تعلم

مكتبة JQUERY

جمع وإعداد :
أحمد إبراهيم

لاي سؤال او استفسار يمكنكم التواصل مع الكاتب عبر الفيسبوك :
https://www.facebook.com/ah.ib.93

كتب اخرى للكاتب في لغات برمجية مختلفة مثل (HTML, CSS
JavaScript, Java, PHP, Quick Basic) تجدونها على هذا الرابط :

https://drive.google.com/open?id=0B2aI_a6mphOUQi1jdzlYSFhITWs

فهرس الكتاب

المقدمة

مكتبة jQuery هي كأي صفحة مستقلة مكتوبة بلغة JavaScript نحتاج أن نستدعيها في صفحة HTML نكتب الكود الخاص بالاستدعاء في رأس الصفحة بين وسمي البداية والنهاية للوسم <head> وكود الاستدعاء هو :-

```
<script type="text/JavaScript" src=" jquery.js"> </script>
```

حيث أن jquery.js هو اسم المكتبة وامتدادها (.js) لكن يمكن أن يكون اسمها مختلف او في مجلد آخر وهنا يجب أن نذكر اسم المجلد والاسم الصحيح الخاص بالمكتبة , وكتبنا الوسمين السابقين لاستدعاء المكتبة ولكتابة شيفرات وأكواد المكتبة فيجب أن نضع الأكواد بين هذين الوسمين في داخل الوسم <head> وهما :-

```
<script type="text/JavaScript">
$ ( document ).ready ( function ( ) {
شيفرة جي كويري هنا
} ) ;
</script>
```

ولا يمكن بأي شكل من الأشكال كتابة أكواد jQuery خارج هذه المنطقة المحددة للمكتبة (لكن يمكن في بعض الأحيان الكتابة في داخل الوسم <body>) وهذا الكود الذي كتبناه في البداية وهو :

```
$ ( document ).ready( function ( ) {
شيفرة جي كويري هنا
} ) ;
```

يعني أن يتم تنفيذ (قراءة) أكواد المكتبة عند أول تحميل الصفحة , بالنتيجة سيكون الشكل العام للصفحة الرئيسية هكذا :-

```
<html>
<head>
<script type="text/JavaScript" src=" jquery.js"> </script>
<script type="text/JavaScript">
$ ( document ).ready ( function ( ) {
شيفرة جي كويري هنا
} ) ;
</head>
<body>
أكواد هوتميل ( الجزء الظاهر من الصفحة )
```

```
</body>
</html>
```

* يمكن كتابة ملف jQuery في صفحة مستقلة وبعد ذلك يتم استدعائها , حيث سنكتب بالصفحة الخاصة بالجكوري الكود مباشرتاً بدون أي وسوم بداية او أي شيء نبدأ مباشرتاً بكتابة ما نريد ونحفظها بالامتداد (name.js) , ثم وفي الصفحة التي نحتاجها نستدعيها وأيضا سوف نستدعي المكتبة ليكون كامل كود الاستدعاء بهذا الشكل

```
<head>
<script  type="text/JavaScript"  src="اسم الصفحة.js"> </script>
<script  type="text/JavaScript"  src="jquery.js"> </script>
</head>
```

وتكون صفحة الجكوري المستقلة بهذا الشكل

```
$ ( document ).ready( function ( ) {
شيفرة جي كويري هنا
 } ) ;
```

الفصل الأول

(المحددات Selectors)

المحددات تستخدم لتحديد مجموعة من عناصر مستند الويب (صفحة html) في البداية ليتم تطبيق شيء من الدوال او حركات jQuery عليها لاحقاً , لكي تقوم بتحديد عدد من عناصر الصفحة في jQuery يجب أن تكتب شيفرة jQuery بالشكل التالي :-

```
jquery ( 'selector' ) ;
```

او بهذا الشكل

```
$ ( 'selector' ) ;
```

ولا يوجد فرق بين الشكلين لأن $ و jquery يعتبران اسمين لتابع واحد , و selector يمثل المحدد الذي يعني مجموعة معينة من عناصر الصفحة مثلا إذا أردنا تحديد جميع الروابط في الصفحة سنستخدم المحدد a والذي يشير إلى وسم الرابط <a> بهذا الشكل

```
$ ( 'a' ) ;
```

وإذا أردنا تحديد كافة الصور الموجودة في الصفحة نستخدم المحدد img والذي يشير إلى الوسم <img /> الخاص بالصور بهذا الشكل

```
$ ( 'img' ) ;
```

وإذا أردنا تحديد كافة الجداول الموجودة في الصفحة والتي تحتوي على صور داخلها نستخدم المحدد td:has(img) بهذا الشكل :-

```
$ ( 'td:has(img) ) ;
```

* هكذا مع المحددات يمكننا وضع أي وسم من وسوم html لنحدده او يمكن وضع اسم id او اسم class لأي عنصر مع ملاحظة أنه نسبق اسم id بعلامة # ونسبق اسم class بالنقطة (.) او يمكن كتابة اسم وسمين ليعني إذا كان وسم بداخل وسم وهو مشابه لتحدد الوسوم في صفحة CSS

* هذا الجدول يوضح كيفية تحديد العناصر (المحددات selectors)

المحدد selector	الوصف Description
*	يعني جميع العناصر الموجودة في المستند
T	يعني جميع عناصر الوسم T في المستند (مثلاً P يعني جميع

	عناصر الوسم P التي في الصفحة وa يعني جميع عناصر الوسم a وimg يعني جميع عناصر الوسم img وهكذا)
.C	يعني جميع عناصر الوسوم التي تنتمي لصف الأنماط C (الوسوم ذات الخاصية class="C" أياً كان نوعها .
.C1.C2.C3 … .Cn	يعني جميع الوسوم التي تنتمي لصفوف الأنماط المذكورة C1 و C2 وC3 و Cn وكل ما تم ذكره .
#X	يعني جميع عناصر الوسوم ذات المعرف X (الوسوم ذات الخاصية id="X" أياً كان نوعها)
T F	يعني جميع عناصر الوسوم F الموجودة ضمن عناصر الوسم T بشكل مباشر وغير مباشر
T>F	يعني جميع عناصر الوسم F الموجودة بشكل مباشر ضمن عناصر الوسم T (عندما نكتب مثلاً div>a فإننا نعني جميع الروابط الموجودة ضمن div بشكل مباشر فلو كان هذا ال div ذاته يحتوي على جدول وأحد خلايا هذا الجدول تحتوي على رابط فلن يكون هذا الأخير من ضمن الروابط التي يعيدها المحدد , عند إذ يمكننا أن نستخدم الصيغة السابقة div a لتحديد كل الروابط التي ضمن وسوم div بشكل مباشر او غير مباشر)
T+F	يعني جميع عناصر الوسم F المجاورة لعناصر الوسم T (يقصد بالعناصر المجاورة العناصر الأبناء من نفس المستوى)
T:has(F)	يعني جميع عناصر الوسم T التي تحتوي داخلها عنصراً ابناً واحداً على الأقل من الوسم F (مثلاً table:has(a) يعني أي جدول يحتوي رابطاً واحداً او أكثر من رابط ضمن محتوياته)
T[A]	يعني جميع عناصر الوسم T التي تحمل الخاصية A مهما كانت قيمة هذه الخاصية A (مثلاً نستطيع أن نكتب المحدد a[href] لتعيد جميع الروابط التي تحتوي على خاصية href مهما كانت قيمتها)
T[A=Value]	يعني جميع عناصر الوسم T التي تحمل الخاصية A شرط أن تكون قيمة هذه الخاصية هي القيمة Value (مثلاً تستطيع أن تكتب المحدد a [target=_blank] لتعيد جميع الروابط التي تفتح في نافذة هدف جديد
T[A^=Value]	يعني جميع عناصر الوسم T التي تعرف الخاصية A شرط أن تكون بداية القيمة التي تحتويها هذه الخاصية هي القيمة Value (فمثلاً لو أردنا تحديد كافة الروابط التي تنقلك لموقع يبدأ بحرفي de فالمحدد a[href^=www.de] يلبي حاجتنا

	تماماً)
T[A$=Value]	يعني جميع عناصر الوسم T التي تعرف الخاصية A شرط أن تكون نهاية القيمة التي تحتويها هذه الخاصية هي القيمة Value (المحدد a [href$=.mp3] علي سبيل المثال يعني جميع الروابط التي تشير إلى ملفات من نمط mp3)
T[A*=Value]	يعني جميع عناصر الوسم T التي تعرف الخاصية A شرط أن تكون القيمة التي تحتويها هذه الخاصية تتضمن القيمة Value (المحدد a[href*=great] يعني جميع الروابط التي يحتوي عنوانها على الكلمة great)
selector:first	يعني أول عنصر مطابق للمحدد selector على مستوى المستند (استخدام المحدد a:first على سبيل المثال يعيد أول عنصر رابط في المستند واستخدام المحدد div img:first يعيد أول عنصر صورة موجود ضمن وسم div في المستند)
selector:last	يعني آخر عنصر مطابق للمحدد selector على مستوى المستند
selector:first-child	يعني أول عنصر أبن مطابق للمحدد selector على مستوى المحدد (إذا استخدمنا على سبيل المثال المحدد div a:first-child فإننا نعني كل أول رابط موجود ضمن كل وسم div في المستند وليس أول حالة مطابقة للمحدد فقط كما هو الحال مع first)
selector:last-child	يعني آخر عنصر ابن مطابق للمحدد selector على مستوى المحدد
selector :only-child	يعني العنصر الابن المطابق للمحدد selector شرط أن يكون وحيداً في وسمه (لو أردنا أن نعيد جميع الروابط الموجودة ضمن الوسم div) شرط أن لا يوجد أكثر من رابط ابن مجاور في كل div فإن استخدام المحدد div a:only-child سيلبي مطلبنا)
selector:nth-child(n)	يعني العنصر الابن رقم n المطابق للمحدد selector (مثلاً لو أردنا تحديد الرابط الرابع الموجود في كل وسم div فإننا نكتب المحدد التالي div a:nth-child(4))
selector:nth-child(event)	يعني العناصر الأبناء ذات الأرقام الزوجية المطابقة للمحدد selector
selector:nth-child(odd)	يعني العناصر الأبناء ذات الأرقام الفردية المطابقة للمحدد selector
selector:nth-child(Xn)	يعني العناصر الأبناء ذات الأرقام التي من مضاعفات X (مثلاً في حال كون الX يحمل القيمة 3 فإن المحدد النهائي

	يصبح div a:nth-child(3n) ويعني جميع الروابط الموجودة في الوسم div على أن يكون ترتيبها من مضاعفات العدد 3 وهذه الروابط هي الرابط الثالث والسادس والتاسع والثاني عشر و... وهكذا)
selector:nth-child(Xn+Y)	يعني Y عنصراً ابناً بعد كل X أبناء (مثلاً في حال كون قيمة Y هي 2 وقيمة X هي 3 سيصبح شكل المحدد النهائي div a:nth-child(3n+2) وهكذا يعني تحديد رابطين بعد كل ثلاثة روابط من أبناء div مما يعني الروابط الرابع والخامس (رابطين بعد الرابط الثالث) والسابع والثامن (رابطين بعد الرابط السادس) والعاشر والحادي عشر (رابطين بعد الرابط التاسع) .... وهكذا)
selector:even	يعني العناصر ذات الأرقام الزوجية المطابقة للمحدد selector على مستوى المستند
selector:odd	يعني العناصر ذات الأرقام الفردية المطابقة للمحدد selector على مستوى المستند
selector:eq(n)	يعني العنصر ذو الترتيب n المطابق للمحدد selector على مستوى المستند
selector:qt(n)	يعني العنصر ذو الترتيب n المطابق للمحدد selector وجميع العناصر المطابقة التي تليه على مستوى المستند
selector:lt(n)	يعني العنصر ذو الترتيب n المطابق للمحدد selector وجميع العناصر المطابقة التي تسبقه على مستوى المستند
:animated	يعني جميع العناصر التي تخضع لحركة حالياً
button	يعني جميع عناصر الأزرار الموجودة في المستند سواءً أكانت من النوع submit او reset او النوع button بشكل عام
:checkbox	يعني جميع عناصر صناديق الاختيار check boxes
:checked	يعني جميع عناصر صناديق الاختيار المحددة (أعني بالمحددة ما يكون حالتها checked سواء أكانت checkbox او radio)
:contains(S)	يعني جميع العناصر التي تحتوي على النص S (مثلاً يعيد المحدد (jQuery) P:contains جميع عناصر النصوص التي تحتوي على كلمة jQuery ويجدر الإشارة هنا أن هذا المحدد حساس لحالة الأحرف الصغيرة والكبيرة)
:disabled	يعني جميع العناصر المعطلة (العناصر غير الفعالة)
:enabled	يعني جميع العناصر الفعالة
:filed	يعني جميع عناصر اختيار الملفات زهو مكافئ للمحدد

