به نام خدا

آموزش JAVASCRIPT (لایه سوم از طراحی وب)

برگرفته از کتاب :

Professional JavaScript for Web Developers

By : Nicholas C.Zakas

آشنایی با مفاهیم پایه ***javascript***

کار با آرایه ها و رشته ها

آشنایی با انواع اشیاء ، متدها و خواصشان

آشنایی با مدل شی گرای مرورگر (***BOM***)

آشنایی با مدل شی گرای سند (***DOM***)

بررسی و روش کنترل رویداد ها در جاوااسکریپت

آموزش کار با فرم ها و جداول از طریق جاوااسکریپت

ارائه انواع مثال ها و نمونه کد ها

و ...

نویسنده : ***احمد بادپی***

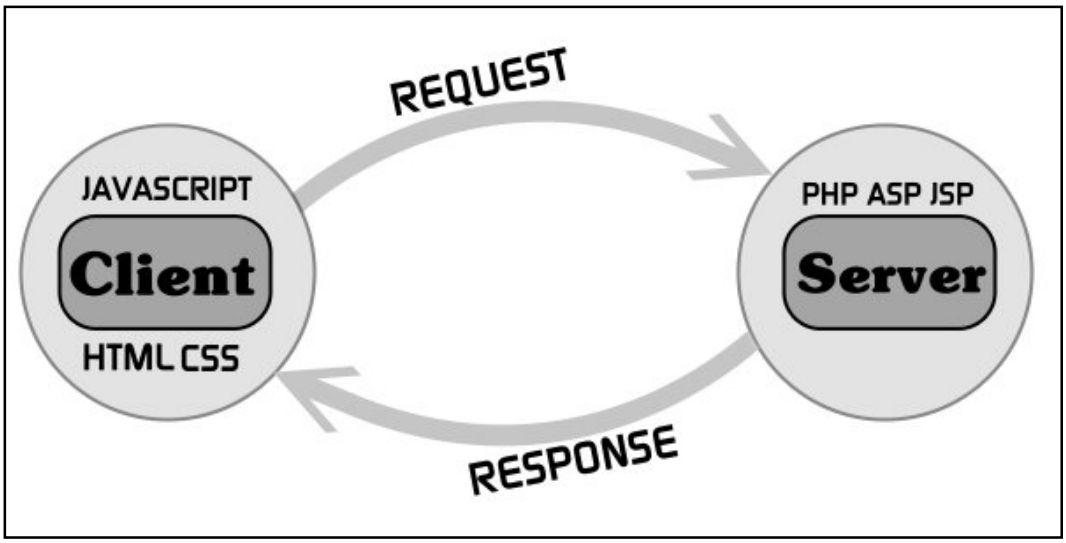
دانشگاه پیام نور مرکز آران و بیدگل

انتقادات و پیشنهادات خود را از طریق ایمیل زیر مطرح نمایید :

ahmadbadpey@gmail.com

انواع زبان های برنامه نویسی تحت وب

همانطور که می دانید کامپیوتر های موجود در شبکه اینترنت را به دو دسته اصلی تقسیم می کنند . کامپیوتر های کاربر (client) و کامپیوتر های سرور (server) . زبان های برنامه نویسی تحت وب نیز به دو دسته تحت کاربر (Client Side) و تحت سرور (server side) تقسیم بندی می شوند .



زبان های تحت کاربر زبان هایی هستند که بوسیله مرورگر و فقط بر روی client ها اجرا می شوند . در واقع برای اجرای این گونه زبان ها به سرور ها نیازی نیست . زبان هایی همچون HTML , CSS , JAVASCRIPT از این دست هستند . از این زبان ها معمولا به تنهایی برای ایجاد سایت های با محتوای ثابت که اصطلاحا به آن ها سایت های **static (ایستا)** می گویند استفاده می شود .
در مقابل این زبان ها ، زبان های تحت سرور وجود دارند که برای اجرا نیاز به سرور ها داشته و می بایست برای اجرا حتما بر روی سرور ها قرار بگیرند . اینگونه زبان ها امکان برقراری ارتباط با پایگاه داده (Database) را دارند. زبان هایی همچون PHP ، ASP و JSP از این دست هستند . از این زبان ها برای ایجاد سایت های با محتوای پویا که اصطلاحا به آن ها سایت های **dynamic (پویا)** گفته می شود استفاده می شود .

زبان **JavaScript** یکی از زبان های مهم برنامه نویسی وب و تحت کاربر (client-side) می باشد. این زبان اولین بار در سال ۱۹۹۵ ارائه شد و وظیفه آن تنها ارزش سنجی عناصر فرم بود.

Document Object Model : DOM

DOM یکی از API ها (رابط یا میانجی برنامه ی کاربردی) برای زبان های مهم HTML و XML به شمار می رود.
DOM تمام عناصر موجود در یک صفحه وب را به صورت درختی از گره ها (node) نمایش می دهد .
DOM با ترسیم درختی فرضی از عناصر موجود در یک صفحه ی وب امکان بی نظیری به طراحان وب برای کنترل بر آن ها می دهد.
گره ها با استفاده از DOM می توانند به راحتی حذف , اضافه و یا جابجا شوند .

Browser Object Model : BOM

یکی دیگر از API های ساخته شده برای HTML که به عنوان یکی از ویژگی های منحصر به فرد مرورگرهای IE و Netscape نیز شناخته می شود BOM است.

از BOM برای دسترسی و دستکاری ویژگی های پنجره یک مرورگر می توان استفاده کرد.
طراحان وب با استفاده از BOM می تواند کارهایی همچون جابجایی پنجره ها و تغییر متن موجود در نوار وضعیت مرورگر و دیگر کارهایی که ارتباط مستقیمی با قسمت Content سند ندارد انجام دهند .
معمولاً BOM با پنجره ها و فریم ها سر و کار داشته و می توان از طریق آن کارهای زیر را انجام داد :

- ☒ باز کردن پنجره های Popup .
- ☒ توانایی بازکردن پنجره های جدید و تغییر اندازه و جابجایی و یا بستن آن ها .
- ☒ بدست آوردن اطلاعاتی از مرورگر (نوع , ورژن و ...) .
- ☒ بدست آوردن اطلاعاتی در مورد سند و موقعیت صفحه ای که در مرورگر باز شده است .
- ☒ بدست آوردن اطلاعاتی در مورد وضوح (resolution) صفحه نمایش کاربر .
- ☒ پشتیبانی از Cookie ها .

به دلیل اینکه هیچ گونه استانداردی برای BOM وجود ندارد هر مرورگر ممکن است به صورتی متفاوت از آن پشتیبانی کند. مانند اشیاء Window و Navigator که هر مرورگر متدها و خاصیت های منحصر به فردی برای آن ها تعریف کرده است.
اینک به چند مفهوم اصلی در زبان JavaScript می پردازیم :

- ☒ **جاوا اسکریپت** Case-Sensitive **است** : یعنی همه چیز مانند نام متغیر ها ، نام توابع ، عملگر ها و هر چیز دیگری نسبت به حروف کوچک و بزرگ حساس است . به عنوان مثال متغیری با نام Test با متغیری با نام test متفاوت است.
- ☒ **متغیرها بدون نوع هستند** : برخلاف زبان هایی همچون java و C , متغیرها نوع خاصی نمی گیرند. در عوض هر متغیر می تواند با کلمه کلیدی var تعریف شده و مقداری را به عنوان مقدار اولیه بپذیرد . در واقع متغیرها "مقدار گرا" هستند. یعنی در هنگامی که تعریف (مقداردهی) می شوند نوعشان نیز مشخص می گردد .
- ☒ **قرار دادن (;) در انتهای هر دستور اختیاری است** : دستورات در جاوا اسکریپت می توانند به (;) ختم شوند یا نشوند. در صورت قرار ندادن (;) ، جاوا اسکریپت انتهای هر خط را به عنوان پایان دستور در نظر خواهد گرفت. اما روش صحیح ، استفاده از (;) در انتهای دستورات است. چون بعضی از مرورگرها از روش اول پشتیبانی نمی کند و ممکن است در اجرای کدها دچار مشکل شوند.
- ☒ **درج توضیحات در جاوا اسکریپت** : برای درج توضیحات در میان کدها از روش های زبان های برنامه نویسی همچون C و **C++** می توان استفاده نمود یعنی از // یا /* */ :

```
//this is a single-line comment

/* this is a multiline
comment */
```

متغیرها (Variables) در جاوا اسکریپت

متغیر ها با کلمه var تعریف می شوند. مانند :

```
Var  test ='ali';
```

در این مثال متغیری با نام test اعلان شده و مقدار اولیه 'ali' را می گیرد.
چون متغیرها بدون نوع هستند مفسر جاوا اسکریپت خود به خود نوع test را String در نظر می گیرد.
ما همچنین می توانیم دو یا چند متغیر را همزمان تعریف کنیم.

```
Var  test 1='ali' , test2='salam' ;
```

باید توجه داشته باشیم متغیرهایی که با یک Var تعریف می شود ممکن است نوع یکسانی نداشته باشند.

```
Var  test_1='ali' , age=25;
```

برخلاف جاوا (Java) متغیرها می توانند مقدار اولیه نگیرند.

```
Var  test ;
```

- ✓ **برخلاف جاوا متغیرها می توانند در زمان های مختلف مقدارهای متفاوتی داشته باشند . این یکی از امتیازات متغیر های بدون نوع در زبان جاوا اسکریپت به شمار می رود.**

```
Var  test ="hi" ;
alert(test);   // hi
Test=55;
alert(test);   // 55
```

نامگذاری متغیر ها :

نامگذاری متغیرها می بایست شرایط زیر را داشته باشد :

1. اولین کاراکتر متغیر می تواند یک حرف , یک (_) Underline و یا یک علامت $ باشد.
2. بقیه کاراکترها می توانند از $, Underline و یا هر حرف و عددی باشند.

تمام متغیر های زیر صحیح هستند :

```
Var  test ;
var  $test ;
var  $1 ;
var  _$test2 ;
```

یکی دیگر از امتیازات و یا شاید جذابیت های Java Script (که در خیلی از زبان های برنامه نویسی دیگر وجود ندارد) این است که لازم نیست که متغیر ها را قبل از مقدار دهی ، اعلان کنیم :

```
var  sTest="hello";
sTest2=sTest + "world";
alert (sTest2);  // hello world
```

در مثال فوق متغیر sTest2 قبل از مقداردهی اعلان نشده است .

موقعی که مفسر به چنین متغیرهای که بدون اعلان شدن مقداردهی می شوند , می رسد یک متغیر سراسری با آن نام ایجاد کرده و مقداری را به آن اختصاص می دهد.

با این وجود پیشنهاد می شود همیشه قبل از به کارگیری متغیرها آن ها را اعلان کنید.

کلمات کلیدی :

جاوااسکریپت تعدادی از کلمات را به عنوان کلمات کلیدی (keywords) می شناسد . این کلمات کلیدی معمولا ابتدا یا انتهای دستورات را مشخص می کنند . کلمات کلیدی به عنوان کلمات رزرو شده شناخته می شوند و نمی توان از آن ها به عنوان نام متغیر ها یا توابع استفاده نمود . در زیر لیست کاملی از این کلمات را مشاهده می کنید :

```
Break       else       new        var
Case        finally    return     void
Catch       for        switch     while
Continue    function   this       with
Default     if         throw
Delete      in         try
Do          instanceof typeof
```

اگر شما از یکی از کلمات فوق برای نامگذاری متغیر ها یا توابع استفاده کنید با خطای Identifier expected روبرو خواهید شد .

کلمات رزرو شده :

جاوااسکریپت تعدادی از کلمات رزرو شده را نیز معرفی کرده است . کلمات رزرو شده نیز نمی توانند به عنوان نام متغیر ها و توابع استفاده شوند . لیست کاملی از این کلمات را در زیر مشاهده می کنید :

```
Abstract    enum        int         short
Boolean     export      interface   static
Byte        extends     long        super
Char        final       native      synchronized
Class       float       package     throws
Const       goto        private     transient
Debugger    implements  protected   volatile
Double      import      public
```

مقادیر اصلی :

در جاوا اسکریپت پنج نوع مقدار اصلی به شرح زیر وجود دارد :

undefined , null , boolean , number , string

عملگر typeof یک پارامتر می گیرد : یا یک متغیر یا یک مقدار و نوع آن را بر می گرداند.

این عملگر یکی از پنج نوع زیر را بر می گرداند.

- UndeFined : اگر نوع متغیر از نوع undefined است.
- Boolean : اگر نوع متغیر از نوع Boolean باشد.
- Number : اگر نوع متغیر از نوع Number باشد.
- String : اگر نوع متغیر از String باشد.
- Object : اگر متغیر یک ارجاع یا از نوع null باشد .

نوع داده Undefind :

این نوع فقط شامل یک مقدار می شود : Undefined .

متغیری که اعلان می شود ولی مقدار دهی اولیه نمی شود به صورت پیش فرض از نوع Undefined خواهد بود .

```
Var  oTemp ;
alert (typeof  oTemp) ;  // outputs "Undefined"
```

نکته اینکه که متغیری که اعلان می شود و مقدار نمی گیرد با متغیری که اصلاً اعلان هم نشده است کاملا متفاوت است . هر چند که عملگر typeof بین این دو تفاوتی قائل نمی شود. و برای هر دو متغیر مقدار Undefined را بر می گرداند , اگر چه فقط یکی از آن ها (oTemp2) تعریف شده است.

```
Var  oTemp ;
alert (typeof  oTemp) ;  // outputs "Undefined"
alert (typeof oTemp2) ;  // outputs  "Undefined"
```

اگر شما از oTemp2 به وسیله ی هر عملگری به غیر از typeof استفاده کنید یک خطا رخ خواهد داد :

```
//make sure this variable isn't defined
//var oTemp2;
//try outputting
alert(oTemp2 == undefined); //causes error
```

مقدار Undefined زمانی که یک تابع مقداری را برنگرداند هم Return می شود.

```
Function Testfunc () {
     // leave the function black
}
alert( TestFunc() == undefined ); //outputs “true”
```

نوع داده Null :

دیگر نوع داده که فقط یک مقدار دارد , null است که مقدار ویژه null را می گیرد.

از نظر Java Script نوع Undefined یکی از مشتقات نوع null است و معادل یکدیگرند :

```
alert(null == undefined); //outputs “true”
```

اگر چه این دو معادل یکدیگرند اما معانی خاصی دارند.

Undefined موقعی به یک متغیر نسبت داده می شود که اعلان شود ولی مقداردهی نشود. در حالی که متغیری وقتی از نوع null است که شامل شیء ای باشد که وجود ندارد.

نوع داده Boolean :

نوع Boolean یکی از پر استفاده ترین نوع داده در زبان های برنامه نویسی است و متغیری از این نوع فقط می تواند یکی از دو مقدار true یا false به عنوان مقدار بپذیرد . اگر چه بر خلاف زبان های برنامه نویسی متداول , در جاوا اسکریپت false با 0 برابر نیست اما در صورت لزوم 0 به false تبدیل خواهد شد . به مثال های زیر توجه کنید :

```
var bFound = true;
var bLost = false;
```

نوع داده Number :

این نوع نیز یکی از پرکاربردترین انواع است. این نوع داده می تواند شامل اعداد Integer , **8** بایتی و اعداد اعشاری **16** بایتی باشد.

به عنوان مثال متغیر زیر از نوع Integer است و مقدار اولیه ی 55 را دارد :

```
var iNum = 55;
```

برای تعریف متغیرهای اعشاری به صورت زیر عمل می شود :

```
var fNum = 5.0;
```

نوع داده String :

این نوع می تواند برای ذخیره صفر یا چندین کاراکتر به کار رود. هر کاراکتر در یک رشته موقعیتی دارد. موقعیت اولین کاراکتر صفر است.

برای تعریف یک متغیر String باید از (') یا (") استفاده کنیم . معمولاً برای تعریف یک کاراکتر از (') و برای تعریف یک رشته از (") استفاده می شود.

```
var sColor1 = "blue";
var sColor2 = 'blue';
```

تبدیل انواع :

جاوا اسکریپت توابعی را برای تبدیل انواع فراهم آورده است.

تبدیل به رشته :

یکی از جذابترین ویژگی هایی که جاوا اسکریپت در رابطه با انواع اصلی Numbers , Boolean و String فراهم کرده است این است که آنها در اصل اشیای کاذب هستند, به این معنی که دارای خاصیت ها و متدهای مشترک و منحصر به فردی می باشند.

به عنوان مثال برای بدست آوردن طول یک رشته می توان از خاصیت Length استفاده نمود :

```
var sColor = "blue" ;
alert (sColor.length) ;   //outputs “4”
```

سه نوع داده Number , Bool و String متدی به نام ()tostring برای تبدیل به رشته دارند.

این متد برای متغیر های از نوع Boolean یکی از مقادیر true و false را بسته به مقدار متغیر بر می گرداند :

```
var bFound = false;
alert(bFound.toString()); //outputs “false”
```

این متد برای متغیر های از نوع number رشته ای حاوی آن عدد را بر می گرداند :

```
var iNum1 = 10;
var fNum2 = 10.0;
alert(iNum1.toString()); //outputs “10”
alert(fNum2.toString()); //outputs “10”
```

تبدیل به یک عدد :

جاوا اسکریپت دو متد برای تبدیل انواع غیر عددی به عددی فراهم کرده است :

- praseInt()
- parseFloat()

✓ **نکته : توجه کنید که حروف I و F باید به صورت حرف بزرگ نوشته شوند .**

این متد ها فقط بر روی رشته های حاوی عدد کار می کنند و بر روی بقیه انواع مقدار NaN را بر می گردانند.

