدث عن طباعة جملة معينة في شريط (status) والذي يعتبر من الكائن window تمثله هذه الصورة :



فإذا أردت عزيزي المتدرب أن تذكر أي جملة معينة فتظهر في شريط (status) تستطيع ذلك من خلال الامر :

```
window.status = " أهلا بك في موقعنا " ;
```

أما إذا كنت تريد أن يظهر في شريط (status) متغير كأن يظهر مثلا أسم زائر موقعك فيكتب بهذا الشكل :

```
window.status = name ;
```

ولو أردت أن تذكر في شريط (status) جملة ومن ثم متغير فتكون الصيغ كما يلي :

١-

```
window.status = name  " أهلا بك يا " ;
```

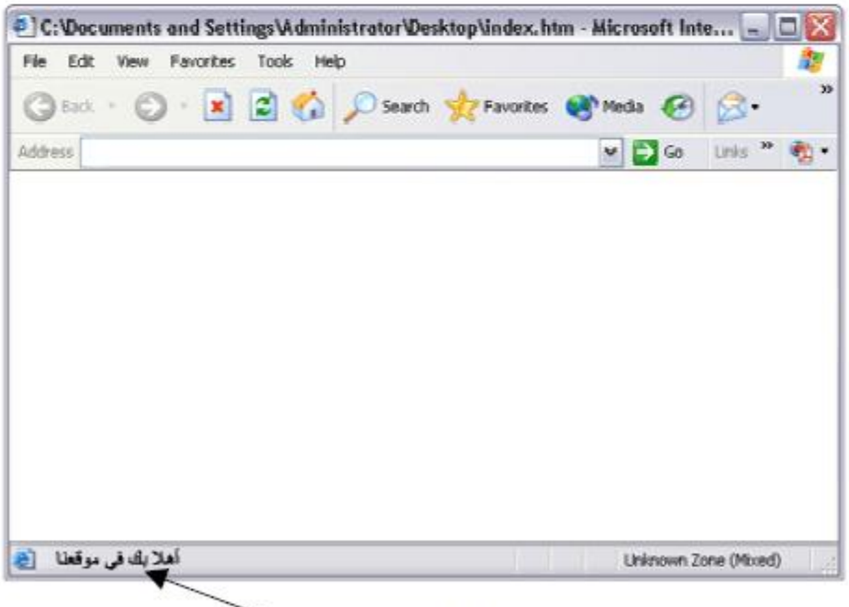
.

٢-

```
window.status =  " شرفت موقعنا "   name ;
```

مثال :

```
<html >
    <head><title>الجافا سكربت</title>
        <script type = "text/javascript">
            window.status = " أهلا بك في موقعنا " ;
        </script>
    </head>
  <body></body>

</html>
```



هنا تنفيذ المثال

لايحتاج الى تحليل فأنت أعلم به . ولكن لنفرض ان لدينا بعض الجمل الموجوده في المصفوفه ونريد عرضها في ال **status** اي يعرض لنا جمل متغيرة باستمرار ... اتركه لك لتفكر به ولكن ألمح لك لبعض الاستخدامات لتنفيذ ذلك.. اولا نحتاج الى تعريف مصفوفة **Array** وايضا الى جملة **for** والبقية عليك .

- لنأخذ ايضا بعض من الكائن **window** كأمثله وسوف اترك لك تحليلها ولكن سوف اضع بعض التعليقات التي سوف تساعدك على تحليل كل مثال ولن اضع لك صورا لها لكي تنفذها انت في نفسك وتتخيلها ..

```
<html >
    <head><title>الجافا سكربت</title>
        <script type = "text/javascript">
          window.open ("www.freewebs.com", "new_web", " toolbar = no ,
          location = no , directories = no , status = no , menubar= no , scrollbars
          = no  , resizable = no , copyhistory = yes , width = 400 , height = 400 " ) ;
        </script>
    </head>
  <body></body>

</html>
```

يقوم هذا البرنامج او الكود بفتح صفحة جديده بمجرد دخول المستخدم على الصفحة التي بها هذا الكود اما ما يحتوي تلاحظ انه يحتوي على الكائن **window** وبداخله اسم الموقع الذي سوف يفتحه للمستخدم اما بقية الاوامر هي خصائص صفحة الاكسبلورر من وجود **toolbar** او عدمه من سماح للمستخدم بالتحكم في حجم الصفحة التي سوف تفتح له بالاضافه الي طول وعرض الصفحة التي تريد ان تفتح للمستخدم ... الخ جرب ان تغير بين ال **yes** وال **no** وانظر ماذا سوف يحدث فالمعلومه تصلك بالتطبيق ..

```
<html >
    <head><title>الجافا سكربت</title>
        <script type = "text/javascript">
                location.reload( )
        </script>
    </head>
  <body></body>

</html>
```

يقوم الكائن **reload** من خلال الامر **location.reload( )** بإعادة تحميل الصفحة اي عملية **refresh** للصفحة .