input [type="file"]	
يعني جميع عناصر الوسوم الخاصة بالعناوين h1 , h2 , h3 , h4 , h5 , h6	**:header**
يعني جميع العناصر المخفية	**:hidden**
يعني جميع وسوم صور النماذج وهو مكافئ للمحدد input[type=image]	**:image**
يعني جميع وسوم النماذج input	**:input**
يعني جميع الوسوم التي لا تطابق المحدد selector (مثلاً يعيد المحدد input:not(:button) جميع عناصر النماذج عدا الأزرار منها	**:not(selector)**
يعني جميع عناصر الوسوم الآباء (التي تحتوي على عناصر وسوم فرعية بما فيها النصوص)	**:parent**
يعني جميع عناصر إدخال كلمات المرور وهو مكافئ للمحدد input[type=password]	**:password**
يعني جميع عناصر أزرار الاختيار radio	**:radio**
يعني جميع أزرار إعادة التعيين	**:reset**
يعني جميع عناصر الاختيار المختارة (selected option)	**:selected**
يعني جميع أزرار الإرسال	**:submit**
يعني جميع مربعات إدخال النصوص وهو مكافئ للمحدد input[type=text]	**:text**
يعني جميع العناصر الظاهرة	**:visible**
يعني اجتماع جميع عناصر الوسوم التي تعيدها المحددات المذكورة ويقصد بالمحدد هنا كل ما هو مذكور في هذا الجدول وهذه القاعدة تعتبر مفيدة جداً في حال الرغبة بتحديد أجزاء كثيرة من الصفحة لا يستطيع محدد واحد أن يقوم بتحديدها	**selector1 , selector2 , ... , selectorn**

* المحددات وكما قلنا في البداية هي فقط تحدد العناصر ولتطبيق تأثير او حركة ما يجب أن نستعمل دوال المكتبة لبتم تطبيقها على العناصر المحددة وتكتب بهذا الشكل

```
$ ( 'a' ) . function ( ) ;
```

حيث function هي اسم الدالة التي نريد تطبيق تأثيرها على العناصر المحددة بالدالة $ التي تأخذ العناصر التي حددناها كأنها مصفوفة وسترسل هذه المحددات (مصفوفة العناصر) إلى الدالة function التي بدورها ستطبق التأثير على العناصر .

ملاحظة / يمكننا تطبيق أكثر من دالة على المحددات وذلك بكتابتها بهذا الشكل

```
$ ( 'selector' ).function1 ( ).function2 ( ) ;
```

ويمكننا كتابة ما نشاء من الدوال بنفس الطريقة فقط نفصل بينها بنقطة .

* تستخدم الدالة addClass لإعطاء اسم class للعناصر المحددة

مثال/

```
$ ( 'a' ).addClass( 'red' ) ;
```

في هذا المثال أعطينا اسم class هو red لجميع الروابط الموجودة في الصفحة أي كما لو أننا كتبنا بداخل كل رابط في صفحة html (clss=red)

مثال /

```
<html>
<head>
<script  type="text/JavaScript"  src=" jquery.js"> </script>
<script  type="text/JavaScript">
$ ( document ).ready ( function ( ) {
$( '#Button1' ).click( function ( ) {
$( '#TextArea1' ).toggle( 'slow' ) ;
} )
} ) ;
</script>
</head>
<body>
<input  type="button"  id="Button1"  />
<textarea  id="TextArea1"  name="S1"> </textarea>
</body>
</html>
```

توليد محتويات HTML جديدة

لتوليد محتوى html جديد في الصفحة نستخدم الصيغة التالية

```
$ ( 'html' )
```

حيث أن html هنا تعني المحتوى الجديد الذي نريد توليده مكتوب بلغة html العادية مثلاً لإضافة طبقة div جديد نكتب الشكل التالي (' ') $ او يمكن كتابته بالشكل التالي $(' ') لاحظ أنه لا يوجد فرق أي بمعنى أنه يمكن عدم إغلاق الوسوم لأن المكتبة ستقوم بهذا بدلاً عنا , ولكن عند استعمال هذه الصيغة فأن مكتبة jQuery تعيد المحتوى الجديد الذي تم إنشائه لكنها لا تضيفه إلى الصفحة لتعطيك بذلك حرية إضافته في المكان المطلوب تماماً من الصفحة فمثلاً يمكن الكتابة بهذا الشكل

```
$( '<div> Hello </div>' ).appendTo( '#id' ) ;
```

ليتم إضافة الوسم إلى نهاية المنطقة التي تمتلك الاسم id .

ملاحظة / مكتبة jQuery لا تمكننا من إنشاء عناصر الوسم script باستخدام هذه الطريقة .

الفصل الثاني

(الدوال functions)

سنقسم الدوال إلى مجموعات حسب الوظيفة الخاصة بكل منها إلى ما يلي :-

أولاً :- دوال التعامل مع خصائص الوسوم Attributes

◆ الدالة attr

كما تعلم فإن لكل وسم في لغة html مجموعة من الخصائص ولجلب قيمة إحدى هذه الخصائص او إضافة قيمة لخاصية معينة نستخدم الدالة attr , لجلب قيمة خاصية معينة نستخدم الصيغة التالية

```
$ ( '#id' ).attr ( 'اسم الخاصية' ) ;
```

هنا سيتم جلب قيم الخاصية التي ذكرنا اسمها الخاصة بالوسم ذو المعرف id , ولإعطاء قيمة إلى خاصية معينة نستخدم الصيغة العامة التالية

```
$ ( '#id' ).attr ( 'اسم الخاصية' , 'القيمة' ) ;
```

هنا سيتم إعطاء القيمة المحددة إلى الخاصية المذكورة الخاصة بالوسم ذو المعرف id .

* مع ملاحظة أن الطرق السابقة تشمل محدد واحد معرف بالخاصية id ولقراءة خاصية معينة لجميع عناصر مجموعة معينة نستخدم الصيغة التالية

```
$ ( 'selector' ).each( function ( ) {
$ ( this ).attr( 'اسم الخاصية' ) ;
} ) ;
```

حيث أن each هي حلقة تكرار و this هي كلمة محجوزة تشير إلى العنصر الحالي أثناء الدوران , ولإضافة قيمة لخاصية معينة نستخدم الصيغة العامة التالية

```
$ ( 'selector' ).each( function ( n ) {
$ ( this ).attr( 'اسم الخاصية' , 'القيمة' ) ;
} ) ;
```

او يمكن إضافة قيم لمجموعة من الخواص لمحدد ما وذلك باستخدام الصيغة العامة التالية

```
$ ( 'selector' ).attr(
```

```
{ 'القيمة':اسم الخاصية , 'القيمة':اسم الخاصية }
) ;
```

ويسمى هذا ما بين القوسين المعقوفين { } بالمتغير (JSON) .

مثال /

```
$ ( ':input' ).attr(
{ value:'  ' , title:'welcome' }
) ;
```

في هذا المثال سيتم إضافة قيم نصية فارغة إلى الخاصية value لجميع عناصر الوسم input وكذلك يتم إضافة القيمة welcome إلى الخاصية title .

* يمكن أن نكتب داخل قوسي الدالة attr اسم function ونضيف داخل هذه الـ function بارامترين الأول يعبر عن الخاصية والثاني يعبر عن القيمة ونختار لهم أي اسمين .

مثال/ لإضافة قيمة جديدة إلى الخاصية title لكن لا يحذف القيمة القديمة للخاصية بل يضيف القيمة الجديدة على القيمة القديمة

```
$ ( '#a' ).attr( 'title' , function( x , y ) {
return y+ "new text" ;
} ) ;
```

◆ الدالة removeAttr

وهي تستخدم لحذف خاصية معينة مع قيمتها من وسم معرف بالخاصية id او لجميع مجموعة الوسوم الواحدة الموجودة في الصفحة , ولها الصيغة العامة التالية

```
$( ( '#id' ).removeAttr( 'اسم الخاصية' ) ;
```

مثال/ لحذف الخاصية target لجميع الروابط في الصفحة

```
$ ( 'a' ).removeAttr( 'target' ) ;
```

◆ الدالة prop

هذه الدالة نفس الدالة attr تماماً ولا يوجد أي فرق (وكل ما نستطيع فعله بالدالة attr يمكن فعله بالدالة prop) , حيث أنها تستخدم لجلب قيمة خاصية معينة وتكون الصيغة العامة لها هي

```
$ ( 'selector' ).prop( 'اسم الخاصية' );
```

وتستخدم أيضاً لإضافة قيمة للخاصية المحددة بالشكل التالي

```
$ ( 'selector' ).prop( 'اسم الخاصية' , 'القيمة' ) ;
```

لكن بالصيغة السابقة يمكن التعامل مع خاصية واحدة ولكي نتعامل مع عدة خواص يجب أن نكتبها داخل المتغير JSON

- الدالة **removeProp**

تستخدم هذه الدالة لإزالة قيمة خاصية معينة أي أنها لا تزيل الخاصية بل تزيل فقط القيمة التي تحملها هذه الخاصية المحددة بخلاف الدالة removeAttr التي تزيل الخاصية مع ما تحمل من قيمة والصيغة العامة لها هي

```
$( ( '#id' ).removeAttr( 'اسم الخاصية' ) ;
```

ثانياً :- دوال التعامل مع الأنماط style

- الدالة **addClass**

تستخدم هذه الدالة لإضافة اسم class للعناصر المحددة , والصيغة العامة لها هي

```
$ ( 'selector' ).addClass( 'اسم الكلاس' ) ;
```

- الدالة **removeClass**

تستخدم هذه الدالة لحذف أحد صفوف الأنماط التي ينتمي إليها كائن معين أي أنها ستحذف اسم class للعنصر المحدد والصيغة العامة لها هي

```
$ ( 'selector' ).removeClass( 'اسم الكلاس' ) ;
```

الدالة toggleClass

تستخدم هذه الدالة للقلب حيث أنها تقوم بإضافة أحد صفوف النمط إلى العناصر التي لا تنتمي إليه من المجموعة التي يعيدها المحدد (أي تضيف اسم class) وتقوم بنفس الوقت بإقاف تطبيق ذات الصف على العناصر التي تنتمي إليه من المجموعة التي يعيدها المحدد (أي تحذف اسم class) , والصيغة العامة لهذه الدالة هي

```
$ ( 'selector' ).toggleClass( 'اسم الكلاس' ) ;
```

الدالة hasClass

تستخدم هذه الدالة للتحقق إن كان اسم هذا الصف (class) موجود في المحدد المذكور أو لا ويمرر اسم الصف (class) المطلوب البحث عنه بين قوسيها حيث أنها ستعيد القيمة true إذا كان موجود وتعيد القيمة false إذا كان غير موجود والصيغة العامة لها هي

```
$ ( 'selsctor' ).hasClass( 'اسم الكلاس' );
```

الدالة css

تسمح لنا بقراءة قيمة خاصية نمط العنصر بشكل مباشر عبر الصيغة العامة التالية

```
$ ( '#id' ).css( 'name' ) ;
```

حيث ستقوم الدالة بإعادة قيمة الخاصية name من خصائص أنماط العنصر الذي يحمل المعرف id , أما لإسناد قيمة خاصية نمط لعنصر معين يمكن استعمال الصيغة التالية

```
$ ( '#id' ).css(
{ "name1":"value1" , "name2":"value2" }
) ;
```

ويسمى ما بين القوسين المعقوفين { } من خصائص بالمتغير او الكائن (JSON) .

مثال/

```
$ ( '#div' ).css( { "color":"red" } ) ;
```

يمكن كتابة نفس المثال السابق لكن بطريقة مختلفة مع ملاحظة أن هذه الطريقة الجديدة تستخدم فقط مع خاصية واحدة ولا يمكن تطبيق أكثر من خاصية بعكس الطريقة في المثال السابق حيث يمكن تطبيق أكثر من خاصية (الطريقة في المثال السابق هي الأفضل)

```
$ ( '#div' ).css( "color" , "red" ) ;
```

مثال/

```
$ ( '#test' ).css( {
        "color" : "red" ,
        "font-size" : "30px" ,
        "background" : "#999"
} ) ;
```

* مع ملاحظة أنه يمكن كتابة هذه الخواص بسطر واحد او بعدة أسطر لا يوجد فرق .