متد ()parseInt از اولین کاراکتر رشته شروع می کند اگر عدد بود آن را بر می گرداند در غیر این صورت مقدار NaN را برمی گرداند.

این روند تا آخرین کاراکتر ادامه پیدا می کند تا اینکه به کاراکتری غیر عددی برسد. به عنوان مثال این متد عبارت "123red" را به صورت 123 بر می گرداند.

```
var iNum1 = parseInt(“1234blue”); //returns 1234
var iNum3 = parseInt("22.5");     //returns 22
var iNum4 = parseInt("blue");     //returns NaN
```

متد ()parseFloat نیز مثل ()parseInt کار می کند و از اولین کاراکتر شروع به جستجو می کند. البته در این متد اولین کاراکتر نقطه حساب نمی شود و آن را به همان صورت بر می گرداند.

اگر دو کاراکتر نقطه در رشته وجود داشته باشند دومین نقطه به عنوان invalid شناخته می شود و عملیات تبدیل متوقف می شود. مثال ها :

```
var fNum1 = parseFloat("1234blue");    //returns 1234.0
var fNum3 = parseFloat("22.5");        //returns 22.5
var fNum4 = parseFloat("22.34.5");     //returns 22.34
var fNum6 = parseFloat("blue");        //returns NaN
```

روش دیگر تبدیل انواع (Type Casting**)**

سه نوع type casting در جاوا اسکریپت وجود دارد :

- Boolean ()
- Number ()
- String ()

تابع Boolean() مقدار True را وقتی بر می گرداند که رشته شامل حداقل یک کارکتر , یک عدد بزرگتر از صفر و یا یک شیء باشد و مقدار False را بر می گرداند هر گاه رشته خالی است یا صفر است یا undefined و یا null باشد :

```
var b1 = Boolean("");                  //false – empty string
var b2 = Boolean("hi");                //true – non-empty string
var b3 = Boolean(100);                 //true – non-zero number
var b4 = Boolean(null);                //false - null
var b5 = Boolean(0);                   //false - zero
var b6 = Boolean(new Object());        //true – object
```

تابع Number() کاری شبیه ParseInt() و ParseFloat() را انجام می دهد اما تفاوت هایی هم دارد .
اگر به یاد داشته باشید متد های ParseInt() و ParseFloat()مقدار گرفته شده را فقط تا اولین کاراکتر بی ارزش بر می گرداندند. مثلا رشته "4.5.6" به 4.5 تبدیل خواهند کرد . اما کاربرد متد Number() برای این رشته مقدار NaN را بر می گرداند زیرا این رشته از نظر متد Number() در کل امکان تبدیل به یک عدد را ندارد .
اگر رشته ای امکان تبدیل به یک عدد را داشته باشد متد Number() برای اینکه از ParseInt() یا ParseFloat() استفاده کند تصمیم می گیرد . در مثال زیر حاصل اجرای متد Number() برای انواع داده ها را نشان می دهد :

```
Number(false)          0
Number(true)           1
Number(undefined)      NaN
Number(null)           0
Number("5.5")          5.5
Number("56")           56
Number("5.6.7")        NaN
Number(new Object())   NaN
Number(100)            100
```

ساده ترین تابع هم String() است که همان چیزی را که می گیرد به عنوان رشته بر می گرداند :

```
var s1 = String(null); //    "null"
```

جاوااسکریپت در مرورگر ها :

حال که تا حدودی با بسیاری از مفاهیم پایه جاوااسکریپت آشنا شدیم می خواهیم طریقه استفاده و قرار دادن آن ها در صفحه را بررسی کنیم . HTML برای استفاده از جاوااسکریپت در صفحات تگی به نام script را فراهم کرده که در ادامه با آن آشنا خواهیم شد .
عموما از این تگ در داخل تگ head صفحه استفاده می شود و می تواند یک ، دو یا سه صفت را بگیرد . صفت language که نوع زبان استفاده شده را تعیین می کند ، صفت اختیاری src که مکان یک فایل خارجی جاوااسکریپت را مشخص می کند و صفت type که نوع MIME TYPE یک فایل خارجی جاوااسکریپت را مشخص می کند و باید برابر عبارت text/javascript قرار داده شود .
مقدار صفت language معمولا برابر javascript یا یکی از نسخه های آن مثلا javascript 1.3 تعیین می شود . (اگر از صفت javascript چشم پوشی شود ، مرورگر ها آخرین نسخه موجود این زبان را در نظر می گیرند.)
کد های جاوااسکریپت در داخل تگ script نوشته می شوند اما در صورتی که همزمان از صفت src نیز استفاده شود در این صورت معمولا مرورگر ها کد های داخل تگ script را نادیده می گیرند . به مثال زیر دقت کنید :

```
<html>
<head>
     <title>Title of Page</title>
          <script language="JavaScript">
               var i = 0;
          </script>
     <script language="JavaScript" src="../scripts/external.js"></script>
</head>
<body>
     <!-- body goes here -->
</body>
</html>
```

در این مثال هر دو نوع تعریف کد های جاوااسکریپت در صفحه نشان داده شده است . تگ اسکریپت اول به صورت inline (درون خطی) به تعریف کد ها پرداخته است . و در تگ script دوم به یک فایل خارجی javascript اشاره شده است .

فایل های خارجی javascript :

فایل های خارجی جاوااسکریپت فرمت بسیار ساده ای دارند . آن ها درواقع فایل های متنی ساده حاوی کد های جاوااسکریپت هستند .
دقت کنید که در فایل های خارجی جاوااسکریپت از هیچ تگ script ی نمی بایست استفاده شود . به عنوان مثال به تکه کد زیر دقت کنید :

```
<html>
<head>
<title>Title of Page</title>
     <script language="JavaScript">
          function sayHi() {
               alert("Hi");
          }
     </script>
</head>
<body>
     <!-- body goes here -->
</body>
</html>
```

ما می توانیم خود تابع sayhi() را در فایلی خارجی مثلا به نام external.js ذخیره کرده و آن را به صورت زیر در صفحه مورد نظر لینک (الحاق) کنیم :

```
<html>
      <head>
      <title>Title of Page</title>
            <script language=”JavaScript” src=”external.js”></script>
      </head>
<body>
      <!-- body goes here -->
</body>
</html>
```

تفاوت های به کارگیری کد به صورت inline و External :

چه موقع ما باید از روش inline و چه موقع باید از روش external برای به کارگیری کد های جاوااسکریپت استفاده کنیم ؟ هر چند که قانون سفت و سختی برای استفاده از هر یک از روش های فوق وجود ندارد اما به دلایل زیر استفاده از روش inline مناسب به نظر **نمی رسد** :

- **امنیت :** هر کسی می تواند با باز کردن source صفحه شما ، کد های شما را ببیند و چه بسا به حفره های امنیتی آن پی برده و در پی پیاده سازی اقصاد شوم خود برآید .
- **ذخیره شدن در حافظه مرورگر ها :** یکی از مزیت های استفاده از روش External این است که فایل های خارجی جاوااسکریپت پس از اولین بارگذاری در حافظه نهان مرورگر (cache) ذخیره شده و در دفعات بعدی فایل خارجی از cache فراخوانی و استفاده خواهند شد .
- **نگه داری کد ها :** چنانچه شما بخواهید از یک کد در چندین صفحه وب استفاده کنید مطمئنا استفاده از روش اول موجب افزایش کد نویسی و در نتیجه حجم صفحه خواهد شد اما ما می توانیم از روش دوم برای چندین فایل استفاده کنیم .

مکان قرار دادن تگ script در صفحه :

معمولا کد ها و توابع تعریفی بوسیله جاوااسکریپت باید در قسمت head صفحه قرار گیرد تا به موقع بارگزاری شده و برای استفاده در قسمت body صفحه آماده استفاده و صدا زدن باشند . معمولا کد هایی که درآن ها توابع از قبل تعریف شده صدا زده می شوند در قسمت body قرار می گیرند .

قراردادن تگ script در داخل قسمت body موجب اجراشدن کد های داخل آن به محض بارگذاری قسمتی از صفحه در مرورگر خواهد شد . برای مثال به تکه کد زیر دقت کنید :

```
<html>
<head>
<title>Title of Page</title>
      <script language=”JavaScript”>
            function sayHi() {
                  alert(“Hi”);
            }
      </script>
</head>
<body>
      <script language=”JavaScript”>
            sayHi();
      </script>
      <p>This is the first text the user will see.</p>
</body>
</html>
```

در کد فوق متد sayHi() دقیقا قبل از نمایش هر گونه متنی در صفحه اجرا خواهد شد. به این معنی که پنجره alert قبل از متن This is the first text the user will see اجرا خواهد شد . این روش برای صدا زدن متد های جاوااسکریپت اصلا پیشنهاد نمی شود و می بایست به جای آن از کنترلگر های حوادث (Event Handler) استفاده کرد . مثلا :

```
<html>
<head>
<title>Title of Page</title>
      <script language=”JavaScript”>
            function sayHi() {
                  alert(“Hi”);
            }
      </script>
</head>
<body>
      <input type=”button” value=”Call Function” onclick=”sayHi()” />
</body>
</html>
```

در اینجا دکمه ای با استفاده از تگ input ایجاد شده است که در صورت کلیک بر روی آن تابع sayHi() فراخوانی می شود . صفت onclick در اینجا یک کنترلگر حادثه است که به رویداد رخ داده پاسخ می دهد .

نکته اینکه ازآنجایی که کد های جاوااسکریپت به محض بارگذاری اجرا هم می شوند ممکن است در این صورت توابعی که از قبل وجود ندارند صدا زده شوند که در این صورت یک خطا رخ خواهد داد . در مثال قبل با عوض کردن جای تگ های script یک خطا رخ خواهد داد :

```
<html>
<head>
<title>Title of Page</title>
</head>
<body>
      <script language=”JavaScript”>
            sayHi();
      </script>
<p>This is the first text the user will see.</p>
      <script language=”JavaScript”>
            function sayHi() {
                  alert(“Hi”);
            }
      </script>
</body>
</html>
```

در صورت اجرای کد فوق یک خطا رخ خواهد داد زیرا تابع قبل از اینکه تعریف شود صدا زده شده است .چون کد ها از بالا به پایین بارگذاری می شوند تابع sayHi() وجود نخواهد داشت تا تگ script دوم تولید نشده است .

مخفی کردن اسکریپت ها از مرورگر های قدیمی :

هنوز کاربران زیادی وجود دارند که از مرورگر هایی استفاده می کنند که با جاوااسکریپت ناسازگار هستند . از آن مهمتر ، تعدادی ا زکاربران گزینه پشتیبانی از جاوااسکریپت را در مرورگر خود غیر فعال کرده اند . از آنجایی که مرورگر های قدیمی دستور <script> را نمی شناسند و نمی توانند آن را تفسیر نمایند در اکثر موارد این مرورگر ها به جای تفسیر اسکریپت ، متن آن را در صفحه نمایش می دهند .

برای جلوگیری از این مشکل ، می توان اسکریپت ها را در داخل توضیحات HTML قرار داد . با این کار مرورگر های قدیمی آن را نادیده گرفته و نمایش نخواهند داد . از طرف دیگر مرورگر های جدبد می دانند که دستورات توضیحی که در بین دستورات آغازین و پایانی <script> منظور شده اند تنها برای مخفی کردن اسکریپت از دید مرور گر های قدیمی تر است و لذا به تفسیر اسکریپت ادامه می دهند .

همان طور که می دانید برای نوشتن یک توضیح در سند HTML کافی است علامت --!> را در ابتدا و علامت <-- را در انتهای آن قرار دهید .

به مثال زیر دقت کنید :

```
<script language=”JavaScript”><!-- hide from older browsers
      function sayHi() {
            alert(“Hi”);
            }
      //-->
</script>
```

- ✓ **به دو slashی که در انتهای دستور فوق آمده دقت کنید . این دو اسلش برای جلوگیری از این که مفسر جاوااسکریپت مرورگر های سازگاز با جاوااسکریپت عبارت <-- را به عنوان یک دستور جاوااسکریپت تفسیر نکند استفاده شده است . عدم استفاده از این دو // موجب ایجاد یک خطا خواهد شد .**

شما روش مخفی کردن اسکریپت ها از مرورگر های ناسازگار با جاوااسکریپت را فراگرفتید اما چگونه می توان برای کاربرانی که از این مرورگر ها استفاده می کنند نیز مطلب جایگزینی نمایش داد ؟ برای اینکار باید از تگی به نام <noscript> استفاده کنیم . مرورگر های سازگار هر چیزی را که بین دستورات آغازین و پایانی <noscript> قرار داشته باشد ، نادیده می گیرند . از طرف دیگر مرورگر های قدیمی این دستور را نمی شناسند و بنابراین آنرا نادیده گرفته و به سراغ دستورات بعدی (که توسط این دستور احاطه شده اند) می روند .

به مثال زیر توجه کنید :

```
<html>
<head>
<title>Title of Page</title>
      <script language=”JavaScript”>
            function sayHi() {
                  alert(“Hi”);
            }
      </script>
</head>
<body>
      <script language=”JavaScript”>
            sayHi();
      </script>
      <noscript>
            <p>Your browser doesn’t support JavaScript. If it did support
            JavaScript, you would see this message: Hi!</p>
      </noscript>
<p>This is the first text the user will see if JavaScript is enabled. If
JavaScript is disabled this is the second text the user will see.</p>
</body>
</html>
```

کار با آرایه ها در جاوااسکریپت

ایجاد آرایه ها با استفاده از کلاس Array

در جاوااسکریپت بر خلاف جاوا (java) ، کلاس درون ساختی به نام array وجود دارد که از آن برای ایجاد آرایه ها (که البته به عنوان یک شیء در نظر گرفته می شوند) استفاده می شود . برای ایجاد یک شی ء از نوع آرایه از دستورات زیر استفاده می کنیم :

```
var aValues = new Array();
```

اگر شما از قبل تعداد عناصر آرایه مورد نظرتان را بدانید می توانید به شکل زیر عمل کنید :

```
var aValues = new Array(20);
```

برای مقداردهی خانه های آرایه به شکل زیر عمل می کنیم :

```
var aColors = new Array();
aColors[0] = “red”;
aColors[1] = “green”;
aColors[2] = “blue”;
```

در آرایه بالا با هر بار اضافه کردن عنصر جدید به صورت خودکار به تعداد خانه های آن افزوده می شود .

اگر شما از قبل مقادیری که قرار است درآرایه قرار بگیرند را بدانید می توانید به صورت عمل کنید :

```
var aColors = new Array(“red”, “green”, “blue”);
```

برای دسترسی به عناصر آرایه به صورت زیر عمل می شود :

```
alert(aColors[1]); //outputs “green”
```

بدست آوردن طول آرایه

برای مشخص کردن تعداد عناصر موجود در آرایه از خاصیتی به نام length استفاده می شود . این مقدار همیشه یک واحد بیشتر از موقعیت آخرین خانه آرایه است .

اگر درآرایه قبلی که سه عنصر داشت ما به یکباره موقعیت مثلا **25** را پر کنیم طول آرایه چه خواهد بود ؟

در این صورت جاوااسکریپت خانه های از **3** تا **24** را با مقدار null پر خواهد کرد و در این صورت طول آرایه هم برابر **26** خواهد بود :

```
var aColors = new Array(“red”, “green”, “blue”);
alert(aColors.length); //outputs “3”
aColors[25] = “purple”;
aColors(arr.length); //outputs “26”
```

راه دیگر ایجاد یک آرایه استفاده از براکت ها **([ ])**و علامت , بین هر عنصر از آرایه است به صورت زیر :

```
var aColors = [“red”, “green”, “blue”];
alert(aColors.length); //outputs “3”
aColors[25] = “purple”;
alert(aColors.length); //outputs “26”
```

تبدیل آرایه به رشته :

آرایه ها از سه متد خاص برای خروجی عناصر خود به صورت رشته ای که با کاما از هم جداشده اند پشتیبانی می کند :

```
var aColors = [“red”, “green”, “blue”];
alert(aColors.toString()); //outputs “red,green,blue”
alert(aColors.valueOf()); //outputs “red,green,blue”
alert(aColors.toLocaleString()); //outputs “red,green,blue”
```

می بینید که حاصل اجرای هر سه کد بالا به صورت زیر خواهد بود .

از تابعی به نام join() برای الحاق عناصر یک آرایه که البته به وسیله یک جداکننده (separator) از هم جدا شده اند استفاده می شود. این تابع یک آرگومان دارد که در واقع رشته ای است که بین هر یک از عناصر وجود دارد . به مثال های زیر دقت کنید :

```
var aColors = [“red”, “green”, “blue”];
alert(aColors.join(“,”)); //outputs “red,green,blue”
alert(aColors.join(“-spring-”)); //outputs “red-spring-green-spring-blue”
alert(aColors.join(“][“)); //outputs “red][green][blue”
```

تبدیل رشته به آرایه :

سوالی که در اینجا پیش می آید این است که آیا اشیایی از نوع string را هم می توان به طریق مشابه به آرایه تبدیل کرد ؟ جواب مثبت است !!!

شی string متدی به نام split() دارد که یک آرگومان می گیرد که همانطور که حدس زدید جداکننده ی رشته برای تبدیل به آرایه را مشخص می کند .