ملاحظة:

يفضل إستخدام أمر **break** أو وضع الـ **location.reload( )** ضمن تكنيك معين من دالة أو حلقة تكرار لكي لا يدوم الـ **refresh** الى ما لا نهاية وبالتالي لا تستطيع إغلاق الصفحة

```
<html >
    <head><title>الجافا سكربت</title>
        <script type = "text/javascript">
            window.print( )
        </script>
    </head>
  <body></body>

</html>
```

يقوم هذا الكود بطباعة الصفحة التي يوضع بها الكائن **window.print( )** اي تشغيل خاصية الطباعة في الجهاز .. لتقوم الطابعة بطباعة الصفحة بمعنى بدلا من ان يذهب المستخدم الى **file** ومن ثم يختار **print** هذا الامر يقوم تلقائيا بذلك.. بمجرد دخول الزائر الى الصفحة عزيزي المتدرب حاول ان تضعه انت ضمن **button** بحيث عندما يضغط المستخدم عليها تقوم بطباعة الصفحة.. ليست صعبه تذكر انه يمكننا عمل **function** ووضع بداخله كائن طباعة الصفحة ومن ثم الذهاب الى كود الايقونة في ال **HTML** ووضع **onclick** ومساوته بإسم ال **function** وبهذا نقول عندما يضغط نفذ تذكر ما شرحناه سابقا في الـ **form** .

```
<html  dir="rtl">
    <head><title>الجافا سكربت</title>
          <script type = "text/javascript">
     document.write(" إصدار متصفحك هو :"+ navigator.appName + "<br />") ;
     document.write (window.screen.availWidth + "<br />" ) ;
     document.write( window.screen.availHeight + "<br />") ;
     document.write( window.screen.colorDepth + "<br />") ;
          </script>
    </head>
  <body></body>

</html>
```

هذا المثال لا يحتاج الى تحليل فهو يحتوي على كائنات المتصفح ووضعناها في جمل طباعة لكي يطبع لنا ما سوف تقوم به او بمعنى اصح ما سوف يرجعه لنا كل كائن وسوف اضع بعض الشرح لك لكي يسهل عليك تحليله ..

- **navigator.appName** هذا الكائن سوف يعود لنا بإسم المتصفح لدينا ولاحظ عند وضعه في جملة الطباعة وضعناه كمتغير .
- **window.screen.availWidth** هذا الكائن يقوم بارجاع عرض الشاشه في جهاز المستخدم اي **RESOLUTION SCREEN** .
- **window.screen.availHeight** هذا الكائن يقوم بارجاع طول الشاشه في جهاز المستخدم اي **RESOLUTION SCREEN** .
- **window.screen.colorDepth** هذا الكائن يقوم بارجاع لنا مقدار العمق في الألوان المستخدمه في جهاز المستخدم أليس المستخدم يختار في جهازه من **settings** في خصائص عرض الشاشه ال **color quality** مثلا **32 bit** او **16 bit** ... الخ

٤- حل النموذج

ويعني ذلك محاولة معرفة قيم المتغيرات المتحكم فيها والتي تعطي أفضل حل ممكن بدون تجاوز القيود المفروضة على المشكلة.

٥- كتابة التقرير

يجب أن يكتب بلغة بسيطة، موضحاً فيه الحل وطريقة تنفيذه.

استخدام التمثيل الكمي في حل المشاكل الإدارية

مثال رقم (١ - ١)

شركة ترغب في تحقيق أقصى ربح ممكن من إنتاج حقائب جلدية، ومعدل ربح الحقيبة الواحدة ١٢ ريال. ويلزم لإنتاج الحقيبة الواحدة أربع ساعات عمل.
ويتوفر لدى الشركة ٤٠ ساعة عمل فقط في الأسبوع الواحد. فما هو عدد الحقائب الممكن إنتاجها في الأسبوع من أجل تحقيق هدف الشركة (أقصى ربح)؟

تعريف المتغيرات:

د: دالة الهدف

س: عدد الوحدات الممكن إنتاجها من الحقائب

صياغة المشكلة رياضياً:

يمكن صياغة المشكلة على مرحلتين:

المرحلة الأولى: تمثيل الهدف

حقق أقصى ربح: $Max \quad Z = f(x)$

المرحلة الثانية: القيود المفروضة على الإنتاج

٤س $\leqslant$ ٤٠ (١)

س $\geqslant$ صفر (٢)

الشرط الثاني يكاد يكون بديهياً ويعني أنه يجب أن يكون العدد المنتج من الحقائب إيجابياً، ويعرف بشرط عدم السالبية.