◆ الدالة width

من خلال هذه الدالة يمكن التعامل مع عرض العنصر في صفحة الـ CSS حيث يمن قراءة عرض العنصر عبر الصيغة التالية

```
$ ( '#id' ).width( ) ;
```

او يمكن إسناد قيمة لعرض العنصر بالشكل التالي

```
$ ( '#id' ).width( 'القيمة' ) ;
```

* وكذلك يمكن من خلال هذه الدالة قراءة عرض النافذة المفتوحة window إذا مررنا كلمة window كمحدد لهذه الدالة (لكن بدون علامات تنصيص) هكذا

```
$ ( window ).width( );
```

او قراءة عرض المستند الحالي document إذا مررنا كلمة document كمحدد لهذه الدالة (لكن بدون علامات تنصيص) هكذا

```
$ ( document ).width( );
```

وهذا مهم لتغير نمط الموقع بحسب حجم الشاشة لدى الزائر .

◆ الدالة height

من خلال هذه الدالة يمكن التعامل مع ارتفاع العنصر في صفحة الـ CSS حيث يمن قراءة ارتفاع العنصر عبر الصيغة التالية

```
$ ( '#id' ).height( ) ;
```

او يمكن إسناد قيمة لارتفاع العنصر بالشكل التالي

```
$ ( '#id' ).height( 'القيمة' ) ;
```

* وكذلك يمكن من خلال هذه الدالة قراءة ارتفاع النافذة المفتوحة window إذا مررنا كلمة window كمحدد لهذه الدالة (لكن بدون علامات تنصيص) هكذا

```
$ ( window ).height( );
```

او قراءة عرض المستند الحالي document إذا مررنا كلمة document كمحدد لهذه الدالة (لكن بدون علامات تنصيص) هكذا

```
$ ( document ).height( );
```

وهذا مهم لتغير نمط الموقع بحسب حجم الشاشة لدى الزائر .

🔹 الدالتين innerWidth و outerWidth

هذه الدالتين هما تماماً نفس الدالة width لكن الفرق هو أن الدالة innerWidth تقيس العرض الداخلي للعنصر ولا تقيس ما هو خارج ذلك من إطار للعنصر او إزاحة خارجية للعنصر (margin) وهي أيضاً لا تقيس الإزاحة الداخلية للعنصر (padding) لكن يمكن أن نجعلها تقيسها وذلك بوضع true بين قوسيها وإذا وضعنا false فهي لن تقيسها لأن false هي القيمة الافتراضية (أي لا يوجد فرق سواء كتبناها او لم نكتبها) , وتكون الصيغة العامة لها بهذا الشكل

```
$ ( 'selector' ).innerWidth( ) ;
```

او بهذا الشكل

```
$ ( 'selector' ).innerWidth( true ) ;
```

أما الدالة outerWidth فهي تقيس كل العرض مع المسافة الخارجية وهي بشكل افتراضي لا تقيس الإزاحة الخارجية للعنصر (margin) لكن يمكن أن نجعلها تقيسها وذلك بوضع true بين قوسيها وإذا وضعنا false فهي لن تقيسها لأن false هي القيمة الافتراضية (أي لا يوجد فرق سواء كتبناها او لم نكتبها) , وتكون الصيغة العامة لها بهذا الشكل

```
$ ( 'selector' ).outerWidth( ) ;
```

او بهذا الشكل

```
$ ( 'selector' ).outerWidth( true ) ;
```

🔹 الدالتين innerHeight و outerHeight

هذه الدالتين هما تماماً نفس الدالة Height لكن الفرق هو أن الدالة innerHeight تقيس الارتفاع الداخلي للعنصر ولا تقيس ما هو خارج ذلك من إطار للعنصر او إزاحة خارجية للعنصر (margin) وهي أيضاً لا تقيس الإزاحة الداخلية للعنصر (padding) لكن يمكن أن نجعلها تقيسها وذلك بوضع true بين قوسيها وإذا وضعنا false فهي لن تقيسها لأن false هي القيمة الافتراضية (أي لا يوجد فرق سواء كتبناها او لم نكتبها) , وتكون الصيغة العامة لها بهذا الشكل

```
$ ( 'selector' ).innerHeight( ) ;
```

او بهذا الشكل

```
$ ( 'selector' ).innerHeight( true ) ;
```

أما الدالة outerHeight فهي تقيس كل الارتفاع مع المسافة الخارجية وهي بشكل افتراضي لا تقيس الإزاحة الخارجية للعنصر (margin) لكن يمكن أن نجعلها تقيسها وذلك بوضع true بين قوسيها وإذا وضعنا false فهي لن تقيسها لأن false هي القيمة الافتراضية (أي لا يوجد فرق سواء كتبناها او لم نكتبها) , وتكون الصيغة العامة لها بهذا الشكل

```
$ ( 'selector' ).outerHeight ( ) ;
```

او بهذا الشكل

```
$ ( 'selector' ).outerHeight( true ) ;
```

ثالثاً :- دوال التعامل مع محتوى عناصر المستند Inner Content

الدالة html

تقوم هذه الدالة بقراءة محتوى عنصر معين عبر الشكل التالي

```
$ ( '#id' ).html( ) ;
```

حيث أنها تعيد شيفرة html التي تمثل المحتوى الموجود داخل العنصر ذو المعرف id وتقوم بإسناد قيمة جديدة للمحتوى (استبداله) بهذا الشكل

```
$ ( '#id' ).html( com ) ;
```

حيث أن com هو شيفرة html التي تمثل المحتوى الجديد .

الدالة text

فهي مماثلة للدالة السابقة عدا أنها تتعامل مع المحتوى كنص عادي وليس كشيفرة html (لا يمكن أن نكتب بين قويها وسوم html) , ولها الشكل التالي في حالة القراءة

```
$ ( '#id' ).text( ) ;
```

والشكل التالي في حالة الإسناد

```
$ ( '#id' ).text( com ) ;
```

* مقارنة بين الدالة html والدالة text

لو كال لدينا في صفحتنا ما يلي

```
<ul  id="myul">
<li> 1 </li>
<li> 2 </li>
</ul>
```

ثم قمنا بإسناد الدالة text كما يلي

```
$ ( '#myul' ).text( ) ;
```

فإنها ستعيد القيمة 2 1 كنص , أما في حالة استدعاء الدالة html بنفس الطريقة فإنها ستعيد المحتوى

```
<li> 1 </li>
<li> 2 </li>
```

الدالة size

تستخدم هذه الدالة لتعيد عدد العنصر او الوسوم الموجودة (المحدد) تعيد لنا عددها بشكل رقم وليس مصفوفة كما تفعل الدالة index .

مثال/

```
$ ( 'ul  li' ).size( ) ;
```

◆ الدالة index

تستخدم هذه الدالة لمعرفة عدد العناصر او الوسوم الموجود (المحددة) وتعيد لنا أعدادها بشكل مصفوفة وليس بشكل رقم كما تفعل الدالة size وهي تأخذ بارامتر وهو الكائن this ليشير إلى هذه الوسوم المحددة .

مثال/

```
$ ( '#div ul > li' ).index( this ) ;
```

◆ الدالة next

تستخدم هذه الدالة إذا كنا قد حددنا عنصر وأردنا أن يجري التطبيق او التأثير على العنصر المجاور له وليس عليه , مثلاً لو كان لدينا وسم وبداخله مثلاً وسمين آخرين الأول يحمل الخاصية id="box1" والثاني يحمل الخاصية id="box2" وأردنا عند الضغط على الصندوق الأول يختفي الصندوق الثاني يمكن أن نكتب ما يلي

```
$ ( '#box1' ).click( function( ) {
$ ( this ).next('#box2').hide( "slow" ) ;
} ) ;
```

◆ الدالة children

تستخدم هذه الدالة إذا كنا قد حددنا عنصر وأردنا أن يجري التطبيق او التأثير على العنصر الابن له وليس

```
$ ( '#box1' ).click( function( ) {
$ ( this ). children ('p' ).hide( "slow" ) ;
} ) ;
```

◆ الدالة is

تستخدم هذه الدالة مع أدوات الشرط حيث من خلالها نسأل هل حالة العنصر المحدد كذا (مثلا مخفي) وستعود بقيمة true إذا كان السؤال صح او تعود بالقيمة false إذا كان خطأ .

مثال/

```
$ ( '#box' ).click(function( ) {
$ ( '#box' ).hide( "slow" ) ;
if ( $ ( this ).is( ':hidden' ) ) {
alert( "yes" ) ;
} else {
alert( "no" ) ;
}
} ) ;
```

◆ الدالتين append و appendTo

بالنسبة للدالة append

تستخدم الدالة append لإضافة محتوى html جديد إلى نهاية العنصر المحدد وتستعمل بالشكل التالي

```
$ ( 'selector' ).append( 'HTML' ) ;
```

ولهذه الدالة عمل مهم آخر يمكننا من نقل او نسخ مجموعة من عناصر المستند من مكانها ضمن المستند إلى مكان آخر ضمنه , والصيغة العامة لها تكون بهذه الشكل

```
$ ( 'targetSelector' ).append( $ ( 'sourceSelector' ) ) ;
```

حيث هنا مررنا مجموعة من العناصر للدالة عن طريق استدعاء محدد sourceSelector ولم نقم بتمرير محتوى HTML بشكل مباشر وفي هذه الحالة ستقوم هذه التعليمة بأخذ العناصر التي يعيدها المعرف sourceSelector ونقلها من مكانها الأصلي إلى المكن الهدف مما يعني حذفها من مكانها الأصلي وإضافتها في المكان الهدف وهو آخر العنصر الذي يعيده targetSelector وإن كان ما يعيده هو عنصراً واحداً أما إن كان أكثر من عنصر فإن الدالة ستقوم بعمليه نسخ بدلاً من النقل مما يعني أنها ستحافظ على المحتوى المصدر وتضيف محتوى مماثل له تماماً آخر كل كائن يعيده المحتوى الهدف , وكمثال على الموضوع تصور أن لدينا المحتوى التالي في المستند

```
<ul>
<li> 11 </li>
<li  id="item12"> 12 </li>
</ul>
<ol>
<li> 22 </li>
</ol>
```

في هذه الحالة فإن الاستدعاء

```
$ ( '#item12' ).append( $ ( 'ol' ) ) ;
```

سيقوم بحذف القائمة الثانية من موضعها الأصلي وإضافة واحدة جديدة مطابقة لها تماماً وجعلها جزءاً من القائمة الأولى تابعاً للعنصر الثاني في نهايته , في حين أن الاستدعاء

```
$ ( 'ul  li' ).append( $ ( 'ol' ) ) ;
```

سيقوم بنسخ القائمة الثانية بدون أن يؤثر عليها وينشأ قائمة مطابقة لها تماماً يضيفها إلى نهاية كل عنصر من عناصر القائمة الأولى

* نستنتج مما سبق الخلاصة التالية إذا كان المحدد الهدف يعيد أكثر من عنصر (مصفوفة من العناصر) فإن استدعاء الدالة append يقوم بنسخ المصدر أما إن كان ما يعيده المحدد الهدف عنصراً واحداً فأن استدعاء الدالة يقوم بنقل المصدر .