حال اگر شما رشته ای دارید که با , از هم جدا شده است می توانید به صورت زیر عمل کنید :

```
var sColors = “red,green,blue”;
var aColors = sColors.split(“,”);
```

اگر هیچ جداکننده ای مشخص نشود ، این تابع آرایه ای را بر می گرداند که هر عنصر آن شامل یکی از کاراکترهای رشته ی مورد نظر است . برای مثال :

```
var sColors = “green”;
var aColors = sColors.split(“”);
alert(aColors.toString()); //outputs “g,r,e,e,n”
```

اضافه کردن مقادیر جدید به آرایه ها :

آرایه ها از متدی به نام concat() پشتیبانی می کنند . این تابع چندین آرگومان می گیرد و به آرایه جاری اضافه می کند و حاصل آن یک آرایه ی جدید خواهد بود . به مثالهای زیر دقت کنید :

```
var aColors = [“red”, “green”, “blue”];
var aColors2 = arr.concat(“yellow”, “purple”);
alert(aColors2.toString()); //outputs “red,green,blue,yellow,purple”
alert(aColors.toString()); //outputs “red,green,blue”
```

برگرداندن عناصر خاصی از آرایه :

از تابعی به نام slice() برای برگرداندن عناصر خاصی از آرایه استفاده می شود . این تابع دو آرگومان می گیرد و از خانه آرگومان اول تا **قبل** از آرگومان دوم را به آرایه جدیدی تبدیل می کند . اگر فقط آرگومان اول منظور گردد این تابع عناصر از آن آرگومان تا انتهای آرایه را بر می گرداند . به مثال های زیر دقت کنید :

```
var aColors = [“red”, “green”, “blue”, “yellow”, “purple”];
var aColors2 = arr.slice(1);
var aColors3 = arr.slice(1, 4);
alert(aColors2.toString()); //outputs “green,blue,yellow,purple”
alert(aColors3.toString()); //outputs “green,blue,yellow”
```

در حالت کلی arr.slice(n,m) **عناصر از خانه** n **تا** m-1 **را برمی گرداند .**

تبدیل آرایه ها به پشته و صف :

یکی از جذابترین ویژگی های آرایه در جاوااسکریپت امکان تبدیل کردن آنها به دیگر ساختمان داده های رایج همچون stack و queue است .

اگر آرایه ای را به عنوان stack در نظر بگیریم می توانیم به راحتی ازتوابع ()push و ()pop برای اضافه و حذف عناصر از انتهای آرایه استفاده کنیم .

تابع ()push امکان اضافه کردن چندین عنصر به آرایه و تابع ()pop امکان حذف آخرین عنصر آرایه و برگرداندن آن به عنوان مقدار بازگشتی تابع را فراهم می کند . البته تابع ()pop عنصری را که برمی گرداند از آرایه حذف می کند . به مثال های زیر دقت کنید :

```
var stack = new Array;
stack.push(“red”);
stack.push(“green”);
stack.push(“yellow”);
alert(stack.toString()); //outputs “red,green,yellow”
var vItem = stack.pop();
alert(vItem); //outputs “yellow”
alert(stack.toString()); //outputs “red,green”
```

جاوااسکریپت توابع دیگری برای دستکاری عناصر ابتدایی آرایه فراهم می کند . تابعی به نام ()shift برای حذف و برگرداندن عنصر اول آرایه استفاده می شود . از طرف دیگر تابعی به نام ()unshift یک عنصر را به ابتدای آرایه اضافه کرده و بقیه عناصر را یک موقعیت به جلو جابجا می کند :

```
var aColors = [“red”, “green”, “yellow”];
var vItem = aColors.shift();
alert(aColors.toString()); //outputs “green,yellow”
alert(vItem); //outputs “red”
aColors.unshift(“black”);
alert(aColors.toString()); //outputs “black,green,yellow”
```

در شکل زیر نحوه عملکرد توابع فوق بر روی یک آرایه عددی نمایش داده شده است :

مرتب سازی آرایه ها :

از دو تابع برای مرتب سازی (ordering) عناصر آرایه استفاده میشود . تابعی به نام ()reverse برای مرتب سازی عکس آرایه استفاده می شود . مثال :

```
var aColors = [“red”, “green”, “blue”];
aColors.reverse();
alert(aColors.toString()); //outputs “blue,green,red”
```

از طرف دیگر تابعی به نام ()sort عناصر آرایه را به صورت صعودی بر حسب مقادیرشان مرتب می کند . در این صورت عناصر آرایه بر حسب کد های کاراکتری شان مرتب می شوند . مثال :

```
var aColors = [“red”, “green”, “blue”, “yellow”];
aColors.sort();
alert(aColors.toString()); //outputs “blue,green,red,yellow”
```

در صورتی که عناصر آرایه اعداد باشند نتیجه کمی عجیب و غریب است :

```
var aColors = [3, 32, 2, 5]
aColors.sort();
alert(aColors.toString()); //outputs “2,3,32,5”
```

حذف و درج در میانه های آرایه :

یکی از پیچیده ترین توابعی که در کار با آرایه ها مورد استفاده قرار می گیرد تابعی به نام splice() است . هدف اصلی این تابع درج یکسری عناصر درمیانه های آرایه است .

راه های گوناگونی برای این استفاده از این متد در رابطه با آرایه و عمل درج پیشنهاد شده است :

- **عمل حذف :** از این متد برای حذف عناصری از میانه های آرایه می توان استفاده کرد . برای این کار از دو پارامتر برای این تابع استفاده می شود : موقعیت اولین عنصر و تعداد عناصر مورد نظر برای حذف . برای مثال arr.splice(0, 2) دو عنصر اول آرایه ای به نام arr را حذف می کند .
- **درج بدون حذف :** شما می توانید ازاین تابع برای درج عناصر جدید با استفاده از سه پارامتراستفاده کنید : موقعیت شروع ، تعداد عناصر حذفی و عناصر جدید برای درج .

شما می توانید هر تعداد پارامتر برای درج را به این تابع بدهید . برای مثال arr.splice(2, 0, “red”, “green”) عناصر red و green را از خانه دوم در آرایه درج می کند .

- **درج عنصر همراه با حذف :** شما می توانید از این تابع برای درج عناصر جدید در یک موقعیت مشخص همزمان با عمل حذف و استفاده از سه پارامتر استفاده کنید : موقعیت شروع حذف ، تعداد عناصر حذفی و عناصر جدید درجی . به عنوان مثال arr.splice(2, 1, “red”, “green”) یک عنصر را از موقعیت ۲ حذف کرده و مقادیر red و green را از همان موقعیت (2) درج می کند .

کار با رشته ها در جاوااسکریپت

ایجاد اشیاء رشته ای (رشته) با استفاده از کلاس string :

از این کلاس برای ایجاد اشیاء رشته ای (به اختصار رشته ها) استفاده می شود . دستور زیر متغیری حاوی رشته Hello World را تولید می کند :

```
var oStringObject = new String(“hello world”);
```

اشیای از نوع string خاصیتی به نام length دارند که تعداد کاراکتر های رشته را بر می گرداند . این شیء از چندین متد نیز پشتیبانی می کند که در ادامه شرح خواهیم داد :

بدست آوردن کاراکتر موجود در یک موقعیت خاص :

charAt() : عددی را به عنوان آرگومان می گیرد و کاراکتر نظیر آن در رشته اصلی را برمی گرداند . مثلا :

```
var oStringObject = new String(“hello world”);
alert(oStringObject.length); //outputs “11”
```

اگر چنانچه می خواهید به جای خود کاراکتر کد کاراکتری آن را بدست آورید از متد charCodeAt() استفاده کنید :

```
var oStringObject = new String(“hello world”);
alert(oStringObject.charCodeAt(1)); //outputs “101”
```

این دستور مقدار 101 که معادل کد کاراکتری حرف e است را بر می گرداند .

الحاق دو رشته :

متد دیگر ()concat است که برای الحاق دو رشته استفاده می شود . برای مثال :

```
var oStringObject = new String(“hello “);
var sResult = oStringObject.concat(“world”);
alert(sResult); //outputs “hello world”
alert(oStringObject); //outputs “hello “
```

به جای استفاده از متد ()concat می توان از عملگر **+** نیز برای الحاق دو رشته استفاده کرد .

بدست آوردن موقعیت یک کاراکتر خاص در رشته :

برای تشخیص اینکه یک کاراکتر خاص در یک رشته هست یا نه می توان از متد های ()indexOf و ()lastIndexOf استفاده می شود .

هر دو این متدها موقعیت زیر رشته ای در رشته دیگر را برمی گرداند که البته در صورت پیدانشدن مقدار1- را بر می گردانند .

تنها تفاوت این دو تابع در این است که ()indexOf جستجو را از ابتدای رشته (موقعیت **0**) شروع می کند ولی دیگری جستجو را از انتهای رشته شروع می کند . برای مثال :

```
var oStringObject = new String(“hello world”);
alert(oStringObject.indexOf(“o”)); //outputs “4”
alert(oStringObject.lastIndexOf(“o”)); //outputs “7”
```

در صورتی که حرف **O** در عبارت بالا فقط یکبار تکرار می شد هر دو این متد ها فقط یک مقدار رابر می گردانند .

مقایسه رشته ها :

متد دیگری که برای رشته ها تعریف شده ()localeCompare است که برای مقایسه رشته ها مورد استفاده قرار می گیرد. (این متد معادل تابع ()strcmp در زبان **C++** است .)

این تابع یک آرگومان رشته ای می پذیرد و یکی از سه مقدار زیر را بر می گرداند :

1. اگر شیء رشته ای کوچکتر از آرگومان باشد 1- را بر می گرداند .
2. اگر برابر باشند 0 را برمی گرداند .
3. اگر شیء رشته ای بزرگتر باشد مقدار 1 را بر می گرداند .

مثال ها :

```
var oStringObject = new String(“yellow”);
alert(oStringObject.localeCompare(“brick”)); //outputs “1”
alert(oStringObject.localeCompare(“yellow”)); //outputs “0”
alert(oStringObject.localeCompare (“zoo”)); //outputs “-1”
```

جدا کردن زیر رشته ای از رشته دیگر :

دو تابع برای جدا کردن زیر رشته ها از رشته اصلی وجود دارد : ()slice و ()substring .

هر دو این متد ها یک یا دو آرگومان را می پذیرند که آرگومان اول محل شروع و آرگومان دوم محل پایان را تعیین می کند . (البته خودآرگومان دوم جزء زیر رشته نخواهد بود .)

اگر آرگومان دوم نادیده گرفته شود length رشته درنظر گرفته خواهد شد .

چیزی که این دو متد بر می گرداند زیر رشته حاصل است :

```
var oStringObject = new String(“hello world”);
alert(oStringObject.slice(3)); //outputs “lo world”
alert(oStringObject.substring(3)); //outputs “lo world”
alert(oStringObject.slice(3, 7)); //outputs “lo w”
alert(oStringObject.substring(3,7)); //outputs “lo w”
```

سوالی که دراینجا پیش می آید این است که چرا دقیقا این دو تابع بک کار را انجام می دهند ؟ در حقیقت تفاوت آن ها در کار با آرگومان های منفی است .

برای متد slice() آرگومان منفی با طول رشته جمع شده و حاصل آن به عنوان آرگومان اصلی درنظر گرفته می شود . در حالی که برای تابع sustring() مقادیر منفی به عنوان صفر درنظر گرفته می شود .(درواقع نادیده گرفته می شوند .)

مثال ها :

```
var oStringObject= new String(“hello world”);
alert(oStringObject.slice(-3)); //outputs “rld”
alert(oStringObject.substring(-3)); //outputs “hello world”
alert(oStringObject.slice(3, -4)); //outputs “lo w”
alert(oStringObject.substring(3,-4)); //outputs “hel”
```

در خط دوم از کد بالا چون آرگومان منفی است طول رشته با **3-** جمع می شود که حاصل **8** است درواقع دستور زیر اجرا میشود :

```
oStringObject.slice(8);
```

که از خانه هشتم رشته تا انتهای آرایه را بر می گرداند . اما در خط سوم آرگومان منفی صفر درنظر گرفته می شود . یعنی :

```
oStringObject.substring(0);
```

در خط چهارم آرگومان دوم با طول رشته جمع شده و حاصل آن یعنی **8** به عنوان آرگومان دوم در نظرگرفته می شود . یعنی :

```
oStringObject.slice(3,8);
```

و در خط پنجم حاصل به صورت زیر محاسبه می شود :

```
oStringObject.substring(3,0);
```

toUpperCase() و toLowerCase()

از توابعی همچون toUpperCase() و toLowerCase() برای تبدیل حروف رشته به حروف بزرگ یا کوچک استفاده می شود که کار آن ها از روی اسمشان کاملا مشخص است :

```
var oStringObject= new String(“Hello World”);
alert(oStringObject.toLocaleUpperCase()); //outputs “HELLO WORLD”
alert(oStringObject.toUpperCase()); //outputs “HELLO WORLD”
alert(oStringObject.toLocaleLowerCase()); //outputs “hello world”
alert(oStringObject.toLowerCase()); //outputs “hello world”
```

اشیای درونی (پیش ساخته) :

جاوااسکریپت شامل تعدادی شی از پیش ساخته است که طراحان می توانند از آن ها در برنامه های خود استفاده کنند . در واقع ما کلاس هایی برای این اشیا نداریم و لازم نیست شی ای از روی آن ها ساخته شود .

شی Math :

یکی از اشیای از پیش ساخته شده جاوااسکریپت است که برای انجام محاسبات عددی و عملیات مربوط به ریاضیات استفاده می شود . این شی شامل یکسری خاصیت و متد است که انجام محاسبات را آسان می کند .

متدهای min() و max() :

از این توابع برای پیداکردن کوچکترین و بزرگترین مقادیر از بین چند عدد استفاده می شود . این متد ها هر تعداد پارامتر را می توانند بپذیرند :

```
var iMax = Math.max(3, 54, 32, 16);
alert(iMax); //outputs “54”
var iMin = Math.min(3, 54, 32, 16);
alert(iMin); //outputs “3”
```

این توابع برای جلوگیری از نوشتن برنامه های اضافی برای پیداکردن min و max اعداد می تواند استفاده شود .
یکی از متد ها ، abs() است که قدر مطلق اعداد گرفته شده را بر می گرداند .
گروهی دیگر از متد ها که برای گرد کردن اعداد اعشاری به صحیح مورد استفاده قرار می گیرند . این توابع شامل ceil() و floor() و round() هستند .

- تابع round() : این تابع عدد گرفته شده را به عدد صحیح بالاتر گرد می کند اگر قسمت اعشاری ا زنصف بیشتر یا مساوی باشد و در غیر این صورت آن را به عدد صحیح پایین تر گرد می کند .
- تابع ceil() : این تابع بدون در نظر گرفتن قسمت اعشاری آن را به کوچکترین عدد صحیح بعدی گرد می کند .
- تابع floor() : این تابع بدون در نظر گرفتن قسمت اعشاری آن را به بزرگترین عدد صحیح قبلی گرد می کند .

به مثال های زیر توجه کنید :

```
alert(Math.ceil(25.5));  //outputs “26”
alert(Math.round(25.5)); //outputs “26”
alert(Math.floor(25.5)); //outputs “25”
```

گروه دیگری از متد ها برای کار با مقادیر توانی وجود دارد :
Log() : برای محاسبه لگاریتم طبیعی عدد گرفته شده به کار می رود .
Pow() : برای محاسبه توان یک عدد به کار می رود که دو آرگومان می گیرد :

```
var iNum = Math.pow(2, 10);
```

sqrt() : جذر یک عدد را حساب می کند:

```
var iNum = Math.sqrt(4);
alert(iNum); //outputs “2”
```

شی Math شامل متد های زیر نیز می باشد :

acos(x) , asin(x) , atan(x) , atan2(x, y) , cos(x) , sin(x) , tan(x)

یکی دیگر از متد های مربوط به شی ء Math که کاربرد زیادی هم دارد random() است . که برای تولید اعداد تصادفی بین 0 و 1 (البته نه خود 0 و 1) مورد استفاده قرار می گیرد .
البته برای تولید اعداد تصادفی در یک محدوده خاص از فرمول زیر استفاده می شود :

```
number = Math.floor(Math.random() * total_number_of_choices +
first_possible_value)
```

به عنوان مثال یرای ایجاد مقادیر تصادفی بین 1 و 10 به صورت زیر عمل می شود :

```
var iNum = Math.floor(Math.random() * 10 + 1);
```

بهترین راه برای ایجاد مقادیر تصادفی استفاده از یک تابع است که به صورت زیر نوشته می شود :

```
function selectFrom(iFirstValue, iLastValue) {
     var iChoices = iLastValue – iFirstValue + 1;
     return Math.floor(Math.random() * iChoices + iFirstValue);
}
//select from between 2 and 10
     var iNum = selectFrom(2, 10);
```

استفاده از این تابع برای انتخاب یک عنصر تصادفی از آرایه بسیار آسان است . برای مثال :

```
var aColors = [“red”, “green”, “blue”, “yellow”, “black”, “purple”, “brown”];
var sColor = aColors[selectFrom(0, aColors.length-1)];
```

در اینجا آرگومان دوم تابع ، طول آرایه منهای 1 است که در واقع موقعیت آخرین عنصر می باشد .

دیگر توابع مفید :

از توابعی همچون encodeURI() و encodeURIComponent() برای encode کردن آدرس های اینترنتی(URI ها) استفاده می شود . در حالت کلی و صحیح یک آدرس نباید شامل کاراکترهای خاص همچون space باشد . این توابع به شما درتبدیل کردن و encode کردن آدرس های اینترنتی نادرست و بی ارزش برای اینکه مرورگر ها آنها را بفهمند استفاده می شود .