أما بالنسبة للدالة appendTo

فهي تماماً نفس الدالة append لكن الفرق الوحيد بينهما هو أن الدالة appendTo يكون الباراميتر الموجود بين قوسيها هو المحدد الهدف الذي ستنقل له العناصر على عكس الدالة append حيث أن الباراميتر الذي يكون بين قوسيها هو المصدر الذي ستأخذ منه العناصر , والصيغة العامة للدالة appendTo هي

```
$ ( 'sourceSelector' ).appendTo( 'targetSelector' ) ;
```

◆ الدالتين prepend و prependTo

تعمل هاتان الداللتان نفس عمل ألدالتي السابقتين (append , appendTo) لكن الفرق الوحيد هو أن ألدالتي السابقتين تعيدان المحتوى في نهاية المحتوى الداخلي للعنصر المحدد (قبل وسم الإغلاق الخاص بكل منهما) أما هاتان الداللتان فإنهما تعيدان المحتوى الجديد في بداية المحتوى

الداخلي للعنصر (بعد وسم البدأ لكل مهما) حيث تعمل prepend بنفس طريقة append وتعمل prependTo بنفس عمل appendTo

◆ الدالتين ( ) before و ( ) insertBefore

تعمل هاتان الداللتان بنفس أسلوب الدالتين السابقتين (prepend , prependTo) تماماً لكن باختلاف أن هاتين الدالتين تضيفان العناصر قبل الهدف تماماً أي قبل وسم البدأ الخاص به أما الدالتين السابقتين تضيفان العناصر بعد وسم البدأ الخاص بالعنصر الهدف

مثال/

```
$ ( 'p  img ' ).before( '<p> I love America </p>' ) ;
```

يقوم المثال السابق بإضافة الجملة (I love America) قبل كل عنصر يعيد المحدد p img , ويقوم المثال التالي بنفس المهمة

```
$ ( '<p> I love America </p>' ).insertBefore( 'p img' ) ;
```

◆ الداللتان ( ) after و ( ) insertAfter

هاتان الداللتان تعملان تماماً مثل الدالتين (append , appendTo) باختلاف واحد ألا وهو أن الداللتان (append , appendTo) تضيفان المحتوى في نهاية العنصر الهدف أي قبل وسم الغلق للعنصر الهدف أما هاتان الداللتان (after , insertAfter) فإنهما تضيفان المحتوى في نهاية العنصر الهدف لكن بعد وسم الغلق للعنصر الهدف .

مثال/

```
$ ( 'a[ href ^=http://www. ]' ).after( '<small  style="color:red"> external
</small>' ) ;
```

يقوم هذا المثال بإضافة الكلمة external بعد كل رابط يشير إلى موقع خارجي , ويقوم المثال التالي بنفس الوضيفة

```
$ ( '<small  style="color:red"> external </small>' )insertAfter (
```

```
'a[ href ^=http://www. ]' ) ;
```

◆ الدالة clone

تستخدم هذه الدالة لنسخ عناصر المستند وهي غالباً ما تستخدم مع الدوال before و after و append

مثال/ يقوم بنسخ عنصر div من المستند ويضيفها إلى نهاية div آخر

```
$ ( '#sourceDiv' ).clone( ).appendTo( '#targetDiv' ) ;
```

رابعاً :- دوال التغليف wrapping

أحياناً نحتاج للقيام بتغليف محتوى ما بوسم معين او مجموعة من الوسوم فمثلاً لنفرض أننا نريد تغليف جميع وسوم <img> بالوسم <a> مثلاً او نريد تغليف جميع العناصر p a بالوسم , ولهذه العملية تؤمن لنا مكتبة jQuery مجموعة من الدوال وهي :-

◆ الدالة wrap

تقوم هذه الدالة بتغليف كل عنصر من مجموعة العناصر التي تطبق عليها بالغلاف الممرر لها كوسيط parameter والصيغة العامة لها هي

```
$ ( 'selector' ).wrap( 'وسم الغلاف' ) ;
```

مثال/ لتغليف كل عنصر صور برابط

```
$ ( '<img [src]>' ).wrap( '<a href="#"> </a>' ) ;
```

◆ الدالة wrapAll

وهي نفس الدالة السابقة باختلاف أن هذه الدالة تقوم بتغليف جميع العناصر المحددة بغلاف واحد فقط .

◆ الدالة wrapInner

تقوم هذه الدالة بتغليف محتوى مجموعة من العناصر بدلاً من تغليفها نفسها , والصيغة العامة لها هي

```
$ ( 'selector' ).wrapInner( 'وسم الغلاف' ) ;
```

ملاحظة/ يمكن الاستفادة من دوال التغليف لنسخ العناصر من مكان إلى آخر ضمن المستند وذلك عن طريق جعل الغلاف الجديد عنصراً من العناصر الموجودة أصلاً في المستند .

خامساً :- دوال حذف عناصر المستند

- الدالة **remove**

تستخدم هذه الدالة لحذف جميع العناصر التي يعيدها المحدد , والصيغة العامة لها هي

```
$ ( 'selector' ).remove ( ) ;
```

- الدالة **empty**

تستخدم هذه الدالة لحذف المحتوى الداخلي الخاص بالناصر المحددة والإبقاء على العناصر نفسها , والصيغة العامة لها هي

```
$ ( 'selector' ).empty ( ) ;
```

سادساً :- دوال التعامل مع عناصر النماذج Form Elements

◆ الدالة val

تستخدم لقراءة القيم من عناصر النموذج وهي اختصار لكلمة value , والصيغة العامة لها هي

```
$ ( 'selector' ).val ( ) ;
```

تعيد هذه الدالة قيمة وحيدة في حال كان المحدد selector يعيد عنصراً واحداً وهي قيمة ذلك العنصر أما في حال كان المحدد يعيد أكثر من عنصر فإن الدالة val تعيد قيمة أول عنصر من هذه العناصر , والمقصود بالقيمة هنا هي القيمة التي تحملها الخاصية value

ملاحظة/ إن الدالة val في حال استدعائها على عنصر تحديد يسمح باختيار أكثر من خيار في وقت واحد فإن الدالة val تعيد مصفوفة تمثل الخيارات المنتقاة من قبل المستخدم .

مثال/ يعيد قيمة الخاصية value للعنصر ذو المعرف firstName

```
$ ( '#firstName' ).val ( ) ;
```

ملاحظة/ إذا كان أول عنصر من المجموعة التي يعيدها المحدد ليس عنصر نموذج فإن خطأ برمجي سيظهر في مستند الويب .

ملاحظة/ الدالة val تعيد قيمة الخاصية value لعناصر صناديق الاختيار checkbox و button و radio بغض النظر عن كون هذه العناصر في حالة تحديد selected او لا ولكن حلاً لهذه المشكلة يتوفر بسهولة بعد مراجعة جدول المحددات وكتابة شيء شبيه بما يلي

```
$ ( '[ name=radioGroup ]:selected' ).val( ) ;
```

* للدالة val استخدام آخر غير قراءة القيم من عناصر النموذج حيث يمكن إسناد القيم لها وفي هذه الحالة ستكون الصيغة العامة لها بهذا الشكل

```
$ ( 'selector' ).val( 'القيمة' ) ;
```

مثال/

```
$ ( '#text' ).val( 'Hello' ) ;
```

وفي هذا المثال سيجعل العنصر ذو المعرف text يحمل الخاصية value وتكون قيمتها تساوي Hello

كما تستخدم الدالة val للقيام بتغيير حالة عناصر button و radio و checkbox وعناصر الوسم <select> إلى حالة التحديد selected , فالجملة التالية مثلا

```
$ ( 'input' ).val( [ 'ahmed' , 'jQuery' , 'JavaScript' ] ) ;
```

تجعل الحالة selected حالة لجميع عناصر النموذج التي تحمل الخاصية value وتكون هذه الخاصية مساوية لأحد هذه القيم ahmed او jQuery او JavaScript .

مثال/

```
<html>
<head>
<script  src="jquery.js"  type="text/JavaScript">
</script>
<script  type="text/JavaScript">
$ ( document ).read( function ( ) {
$ ( '*' ).val( [ 'Ahmed' , 'jQuery' , 'JavaScript' ] ) ;
} ) ;
</script>
</head>
<body>
<select  id="select1">
<option  value="x"> Ahmed </option>
<option  value="x"> jQuery </option>
<option> JavaScript </option>
</select>
</body>
</html>
```

الفصل الثالث

(الأحداث Events)

دوال الأحداث في jQuery مشابهه للأحداث المستخدمة مع JavaScript مع اختلاف بسيط وهو مثلاً إذا أردنا استجابة الحدث لضغط زر ما فبدلا من أن نكتب onclick سنكتب click أي فقط نحذف on وهذا ينطبق على جميع دوال الأحداث الأخرى , والجدول التالي يوضح هذه الأحداث

* دوال الأحداث الخاصة بالنافذة (المستند ككل)

اسم دالة الحدث	لحظة تفجير الحدث
load	عند تحميل المستند
unload	عند غلق المستند

* دوال الأحداث الخاصة بعناصر النموذج

اسم دالة الحدث	لحظة تفجير الحدث
focusin او focus	عند انتقال التركيز إلى العنصر (مثلاً في حالة الانتقال بين مجموعة من حقول الإدخال فإن الحدث focus يقع على حقل الإدخال التالي بمجرد الانتقال إليه عن طريق مفتاح الجدول Tab مثلاً)
focusout	عند ابتعاد التركيز عن العنصر (عكس الدالة السابقة)
blur	عند خسارة العنصر للتركيز وهو معاكس تماماً للحدث focus
change	عند حدوث تغيير على العنصر
submit	عند إرسال القيم إلى الخادم

select	عند اختيار بند من محتويات العنصر بمعنى تحديد او تظليل النص او العنصر (نرى هذا الحدث في مثل حالة العنصر <select>)
scroll	عند تحريك أشرطة التمرير (نرى هذا الحدث في مثل حالة العنصر textarea)

ملاحظة/ focusin هي نفس focus لكن يفضل استخدام focusin لأنها تتوافق مع متصفحات أكثر , وكذلك فإن focusout هي نفس blur لكن يفضل استخدام focusout لأنها تتوافق مع متصفحات أكثر .

* الحدث submit عند استخدامه يجب ان نحدد الـ form وليس زر الإرسال , لاحظ هذا المثال:-

```
$("#myform").submit(function( ) {
        alert("yes");
}
```

* دوال الأحداث الخاصة بلوحة المفاتيح

اسم دالة الحدث	لحظة تفجير الحدث
keydown	عند الضغط على مفتاح من لوحة المفاتيح
keypress	بعد الضغط على مفتاح من لوحة المفاتيح وتحريره
keyup	عند تحرير مفتاح من لوحة المفاتيح

* دوال الأحداث الخاصة بأزرار الفأرة

اسم دالة الحدث	لحظة تفجير الحدث
click	عند النقر المفرد
dblclick	عند النقر المزدوج
mousedown	عند الضغط على زر الفأرة (أثناء الضغط)
mouseup	عند تحرير زر الفأرة (الانتهاء من الضغط)
mousemove	عند مرور مؤشر الفأرة (وكل حركة فوق العنصر)
hover	عند مرور مؤشر الفأرة على العنصر
mouseenter	عند مرور مؤشر الفأرة على العنصر
mouseover	عند دخول مؤشر الفأرة
mouseout	عند خروج مؤشر الفأرة (يقصد بدخول مؤشر الفأرة أول مرور له فوق العنصر ويقصد بالخروج لحظة ابتعاده عنه)
mouseleave	عند خروج مؤشر الفأرة من العنصر

* دوال الأحداث الأخرى

اسم دالة الحدث	لحظة تفجير الحدث
error	عند حدوث خطأ
resize	عند تغيير الحجم

جميع الدوال في الجدول أعله لها الشكل العام التالي

```
$ ( 'selector' ).event( function ) ;
```

حيث تقوم بتعيين الدالة function استجابة للحدث event لجميع العناصر التي يعيدها المحدد selector .

مثال/

```
<html>
<head>
<script type="text/JavaScript" src="jquery.js">
</script>
<script type="text/JavaScript">
$ ( document ).ready( function ( ) {
$ ( '#Button1' ).click( function ( ) {
alert( "تم النقر على الزر" ) ;
} ) ;
$ ( 'img' ).mousemove ( function ( ) {
alert ( "تم مرور المؤشر فوق الصورة" ) ;
} ) ;
} ) ;
</script>
</head>
<body>
<input id="Button1" type="button" />
<img src="mypic.gif" />
</body>
</html>
```

كما لاحظت في هذا المثال فإننا لا نكتب او لا نستدعي الحدث من داخل الوسوم الخاصة به بل تكون الكتابة فقط في جزء كود jQuery , ويمكن تطبيق أي حدث من هذه الأحداث بنفس الطريقة لهذا المثال لا يوجد فرق .

* كما تعرف أن الحدث hover يستخدم ليدل إذا أصبح مؤشر الفأرة فوق العنصر المحدد نفذ هذا التأثير لكن إذا أردناه أن يطبق تأثير معين عند ابتعاد مؤشر الفأرة من العنصر يمكن أن نسند له function آخر وبداخله نضع تأثير ابتعاد المؤشر

مثال/

```
$ ( '.my' ).hover( function( ) {
$ ( '.my' ).fadeTo( 200 , 1 ) ;
} , function( ) {
$ ( '.my' ).fadeTo( 400 , 0.3 ) ;
} ) ;
```

◆ الحدث toggle

يستخدم هذا الحدث عندما نريد تطبيق تأثير معين عند الضغط بزر الفأرة عليه وعند الضغط مرة أخرى يتم تنفيذ تأثير آخر وعند الضغط مجدداً سيتم تنفيذ تأثير ثالث وهكذا نضيف التأثيرات بعدد ما نشاء حيث سنضيف كل تأثير بـ function جديد توضع الـ functions بين قوسي الحدث ونفصل بين الواحدة والأخرى بفارزة .

مثال/

```
$ ( '#test' ).toggle(
function( ) {
$ ( this ).css( "color","red" ) ;
} ,
function( ) {
$ ( this ).css( "color","blue" ) ;
} ) ;
```

◆ الدالة bind

تستخدم لتحديد دالة معينة كاستجابة لحدث محدد ولها الشكل العام التالي

```
$ ( 'selector' ).bind( 'اسم الحدث' , function ) ;
```

مثال/ لاحظ أن هاتان التعليمتان تقومان بنفس العمل تماماً

```
$ ( 'img' ).click( myFunction ) ;

$ ( 'img' ).bind( 'click' , 'myFunction' ) ;
```

عمل الدالة bind على نفس العنصر أكثر من مرة يؤدي إلى تنفيذ جميع الدوال التي تربطها بالعنصر عند تفجير الحدث وهذا ممكن أيضاً باستخدام الدوال البسيطة لذا فإن التعليمة التالية

```
$ ('img' ).bind( 'click' , F1 ).bind( 'click' , F2 ).bind( 'click' , F3 )
```

تقوم باستدعاء الدوال F1 , F2 , F3 بالترتيب عند تفجير حدث النقر الخاص بعناصر الصور في المستند وإن التعليمة التالية تقوم بالمثل أيضاً

```
$ ( 'img' ).click( F1 ) .click( F2 ) .click( F3 )
```

◆ الدالة unbind

هذه الدالة هي نقيضه الدالة bind حيث أنها تعمل على إلغاء ربط أحد الدوال بحدث معين ولها نفس الشكل العام الخاص بالدالة bind لذا فإن التعليمتين التاليتين

```
$ ( 'img' ).bind( 'click' , F1 ) .bind( 'click' , F2 ) .bind( 'click' , F3 ) ;

$ ( 'img' ).unbind( 'click' , F2 ) ;
```

ستقومان باستدعاء الدالتين F1 و F3 عند تفجير حدث النقر الخاص بعناصر الصور في المستند دون تطبيق استدعاء الدالة F2 وذلك لأن الدالة unbind قامت بإلغاء ربطها بالحدث .

◆ الدالة one

هي تماماً نفس الدالة bind لكن الفرق بينهما هو أن الدالة bind تستدعي مجموعة من الدوال عند تفجير حدث واحد فقط بينما تستطيع الدالة one أن تستدعي مجموعة من الدوال عند تفجير مجموعة من الأحداث . لذا فإن التعليمة التالية

```
$ ( 'img' ).one( "click , mousemove , dblclick" , 'myFunction' ) ;
```

تقوم باستدعاء الدالة myFunction عند تفجير أي من الأحداث dblclick او mousemove او click

◆ الكائن event

يكتب هذا الكائن اختصاراً e وهو يمكن أن يكتب كوسيط أخياري داخل قوسي أي واحدة من دوال الأحداث , حيث أن هذا الوسيط (e) يحمل معلومات هامة تخص الحدث .

مثال/

```
$ ( 'a' ).click( function ( e ) {
alert ( "Hello" ) ;
} ) ;
```

لا يختلف عمل الأسطر أعلاه عن عملها بدون تمرير الوسيط e أبداً فهي تقوم بعرض الرسالة Hello عند النقر على أي رابط في الصفحة ولكن ما يميزها هو أن وسيطها e يحمل قيماً خاصة تمثل معلومات عن الحدث click يمكن الاستفادة منها في جسم الدالة فمثلاً يمكن أن نكتب

```
$ ( 'a' ).click( function ( e ) {
alert ( e.type ) ;
} ) ;
```

وستكون نتيجة النقر على الروابط هي رسالة تحتوي على الكلمة click وهي هنا قيمة الخاصية type .

ملاحظة/ على الرغم من أن الصيغة ذاتها تستخدم لتعرف الكائن e في جميع دوال الأحداث إلا أن خاصية الكائن e تختلف من حدث لآخر .

* هذا الجدول فيه أهم خواص الكائن e

الخاصية	المعلومات التي تحتويها
altKey	تحتوي على القيمة true في حال كون المفتاح alt مضغوطاً لحظة تفجير الحدث وتحتوي false في الحالة المعاكسة
ctrlKey	تحتوي على القيمة true في حال كون المفتاح ctrl مضغوطاً لحظة تفجير الحدث وتحتوي false في الحالة المعاكسة
keyCode	تتعلق بالحدثين keyup و keydown وتعيد هذه الخاصية قيمة ASCII (الترميز الستعشري) الخاص بالمفتاح المضغوط (وفي حالة الضغط على حرف ما فإنها تعيد قيمة الحرف الكبير -كابيتل لتر- او upper case فمثلاً هذه الخاصية ستحتوي القيمة 65 في حالة ضغط المفتاح a او A)
pageX	تتعلق بأحداث الفأرة وتعيد الإحداثيات الأفقية X الخاصة بمؤشر الفأرة بالنسبة للصفحة (بعد مؤشر الفأرة عن حافة الصفحة اليسرى عادةً)
pageY	تتعلق بأحداث الفأرة وتعيد الإحداثيات العمودية Y الخاصة بمؤشر الفأرة بالنسبة للصفحة (بعد مؤشر الفأرة عن حافة الصفحة العلوية عادةً)
screenX	تتعلق بأحداث الفأرة وتعيد الإحداثيات الأفقية X الخاصة بمؤشر الفأرة بالنسبة للشاشة (بعد مؤشر الفأرة عن حافة الشاشة اليسرى عادةً)
screenY	تتعلق بأحداث الفأرة وتعيد الإحداثيات العمودية Y الخاصة بمؤشر الفأرة بالنسبة للشاشة (بعد مؤشر الفأرة عن حافة الشاشة العلوية عادةً)
relatedTarget	تتعلق ببعض أحداث الفأرة وتعيد معرف العنصر الذي دخل إليه / خرج عنه مؤشر الفأرة لحظة تفجير الحدث

shiftKey	تحتوي القيمة true في حال كون المفتاح shift مضغوطاً لحظة تفجير الحدث وتحتوي false في الحالة المعاكسة
type	تستعمل مع جميع الأحداث وتعيد اسم الحدث
which	بالنسبة لأحداث لوحة المفاتيح تعيد هذه الخاصية قيمة ASCII (الترميز الستعشري) الخاص بالمفتاح المضغوط , أما بالنسبة لأحداث الفأرة تعيد رقماً يمثل زر الفأرة المضغوط حيث أن الرقم 1 يمثل زر الفأرة الأيسر والرقم 2 يمثل زر الفأرة الأيمن

◆ الدالة preventDefault

هذه الدالة خاصة بالكائن e وهي تسمح بإلغاء الاستجابة الافتراضية لحدث ما , مثلاً عند النقر على رابط سينقلنا مباشرتاً إلى عنوان الصفحة المذكور لكن بواسطة هذه الدالة يمكن إلغاء الانتقال هذا

مثال/

```
$ ( 'a' ).click( function ( e ) {
e.preventDefault ( ) ;
} ) ;
```

وفي هذه الحالة لن يتم الانتقال إلى أي مكان عند النقر على أي رابط في الصفحة .

الفصل الرابع

(الحركات Animations)

المؤثرات البسيطة

◆ الدالة show

تستخدم هذه الدالة لإظهار العناصر المخفية والصيغة العامة لها هي

```
$ ( 'selector' ).show( 'speed' , 'F' ) ;
```

ويمكن عدم تمرير أي بارامتر بين قوسيها , او يمكن تمرير بارامتر واحد فقط .

* إن speed تعني السرعة ويمكن كتابة السرعة بطريقتين

1- صيغة نصية (توضع بين علامات تنصيص) :

أ- slow للسرعة البطيئة .

ب- normal للسرعة الطبيعية .

ج - fast للسرعة السريعة .

2- صيغة رقمية (لا توضع بين علامات تنصيص) تمثل زمن تنفيذ المؤثر بالملي ثانية , كل 1 ثانية يساوي 1000 ملي ثانية .

وأن F تعني اسم دالة أخرى موجودة في المستند يتم تنفيذها بعد انتهاء هذا المؤثر .

◆ الدالة hide

تستخدم هذه الدالة لإخفاء العناصر والصيغة العامة لها هي

```
$ ( 'selector' ).hide( 'speed' , 'F' ) ;
```

ويمكن عدم تمرير أي باراميتر بين قوسيها , او يمكن تمرير باراميتر واحد فقط .

* إن speed تعني السرعة ويمكن كتابة السرعة بطريقتين
1- صيغة نصية (توضع بين علامات تنصيص) :
أ- slow للسرعة البطيئة .
ب- normal للسرعة الطبيعية .
ج - fast للسرعة السريعة .
2- صيغة رقمية (لا توضع بين علامات تنصيص) تمثل زمن تنفيذ المؤثر بالملي ثانية , كل 1 ثانية يساوي 1000 ملي ثانية .

وأن F تعني اسم دالة أخرى موجودة في المستند يتم تنفيذها بعد انتهاء هذا المؤثر .

◆ الدالة toggle

تستخدم هذه الدالة لقلب العناصر حيث تقوم بإظهار العناصر إن كانت مخفية وإخفائها إن كانت ظاهرة والصيغة العامة لها هي

```
$ ( 'selector' ).toggle( 'speed' , 'F' ) ;
```

* إن speed تعني السرعة ويمكن كتابة السرعة بطريقتين
1- صيغة نصية (توضع بين علامات تنصيص) :
أ- slow للسرعة البطيئة .
ب- normal للسرعة الطبيعية .
ج - fast للسرعة السريعة .
2- صيغة رقمية (لا توضع بين علامات تنصيص) تمثل زمن تنفيذ المؤثر بالملي ثانية , كل 1 ثانية يساوي 1000 ملي ثانية .

وأن F تعني اسم دالة أخرى موجودة في المستند يتم تنفيذها بعد انتهاء هذا المؤثر .

مؤثرات التلاشي

◆ الدالة fadeIn

تستخدم هذه الدالة للظهور المتلاشي (أي أن يكون العنصر مخفي وهذه الدالة تظهره) والصيغة العامة لها هي

```
$ ( 'selector' ).fadIn( speed , 'F' ) ;
```

* إن speed تعني السرعة وتكتب بصيغة رقمية (لا توضع بين علامات تنصيص) تمثل زمن تنفيذ المؤثر بالملي ثانية , كل 1 ثانية يساوي 1000 ملي ثانية .
وأن F تعني اسم دالة أخرى موجودة في المستند يتم تنفيذها بعد انتهاء هذا المؤثر .

◆ الدالة fadeOut

تستخدم هذه الدالة للإخفاء المتلاشي والصيغة العامة لها هي

```
$ ( 'selector' ).fadeOut( speed , 'F' ) ;
```

* إن speed تعني السرعة وتكتب بصيغة رقمية (لا توضع بين علامات تنصيص) تمثل زمن تنفيذ المؤثر بالملي ثانية , كل 1 ثانية يساوي 1000 ملي ثانية .
وأن F تعني اسم دالة أخرى موجودة في المستند يتم تنفيذها بعد انتهاء هذا المؤثر .

◆ الدالة fadeTo

تستخدم هذه الدالة أيضاً للإخفاء المتلاشي لكن من خلال هذه الدالة يمكن تحديد الشفافية والصيغة العامة لها هي

```
$ ( 'selector' ).fadeTo( 'speed' , 'opacity'  , 'F' ) ;
```

* إن speed تعني السرعة ويمكن كتابة السرعة بطريقتين
1- صيغة نصية (توضع بين علامات تنصيص) :
أ- slow للسرعة البطيئة .
ب- normal للسرعة الطبيعية .
ج - fast للسرعة السريعة .
2- صيغة رقمية (لا توضع بين علامات تنصيص) تمثل زمن تنفيذ المؤثر بالملي ثانية , كل 1 ثانية يساوي 1000 ملي ثانية .