متد encodeURI() معمولا برای آدرس های کامل (به عنوان مثال http://itcom.pnuab.ac.ir/illegal value.htm) مورد استفاده قرار می گیرد در حالی که دیگری برای قسمتی از آدرس ها همچون illegal value.htm مورد استفاده قرار می گیرد . تفاوت اصلی بین این دو تابع این است که تابع اول کاراکتر های خاصی که به عنوان جزئی از آدرس هستند همچون (:) ، / ، ؟ و... را encode نمی کند درحالی که تابع دوم تمام کاراکتر های غیر استاندارد را encode خواهد کرد . برای مثال :

```
var sUri = “http://www.wrox.com/illegal value.htm#start”;
alert(encodeURI(sUri));
alert(encodeURIComponent(sUri));
```

حاصل اجرای کد بالا به صورت زیر خواهد شد :

```
http://www.wrox.com/illegal%20value.htm#start
http%3A%2F%2Fwww.wrox.com%2Fillegal%20value.htm%23start
```

طبیعتا دو تابع برای decode کردن آدرس های اینترنتی استفاده می شود همچون :

- ✓ decodeURI()
- ✓ decodeURIComponent()

به عنوان مثال :

```
var sUri = “http%3A%2F%2Fwww.wrox.com%2Fillegal%20value.htm%23start”;
alert(decodeURI(sUri));
alert(decodeURIComponent(sUri));
```

حاصل اجرای این کد به صورت زیر خواهد بود :

```
http%3A%2F%2Fwww.wrox.com%2Fillegal value.htm%23start
http://www.wrox.com/illegal value.htm#start
```

آخرین تابعی که به نظر قدرتنمند می آید eval() است . این تابع که شبیه به مفسر جاوااسکریپت کار می کند آرگومانی از نوع رشته می گیرد که در واقع یک برنامه به زبان جاوااسکریپت است و این تابع آن را همانند سایر برنامه ها اجرا می کند . برای مثال :

```
eval(“alert(‘hi’)”);
```

این تکه کد در حقیقت معادل دستور زیر است :

```
alert(“hi”);
```

موقعی که مفسر جاوااسکریپت به این تابع می رسد آرگومان آن را به عنوان یک دستور خیلی ساده تفسیر کرده و اجرا می کند .
این به این معنی است که شما می توانید از داخل آرگومان های این تابع به تمام متغیرهای خارج آن دسترسی داشته و از آن ها استفاده کنید :

```
var msg = “hello world”;
eval(“alert(msg)”);
```

همچنین شما می توانید آرگومان تابع eval() را یک تابع تعریف کرده و سپس آن را خارج از تابع eval() صدا بزنید . برای مثال :

```
eval(“function sayHi() { alert(‘hi’); }”);
sayHi();
```

کار با تاریخ و زمان در جاوااسکریپت

یکی از ویژگی هایی که جاوااسکریپت دارد جمع آوری اطلاعات از سیستم کاربر و نمایش آنها در صفحات وب است. همانطور که می دانید HTML به تنهایی قادر به انجام چنین کاری نیست اما با کمک زبانهای دیگر تحت وب مانند Javascript ، می تواند تا حدودی این مشکل را برطرف کند. شئ هایی در جاوااسکریپت وجود دارند که توسط متدهای مختلف، اطلاعات مورد نیاز را از سیستم گرفته و در اختیار کاربران قرار می دهند. یکی از این object ها و شئ ها ، Date می باشد که به کمک آن می توانیم تاریخ و زمان سیستم را هنگام اجرای کد دریافت کنیم، سپس آنرا نمایش دهیم و یا اینکه در یک متغیر ذخیره کنیم تا در صورت لزوم از آن بهره گیریم.
برای ایجاد شی ای از این نوع میتوان به شکل زیر عمل کرد :

```
var d = new Date();
```

شی Date() تعداد هزارم ثانیه های گذشته از ساعت 12:00:00 روز 01/01/1970 تا زمان و تاریخ کنونی را در خود نگه داری می کند . این شی دارای متدی به نام valueOf() می باشد که این مقدار را بر می گرداند. به عنوان مثال به کد زیر نگاه کنید :

```
<script type="text/javascript">
      var d=new Date();
      document.write(d.valueOf());
</script>
```

حاصل اجرای کد فوق می تواند عددی به شکل زیر باشد :

```
1269938333117
```

این شیء دارای متد هایی است که از آن ها برای بدست آوردن جزئیات بیشتری از تاریخ و زمان استفاده نمود . بعضی از این متد ها و خواص را در جدول زیر مشاهده می کنید :

نام متد	توضیحات
getDate()	روزی از ماه را بر می گرداند که می تواند مقداری از 1 تا 31 باشد .
getMonth()	ماهی از سال را بر می گرداند که مقداری از 0 تا 11 می باشد
getFullYear()	سال را در قالب 4 عدد بر می گرداند
getHours()	ساعتی از روز را بر می گرداند که می تواند مقداری از 0 تا 23 باشد
getMinutes()	دقیقه را بر می گرداند که مقداری از 0 تا 59 است .
getSeconds()	ثانیه را بر می گرداند که مقداری از 0 تا 59 است
getDate()	روزی از هفته را بر می گرداند که می تواند مقداری از 0 تا 6 باشد . (**6** به معنی **sunday**)
getTime()	تعداد میلی ثانیه های گذشته از تاریخ **1/1/1970** را بر می گرداند
valueOf()	تعداد میلی ثانیه های گذشته از تاریخ **1/1/1970** را بر می گرداند
toString()	رشته ای حاوی اطلاعاتی همچون مخفف نام روز جاری ، ماه جاری و ... را بر می گرداند.

علاوه بر متد های فوق ، شی ء Date از متدی به نام getTimezoneOffset() که اختلاف بین زمان محلی و زمان واحد جهانی رابر حسب دقیقه بر می گرداند نیز پشتیانی می کند . به عنوان مثال این متد مقدار ۲۱۰ را برای وقت محلی ایران بر می گرداند . (که همان اختلاف ۳:۳۰ دقیقه ای ساعت تهران نسبت به زمان واحد جهانی است .)

BOM (Browser Object Model) : مدل شیء گرای مرورگر

ما نمی توانیم بدون صحبت درباره BOM با جاوا اسکریپت کار کنیم. BOM اشیایی که با پنجره ی مرورگر ارتباط و تعامل مستقیم دارند را فراهم می کند , مانند شکل زیر :

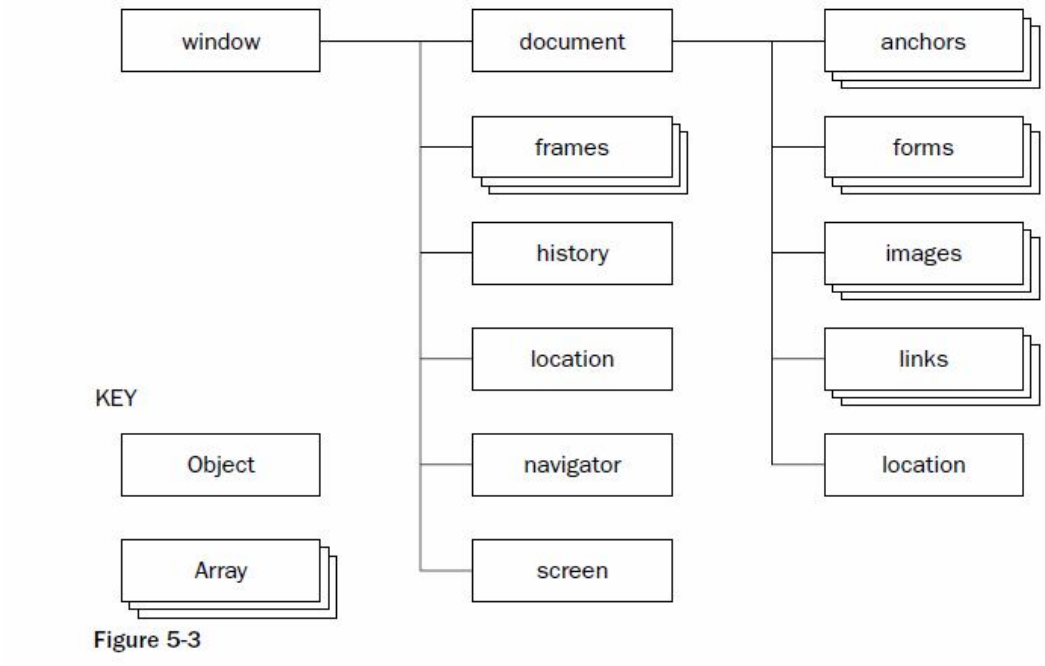


Figure 5-3

- ✓ **توجه : BOM مجموعه ای از اشیای مرتبط با هم را فراهم می کند.**

شیء window :

شی ء Window تمامی پنجره های مرورگر را شامل می شود اما نه لزوماً محتوایی که در آن نمایش داده می شود.
از این شیء برای جابجایی ، تغییر اندازه و دیگر اثرات بر روی پنجره ها استفاده می کنیم.

دستکاری پنجره ها :

چهار متد برای دستکاری پنجره مرورگر برای شیء Window وجود دارد :

۱) moveBy(dx,dy) :
پنجره را نسبت به موقعیت کنونی به اندازه x در جهت افقی و به اندازه y در جهت عمودی جابجا می کند. عدد های منفی هم برای x,y مجازند.

۲) moveTo(x,y) :
گوشه بالای چپ مرورگر را به موقعیت x,y می برد. مقادیر منفی نیز مجاز هستند.

۳) resizeBy(w,h) :
عرض پنجره مرورگر را به اندازه w و ارتفاع آنرا به اندازه h نسبت به size کنونی تغییر می دهد. مقادیر منفی نیز مجازند.

۴) resizeTo(w,h) :
عرض مرورگر را به w و ارتفاع آن را به h تغییر می دهد. **مقادیر منفی مجاز نیستند .**

مثال ها :

```
window.moveBy(10, 20)
// پنجره را نسبت به مکان فعلی 10px پیکسل به سمت راست و 20px به سمت پایین جابجا می کند .

window.resizeTo(150, 300)
عرض پنجره را به 150px و ارتفاع آن را به 300px تغییر می دهد .

window.resizeBy(150, 0)
فقط 150px به عرض کنونی پنجره اضافه می کند .

window.moveTo(0, 0)
پنجره را به گوشه بالا و سمت چپ صفحه نمایش هدایت می کند .
```

پیمایش و باز کردن پنجره های جدید

برای باز کردن پنجره های جدید با استفاده از جاوا اسکریپت از متد open() استفاده می شود که چهار آرگومان می گیرد :

1. آدرس صفحه
2. نام صفحه
3. رشته ای از ویژگی های
4. و یک مقدار Booelan

عموماً فقط از سه آرگومان اول استفاده می شود. اگر پنجره ای از قبل با نامی که برای آرگومان دوم انتخاب کرده اید وجود داشته باشد صفحه در آن پنجره باز خواهد شد , در غیر این صورت در پنجره ای جدید باز می شود.
اگر آرگومان سوم مشخص نشود پنجره با تنظیمات پنجره اصلی مرورگر باز خواهد شد.

ویژگی های آرگومان سوم مشخص می کند که پنجره ی جدید چه خصوصیاتی داشته باشد که در زیر بیان می کنیم :
خصوصیات با (=) مقدار دهی می شود و با (,) از هم جدا می شود.

برخی از خصوصیات مجاز قابل استفاده عبارتند از :

- Left : فاصله از چپ
- Top : فاصله از بالا
- Width : عرض پنجره
- Height : ارتفاع پنجره
- Resizable , (Yes,No) : آیا پنجره قابل تغییر اندازه باشد یا خیر
- Scrollable , (Yes,NO) : scroll دار بودن یا نبودن
- Toolbar , (Yes,NO) : آیا شامل نوار ابزار باشد .
- Status , (Yes,No) : آیا نوار وضعیت داشته باشد
- Location , (Yes,No) : آیا نوار آدرس داشته باشد .

✓ **در رشته ای از خصوصیات نباید هیچ فضای خالی وجود داشته باشد.**

متد Open شیء ای از نوع Window را بر می گرداند که تمام متد ها و خاصیت هایی که شیء Window دارد را داراست.
برای بستن پنجره از متد close() استفاده می شود. این متد فقط می تواند پنجره ای که توسط جاوا اسکریپت باز شده است را مستقیما ببندد نه پنجره ی اصلی.

پنجره های System Dialog :

شیء Window چندین تابع برای نمایش پیغام و گرفتن جواب از کاربران را دارد.

☒ alert() : این تابع یک آرگومان از نوع متن می گیرد و آن را در قالب یک پنچره کوچک که یک دکمه Ok دارد نمایش می دهد :

```
<scritp type="text/javascript" >
     alert('Hello world');
</script>
```

از این پنجره معمولاً برای نمایش یک پیغام به صورت هشدار استفاده می شود .

☒ confirm() : این تابع هم مانند تابع بالاست . تنها تفاوت این دو وجود یک دکمه Cancel در پنجره ی باز شونده است .

```
<scritp type="text/javascript" >
     confirm('Are you sure ? ') ;
</script>
```

در صورتی که کاربر دکمه Ok را بزند مقدار True و در صورت زدن دکمه ی Cancel مقدار False را بر می گرداند .

☒ prompt() : پنجره ی این متد چهار قسمت دارد. دکمه ی Ok , دکمه ی Cancel , یک متن و یک text field برای وارد کردن یک رشته توسط کاربر .

این متد دو آرگومان می گیرد :

1. عنوان سوال یا متنی که به کاربر نشان داده می شود .
2. مقدار پیش فرض برای Text field

```
<scritp type="text/javascript" >
     Prompt('what is your name','ali');
</script>
```

در صورتی که کاربر دکمه ی Ok را بزند تابع مقدار وارد شده در Text field را بر می گرداند و در صورت زدن دکمه ی Cancel مقدار Null را بر می گرداند.

خاصیت statusbar :

این قسمت پنجره فرآیند بارگزاری و پایان بارگزاری را به کاربر نشان می دهد. هر چند که می توانیم از دو خاصیت به نام های status و defaultStatus برای تغییر آن استفاده کنیم.

همانطور که حدس زدید از خاصیت Status برای تغییر متن Statusbar برای چند لحظه استفاده می شود در حالی که از defaultstatus برای تغییر Statusbar تا زمانی که کاربر در صفحه هست استفاده می شود .

برای تغییر لحظه ای نوار وضعیت مثلاً وقتی کاربر ، ماوس را روی یک لینک قرار می دهد می توان از کد زیر استفاده نمود :

```
<a href="books.htm" onmouseover="window.status='Information on Wrox books.'
">Books</a>
```

اجرای مکرر کد ها از طریق متدهای Timeouts و Intervals :

از این دو تابع برای اجرای یک تکه کد بعد از بازه زمانی خاصی استفاده می شود .

- **setTimeouts :** کد گرفته شده را پس از عددی بر حسب میلی ثانیه اجرا می کند. در حالی که Intervals کد گرفته شده را مکرراً بعد از مدتی بر حسب میلی ثانیه چندین بار تکرار می کند. این متد دو آرگومان می گیرد :
  1. کدی که باید اجرا شود .
  2. مدت زمانی که باید بعد از آن کد اجرا شود .

آرگومان اولی هم می تواند به صورت یک رشته از کدها و هم نام یک تابع باشد. هر سه کد زیر بعد از یک ثانیه یک پنجره هشدار را نمایش می دهند :

```
<scritp type="text/javascript" >
     setTimeout("alert('Hello world!')", 1000);
</script>

-----------------------------------------------------

<scritp type="text/javascript" >
     setTimeout(function() { alert("Hello world!"); }, 1000);
</script>

-----------------------------------------------------

<scritp type="text/javascript" >

     function sayHelloWorld() {
          alert("Hello world!");
     }
     setTimout(sayHelloWorld, 1000);

</script>
```

برای جلوگیری از اجرای تابع setTimeout() از متد ClearTimeout() به صورت زیر استفاده می شود :

```
<scritp type="text/javascript" >
     var iTimeoutId = setTimeout("alert('Hello world!')", 1000);
     clearTimeout(iTimeoutId);
</script>
```

- **setIntervals** : مانند تابع قبلی است جز اینکه کد گرفته شده را بعد از گذشت بازه ی زمانی مشخص تکرار می کند. برای جلوگیری از اجرای این متد , از تابعی به نام ClearInterval استفاده می شود :

```
setInterval("alert('Hello world!') ", 1000);
-----------------------------------------------
setInterval(function() { alert("Hello world!"); }, 1000);
-----------------------------------------------
function sayHelloWorld() {
alert("Hello world!");
}
setInterval(sayHelloWorld, 1000);
```

شیء history :

ممکن است بخواهیم به تاریخچه ی مرورگر دسترسی داشته باشیم , البته هیچ راهی برای دسترسی به آدرس صفحات که در History وجود دارند , نیست. برای این کار از متدها و خاصیت های شیء History مربوط به شیء Window استفاده می کنیم :

متد ()Go فقط یک پارامتر می گیرد : تعداد صفحاتی که باید به جلو یا به عقب پیمایش شوند. اگر عدد منفی باشد به صفحات قبل و اگر عدد مثبت باشد به صفخات جلو می رویم. برای مثال جهت رفتن به یک صفحه عقب از کد زیر استفاده می کنیم :

```
window.history.go(-1);
```

و برای رفتن به جلو :

```
window.history.go(+1);
```

همچنین می توانیم از متد های ()back و ()forward به جای کدهای بالا استفاده کنیم.

```
//go back one
history.back();
//go forward one
history.forward();
```

همچنین از خاصیت lenght برای تعداد صفحات موجود در history استفاده کنیم :

```
alert("There are currently " + history.length + " pages in history.");
```

شیء Document :

این شیء که تنها شیء مشترک بین مدل های شیءگرای BOM و DOM است. و نیز دارای خصوصیاتی است.
یکی از خاصیت های این شیء URL است که برای تنظیم و دسترسی به آدرس کنونی صفحه استفاده می شود .

```
document.URL = "http://www.tavoosebehesht.ir/";
```

همچنین این شیء دارای یکسری خصوصیات مجموعه ای برای دسترسی به انواع عناصر داخل صفحه ی بارگزاری شده است.
برخی از خاصیت ها به شرح زیر است :

مجموعه	توضیحات
anchors	دسترسی به لینک های صفحه
embeds	دسترسی به تمامی عناصر embed صفحه
forms	دسترسی به تمامی فرم های صفحه
images	دسترسی به تمامی عناصر عکس صفحه
links	دسترسی به تمامی لینک های صفحه

هر مجموعه می تواند بوسیله ی عدد یا نام ، index گذاری شوند. به این معنی که شما می توانید به صورت زیر به اولین عنصر عکس صفحه دسترسی داشته باشید :

```
Document.images[0];
Or
Document.images['image-name'] ;
```

با این روش ما می توانیم به آدرس آن ها هم دسترسی داشته باشیم , به صورت زیر :

```
document.images[0].src
```

از دیگر متدهای این شیء می توان به Write() و Writeln() برای چاپ یک متن اشاره کرد .

شیء location :

یکی دیگر از شیء ها برای دسترسی به آدرس صفحه جاری ، location است. ما توسط خاصیت location.href می توانیم برای تنظیم یا بدست آوردن URL استفاده کنیم :

```
document.href= "http://www.tavoosebehesht.ir/";
```

متد assign() هم همین کار را می کند.

از متد reload() برای بارگزاری مجدد صفحه استفاده می شود. ما می توانیم تعیین کنیم که بارگزاری مجدد از روی Cache یا Server باشد. این کار با یکی از آرگومان false برای بارگزاری مجدد از Catch و true برای بارگزاری مجدد از Server استفاده می شود.

شیء Navigator :

این شیء یکی از اشیای قدیمی مدل شیءگرای BOM است.

از این شیء برای دسترسی و بدست آوردن اطلاعاتی در مورد نوع و نسخه مرورگر استفاده می شود. بعضی از خاصیت های آن به شرح زیر است :

خاصیت ها	توضیحات
appcodeName	رشته ای حاوی کد رشته ای مرورگر
appName	نام عمومی مرورگر
appMinotVersion	اطلاعات اضافی مرورگر
appVersion	نسخه مرورگر
browserLanguage	نوع زبان مرورگر یا سیستم عامل
cookieEnabled	مشخص می کند میکند آیا کوکی ها فعال هستند یا خیر
cpuClass	کلاس cpu را مشخص میکند
javaEnabled	فعال بودن جاوا
Language	زبان مرورگر را مشخص میکند
mimeType	آرایه ای از mimetype های ثبت شده در مرورگر
Platform	نوع platform ی که کامپیوتر کاربر بر روی آن قرار دارد را مشخص می کند .

شی ء screen :

از این شی ء برای دسترسی به اطلاعات مربوطه به صفحه نمایش کاربر استفاده می شود . این شی ء شامل خواص زیر است :

خاصیت	توضیحات
availHeight	ارتفاع قابل دسترس از ویندوز
availWidth	عرض قابل دسترس از ویندوز
colorDepth	تعداد بیت ها برای نمایش رنگ ها
height	ارتفاع صفحه
width	عرض صفحه

از دو خاصیت اول می توان برای بدست آوردن سایز جدید پنجره استفاده نمود . به طور مثال برای fullscreen کردن صفحه نمایش می توان از کد زیر استفاده نمود :

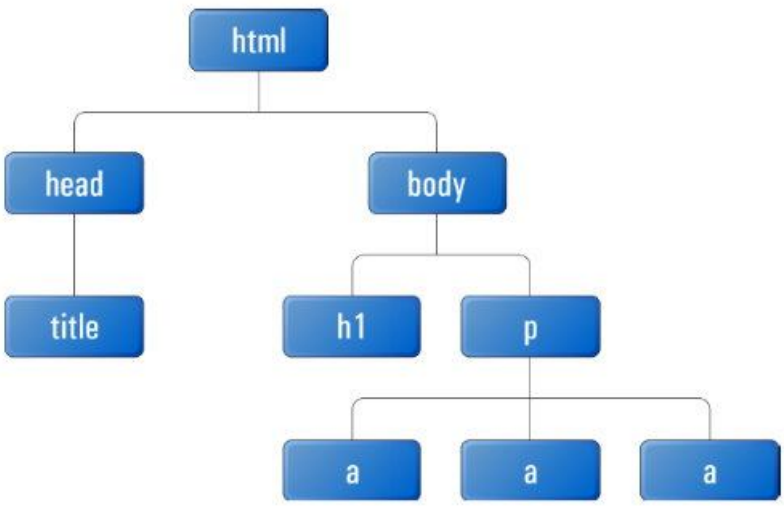
```
Window.MoveTo(0,0);
Window.resizeTo(screen.availWidth,screen.availHegiht);
```

DOM BASIC (اساس مدل شی گرای سند)

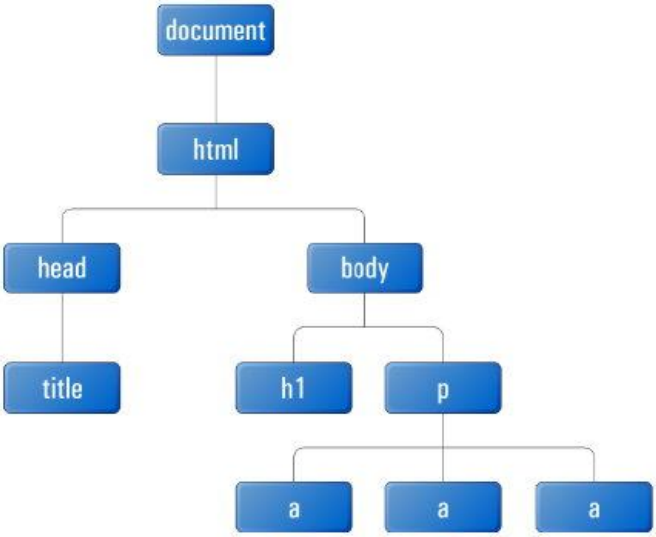
مدل شی گرای DOM به طراحان وب امکان دسترسی و دستکاری عناصر یک صفحه HTML را می دهد . این مدل عناصر موجود در یک صفحه HTML را به صورت درختی از گره ها ترسیم می کند به شکل زیر :

```
<html>
      <head>
            <title>DOMinating JavaScript</title>
      </head>
      <body>
      <h1>DOMinating JavaScript</h1>
      <p>If you need some help with your JavaScript, you might like to read articles
      from <a href=http://www.danwebb.net/ rel="external">DanWebb</a>,
      <a href="http://www.quirksmode.org/" rel="external">PPK</a>
      and
      <a href="http://adactio.com/" rel="external">Jeremy Keith</a>.
      </p>
      </body>
</html>
```

این کد را می توان در قالب درخت زیر نمایش داد :



همانطور که می بینید می توان هر یک از عناصر موجود در صفحه را در قالب یک node (گره) نمایش داده می شود . اما همیشه در DOM گرهی ویژه به نام document وجود دارد که در بالاترین سطح درخت قرار گرفته و سایر گره ها را شامل می شود . با این فرض درخت فوق به شکل زیر تبدیل خواهد شد :

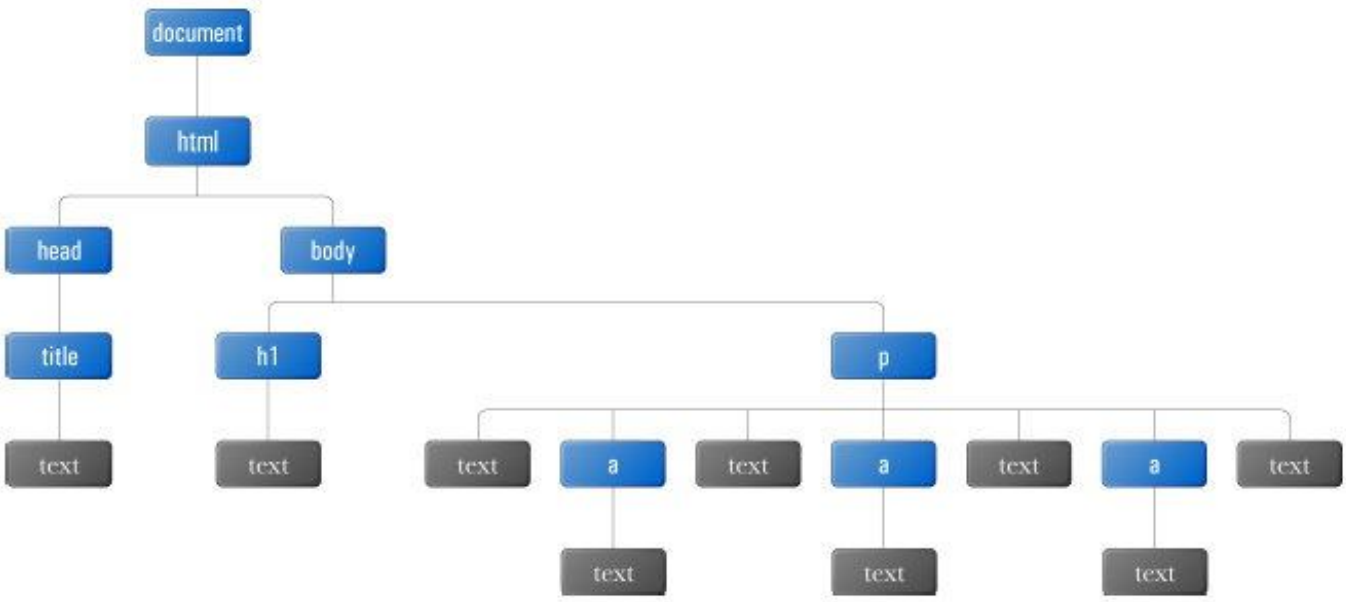


در درخت بالا هر مستطیل به عنوان یک گره (node) محسوب می شود . گره ها انواع مختلفی دارند که بعضی از آن ها به شرح زیر است :

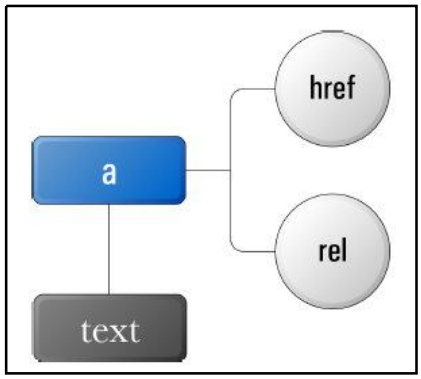
Document : بالاترین گرهی که همه گره های دیگر به آن متصل هستند (فرزند آن هستند.) به این نوع گره ، document Node (گره سند) گفته می شود .

Element : گرهی که شامل یک عنصر از صفحه باشد . این گره شامل یک تگ آغازی و یک تگ پایانی مانند <tag></tag> یا <tag /> است . این نوع گره تنها نوعی است که می تواند شامل فرزندان از انواع دیگر باشد . به این گره ها ، element Node (گره عنصری) گفته می شود .

Text : این نوع گره ها به متن داخل یک تگ آغازی و تگ پایانی اشاره دارند . این نوع گره ها هم نمی توانند فرزند داشته باشند . به این نوع گره ها ، text Node (گره متنی می گویند.) . اگر گره های متنی را هم به مثالی که بررسی کردیم اضافه کنیم درخت ما به شکل زیر تبدیل خواهد شد :



Attr : گرهی که به یک صفت از یک عنصر اشاره می کند و فاقد فرزند می باشد . به این نوع گره ها ، attribute Node (گره صفتی) گفته می شود . در درخت DOM معمولا این گره ها را به صورت دایره ای و متصل به گره های عنصری نمایش می دهند . به عنوان مثال هر یک از عناصر لینکی که در مثال بالا مشاهده می شود دارای صفت های href و rel هستند که می توان آن ها را به صورت زیر نمایش داد :



Comment : به گره های توضیحی اشاره می کند و فاقد فرزند است . (در واقع به تگ comment صفحه اشاره می کند .)
غالبا گرهی اصلی به عنوان راس این درخت وجود دارد که همان document است .
گره ها از نظر جاوااسکریپت به عنوان یک شی ء در نظر گرفته می شود که این اشیا می توانند خاصیت ها و متد هایی داشته باشند .
بعضی از آن ها به شرح زیر هستند :

خاصیت / متد	نوع / نوع بازگشتی	توضیحات
nodeName	**String**	نام گره را بر می گرداند . این خاصیت بستگی به نوع گره دارد .
nodeValue	**String**	مقدار گره را بر می گرداند . این خاصیت بستگی به نوع گره دارد .
nodeType	**Number**	یکی از انوع گره را بر می گرداند .
ownerDocument	**Document**	اشاره به شی **document** ی که گره جزئی از آن است دارد
firstChild	**Node**	اشاره به اولین گره از لیست گره ها دارد .
lastChild	**Node**	اشاره به آخرین گره از لیست گره ها دارد .
childNodes	**NodeList**	لیست (آرایه) ای از تمام گره های داخل یک گره
previousSibling	**Node**	اشاره به گره همزاد (برادر) قبلی دارد . اگر همزاد قبلی وجود نداشت باشد مقدار **null** را بر میگرداند .
nextSibling	**Node**	اشاره به گره همزاد (برادر) بعدی دارد . اگر همزاد بعدی وجود نداشت باشد مقدار **null** را بر میگرداند .
hasChildNodes()	**Boolean**	در صورتی که آرایه **childNodes** دارای یک یا بیشتر از یک عضو (گره) باشد True را بر میگرداند
appendChild(*node*)	**Node**	آرگومان **node** را به انتهای آرایه **childNodes** اضافه می کند.
removeChild(*node*)	**Node**	آرگومان **node** را از انتهای آرایه **childNodes** حذف می کند.
replaceChild (*newnode, oldnode*)	**Node**	در آرایه **childNodes** ، **oldnode** را با **newnode** جابجا می کند .
insertBefore (*newnode, refnode*)	**Node**	در آرایه **childNodes** ، **newnode** را قبل از **refnode** قرار می دهد .

استفاده از DOM :

دسترسی به گره ها :

تکه کد زیر را در نظر بگیرید :

```
<html>
      <head>
            <title>DOM Example</title>
      </head>
      <body>
            <p>Hello World!</p>
            <p>Isn’t this exciting?</p>
            <p>You’re learning to use the DOM!</p>
      </body>
</html>
```

اولا برای دسترسی به عنصر HTML می توان از documentElement که یکی از خاصیت های شی ء document است استفاده کنیم . به صورت زیر :

```
var oHtml = document.documentElement ;
```

حال ما می توانیم با استفاده از این متغیر به عناصر head و body به صورت زیر دسترسی داشته باشیم :

```
var oHead = oHtml.firstChild;
var oBody = oHtml.lastChild;
```

راه دیگر به صورت زیر است :

```
var oHead = oHtml.childNodes[0];
var oBody = oHtml.childNodes[1];
```

برای بدست آوردن تعداد فرزندان یک گره :

```
alert(oHtml.childNodes.length); //outputs “2”
```

ما می توانیم از متدی موسوم به item() برای دسترسی نیز استفاده کنیم :

```
var oHead = oHtml.childNodes.item(0);
var oBody = oHtml.childNodes.item(1);
```

DOM همچنین از دستور document.body را برای دسترسی به عنصر body صفحه استفاده می کند .

```
var oBody = document.body;
```

ما می توانیم صحت رابطه های سه متغیر oHead ، oBody و oHtml را به صورت زیر نشان دهیم :

```
alert(oHead.parentNode == oHtml);       //outputs “true”
alert(oBody.parentNode == oHtml);       //outputs “true”
alert(oBody.previousSibling == oHead);  //outputs “true”
alert(oHead.nextSibling == oBody);      //outputs “true”
alert(oHead.ownerDocument == document); //outputs “true”
```

دسترسی به صفات عناصر:

DOM برای دسترسی و دستکاری صفات یک عنصر سه متد تعریف کرده است :

getAttribute(name) : مقدار صفتی به نام name را از عنصری خاص بر می گرداند .
setAttribute(name,new Value) : مقدار صفتی به نام name را برابر new Value قرار می دهد .
removeAttribute(name) : صفتی به نام name را از عنصری مشخص حذف می کند .