وأن opacity تمثل درجة الشفافية للعنصر التي ينتقل إليها أثناء تنفيذ المؤثر ويستقر عليها بعد انتهاء تنفيذ المؤثر ويمكن أن تكون درجة الشفافية بين 0 (شفافة لدرجة عدم الظهور) و 1 (ظهور بشكل طبيعي) والدرجات الأخرى تكون بشكل أرقام عشرية بين الصفر والواحد , وأن F تعني اسم دالة أخرى موجودة في المستند يتم تنفيذها بعد انتهاء هذا المؤثر وهذا الباراميتر

اختياري أي يمكن أن لا نضعه بخلاف البارامترين السابقين فهما إجباريان أي لا تعمل الدالة إذا لم نضعهم .

ملاحظة/ الدالة fadeTo لا تقوم بحذف العناصر بعد انتهاء الحركة كما تفعل الدالة fadeOut .

مثال/

```
$ ( '#div' ).fadeTo( 4000 , 0.4 ) ;
```

مؤثرات الانزلاق

◆ الدالة slideDown

تستخدم هذه الدالة لتطبيق تأثير الانزلاق السفلي (أي أن يكون العنصر مخفي وتظهره هذه الدالة بالانزلاق السفلي) والصيغة العامة لها هي

```
$ ( 'selector' ).slideDown( speed , 'F' ) ;
```

* إن speed تعني السرعة وتكتب بصيغة رقمية (لا توضع بين علامات تنصيص) تمثل زمن تنفيذ المؤثر بالملي ثانية , كل 1 ثانية يساوي 1000 ملي ثانية .
وأن F تعني اسم دالة أخرى موجودة في المستند يتم تنفيذها بعد انتهاء هذا المؤثر .

◆ الدالة slideUp

تستخدم هذه الدالة لتطبيق تأثير الانزلاق العلوي (أي أن يكون العنصر ظاهر وتخفيه هذه الدالة بالانزلاق العلوي) والصيغة العامة لها هي

```
$ ( 'selector' ).slideUp( speed , 'F' ) ;
```

* إن speed تعني السرعة وتكتب بصيغة رقمية (لا توضع بين علامات تنصيص) تمثل زمن تنفيذ المؤثر بالملي ثانية , كل 1 ثانية يساوي 1000 ملي ثانية .
وأن F تعني اسم دالة أخرى موجودة في المستند يتم تنفيذها بعد انتهاء هذا المؤثر .

◆ الدالة slideToggle

وهي تستخدم كدالة قلب للدالتين السابقتين (أي بمعنى دمج الدالتين) حيث أنها تقوم بإخفاء العناصر الظاهرة باستخدام الدالة slideUp وإظهار العناصر المخفية باستخدام الدالة slideDown , والصيغة العامة لها هي

```
$ ( 'selector' ).slideToggle( speed , 'F' ) ;
```

* إن speed تعني السرعة وتكتب بصيغة رقمية (لا توضع بين علامات تنصيص) تمثل زمن تنفيذ المؤثر بالملي ثانية , كل 1 ثانية يساوي 1000 ملي ثانية .
وأن F تعني اسم دالة أخرى موجودة في المستند يتم تنفيذها بعد انتهاء هذا المؤثر .

إيقاف الحركة

◆ الدالة stop

تستخدم هذه الدالة لإيقاف تنفيذ أي حركة قيد التنفيذ والصيغة العامة لها هي

```
$ ( 'selector' ).stop( ) ;
```

◆ الدالة delay

تستخدم هذه الدالة لإيقاف تنفيذ تأثير حركة او أي مؤثر فترة زمنية معينة ثم يمكن أن يعود لينفذ تأثير مؤثر آخر والصيغة العامة لها هي

```
$ ( 'selector' ).delay( speed ) ;
```

* إن speed تعني السرعة وتكتب بصيغة رقمية (لا توضع بين علامات تنصيص) وهي تمثل زمن توقف المؤثر عن العمل .

مثال/

```
$ ( '#div' ).slideDown( 3000 ).delay( 4000 ).slideUp( 3000 ) ;
```

إنشاء حركات خاصة

◆ الدالة animate

يمكن إنشاء حركات خاصة بنا غير تلك الافتراضية التي تقدمها المكتبة وذلك من خلال الدالة animate والصيغة العامة لها هي

```
$ ( 'selector' ).animate( { properties } , 'speed' ) ;
```

وفكرة هذه الدالة هي الانتقال من الشكل الطبيعي للعنصر إلى الشكل الجديد الذي يحمل الخصائص properties , حيث أن هذه الخصائص properties يمكن أن تحمل خصائص الأنماط الخاصة بالعنصر والتي تأخذ قيماً رقمية فقط في العادة مثل height و opacity و top و size-font .. الخ , او أن تأخذ أسماء إحدى دوال المكتبة الجاهزة الخاصة بالحركة مثل show و toggle و hide ... الخ .

* إن speed تعني السرعة ويمكن كتابة السرعة بطريقتين

1- صيغة نصية (توضع بين علامات تنصيص) :

أ- slow للسرعة البطيئة .

ب- normal للسرعة الطبيعية .

ج - fast للسرعة السريعة .

2- صيغة رقمية (لا توضع بين علامات تنصيص) تمثل زمن تنفيذ المؤثر بالملي ثانية , كل 1 ثانية يساوي 1000 ملي ثانية .

مثال/

```
$ ( '#div' ).animate( {
"width" : "300px" ,
 "height" : "200px" ,
"margin-top" : "150px"
"padding-right" : "20px"
```

```
} , 1000 ) ;
```

مثال/ لتكبير حجم العنصر ثلاثة أضعاف

```
$ ( 'اسم العنصر المحدد' ).animate(
{ width: $( this ).width ( ) * 3 , height: $( this ).height ( ) * 3 } , 'slow' ) ;
```

* يمكن أن نستخدم أحد العلامتين + او - ونضع بعدهم علامة = لتدل على أن يتم جمع او إنقاص القيمة القديمة وإضافة القيمة الجديدة عليها .

مثال/

```
<html>
<head>
<script  type="text/JavaScript"  src="jquery.js"> </script>
<script  type="text/JavaScript">
$ ( document ).ready( function ( ) {
        $ ( '#box' ).animate( { left :"+=50px" } , 2000 ) ;
} ) ;
</script>
<style  type="text/css">
#box {
position:absolute; left:2px; top:2px ;
border:1px solid #ccc ;
height:200px ;  width:300px ;
}
</style>
</head>
<body>
<div  id="box"> Hi </div>
</body>
</html>
```

* وكما لاحظت في المثال عندما نستخدم حركة عن جهة معينة (مثلاً استخدمنا هنا left او أي اتجاه آخر) فيجب أن تكون خاصية position موجودة للعنصر في الـ CSS .

الفصل الخامس

(التخاطب مع الخادم عبر تقنية AJAX)

جلب المحتويات من الخادم

◆ الدالة load

تستخدم هذه الدالة لتأمين إرسال الطلبات غير المتزامنة للخادم server والصيغة العامة لها هي

```
$ ( 'selector' ).load( url , Json , F ) ;
```

حيث أن الوسيط url يمثل اسم الصفحة التي سيرسل إليها الطلب في الخادم في حين أن الوسيط الاختياري Json هو كائن يمثل مجموعة من الوسطاء التي سترسل إلى الصفحة ذاتها في الخادم أما الوسيط الاختياري F فهو يمثل اسماً لدالة موجودة في الصفحة تستدعى لحظة انتهاء تنفيذ الطلب وبعد قدوم الاستجابة من الخادم ويمكن تمرير استجابة الخادم كوسيط لهذه الدالة .

ملاحظة/ الدالة load تقوم باستدعاء محتوى الكائنات التي يعيدها المحدد selector بالمحتوى الذي سيعود من الخادم server نتيجة تنفيذ الطلب على شكل HTML (في الحقيقة ستعود الصفحة المطلوبة كاملة) .

مثال/

```
$ ( '#divA' ).load( 'mypage.php  a[ href=www.aw.com ]' ) ;
```

حيث تقوم الشيفرة السابقة باستبدال محتوى الوسم ذو المعرف divA بالمحتوى الذي يطابق المحدد a[href=www.aw.com] والذي ستعيده الدالة load من الصفحة mypage.php .

كمثال على تمرير وسطاء للصفحة الهدف في الخادم لنفرض أن الصفحة التي نريد استعادة قيم من محتواها تملك العنوان التالي

/mypage.php?p1=v1&p2=v2

عند إذ يمكننا أن نكتب الشيفرة التالية

```
$ ( '#divA' ).load( '/mypage.php' , { p1:'v1' , p2:'v2' }) ;
```

حيث أن الوسيط الثاني هنا هو عبارة عن كائن JSON يمثل وسطاء الصفحة الهدف (أي سترسل هذه المعلومات إلى الصفحة الهدف) .

ملاحظة/ مستعرض الإنترنت إكسبلورر يقوم بتخزين نسخة مؤقتة من الصفحات المطلوبة cache لذا إن كانت الصفحة التي تنوي جلب المعلومات منها متجددة بشكل سريع مثل صفحات أسعار العملات مثلاً عندها أحرص على تزويد وسيط ذا قيمة تتغير عشوائياً للدالة لأن هذا يجعل المستعرض سالف الذكر يعتبر كل طلب منها صفحة مستقلة عن الأخرى وبالتالي لن يؤثر عمل cache لإحدى الصفحات على عمل المتبقي منهن .

مثال/

```
$ ( '#div' ).load( "test.php") ;
```

حيث سيقوم هذا المثال بجلب محتويات الصفحة المسماة test.php ويعرضها في الوسم ذو المعرف div , يمكننا أن نعرض جزء محدد من الصفحة test.php وذلك بكتابة محدد للعنصر الذي سيظهر بعد اسم الصفحة

مثال/

```
$ ( '#div' ).load( "test.php  #id") ;
```

لكن لو افترضنا أن هذه الصفحة غير موجودة ففي هذه الحالة سوف لن يعرض لنا أي شيء لكن إذا أردناه أن يعرض لنا الخطأ فيمكن أن نضيف له function ونسند لها ثلاثة بارامترات وبالترتيب حيث أن الأول يعبر عن محتوى الملف الذي سنجلبه وهو responsText والثاني يعبر عن حالة الإرسال (تمت بنجاح او لا) وهو status والثالث يعبر عن الكائن الخاص بال AJAX وهو xmlHttpRequest , ويمكن التعبير عن كل باراميتر بأي اسم , حيث يمكن الاستفادة من هذه البارامترات من خلال أداة شرط

مثال/

```
$ ( '#div' ).load( "test.php" , function( r , s , xhr ) {
if ( s == 'error' ) {
$ ( this ).text( "Error File Not Found" ) ; }
} ) ;
```

في هذا المثال سيتم عرض العبارة Error File Not Found إذا تحقق الشرط وهو وجود خطأ وكما لاحظت هنا استخدمنا البارامتر الثاني وهو status لكن يمكن أن نستخدم البارامتر الثالث وهو xmlHttpRequest حيث إذا كتبنا بعده نقطة ثم status فإنه سيعبر عن رقم الخطأ أما إذا كتبنا بعده نقطة ثم statusText فإنه سيعبر عن الخطأ لكن بشكل نصي , لاحظ هذا المثال

```
$ ( '#div' ).load( "test.php" , function( r , s , xhr ) {
if ( s == 'error' ) {
$ ( this ).text( "Error Number" + xhr.status + "  " + xhr.statusText  ) ; }
} ) ;
```

في هذا المثال سيعرض رقم الخطأ و تعبير نصي عن الخطأ , البارامتر الأول كما قلنا فهو يعبر عن محتوى الملف الذي سنجلبه فإن كان موجود سيعرض في المكان الذي حددناه له لكن إن لم يجد الصفحة او المحتوى فإنه سيعرض رسالة الخطأ ولإظهار هذه الرسالة نستخدم الدالة html ولا يصح أن نستخدم الدالة text كما فعلنا في المثال السابق لوجود وسوم html وبالتالي يمكن تغيير المثال السابق لهذا الشكل

```
$ ( '#div' ).load( "test.php" , function( r , s , xhr ) {
if ( s == 'error' ) {
$ ( this ).html( r ) ; }
} ) ;
```

تمرير وسطاء للخادم من حقول النموذج

الدالة serialize

تستخدم هذه الدالة احياناً مع الدالة السابقة load حيث تمرر لها قيماً من أحد نماذج الصفحة Form كوسطاء للصفحة الهدف فمثلاً لاستدعاء الصفحة myPage.php مع وسطاء قيمهم قادمة من النموذج ذو المعرف myForm يمكن أن نكتب ما يلي :

```
$ ( '#divA' ).load( '/myPage.php' , $ ( '#myForm' ).serialize ( ) ) ;
```

وستتولى الدالة serialize في هذا المثال توليد كائن JSON المناسب للدالة load والذي سيمثل وسطاء الصفحة الهدف myPage.php .