این متد ها برای دسترسی و دستکاری مستقیم صفت های یک عنصر بسیار مناسب اند . بنابراین برای به دست آوردن مقدار صفت ID تگی مشخص می توان به صورت زیر عمل نمود :

```
var sId = oP.getAttribute(“id”);
```

و برای تغییر مقدار صفت Id به صورت زیر عمل می کنیم :

```
oP.setAttribute(“id”, “newId”);
```

دسترسی به گره های خاص :

ما تا اینجا با دسترسی به گره های فرزند و پدری آشنا شدیم . اما اگر بخواهیم به یک گره خاص ، آن هم در عمق یک درخت دسترسی داشته باشیم چه ؟ برای آسانی این کار ، DOM چندین متد برای دسترسی مستقیم به node ها فراهم آورده است .

getElementsByTagName() :

از این متد برای دسترسی به لیستی از عناصر خاص استفاده می شود .

```
var oImgs = document.getElementsByTagName(“img”);
```

دستور فوق لیستی از تمام عناصر img صفحه را در oImgs ذخیره می کند .
فرض کنید می خواهیم به اولین عنصر عکس اولین پاراگراف صفحه دسترسی داشته باشیم :

```
var oPs = document.getElementsByTagname(“p”);
var oImgsInP = oPs[0].getElementsByTagName(“img”);
```

ما می توانیم از دستور زیر برای دسترسی به تمام عناصر صفحه استفاده کنیم :

```
var oAllElements = document.getElementsByTagName(“*”);
```

getElementsByName() :

DOM برای دسترسی به عناصری که صفت name آنها برابر با مقداری خاص است از این متد استفاده می کند . به مثال زیر توجه کنید :

```
<html>
   <head>
   <title>DOM Example</title>
   </head>
<body>
   <form method=”post” action=”dosomething.php”>
   <fieldset>
         <legend>What color do you like?</legend>
               <input type=”radio” name=”radColor” value=”red” /> Red<br />
               <input type=”radio” name=”radColor” value=”green” /> Green<br />
               <input type=”radio” name=”radColor” value=”blue” /> Blue<br />
         </fieldset>
   <input type=”submit” value=”Submit” />
   </form>
</body>
</html>
```

این صفحه رنگ مورد علاقه کاربر را سوال می کند . radiobutton ها اسم یکسانی دارند . اما ما می خواهیم فقط مقدار radiobutton ی که انتخاب شده است را پیدا کنیم . برای ایجاد ارجاعی به عناصر radiobutton می توان از کد زیر استفاده نمود .

```
var oRadios = document.getElementsByName(“radColor”);
```

حال می توانید از همان روش قبلی برای به دست آوردن مقدار هر از radiobutton ها به روش زیر عمل کنید :

```
alert(oRadios[0].getAttribute(“value”)); //outputs “red”
```

getElementById() :

از این متد برای دسترسی به عناصر به وسیله خاصیت id آنها استفاده می شود . می دانیم که خاصیت id باید یکتا باشد به این معنی که هیچ دو عنصری نمی توانند داخل یک صفحه id یکسانی داشته باشند . این سریعترین و رایجترین راه برای دسترسی به عنصری خاص از صفحه است . به کد زیر نگاه کنید :

```
<html>
<head>
   <title>DOM Example</title>
</head>
<body>
   <p>Hello World!</p>
   <div id=”div1”>This is my first layer</div>
</body>
</html>
```

اگر ما از متد getElementsByTagName() برای دسترسی به عنصر div این صفحه با شناسه div1 بخواهیم استفاده کنیم باید به صورت زیر عمل کنیم :

```
var oDivs = document.getElementsByTagName(“div”);
var oDiv1 = null;
for (var i=0; i < oDivs.length; i++){
   if (oDivs[i].getAttribute(“id”) == “div1”) {
         oDiv1 = oDivs[i];
         break;
   }
}
```

اما ما می توانیم همین کار را به صورت زیر و با استفاده از متد getElementById() انجام دهیم :

```
var oDiv1 = document.getElementById(“div1”);
```

می بینید که استفاده از حالت دوم بسیار ساده تر ، کوتاه تر و بهینه تر است .

ایجاد و دستکاری گره ها :

ما می توانیم از DOM برای اضافه کردن ، حذف کردن و جابه جا کردن و دیگر دستکاری ها استفاده کنیم .

ایجاد گره های جدید:

برای ایجادگره های جدید از متد های زیر استفاده می شود :

createAttribute(name) : برای ایجاد یک صفت جدبد با name گرفته شده به کار می رود

createComment(text) : برای ایجاد یک توضیح

createElement(tagname) : برای ایجاد یک عنصر جدید استفاده می شود .

createTextNode(text) : ایجاد یک متن ساده با عنوان text

createElement(), createTextNode(), appendChild()

فرض کنید تکه کد زیر را داریم :

```
<html>
        <head>
                <title>createElement() Example</title>
        </head>
        <body>
        </body>
</html>
```

حال می خواهیم عبارت زیر را در این صفحه چاپ کنیم :

```
<p>Hello World !</p>
```

اولین کار ایجاد یک عنصر p است .

```
var oP = document.createElement("p");
```

حال یک متن ساده ایجاد می کنیم :

```
var oText = document.createTextNode("Hello World!");
```

حال باید متن را به عنصر p ، append کنیم . برای این کار از متد appendchild() استفاده می کنیم . از این متد برای اضافه کردن یک فرزند به انتهای لیست فرزندان یک گره استفاده می شود .

```
oP.appendChild(oText);
```

پاراگرافی که را ما ایجاد کرده ایم باید به صفحه و قسمت body و یا یکی از زیر مجموعه های آن append کنیم . برای این کار :

```
oP.appendChild(oText);
```

removeChild(), replaceChild(), insertBefore()

طبیعتا وقتی می توانیم گرهی را اضافه کنیم می توانیم آن ها را حذف کنیم . برای حذف گره ها ازمتد removeChild() استفاده می کنیم . این متد یک آرگومان می گیرد که در واقع گرهی است که باید حذف شود . به شکل زیر :

```
var oP = document.body.getElementsByTagName("p")[0];
document.body.removeChild(oP);
```

برای جابجایی گره ها از متد replaceChild() استفاده می شود . از این تابع به صورت زیر استفاده می شود :

```
var oNewP = document.createElement(“p”);
var oText = document.createTextNode(“Hello Universe! “);
oNewP.appendChild(oText);
var oOldP = document.body.getElementsByTagName(“p”)[0];
oOldP.parentNode.replaceChild(oNewP, oOldP);
```

برای اضافه کردن یک عنصر به قبل از عنصر دیگری از ()insertBefore استفاده می شود . این متد دو آرگومان می پذیرد و آرگومان اول را قبل از آرگومان دوم قرار می دهد .

createDocumentFragment()

به محض اینکه ما تعدادی گره جدید به سند اضافه می کنیم صفحه برای نمایش تغییرات ، update میشود . این رفتار برای تعداد تغییرات کم مناسب است . اما هنگامی که تغییرات زیاد باشد و صفحه بخواهد این رفتار را یک به یک در صفحه نمایش دهد ممکن است این عمل به کندی انجام شود .

برای رفع این مشکل می توانید از یک تکه (documentFragment) برنامه استفاده کنید .شما می توانید تمام گره های جدید را به تکه برنامه اضافه کرده و سپس آن را در صفحه اصلی قرار دهید . فرض کنید می خواهیم چندین پاراگراف را در صفحه ایجاد کنیم . در صورت استفاده از روش های قبلی این امر موجب رفرش هر باره صفحه خواهد شد .

اما بهتر است به روش زیر عمل کنیم :

```
var arrText = [“first”, “second”, “third”, “fourth”, “fifth”, “sixth”];
var oFragment = document.createDocumentFragment();
        for (var i=0; i < arrText.length; i++) {
                var oP = document.createElement(“p”);
                var oText = document.createTextNode(arrText[i]);
                oP.appendChild(oText);
                oFragment.appendChild(oP);
        }
document.body.appendChild(oFragment);
```

ویژگی های منحصر به فرد DOM برای HTML :

یکی از ویژگی های DOM این است که DOM امکان تنظیم و دستکاری صفات مربوط به عناصر HTML را فراهم می آورد . از جمله این ویژگی ها می توان به در نظر گرفتن صفات عناصر به عنوان خاصیت های هر شیءاشاره کرد که برای این کار متد ها و خاصیت های ارائه شده است .

ما می توانیم به صفات عناصر به عنوان خاصیت های آن دسترسی داشته باشیم . فرض کنید کد زیر را داریم :

```
<img src=”mypicture.jpg” border=”0” />
```

برای دسترسی و تنظیم src و border می توانیم از متد های ()getAttribute و یا ()setAttribute استفاده کنیم :

```
alert(oImg.getAttribute(“src”));
alert(oImg.getAttribute(“border”));
oImg.setAttribute(“src”, “mypicture2.jpg”);
oImg.setAttribute(“border”, “1”);
```

ما می توانیم از نام صفات هم به عنوان خاصیت هر یک از اشیا برای get و set کردن استفاده کنیم :

```
alert(oImg.src);
alert(oImg.border);
oImg.src = “mypicture2.jpg”;
oImg.border = “1”;
```

- ✓ **نکته : بر خلاف بسیاری از صفات تگ ها ، ما نمی توانیم از خود صفت class به عنوان یک خاصیت استفاده کنیم . چون این کلمه جزء کلمات رزرو شده است و باید به جای آن از کلمه className استفاده کنیم .**

متد های مربوطه به جداول :

فرض کنید که می خواهیم جدول زیر را به صورت پویا و با استفاده از جاوااسکریپت ایجاد کنیم :

```
<table border="1" width="100%">
        <tbody>
                <tr>
                     <td>Cell 1,1</td>
                     <td>Cell 2,1</td>
                </tr>
                <tr>
                     <td>Cell 1,2</td>
                     <td>Cell 2,2</td>
                </tr>
        </tbody>
</table>
```

اگر برای ایجاد این جدول بخواهیم از متد های رایج DOM استفاده کنیم کد ما به صورت ذیل بسیار طولانی و گاهی اوقات سردرگم کننده خواهد شد :

```
//create the table
        var oTable = document.createElement("table");
        oTable.setAttribute("border", "1");
        oTable.setAttribute("width", "100%");

//create the tbody
        var oTBody = document.createElement("tbody");
        oTable.appendChild(oTBody);

//create the first row
        var oTR1 = document.createElement("tr");
        oTBody.appendChild(oTR1);
        var oTD11 = document.createElement("td");
        oTD11.appendChild(document.createTextNode("Cell 1,1"));
        oTR1.appendChild(oTD11);
        var oTD21 = document.createElement("td");
        oTD21.appendChild(document.createTextNode("Cell 2,1"));
        oTR1.appendChild(oTD21);

//create the second row
        var oTR2 = document.createElement("tr");
        oTBody.appendChild(oTR2);
        var oTD12 = document.createElement("td");
        oTD12.appendChild(document.createTextNode("Cell 1,2"));
        oTR2.appendChild(oTD12);
        var oTD22 = document.createElement("td");
        oTD22.appendChild(document.createTextNode("Cell 2,2"));
        oTR2.appendChild(oTD22);

//add the table to the document body
        document.body.appendChild(oTable);
```

برای آسانی اینکار DOM یکسری خاصیت ها و متد های منحصر به فردی برای عناصر اصلی جداول همچون table, tody, tr ایجاد کرده است .

متد ها و خاصیت های منحصر به فرد جدول به شرح زیر می باشد :

caption : اشاره به عنصر caption جدول دارد . (البته اگر وجود داشته باشد.)

tBodies : مجموعه (آرایه) ای از عناصر tbody

tFoot : اشاره به عنصر tfoot جدول

tHead : اشاره به عنصر thead جدول

Rows : مجموعه ای از تمام ردیف های جدول

createThead() : ایجاد و قرار دادن یک عنصر جدید thead در جدول

createTfoot() : ایجاد و قرار دادن یک عنصر جدید tfoot در جدول

createCaption() : ایجاد و قرار دادن یک عنصر جدید caption در جدول

deleteThead() : حذف عنصر thead از جدول

deleteTfoot() : حذف عنصر tfoot از جدول

deleteCaption() : حذف عنصر Caption از جدول

deleteRow(position) : حذف ردیفی از جدول که در موقعیت position قرار دارد

insertRow(position) : قرار دادن ردیفی در موقعیت position

متد ها و خاصیت های tbody :

Rows :مجموعه از ردیف ها در عنصر tbody

deleteRow(position) : حذف ردیفی در موقعیت position

insertRow(position) : قراردادن ردیفی در موقعیت position مجموعه ای از ردیف ها

متد ها و خاصیت های tr :

Cells : مجموعه ای از سلو ل ها در یک ردیف

deleteCell(position) : حذف سلولی در موقعیت position

insertCell(position) : قرار دادن سلولی در موقعیت position مجموعه ای از سلول ها .

برای ایجاد جدول قبلی کد ما به صورت زیر خواهد بود :

```
//create the table
        var oTable = document.createElement("table");
        oTable.setAttribute("border", "1");
        oTable.setAttribute("width", "100%");
//create the tbody
        var oTBody = document.createElement("tbody");
        oTable.appendChild(oTBody);
//create the first row
        oTBody.insertRow(0);
        oTBody.rows[0].insertCell(0);
        oTBody.rows[0].cells[0].appendChild(document.createTextNode("Cell
        1,1"));
        oTBody.rows[0].insertCell(1);
        oTBody.rows[0].cells[1].appendChild(document.createTextNode("Cell
        2,1"));
//create the second row
        oTBody.insertRow(1);
        oTBody.rows[1].insertCell(0);
        oTBody.rows[1].cells[0].appendChild(document.createTextNode("Cell
        1,2"));
        oTBody.rows[1].insertCell(1);
        oTBody.rows[1].cells[1].appendChild(document.createTextNode("Cell
        2,2"));
//add the table to the document body
        document.body.appendChild(oTable);
```

کار با فرم ها و عناصر فرم از طریق جاوا اسکریپت :

فرم ها در صفحات تنها عناصری هستند که کاربران می توانند به صورت مستقیم یکسری اطلاعات را در آن ها وارد نمایند .

برای ایجاد یک فرم از تگ form و برای ایجاد عناصر آن از تگ هایی همچون input ، select ، textarea و .. استفاده می شود که مرورگر ها بوسیله آن ها قادر به نمایش فیلد های یک خطی ، چند خطی ، منوهای بازشو ، دکمه ها و ... هستند .

اساس یک عنصر فرم در صفحه :

یک فرم در صفحه بوسیله تگ form که دارای صفت های زیر می باشد ایجاد می شود :

Method : مشخص می کند که مرورگر از چه روشی برای ارسال داده های فرم استفاده کند که می تواند دو مقدار GET و POST را بگیرد .

Action : فرم ها پس از ارسال باید به یک صفحه پردازشگر که البته به یکی از زبان های server side (تحت سرور) نوشته می شوند هدایت شوند . این صفت آدرس (URL) صفحه پردازشگر فرم را مشخص می کند .

Enctype : نوع Encoding داده های فرم را هنگام ارسال مشخص می کند که در حالت پیش فرض برابر application/x-url-encoded است . اما در حالتی که داخل فرممان عنصری از نوع file که کاربران را قادر به آپلود فایل هایشان می کند باشد باید آن را برابر multipart/form-data قرار دهیم .

Accept : لیستی از MIME type های فایل هایی که قرار است کاربر بتواند آپلود کند را مشخص می کند .

Accept-charset : لیستی ازمجموعه کاراکتری هایی را که سرور باید در هنگام دریافت اطلاعات استفاده کند را مشخص میکند .

اسکریپت نویسی برای دسترسی به عناصر فرم :

کدنویسی برای عناصر فرم نسبت به عناصر دیگر کمی متفاوت است .

ایجاد ارجاع (reference) به عناصر مورد نظر :

قبل از سر و کار داشتن با عناصر form باید ارجاعی به فرم مورد نظرمان در صفحه ایجاد کنیم . این کار از چندین راه انجام می شود .

راه اول استفاده از متد getElementById() است که از Id فرم برای دسترسی به آن استفاده می کند .

راه دوم استفاده از آرایه ی forms[] است که به عنوان یکی از خاصیت های شی document در DOM معرفی شده است .

برای این کار می توان از اندیس عددی که بستگی به مکان فرم مورد نظر در صفحه دارد استفاده کرد . به عنوان مثال :

```
var oForm = document.forms[0] ;  // get the first form
var oOtherForm = document.forms["formZ"] ;  // get the form whose name is
"formZ"
```

دسترسی به عناصر داخل یک فرم :

هر عنصر داخل یک فرم مثل یک دکمه ، یک فیلد یک خطی و ... با استفاده از آرایه ای به نام elements[] که یکی از خاصیت های یک شیء از نوع فرم است قابل دسترسی هستند .

شما می توانید از این آرایه و با استفاده از اندیس عددی یا اسمی مورد نظر به عناصر مختلف فرم دسترسی داشته باشید .

```
var oFirstField = oForm.elements[0] ; // get the first form field
var oTextbox1 = oForm.elements["textbox1"] ; // get the field with the name
"textbox1"
```

خط اول از کد بالا متغیری را تعریف می کند که به اولین عنصر از فرمی به نام oForm اشاره می کند .

خط دوم نیز متغیری را تعریف می کند که به عنصری به نام textbox1 از فرمی به نام oForm اشاره می کند .

یک روش دیگر (که اصطلاحا به آن روش میانبر می گویند) برای دسترسی به عناصری که نام مشخصی دارند استفاده می شود به شکل زیر است :

```
var oTextbox1 = oForm.textbox1;  //get the field with the name "textbox1"
```

کد بالا متغیری تعریف می کند که به عنصری با نام (یا Id) textbox1 از فرمی به نام oForm اشاره می کند .

اگر اسم عنصر مورد نظر دارای چند space باشد باید در اطراف آن از براکت ([]) استفاده کنیم :

```
var oTextbox1 = oForm.textbox1; //get the field with the name "textbox1"
```

ویژگی ها و خاصیت های عناصر form :

تمامی عناصر فرم (به جز عنصری از نوع Hidden) شامل یکسری خواص و رویدادهای مشترکی هستند که در زیر بیان می کنیم :

خاصیت disabled : از این خاصیت هم برای تشخیص اینکه کدام عنصر در حالت غیر فعال قرار دارد و هم برای فعال یا غیر فعال کردن یک عنصر از قبل فعال استفاده می شود .

خاصیت form : اشاره به فرمی دارد که عنصر مورد نظر ما ، داخل آن قرار دارد .

متد ()focus : این متد موجب می شود focus (تمرکز) صفحه بر روی عنصر مورد نظر قرار گیرد .

متد ()blur : این متد عکس متد بالا است و موجب می شود focus (تمرکز) صفحه از روی عنصر مورد نظر برود .

رویداد blur : این رویداد موقعی که focus (تمرکز) صفحه از روی عنصر مورد نظر برود رخ می دهد .

رویداد focus : عکس رویداد بالا عمل می کند و موقعی که focus (تمرکز) بر روی عنصر مورد نظر قرار بگیرد رخ می دهد .

برای مثال :

```
var oField1 = oForm.elements[0];
var oField2 = oForm.elements[1];
//set the first field to be disabled
oField1.disabled = true;
//set the focus to the second field
oField2.focus();
//is the form property equal to oForm?
alert(oField1.form == oForm); //outputs "true"
```

- ✓ **نکته : عناصر از نوع hidden فقط از خاصیت form که در بالا ذکر شد پشتیبانی می کند .**

Submit (ارسال) فرم بوسیله جاوااسکریپت :

در HTML فرستادن فرم از طریق یک دکمه از نوع submit یا عکسی که در نقش دکمه submit عمل می کند انجام می شود .

مثال :

```
<input type="submit" value="Submit" />
<input type="image" src="submit.gif" />
```

در صورت کلیک بر روی هر یک از دکمه های بالا فرم به صورت معمولی ارسال می شود .

اگر شما دکمه Enter را هم از صفحه کلید فشار دهید مرورگر فرم را مثل حالتی که دکمه کلیک می شود ارسال می کند .

شما برای تست ارسال شدن فرم می توانید از کد ساده زیر در تگ آغازین فرم مورد نظرتان استفاده کنید :

```
<form method="post" action="javascript:alert('Submitted')">
```

اگر شما می خواهید که از هیچ یک از راه های فوق استفاده نکنید می توانید از متدی به نام ()submit استفاده کنید .

این متد جزئی از تعریفات DOM برای یک عنصر form است و می تواند هر جایی از صفحه استفاده شود. برای این کار اولا باید ارجاعی به فرم مورد نظر ایجاد کرد (طبق روش هایی که قبلا ذکر شد) :

```
oForm = document.getElementById("form1");
oForm = document.forms["form1"];
oForm = document.forms[0];
```

بعد از این کار شما می توانید به راحتی از این متد استفاده کنید :

```
oForm.submit();
```

ارسال form فقط یکبار !!!

یکی از مشکلاتی که طراحان در فرم ها با آن روبرو هستند این است که بسیاری از کاربران برای اطمینان از اینکه فرم به درستی ارسال شود چندین بار بر روی دکمه submit کلیک می کنند . مشکلی که در اینجا هست این است که به ازای هر بار کلیک کاربر بر روی دکمه یک Request (درخواست) اضافی به سرور ارسال می شود .

راه حل این مشکل بسیار ساده است : بعد از اینکه کاربر دکمه را کلیک کرد ، ما آن را غیر فعال (disabled) می کنیم .
برای انجام اینکار می توان به جای استفاده از دکمه submit معمولی زیر :

```
<input type="submit" value="Submit" />
```

از کد زیر استفاده کرد :

```
<input type=”button” value=”Submit” onclick=”this.disabled=true;
this.form.submit()” />
```

موقعی که این دکمه کلیک می شود اولا خود دکمه غیر فعال می شود و سپس فرمی را که جزئی از آن است را ارسال می کند . توجه کنید که دراینجا کلمه کلیدی this به دکمه اشاره دارد و form به فرم دربرگیرنده دکمه اشاره می کند .
همانطور که یک فرم را می توانیم بوسیله متد ()submit ارسال کنیم می توانیم آن را به وسیله متدی به نام ()reset نیز reset (پاک سازی) کنیم :

```
<input type=”button” value=”Reset” onclick=”document.forms[0].reset()” />
```

کار با textbox ها :

دو نوع text box در html مورد استفاده قرار می گیرد .
یک خطی :

```
<input type="text"/>
```

و چند خطی :

```
<textarea>Content</textarea>
```

برای درست کردن یک textbox یک خطی می بایست صفت type عنصر input را برابر text قرار دهیم . صفت size طول textbox را بر حسب تعداد کاراکترها مشخص می کند . مقدار صفت value مقدار پیش فرض موجود داخل textbox را مشخص می کند . صفت maxlength حداکثر تعداد کاراکتر هایی که بتوان در textbox را وارد کرد را مشخص می کند .

```
<input type="text" size="25" maxlength="50" value="initial value" />
```

عنصر textarea برای ایجاد فیلد های چند خطی مورد استفاده قرار می گیرد . از صفت های rows و cols برای مشخص کردن طول و عرض textarea استفاده می شود .

```
<textarea rows="25" cols="5">initial value</textarea>
```

بر خلاف input این عنصر امکان مشخص کردن حداکثر تعداد کاراکتر های ورودی را ندارد .

بازیابی و تغییر مقدار یک textbox :

اگر چه هر دو عنصر بالا تفاوت هایی دارند اما هر دوی آن ها از خاصیتی به نام value برای بازیابی مقدار وارد شده در آن ها پشتیبانی می کنند .
به عنوان مثال برای بازیافت مقدار وارد شده در فیلدی به نام (یا Id) txt1 می توان به صورت زیر عمل کرد :

```
var oTextbox1 = document.getElementById("txt1");
```

چون مقداری که خاصیت value برمی گرداند یک رشته ساده است می توان از تمامی متد ها و خواصی که قبلا برای رشته ها اشاره کردیم استفاده کرد .

```
alert ('oTextbox1.length');
```

از این خاصیت برای قراردادن مقادیر جدید در textbox ها نیز می توان استفاده کرد . به عنوان مثال با دستور زیر می توان مقادیر جدیدی را به oTextbox1 (که در بالا ذکر شد) اضافه کنیم :

```
oTextbox1.value='first textbox';
```

انتخاب متن های داخل textbox ها :

هر دو نوع فیلد بالا از متدی به نام ()select برای انتخاب تمامی متن داخل آن ها پشتیبانی می کنند .
برای این کار اولا تمرکز (focus) صفحه باید بر روی آن قرار گیرد . برای اطمینان از این امر باید همیشه قبل از متد ()select از متدی به نام ()focus استفاده نمایید . (البته این کار در تمامی مرورگر ها الزامی نیست اما بهتر است همیشه انجام شود .)
به عنوان مثال برای انتخاب تمامی متن موجود در textbox بالا :

```
oTextbox1.focus();
oTextbox1.select();
```

رویداد های textbox ها :

هر دو نوع فیلد بالا علاوه بر پشتیبانی از رویداد های blur و focus از دو رویداد جدید به نام های change و select نیز پشتیبانی می کنند .
Change : این رویداد وقتی رخ می دهد که کاربر بعد از تغییر متن داخل textbox ها ، آن ها را از حالت تمرکز صفحه خارج کند .
Select : این رویداد وقتی رخ می دهد که یک یا چند کاراکتر از رشته های داخل یک textbox چه به صورت دستی یا توسط متد ()select انتخاب شوند .
تفاوت رویداد های change و blur این است که رویداد blur تنها زمانی رخ می دهد که تمرکز صفحه از عنصر مورد نظر خارج شود و رویداد change نیز وقتی رخ می دهد که علاوه بر تغییر متن داخل textarea ها ، تمرکز صفحه نیز از آن ها خارج می شود .
اگر متن داخل textbox ثابت باشد و فقط تمرکز صفحه از عنصر برود blur رخ می دهد اما اگر متن هم تغییر کرده باشد ابتدا رویداد change و به دنبال آن blur رخ خواهد داد .

انتخاب خودکار متن درون textbox ها :

برای انتخاب خودکار متن درون یک textbox هنگامی که تمرکز صفحه بر روی آن ها می رو د می توان به راحتی از دستور this.select() در رویداد onFocus عنصر مورد نظر استفاده نمود .
به عنوان مثال :

```
<input type="text" onfocus="this.select();" />
<textarea onfocus="this.select()"></textarea>
```

چرخش Tab بین عناصر فرم به صورت خودکار :

بعد از تکمیل textfield هایی که تعداد کاراکتر های مشخصی را قبول می کنند می توانید کنترل (تمرکز) صفحه را به دیگر عناصر صفحه منتقل کنید .

برای این کار می توانیم از صفت maxlength درتگ های input استفاده کنیم :

```
<input type="text" maxlength="4" />
```

کاری که باید در اینجا انجام دهیم تشخیص وارد شدن حداکثر کاراکتر ها و فراخوانی متد ()focus برای عنصر فرم بعدی است . برای این کار از تابعی به نام test استفاده می کنیم :

```
function test(oTextbox){
 var oForm = oTextbox.form;

//make sure the textbox is not the last field in the form
  if (oForm.elements[oForm.elements.length-1] != oTextbox
     && oTextbox.value.length == oTextbox.getAttribute("maxlength")) {
         for (var i=0; i < oForm.elements.length; i++) {
            if (oForm.elements[i] == oTextbox) {
               for(var j=i+1; j < oForm.elements.length; j++) {
                  if (oForm.elements[j].type != "hidden") {
                       oForm.elements[j].focus();
                       return;
                  }
               }
             return;
            }
        }
     }
};
```

تابعی که ما نوشتیم باید بعد از هر بار وارد کردن کاراکتر داخل textbox فراخوانی می شود . برای اینکار از رویداد onKeyUp استفاده خواهیم کرد به صورت زیر :

```
<input type='text' maxlength='4' onKeyUp='test(this)' />
```

محدود کردن کاراکتر های ورودی در یک textarea :

اگر چه یک textfield دارای صفتی به نام maxlength برای محدودکردن کاراکتر های ورودی است اما یک textarea فاقد این صفت است . اما ما می توانیم توسط یک کد ساده javascript اینکار را انجام دهیم .

برای این کار ابتدا تابعی به نام ()isNotMax تعریف خواهیم کرد . به صورت زیر :

```
Function isNotMax(oTextbox){
  Return oTextbox.value.length != oTextarea.getAttribute('maxlength') ;
}
```

همانطور که می بینید این تابع خیلی ساده است . فقط تعداد کاراکتر های وارد شده در textbox را با صفت maxlength عنصر مورد نظر مقایسه می کند و در صورتی که برابر نباشد True و در غیراینصورت False را بر میگرداند .

توجه داشته باشید صفت maxlength برای textarea صفتی غیر استاندارد است اما ما می توانیم توسط متد getAttribute مقدار آن را بدست آوریم .

در مرحله بعد ما باید این تابع را در رویداد onKeyPress عنصرمان فراخوانی می کنیم . این رویداد قبل از وارد کردن هر کاراکتر رخ خواهد داد که دقیقا زمانی است که باید به حداکثر رسیدن تعداد کارکتر های ورودی را چک کنیم . چیزی مثل کد زیر :

```
<textarea rows='10' cols='25' maxlength='150' onKeyPress='return
isNotMax(this)'></textarea>
```

توجه کنید که مقدار برگشتی از تابع به کنترل کننده ی رویداد onKeyPress فرستاده می شود . البته این شیوه از راه های قدیمی کنترل رفتار پیش فرض یک رویداد است .

موقعی که تعداد کاراکتر های ورودی از MAX کمتر باشد تابع True به معنی ادامه رفتار عادی رویداد را بر می گرداند در غیر این صورت موجب جلوگیری از رفتار عادی رویداد و در نتیجه کاراکتر های بیش از حد مجاز خواهد شد .

کار با listbox ها و combobox ها :

Listbox ها و combobox ها در HTML بوسیله تگی به نام select ایجاد می شوند که به صورت پیش فرض مرورگر ها این عنصر را به صورت combobox نشان می دهند .

```
<select name="selAge" id="selAge">
      <option value="1">18-21</option>
      <option value="2">22-25</option>
      <option value="3">26-29</option>
      <option value="4">30-35</option>
      <option value="5">Over 35</option>
</select>
```

مقدار صفت value آیتمی که توسط کاربر انتخاب می شود به سرور فرستاده می شود .

برای نشان دادن یک listbox فقط کافی است صفتی به نام size را با مقداری که مشخص کننده ی تعداد آیتم های قابل نمایش به صورت پیش فرض است به تگ select اضافه کنید . به عنوان مثال کد زیر listbox ی با 5 آیتم نمایشی بصوریت پیش فرض را نمایش می دهد :

```
<select name="selAge" id="selAge" size="3">
      <option value="1">18-21</option>
      <option value="2">22-25</option>
      <option value="3">26-29</option>
      <option value="4">30-35</option>
      <option value="5">Over 35</option>
</select>
```

برای دسترسی به هر دو نوع عنصر فوق می توان طبق قواعدی که قبلا گفتیم عمل کنید :

```
oListbox = document.getElementById("selAge");
```

DOM برای تمامی عناصر select آرایه ای به نام option که هر خانه آن اشاره به option ی از آن عنصر دارد تعریف کرده است . ما می توانیم برای نمایش متن (عنوان) هر option و مقدار صفت value آن ها از روش های قبلی استفاده کنیم . مثلا :

```
oListbox = document.getElementById("selAge");
```

علاوه بر این هر option دارای خاصیتی به نام index است که در واقع موقعیت آن را در آرایه option مشخص می کند .

```
alert(oListbox.options[1].index); //outputs "1"
```

البته چون option یک آرایه است می توانیم از خاصیتی به نام length برای مشخص کردن تعداد کل option های select استفاده کنیم .

اما حال از کجا بفهمیم که کدام option (آیتم) توسط کاربر انتخاب شده است ؟

بازیافتن یا تغییر دادن option (ها)ی انتخاب شده :

عنصر select دارای خاصیتی به نام selectedIndex است که Index آیتم انتخاب شده را در خود نگه می دارد . و در صورتی که هیچ آیتمی انتخاب نشده باشد مقدار 1- را بر می گرداند .

اما همانطور که می دانید با اضافه کردن صفتی مانند multiple='multiple' به عنصر select امکان انتخاب بیش از یک آیتم در آنِ واحد امکان پذیر است .

در این صورت خاصیت selectedIndex حاوی اولین عنصر انتخاب شده از list خواهد بود اما این کمکی به ما نمی کند .

چون ما به تمام index تمام آیتم های انتخاب شده احتیاج داریم :

برای این کار ما احتیاج به یک تابع داریم .

این تابع در طول آیتم های یک listbox چرخش کرده و مقدار خاصیتی به نام selected که مشخص کننده ی انتخاب شدن یا نشدن آیتم است را بررسی کرده و index آن option را به آرایه ای اضافه می کند . خاصیت selected فقط می تواند یکی از مقادیر true (انتخاب شده) یا Fasle (انتخاب نشده) را در بر دارد .

```
function getSelectedIndexes (oListbox) {
   var arrIndexes = new Array;
      for (var i=0; i < oListbox.options.length; i++) {
         if (oListbox.options[i].selected) {
              arrIndexes.push(i);
         }
       }
    return arrIndexes;
};
```

از این تابع می توان هم برای بدست آوردن آیتم های انتخاب شده و هم تعداد آن ها استفاده کرد .

اضافه کردن option ها :

ما می توانیم از طریق جاوااسکریپت ، آیتم های جدیدی به list ها اضافه کنیم .

برای این کار تابعی با سه آرگومان می نویسیم :

list ی که می خواهیم روی آن کار کنیم ، نام آیتمی که می خواهیم اضافه کنیم و مقداری که می خواهیم اضافه کنیم .

بعد توسط متد های قبلی DOM یک عنصر option جدید ایجاد کرده و بعد آن را به عنصر select اضافه می کنیم

```
Function test (oListbox, sName, sValue) {
   var oOption = document.createElement("option");
   oOption.appendChild(document.createTextNode(sName));
          if (arguments.length == 3) {
               oOption.setAttribute("value", sValue);
          }
   oListbox.appendChild(oOption);
}
```

چون صفت value برای یک option اختیاری است می توانیم در صورتی که value برای تابع فرستاده شده است آن را به option اضافه کنیم . برای چک کردن اینکه value فرستاده شده یا نه از دستور arqument.length استفاده می کنیم .

حذف option ها :

جاوااسکریپت علاوه بر امکان اضافه کردن option ها ، امکان حذف آن ها را نیز فراهم می کند .

یکی از روش های قدیمی برای اینکار استفاده از آرایه ی options و قراردادن مقدار null برای عنصری ار آن که می خواهیم حذف کنیم است .

```
oListbox.remove(0); //remove the first option
```

روش بهتر و جدیدتر استفاده از متدی به نام ()remove است که آرگومان (index) عنصر مورد نظر برای حذف را می گیرد :

```
Function test (oListbox, iIndex) {
   oListbox.remove(iIndex);
}
```

اگر می خواهید هر یک از option های موجود در یک listbox را حذف کنید می توانید متد ()remove برای هر کدام از آن ها صدا بزنید .

```
Function test (oListbox) {
   for (var i=oListbox.options.length-1; i >= 0; i--) {
      ListUtil.remove(oListbox, i);
   }
}
```

کد بالا برای حذف ، آیتم ها را بر عکس طی می کند . این کار الزامی است چرا که با هر بار حذف شدن یک آیتم از لیست خاصیت index هر option دوباره چینی می شود .