ملاحظة/ في حال تزويد قيمة للوسيط الاختياري JSON ستقوم الدالة بإرسال الطلب بالطريقة POST وفي حال عدم تزويد قيمة لذات الوسيط ستقوم الدالة بإرسال الطلب بالطريقة GET.

مثال/ نكتب في صفحة النموذج الكود التالي

```
<html>
<head>
<script type="text/ecmascript"  src="jquery.js"> </script>
<script type="text/javascript">
```

```
$( document ).ready( function( ) {
$( '#myForm' ).submit( function( e )  {
var send = $( this ).serialize( ) ;
$.ajax( {
url:"1.php" ,
type:"POST" ,
data: send ,
dataType:"json" ,
success: function( r , s , xhr )  {
$( '#div' ).html( r.name + "<br>" + r.comment ) ;
}
} ) ;
return  false ;
} ) ;
} ) ;
</script>
</head>
<body>
<form id="myForm" method="post">
Name: <input type="text" name="name" /> <br>
Comment: <textarea name="comment"></textarea> <br>
<input type="submit" value="Submit Comment" />
</form>
<div id="div"> </div>
</body>
</html>
```

ونكتب في صفحة الـ php التي ستعالج البيانات الكود التالي

```
<?php
$n = $_REQUEST['name'];
$c = $_POST['comment'];
$arr = array('name'=>$n , 'comment'=>$c);
echo json_encode($arr);
?>
```

إرسال طلبات من النوع GET

تسمح لنا مكتبة jQuery بإرسال طلبات غير متزامنة من نوع محدد من النوعين GET و POST عبر مجموعة من الدوال منها :-

- الدالة **$.get**

تقوم هذه الدالة بإرسال طلب غير متزامن من النوع GET (تستخدم هذه الدالة في الغالب لإرسال البيانات القليلة مثل عنوان صفحة او ما شابه) إلى الخادم وتعيد كائناً يمثل الاستجابة التي أعادها الخادم والصيغة العامة لها هي

```
$.get ( url , JSON , F ) ;
```

حيث أن الوسيط url يمثل اسم الصفحة التي سيرسل إليها الطلب في الخادم ويمثل الوسيط الاختياري JSON مجموعة من الوسطاء التي سترسل إلى الصفحة ذاتها في الخادم أما الوسيط الاختياري F فهو يمثل اسماً لدالة موجودة في الصفحة تستدعى لحظة انتهاء تنفيذ الطلب ويمكن تمرير استجابة الخادم كوسيط لهذه الدالة .

وهنا يجدر التنبيه أن هذه الدالة تعتبر من دوال jQuery الوظيفية (مجموعة من دوال المكتبة تستدعى مباشرتاً دون الحاجة للمحددات ويكون لهذه الدوال الصيغة التالية $.X حيث X هنا يمثل اسم الدالة) .

وكمثال لاستخدام هذه الدالة يمكن أن نكتب ما يلي

```
$.get ( '/myPage.php' , { p1:'10' , p2:'cat' } , Function( data ) { alert( data ) ; } ) ;
```

حيث أن تنفيذ هذه الشيفرة يرسل طلبا غير متزامن إلى الصفحة :

```
/myPage.php?p1=10&p2=cat
```

ويأخذ القيمة التي تعيدها هذه الصفحة ليسندها للكائن data الذي يمثل وسيطاً لدالة بسيطة تقوم بعرض قيمة هذا الوسيط في صندوق رسالة .

* يمكن تمرير ثلاث بارامترات إلى الدالة F وهذه البارامترات الثلاثة هي نفسها التي شرحناها مع الدالة load

* وكما تلاحظ فإننا لم نحدد مكان (محدد) لتعرض فيه الصفحة الهدف ولتحديد مكان يمكن أن نكتب المحدد داخل الـfunction

مثال/

```
$.get ( '/myPage.php' , { p1:'10' , p2:'cat' } , Function( d , s , xhr ) {

$('#id').html( d ) ;

} ) ;
```

في هذا المثال ستعرض الصفحة myPsge.php في المحدد #id

إرسال الطلبات من النوع POST

الدالة $.post

وهي تشبه الدالة السابقة حيث أنها ترسل طلبات غير متزامنة لكن من النوع POST (تستخدم هذه الدالة في الغالب لإرسال البيانات الكبيرة مثل التعليقات او ما شابه) إلى الخادم والصيغة العامة لها هي

```
$.post ( url , JSON , F ) ;
```

حيث أن الوسيط url يمثل اسم الصفحة في الخادم , و JSON يمثل وسطاء هذه الصفحة , و F يمثل اسماً لدالة تستدعى عند انتهاء تنفيذ الطلب ويمكن أن تمرر استجابة الخادم كوسيط لها .

إرسال الطلبات لخادم معلومات نوع الاستجابة

من الممكن أن يعيد التطبيق الموجود في الخادم استجابة في أي شكل يحدده مبرمج هذا التطبيق , فإن كنا على بينة ويقين إن التطبيق الذي نرسل إليه الطلب سيعيد استجابته بالتمثيل JSON فإن مكتبة jQuery توفر دالة خاصة لمثل هذه الحالة وهي :-

الدالة $.getJSON

وهي تشبه كثيراً الدالة الوظيفية $.get عدا أنها مصممة لاستعادة كائنات JSON والصيغة العامة لها هي

```
$.getJSON ( url , JSON , F ) ;
```

حيث أن الوسيط url يمثل الصفحة الهدف في الخادم وJSON يمثل وسطاء هذه الصفحة , أما الوسيط F فيمثل اسماً لدالة في المستند تستدعى لحظة انتهاء الطلب ويمكن أن يمرر لها وسيط يعبر عن استجابة الخادم ويكون هذا الوسيط كائناً من كائنات JSON .

الدوال الوظيفية

من الدوال الوظيفية الدوال التالية :-

الدالة $.getScript

تقوم هذه الدالة بجلب ملف JavaScript من الخادم وتنفيذه داخل الصفحة بشكل غير متزامن والصيغة العامة لها هي

```
$.getScript ( url , F ) ;
```

حيث أن الوسيط url يمثل ملف JavaScript في الخادم ويمثل الوسيط F اسماً لدالة موجودة في المستند تستدعى لحظة اكتمال الطلب .

الدالة $.ajax

من خلال هذه الدالة يمكننا التحكم الكامل بكل ما يتعلق بطلبات AJAX وتعتبر هي الدالة الأم لجميع دوال AJAX والصيغة العامة لها هي

```
$.ajax ( options ) ;
```

حيث أن الوسيط options هو عبارة عن كائن JSON يمثل مجموعة الخيارات الخاصة بالطلب والتي سيتم تنفيذه بناءاً عليها .

* الجدول التالي يوضح خيارات الدالة $.ajax

الاسم	النوع	الوصف
url	نص	رابط الصفحة التي سيرسل لها الطلب في الخادم
type	نص	طريقة إرسال الطلب GET او POST
data	كائن JSON	كائن JSON يمثل مجموعة وسطاء الصفحة التي سيرسل لها الطلب (المعلومات التي سترسل إلى الصفحة الهدف وفي الغالب تكون مع رابط الصفحه)
dataType	نص	قيمة من القيم التالية التي تعبر عن نوع البيانات التي نتوقع أن يعيدها الخادم (نوع الصفحة التي سنستقبلها) والأنواع هي : XML و JSON و ACRIPT و TEXT و html و script
timeout	رقم	يمثل الزمن الأعظمي لتنفيذ الطلب بالملي ثانية فإن لم يتم اكتمال الطلب خلال هذا الزمن فإن الدالة تقوم بإلغاء الطلب معتبرة حدوث خطأ ما
success	دالة	دالة يتم استدعائها لحظة اكتمال الطلب بنجاح ويمكن أن تأخذ ثلاث بارامترات وبالترتيب الأول هو data يمثل بيانات الملف الذي تم استدعائه والثاني هو statutsText يمثل رسالة تحمل هل تمت العملية بنجاح او لا بصيغة نصية والبارامتر الثالث هو xmlHttpRequest يمثل معامل AJAX وهذا الباراميتر إذا وضعنا بعده نقطة ثم status فإنه سيعطي رقم الحالة أما إذا وضنا بعده نقطة وثم statusText فإنه سيعطي الحالة بصيغة

		نصية , ويمكن أن نعطي لهذه البارامترات أي اسم لكن بالترتيب
error	دالة	دالة يتم استدعائها لحظة حصول خطأ ما في الطلب
complete	دالة	دالة يتم استدعائها لحظة اكتمال الطلب سواءً أتم بنجاح أم لا
beforeSend	دالة	دالة يتم استدعائها قبل إرسال الطلب تماماً
async	قيمة منطقية	عندما تسند إليها القيمة true يتم إرسال الطلب بشكل غير متزامن (أي يمكن أن تجري عمليات أخرى في الموقع أثناء معالجة الطلب وعدم التوقف إلى أن ينتهي الطلب) وهي الحالة الافتراضية ويفضل استخدامها والعكس في حالة إسناد القيمة false وهي عادتاً تستخدم إذا كان هناك ضغط على الخادم والسرعة بطيئة جداً .
processData	قيمة منطقية	بشكل افتراضي يتم ترميز البيانات المرسلة بواسطة الوسيط data إلى ترميز ملائم مستعرض الويب وهي الحالة ذاتها التي تحدث عند إسناد القيمة true لهذه الخاصية والعكس عند إسناد القيمة false
statusCode	كائن JSON	ينفذ إذا كان هناك خطأ وهو سيوقف العمليات التي بعده حيث يمكن أن نضع داخله مجموعة شروط ونطلب منه مثلاً إذا كان الخطأ الفلاني اعرض الرسالة التالية (وهنا نكتب رقم الخطأ)
contentType		تأخذ القيمة application/x-www-form-urlencoded وتكون هذه القيمة هي أفتراضية وهي لها علاقة بطريقة الإرسال .
username		تستخدم هذه الخاصية إذا طلبنا صفحة وكانت هذه الصفحة محمية باسم مستخدم وكلمة سر فهذه الخاصية ستحمل اسم المستخدم .
Password		تستخدم هذه الخاصية إذا طلبنا صفحة وكانت هذه الصفحة محمية باسم مستخدم وكلمة سر فهذه الخاصية ستحمل كلمة السر

وكمثال على استخدام هذه الدالة قد نكتب شيء مماثل لما يلي :-

```
$.ajax ( {
url : 'myPage.php' ,
type : 'GET' ,
data : { id:10&name:”ahmed” } ,
dataType : 'XML' ,
beforSend : function( ) {
$ ( '#div' ).text( "loading … " ) ;
} ,
statusCode : {
404 : function( r , s , xhr ) {
$ ( '#div' ).text( "number is"+ xhr.status+"  "+ xhr.statusText ) ; } ,
 200 : function( r , s , xhr ) {
$ ( '#div' ).text( "Ok" ) ; }
} ,
contentType : “application/x-www-form-urlencoded” ,
```

```
success : function( d , st , xhr ) {
$ ( '#div' ).text( d ) ;
$ ( '#div' ).text( st ) ;
$ ( '#div' ).text( "number is"+ xhr.status+"  "+ xhr.statusText ) ;
}
} ) ;
```

ولاحظ بأن المحدد #div الذي استخدمناه في المثال هو الذي ستعرض فيه الصفحة الهدف myPage.php

◆ الدالة $.ajaxSetup

من خلال هذه الدالة يمكن تعيين إعدادات افتراضية لكل الطلبات المرسلة عبر الدالة $.ajax والصيغة العامة لها هي

```
$.ajaxSetup ( properties ) ;
```

حيث أن الوسيط properties هو كائن JSON يمثل مجموعة الخاصيات التي ستصبح افتراضيةً للدالة $.ajax ويمكن أن يحتوي الوسيط properties الخيارات ذاتها الموضحة في الجدول السابق , وكمثال على عمله يمكن أن نكتب شيء مشابهاً لما يلي لتغيير الإعدادات الخاصة بالدالة $.ajax

```
$.ajaxSetup (
{
type : 'GET' ,
dataType : 'XML' ,
error : function ( err ) {
alert ( 'erro message is :' + msg ) ;
} ,
timeout : 10000
}
) ;
```