به این دلیل همیشه بهتر است اول عنصری با بیشترین index حذف شود و سپس عناصر با index پایین تر.

رویداد ها در جاوااسکریپت

تعاملات جاوااسکریپت با HTML از طریق رخداد رویداد هایی که به واسطه دستکاری هایی که کاربر یا مرورگر بر روی صفحه انجام می دهد ، انجام می شود .

موقعی که صفحه بارگذاری می شود رویدادی رخ داده است ، موقعی که کاربر بر روی دکمه ای کلیک میکند ، باز هم رویدادی رخ داده است . طراحان می توانند از این رویداد ها برای اجرای کد هایی که به رویداد ها پاسخ می دهند استفاده کنند مثلا دکمه ای موجب بستن پنجره شود ، پیغامی را به کاربر نمایش دهد ، داده ها را ارزش سنجی کند و ... .

رویداد ها در واقع عملیات خاصی هستند که یا توسط کاربر یا توسط خود مرورگر انجام می شوند . این روبداد ها نام هایی همچون click ، load ، mouseover و ... دارند . اصطلاحا به تابعی که در پاسخ به یک رویداد صدا زده میشود event Handler (کنترلگر حادثه) می گویند . به عنوان مثال تابعی که برای پاسخ به رویداد click صدا زده می شود کنترلگر onclick نامیده می شود .

برای مشخص کردن کنترلگر های حادثه به دو روش می توان عمل کرد : **از طریق جاوااسکریپت یا از طریق** HTML .

برای مشخص کردن یک کنترلگر از طریق جاوااسکریپت ابتدا باید به شیء مورد نظر ارجاعی ایجاد کرده و سپس تابعی را به کنترلگر حادثه آن (که به صورت یک خاصیت برای آن تعریف شده است) منتسب می کنیم . برای مثال :

```
Var oDiv = document.getElementById('div1');
 oDiv.onclick= function (){
   alert('I Was Clicked !!!');
 }
```

دقت کنید که در این روش باید نام کنترلگر حادثه به صورت کوچک نوشته شود .

در روش دوم شما یک صفت کنترلگر حادثه را که اسکریپتی را به عنوان مقدار می پذیرد در تگ مربوطه قرار دهیم . به صورت زیر :

```
<div onclick='alert("I Was Clicked !!!")'></div>
```

در این روش نام کنترلگر حادثه می تواند به هر شکلی نوشته شود . در نتیجه onclick معادل است با : OnClick یا ONCLICK .

انواع رویداد ها :

رویداد هایی که در مرورگر رخ میدهند معمولا به چند دسته زیر تقسیم بندی می شوند :

- رویداد های mouse که وقتی کاربر از طریق ماوسش کارهایی را انجام می دهند ، رخ می دهند .
- رویدادهای keyboard که وقتی کاربر بوسیله keyboard دکمه ای را فشار می دهند رخ می دهند .
- رویداد های HTML که موقعی که تغییراتی در پنجره مرورگر انجام می شوند رخ می دهند .
- رویداد های تغییر که زمانی که تغییراتی در ساختار DOM صفحه انچام می شود رخ می دهند .

رویدادهای Mouse : رایج ترین رویداد هایی هستند که رخ می دهند و به شرح زیر می باشند :

- ✓ Onclick : موقعی که کاربر دکمه چپ mouse را فشار می دهد رخ می دهد . (نه دکمه راست) . هنگامی که تمرکز صفحه بر روی یک دکمه باشد و کاربر کلید Enter را هم بزند این رویداد رخ میدهد .
- ✓ Ondblclick : موقعی که کاربر دو بار دکمه چپ mouse را کلیک می کند رخ می دهد .
- ✓ Onmousedown : موقعی که کاربر هر دکمه ای از mouse را فشار دهد رخ می دهد .
- ✓ onMouseOut : موقعی رخ میدهد که نشانگر موس بر روی عنصر است و کاربر آن را به بیرون از محدوده عنصر هدایت میکند .
- ✓ onmouseover : موقعی رخ می دهد که نشانگر موس از خارج از عنصر بر روی ان هدایت می شود .
- ✓ onmouseup : موقعی رخ می دهد که هردکمه ای از mouse رها می شود .
- ✓ Onmousemove : مکررا هنگامی که نشانگر موس بر روی عنصری است رخ می دهد .

تمامی عناصر موجود دریک صفحه از رویدادهای فوق به خوبی پشتیبانی می کنند .

ترتیب اجرایی رویداد ها :

قبل از رخداد رویداد click همیشه ابتدا رویداد mousedown و در پی آن mouseup و آخر سر click رخ می دهد .

درهنگام اجرای رویداد dblclick رویداد های زیر به ترتیب اجرا می شوند :

۱. mousedown
۲. mouseup
۳. click
۴. mousedown
۵. mouseup
۶. click
۷. dblclick

هنگام جا به جا شدن نشانگر موس از یک عنصر بر روی عنصر دیگر ، ابتدا رویداد mouseout سپس رویداد mousemove برای عنصر بین این دو و آخر سر رویداد mouseover رخ می دهد .

رویدادهای keyboard :

رویداد های keyboard به واسطه عملیاتی که کاربر بر روی صفحه کلید انجام می دهد رخ می دهند . رویداد های صفحه کلید به شرح زیر می باشند :

- onkeydown : هنگامی که کلیدی از صفحه کلید زده می شود رخ می دهد . این رویداد مکررا زمانی که دکمه ای پایین نگه داشته باشد نیز رخ می دهد .
- Onkeypress : هنگامی که کلیدی از صفحه کلید زده می شود و به موجب آن یک کاراکتر برگردانده می شود رخ می دهد . این رویداد مکررا زمانی که کاربر دکمه ای را پایین نگه میدارد نیز رخ می دهد .
- Onkeyup : هنگامی رخ میدهد که دکمه ای که پایین بوده است رها شود .

ترتیب اجرایی رویداد های keyboard :

موقعی که کاربر یک کلید کاراکتری را در صفحه کلید فشار می دهد رویداد های زیر به ترتیب اجرا می شوند :

۱- keydown
۲- Keypress
۳- Keyup

اگر کلیدی غیر کاراکتری مثل shift فشار داده شود رویداد های زیر به ترتیب اجرا می شوند :

۱- Keydown
۲- Keyup

اگر کاربر کلیدی کاراکتری را فشارداده و پایین نگه دارد رویداد های keypress و keydown مکررا یکی پس از دیگری رخ می دهند تا زمانی که کلید رها شود .

اگر کاربر کلیدی غیر کاراکتری را فشارداده و پایین نگه دارد فقط رویداد keydown مکررا اجرا میشود .

دیگر رویداد ها :

از دیگر رویداد هایی که ممکن است در صفحه و بر روی بعضی از عناصر رخ دهد می توان به موارد زیر اشاره نمود :

- Load : موقعی رخ می دهد که صفحه به طور کامل بارگذاری شود یا اینکه یک عنصر img یا object به طور کامل بارگذاری شوند . برای فعال کردن کنترلگر های حادثه onload برای صفحه آن را در دستور <body> قرار می دهیم . برای مثال در عبارت زیر از این حادثه استفاده کرده ایم تا پس از خاتمه بار شدن صفحه پیغام loading complete نمایش داده شود :

```
<body onload='alert("loading complete !!!")'></body>
```

- Unload : هنگامی رخ می دهد که کاربر صفحه بار شده جاری را ببند . این می تواند به موجب زدن دکمه X (close) پنجره یا وارد کردن یک آدرس جدید درنوار آدرس مرورگر باشد .
- Abort : این رویداد برای یک object هنگامی که کاربر قبل از بارگذاری کامل آن ، عمل بارگذاری را متوقف کند رخ می دهد .
- Error : این رویداد برای یک صفحه هنگامی که درآن یک خطا رخ می دهد ، برای یک عکس هنگامی که نتواند بارگذاری شود و برای یک عنصر object هنگامی که نتواند بارگذاری شود رخ می دهد .
- Select : این رویداد هنگامی رخ می دهد که کاربر یک یا چند کاراکتر را از داخل یک ناحیه متنی (منظور تگ های textarea و input) انتخاب کند رخ می دهد .

- Change : بر روی یک ناحیه متنی ، هنگامی که مقدار داخل ناحیه متنی تغییر کرده و در ادامه تمرکز (focus) صفحه از روی عنصر برود و برای یک عنصر select هنگامی که مقدار آن تغییر می کند رخ می دهند .
- Submit : برای عنصر form هنگامیکه دکمه submit مربوط به فرم کلیک می شود رخ میدهد .
- Reset : برای عنصر form هنگامیکه دکمه reset مربوط به فرم کلیک می شود رخ میدهد.
- Focus : برای عنصری هنگامیکه تمرکز صفحه بر روی آن برود رخ می دهد .
- Blur : برای عنصری هنگامیکه تمرکز صفحه از روی آن برود رخ می دهد .

شی ء event :

شی ء event که در نسخه ۱،۲ و بالاتر جاوااسکریپت در دسترس قرار گرفته است ، شی ء خاصی است که به همراه هر حادثه برای کنترلگر ان حادثه فرستاده میشود . درواقع کنترلگر حادثه می تواند آن را به عنوان یکی از پارامتر ها دریافت کند و خاصیت های شی ء event اطلاعاتی را در مورد آن حادثه در دسترس برنامه نویسان قرار می دهد .

بعضی از اطلاعاتی که این شی ء در اختیار قرار می دهد به شرح زیر است :

- شی ای که موجب رخداد رویداد شده است
- اطلاعاتی در مورد نشانگر mouse درهنگام رخداد رویداد
- اطلاعاتی در مورد صفحه کلید درهنگام رخداد رویداد

برای دسترسی به این شی ء می توان به چندین طریق عمل کرد :

در Internet Explorer ، این شیء به عنوان یکی از خواص شی ء window قابل دسترسی است . این بدین معنی است که یک کنترلگر حادثه به طریق زیر می تواند به شی ء event دسترسی پیدا کند :

```
oDiv.onclick = function () {
      var oEvent = window.event;
}
```

اگر چه این شی ء به عنوان یکی از خواص window شناخته می شود اما فقط زمانی قابل دسترسی است که رویدادی رخ داده باشد . بعد از اینکه کنترلگر حادثه به طور کامل اجرا شد ، شی event نیز از بین خواهد رفت .

اما در استاندارد های DOM می توان از روش دسترسی به آرگومان تابع برای دسترسی به شی ء event استفاده کنیم . به عنوان مثال :

```
oDiv.onclick = function () {
      var oEvent = arguments[0];
}
```

البته می توان نامی برای این آرگومان نیز مشخص کرد و از آن برای دسترسی استفاده نمود :

```
oDiv.onclick = function (oEvent) {
}
```

خواص و متدهای شیء event :

این شی ء شامل خواص و متد هایی است که در ادامه بررسی خواهیم کرد :

Shiftkey : اگر دکمه shift زده شده باشد true و در غیر این صورت false را بر می گرداند .

altkey : اگر دکمه alt زده شده باشد true و در غیر این صورت false را بر می گرداند .

ctrlkey : اگر دکمه ctrl زده شده باشد true و در غیر این صورت false را بر می گرداند .

Button : مشخص می کند که کدام یک از دکمه های mouse زده شده اند . مقادیری که این خاصیت بر می گرداند به شرح زیر است :

- ۰ : هیچ دکمه ای زده نشده است
- ۱ : دکمه چپ زده شده است .
- ۲ : دکمه راست زده شده است .
- ۳ : دکمه چپ و راست با هم زده شده اند .
- ۴ : دکمه وسط زده شده است .
- ۵ : دکمه های چپ و وسط با هم زده شده اند .
- ۶ : دکمه های راست و وسط با هم زده شده اند .
- ۷ : هر سه دکمه با هم زده شده اند .

clienX و clientY : مختصات نشانگر ماوس در لایه داخلی در هنگام رخداد حادثه .

screenX و screenY : مختصات نشانگر ماوس نسبت به گوشه بالا و چپ صفحه نمایش .

pageX و pageY : مختصات x,y ماوس را نسبت به گوشه چپ و بالای صفحه در هنگام رخداد حادثه را مشخص می کند .

type : نوع رویدادی که رخ داده است را بر می گرداند مثلا click و mouseover و...

keyCode : عددی است که مشخص می کند کدام دکمه از صفحه کلید فشار داده شده است .

Target : شی ای را که در معرض حادثه قرار گرفته است را مشخص می کند (مانند یک سند یا یک پیوند) .

کار با Cookie (کوکی) ها :

کوکی ها در واقع متغیر هایی هستند که در قالب یک فایل متنی ساده بر روی کامپیوتر کاربر ذخیره می شوند و در هر بار درخواست صفحه جدید از سرور با همان کامپیوتر، این فایل هم برای سرور فرستاده می شود . ما می توانیم ا زکوکی ها برای ذخیره یکسری اطلاعات خاص کاربران صفحاتمان استفاده کنیم و در صورت نیاز آن ها را در صفحات دیگر مورد استفاده قرار دهیم .

برای ایجاد کوکی ها در جاوااسکریپت از خاصیت cookie شی ء document به شکل زیر استفاده می کنیم :

```
document.cookie="name=value ; expires=Date ; path = path ; domain=domain";
```

و برای بازیابی تمامی کوکی های از قبل ایجاد شده به شکل زیر عمل خواهیم کرد :

```
Var x = document.cookie;
```

همانطور که در دستور ابتدایی می بینید برای ایجاد کوکی می بایست رشته ای حاوی یکسری خواص و مقادیرشان را در قالب جفت های name=value (که با ; از هم جدا شده اند) به خاصیت cookie نسبت دهیم . در جدول زیر هر یک از این قسمت ها را شرح می دهیم .

خاصیت	توضیحات	مثال
name=*value*	این دستور نام و مقدار کوکی را مشخص می کند .	**name=ali**
expires=*date*	این خاصیت اختیاری زمان انقضای کوکی را مشخص میکند . مقداری که به این خاصیت داده می شود می بایست تاریخی به فرمت بازگشتی از متد **toGMTString()** شی ء **Date** باشد . در صورتی که این خاصیت مشخص نشود هنگامی که کاربر پنجره مرورگر را ببندد کوکی نیز از بین خواهد رفت .	**expires= 13/06/2003 00:00:00**
path=*path*	این خاصیت اختیاری نام مسیری از سایت را که می تواند به کوکی دسترسی داشته باشد را مشخص می کند .	**path=/tutorials/**
domain=*domain*	این خاصیت اختیاری نام سایتی که می تواند از کوکی استفاده کند را مشخص می کند .	**domain = mysite.com**

در مثال زیر یک کوکی با نام username و با مقدار ali که در تاریخ 15/02/2010 از بین می رود ایجاد می شود :

```
document.cookie = " username = ali ; expires = 15/02/2010 00:00:00 ";
```

در مثال زیر یک کوکی با نام myCookie و با مقدار this is my cookie ایجاد شده است :

```
document.cookie = “myCookie=” + escape(“This is my Cookie”);
```

- ✓ **نکته : در کد فوق تابع escape یک رشته را دریافت کرده و تمامی کاراکتر های بی ارزش آن را به کد معادلش تبدیل می کند . قبل از کد معادل یک علامت % قرار می گیرد . به عنوان مثال این تابع کاراکتر space را به کد 20% تبدیل می کند . این تابع معادل تابع encodeURIComponent() است .**

حذف یک کوکی :

برای حذف یک کوکی می توان از تابعی که زمان انقضای کوکی را به یک ثانیه قبل تنظیم می کند استفاده کنیم . این تابع به صورت زیر است :

```
function delete_cookie ( cookie_name )
{
  var cookie_date = new Date ( );  // current date & time
  cookie_date.setTime ( cookie_date.getTime() - 1 );
  document.cookie = cookie_name += "=; expires=" + cookie_date.toGMTString();
}
```

حال کافی است برای حذف یک کوکی نام آن را برای تابع فوق بفرستیم . دستور زیر کوکی با نام username را حذف می کند :

```
delete_cookie ("username") ;
```

بازیابی کوکی ها :

حال که با ایجاد و حذف کردن کوکی ها آشنا شدیم نحوه بازیابی (دسترسی) به آنها را بیان می کنیم . برای بازیابی کوکی هایی که قبلا ایجاد شده اند باز هم از خاصیت cookie شی document به صورت زیر استفاده می کنیم :

```
var x = document.cookie;
```

این دستور لیستی (رشته) از جفت های name=value تمامی کوکی های قابل دسترس برای سند جاری را که با ; از هم جدا شده اند بر می گرداند . به عنوان مثال متغیر x می توانید حاوی رشته ای به صورت زیر باشد :

```
"username=ali; password=abc123"
```

در این مثال دو کوکی از قبل ایجاد شده است : یکی با نام username و مقدار ali و دومی با نام password با مقدار abc123 . اکنون x یک متغیر رشته ای ساده است که می توانیم برای دسترسی به هر یک از کوکی ها و مقدارشان ابتدا x را بوسیله متد split شی string به آرایه ای تبدیل کرده و بوسیله متد های خاص آرایه به آن ها دسترسی داشته باشیم . به عنوان مثال برای چاپ مقدار کوکی های فوق می توان به صورت زیر عمل کرد :

```
var allCookie    = document.cookie;
Var cookieParts  = allCookie.split(";");
Var fistCookie   = cookieParts[0];
Var secondCookie = cookieParts[1];

Var nameOfFirstCookie = firstCookie.split("=")[0];
Var valueOfFirstCookie = firstCookie.split("=")[1];

Var nameOfSecondCookie = firstCookie.split("=")[0];
Var valueOfSecondCookie = firstCookie.split("=")[1];
```