الأحداث الخاصة بطلبات AJAX

* الجدول التالي يوضح هذه الأحداث

اسم الحدث	لحظة التفجير
ajaxComplete	عند انتهاء أي طلب من طلبات AJAX
ajaxError	عند فشل أي طلب من طلبات AJAX
ajaxSend	قبل إرسال أي طلب من طلبات AJAX
ajaxStart	عند بداية إرسال أي طلب من طلبات AJAX
ajaxStop	عند إنتهاء تنفيذ جميع طلبات AJAX
ajaxSuccess	عند انتهاء تنفيذ أي من طلبات AJAX بنجاح

الفصل السادس

(الإضافات plugins)

الإضافة هي عبارة عن ملف JavaScript يتم تضمينه بعد المكتبة ويبنى بالاعتماد عليها لذا فهو يسمى أي اسم ويكون بالامتداد js لكن فريق عمل مكتبة jQuery ينصح بأن تكون تسمية الإضافة بهذا الشكل

jQuery.الاسم.js

لكن يمكن التسمية بدون كلمة jQuery ويكون الشكل العام لشيفرة الإضافة بهذا الشكل :-

```
(function( $ ) {
    هنا ستكون شيفرة الإضافة
} ) ( jQuery ) ;
```

وكما تعرف فأن هناك نوعين من الدوال دوال تستخدم معها المحددات ونوع آخر سميناه الدوال الوظيفية لذا إن أردنا أن ننشأ دالة من النوع الأول ستكون الإضافة بهذا الشكل

```
(function( $ ) {
$.fn.F = function ( parameters ) {
جسم الدالة
}
} ) ( jQuery ) ;
```

حيث أن F هو اسم الدالة الجديدة و parameters هي قائمة الوسطاء التي تأخذها هذه الدالة , أما عند بناء دالة وظيفية جديدة (من النوع الثاني) يكون الشكل العام للدالة هكذا

```
(function( $ ) {
$.F = function ( parameters ) {
جسم الدالة
} ;
} ) ( jQuery ) ;
```

حيث أن F هو اسم الدالة الجديدة و parameters هي قائمة الوسطاء التي تأخذها هذه الدالة .

مثال/ لكتابة إضافة من النوع الثاني

```
( function( $ ) {
$.ourMessage = function( msg ) {
var  m = msg.toString( ) ;
alert( m ) ;
} ;
} ) ( jQuery ) ;
```

في هذا المثال ستقوم الدالة بعرض رسالة معلومات تحتوي على قيمة الوسيط الممرر إليها والآن يمكن أن نستدعي هذه الدالة في صفحتنا في شيفرة jQuery كما يلي

```
<html>
<head>
<script  type="text/JavaScript"  src="jquery.js"></script>
<script  type="text/JavaScript"  src="اسم الملف.js"></script>
<script  type="text/JavaScript">
$.ourMessage( "Hello world" ) ;
</script>
</head>
<body>
</body>
</html>
```

مثال/ لكتابة إضافة من النوع الأول

```
( function( $ ) {
$.fn.Ahmed = function( spd , del ) {
return  this.hide( spd ).delay( del ).show( spd ) ;
}
} ) ( jQuery ) ;
```

كما لاحظت هنا استخدمنا this وهي تشير إلى هذا المحدد أي طبق التأثير على المحدد الذي سيتم تحديده , وكما لاحظت استخدمنا أيضاً الدالة return وهي تستخدم إذا كانت لدينا أكثر من دالة في داخل الإضافة حيث نكتبها مع الوظيفة الأساسية التي يتم عليها التنفيذ (لاحظ أنه يمكن كتابة أكثر من دالة داخل الإضافة الواحدة) , والآن يمكن أن نستدعي هذه الدالة في صفحتنا في شيفرة jQuery كما يلي

```
<html>
<head>
<script  type="text/JavaScript"  src="jquery.js"></script>
<script  type="text/JavaScript"  src="اسم الملف.js"></script>
<script  type="text/JavaScript">
```

```
$ ( 'a' ).click( function( ) {
$ ( this ).Ahmed( 2000 , 1000 ) ;
} ) ;
</script>
</head>
<body>
<a >Example</a>
</body>
</html>
```

* يمكننا أن نكون الإضافات وإعطاء قيم افتراضية تنفذ إذا لم ندخل للدالة قيم وإذا أعطيناها قيم ستطيق القيم الجديدة وذلك من خلال استخدام الكائن JSON

مثال/ يكتب داخل ملف الإضافة

```
( function( $ ) {
$.fn.Ahmed = function( op ) {
var  set = {
        "HS" : 2000 ,
        "DS" : 1000'
        "SS" : 2000
} ;
$.extend( set , op ) ;
this.hide( set['HS'] ).delay( set['DS'] ).show( set['SS'] ) ;
}
} ) ( jQuery ) ;
```

كما لاحظت كتبنا المتغير وسميناه set وهو يعبر عن الكائن JAON (ويمكن إعطائه أي اسم) ووضعنا في داخله الخصائص وبجانبها القيم الافتراضية وكما لاحظت وضعنا بين قوسي الدالة باراميتر واحد وسميناه op (ويمكن إعطائه أي اسم) وللربط بين هذا البارميتر والمتغير set استخدمنا الدالة extend حيث أنها تأخذ بارامترين الأول يعبر عن الكائن JSON والثاني هو الباراميتر وبذلك ستتحول الخصائص وقيمها من الكائن إلى الباراميتر داخل الدالة وكما لاحظت كتبنا مثلا داخل الدالة hide اسم الكائن حيث نتعامل معه كمصفوفة ووضعنا بداخل قوسي المصفوفة رتبتها وهي هنا كما سميناها HS (ويمكن تسميتها بأي اسم) حيث ستأخذ hide القيمة الافتراضية المسندة لهذه الرتبة HS ونفس الكلام بالنسبة للدالتين الأخرتين delay و show , والآن يمكن أن نستدعي هذه الدالة في صفحتنا في شيفرة jQuery كما يلي

```
<html>
<head>
<script  type="text/JavaScript"  src="jquery.js"></script>
<script  type="text/JavaScript"  src="اسم الملف.js"></script>
<script  type="text/JavaScript">
$ ( 'a' ).click( function( ) {
```

```
$ ( this ).Ahmed( {
HS : 3000 ,
SS : 4000
} ) ;
} ) ;
</script>
</head>
<body>
<a >Example</a>
</body>
</html>
```

وفي هذا المثال ستأخذ الدالة delay القيمة الافتراضية وهي 1 ثانية أما الدالتين show و hide ستأخذ القيم المسندة لها هنا وهي 4 ثانية للدالة show و 3 ثانية للدالة hide , وكم لاحظت نضيف القيم للدالة بشكل كائن JSON

ملاحظات عامة

ملاحظة1 // الكائن this يستخدم للإشارة إلى العنصر الذي يكتب بداخله هذا الكائن , مثلاً إذا أردنا تطبيق تأثير معين على عنصر عند حدوث حدث ما على نفس العنصر فيمكن أن نستخدم this لاحظ هذا المثال سنكتبه بطريقتين

(1

```
$ ( '#test' ).hover( function ( ) {
$ ( '#test' ).css({ color:'#ff0000' }) ;
}) ;
```

(2

```
$ ( '#test' ).hover( function ( ) {
$ ( this ).css({ color:'#ff0000' }) ;
}) ;
```

كما لاحظت في المثال السابق لا يوجد فرق بين الطريقتين لكن لاحظ هذا المثال ستتجلى فائدة الكائن this بوضوح

مثال/

```
<html>
<head>
<script  type="text/JavaScript"  src="jquery.js"> </script>
<script  type="text/JavaScript">
$ ( document ).ready( function ( ) {
        $( 'h1' ).click( function ( ) {
        $( 'h1' ).hide( 1000 ) ;
        } ) ;
} ) ;
</script>
</head>
<body>
<h1> 1 </h1>
<h1> 2 </h1>
<h1> 3 </h1>
<h1> 4 </h1>
</body>
```

```
</html>
```

في هذا المثال إذا تم الضغط على أي رقم من هذه الأرقام (<h1>) سوف يؤدي ذلك إلى اختفاء كل الأرقام (<h1>) , لكن إذا أردنا أن يطبق تأثير الاختفاء فقط على العنصر الذي يتم الضغط علية هنا يجب أن نستخدم الكائن this لتصبح شيفرة jQuery بهذا الشكل

```
<script  type="text/JavaScript">
$ ( document ).ready( function ( ) {
        $( 'h1' ).click( function ( ) {
        $( this ).hide( 1000 ) ;
        } ) ;
} ) ;
</script>
```

ملاحظة2 // الكائن JSON هو عبارة عن تنظيم او ترتيب لمجموعة الخواص والقيم التي نريد أن نضيفها إلى دالة او إلى أي شيء حيث أنه يكتب بشكل قوسين معقوفين { } وبداخلهم نكتب الخاصية بين علامات تنصيص ثم نقطتين فوق بعض : ثم قيمتها بين علامات تنصيص ثم فارزة وبعدها نكتب الخاصية مع قيمتها كما فعلنا مع الخاصية الأولى وهكذا نستمر بإضافة القيم للخواص بعدد ما نشاء , سيصبح الشكل العام بهذا الشكل

```
{
"key1":"value1" ,
"key2":"value2" ,
"key3":"value3" ,
.
.
"key n":"value n" ,
}
```

* ويمكن أن نكتب اسم الخاصية او المفتح (key) بدون علامة تنصيص لكن يجب كتابة القيمة value بين علامات التنصيص .

ملاحظة3 // في بعض الأحيان نريد أن نستخدم في صفحتنا مكتبات JavaScript أخرى وأيضاً نريد استخدام مكتبة jQuery لكن في هذه الحالة سيحصل تضارب بين المكتبات لأننا سنعرف أكود مكتبة jQuery بالعلامة $ وهذا متغير عام تستخدمه جميع المكتبات وهنا سيحصل التضارب بين أكود المكتبات لذا سنستخدم الدالة noConflict( ) لتحويل هذه العلامة في مكتبة jQuery إلى علامة او حرف او كلمة نحن نختارها لنكتبها مكانها (وكما قلنا سابقاً إن العلامة $

تعادل كلمة jQuery) لذلك سنعرف متغير جديد ونعطيه اسم ليكون اسمه هو العلامة البديلة للعلامة السابقة $ بهذا الشكل

```
var  J = jQuery.noConflict ( ) ;
```

وبذلك سيكون الحرف J هو العلامة البديلة للعلامة $ وسنكتب الحرف J في أي مكان كنا نكتب فيه العلامة $ داخل المكتبة , بمعنى لو أردنا تحديد عنصر وإجراء تأثير ما علية لا نكتب $ بل سنكتب J بدلا عنه

```
J ( 'selector' ).function( ) ;
```

كنا سابقاً نكتب الكود التالي ونضع بداخله أكواد jQuery

```
$ ( document ).ready( function ( ) {
شيفرة جكوري هنا
} ) ;
```

وكما قلنا سابقاً فإن هذا الكود يعني أن يتم تحميل المكتبة عند ما يبدأ بتحميل الصفحة لكن الآن وبوجود مكتبة أخرى لا يصح أن نكتب هذه التعليمة بل سنلغيها وسيكون الكود التالي بدلاً عنها

```
$ ( function ( ) {
شيفرة جكوري هنا
} ;
```

لكن بما أننا عرفّنا أن الحرف J هو سيكون البديل للعلامة $ سيكون الكود السابق بهذا الشكل

```
J ( function ( ) {
شيفرة جكوري هنا
} ;
```

وبالنتيجة سيكون شكل الصفحة كاملاً بصورة عامة هكذا

```
<html>
<head>
<script  type="text/JavaScript"  src=" jquery.js"> </script>
<script  type="text/JavaScript">
var J = jQuery.noConflict ( ) ;
J ( function ( ) {
//  شيفرة جي كويري هنا
//  J ( 'selector' ).function ( ) ;
} ) ;
</head>
<body>
( أكواد هوتميل ( الجزء الظاهر من الصفحة
</body>
</html>
```

* كتب اخرى من اعداد احمد ابراهيم

1- كتاب المختصر المفيد في لغة PHP
2- كتاب المختصر المفيد في لغة HTML
3- كتاب المختصر المفيد في لغة CSS
4- كتاب المختصر المفيد في لغة JavaScript
5- تصميم قالب Wordpress
6- تعلم لغة Quick Basic