PROGRAMMING AND WEB DESIGNING

2nd edition

Learning Java Script

with more than 50 projects

HADI AHMADI

دهلران پی سی

آموزش فارسی

Java Script

به انضمام

سورس و پروژه های قابل اجرا

نویسنده :

هادی احمدی

۰۹۳۷۵۲۷۰۰۹۶

HadiAhmadi105@Gmail.com

با سپاس

از تمام عزیزانی که با انتقادها و راهنمایی های خود

محرکی برای نوشتن ویرایش دوم شدند.

استفاده از این منبع با ذکر صلوات بر

محمد و آل محمد (ص)

رایگان می باشد.

شما می توانید:

این کتاب را به صورت رایگان دریافت کنید.

به صورت رایگان با دوستانتان به اشتراک بگذارید.

رای دانلود در سایت/وبلاگ خود قرار دهید .

شما اجازه ندارید:

مطالب این کتاب را به اسم خود کپی و منتشر کنید .

از این کتاب استفاده تجاری کنید.

در محتوای این کتاب دست ببرید.

در داخل صفحات این کتاب آرم یا لوگوی خود را قرار دهید.

در آخر با توجه به زحمات بسیار زیادی که در نگارش این کتاب کشیده شده است, به نام خود زدن این منبع کاری به دور از عدل و انصاف خواهد بود.

ویژگی های ویرایش دوم

- ✓ برطرف کردن اشتباهات املایی و نگارشی ویرایش اول.
- ✓ ساختار دادن به متن کتاب و سلسله مراتبی کردن مطالب.
- ✓ توجه ویژه به شکیل بودن و زیبایی نگارش.
- ✓ اضافه کردن فصول مختلفی که در ویرایش اول کمتر به آنها توجه شده بود.
- ✓ بسط دادن مطالب موجود در ویرایش قبل و دسته بندی کردن آنها در بخش های مرتبط.
- ✓ اضافه کردن فصولی که در دیگر منابع آموزشی جاوا اسکریپت کمتر بدانها بها داده می شود از جمله : اضافه کردن یک فصل با عنوان مدل شی گرای سند (DOM) , اضافه کردن فصول مربوط به فریم ها و فرم ها و غیره .
- ✓ مثال محور بودن کتاب , به طوری که در هر فصل بیش از بیست مثال متنوع آورده شده است که همین امر باعث سهولت در فرایند یادگیری و در دراز مدت به خاطر سپردن مطالب و همچنین درک کاربردی بودن مطالب می گردد.

توجه

مطالعه این کتاب برای افراد مبتدی که هیچ آشنایی با برنامه نویسی جاوا اسکریپت ندارند و همچنین افرادی که تا سطح متوسط با این زبان آشنایی دارند توصیه می شود.

فهرست مطالب **صفحه**

مقدمه

کتابی که پیش رو دارید ویرایش دوم کتابی است با همین عنوان که در تابستان ۱۳۹۲ نگارش و راهی اینترنت شد. در ابتدا قصد بر نوشتن کتابی مجزا از کتاب اول بود ولی با مورد توجه قرار گرفتن ویرایش اول این کتاب تصمیم به بسط و گسترش آن نمودم و حال بعد از گذشت چند ماه کار تدوین آن را به پایان رساندم. این ویرایش که فقط در عنوان با ویرایش اول مشترک است به نوعی کتابی دیگر است با ساختاری تازه تر و با رعایت سلسله مراتبی که در منابع آموزشی لاتین جاوا اسکریپت موجود است و به دلایل زیر تصمیم به انتشار آن گرفتم:

1- **نبودن یک منبع جامع که بیشتر سرفصل های جاوا اسکریپت را شامل شود.**
2- **تعدد منابع فارسی که به دلیل نداشتن ساختار و سلسله مراتب مناسب بیشتر باعث سردرگمی خواننده می شود.**
3- **لاتین بودن منابع جامع موجود که بری کسانی که به این زبان تسلط ندارند فرایند یادگیری را سخت و کسل کننده می کنند.**
4- **و . . .**

امید است که با انتشار این کتاب گامی کوچک در آموزش و گسترش برنامه نویسی وب برداشته باشم.

با تشکر

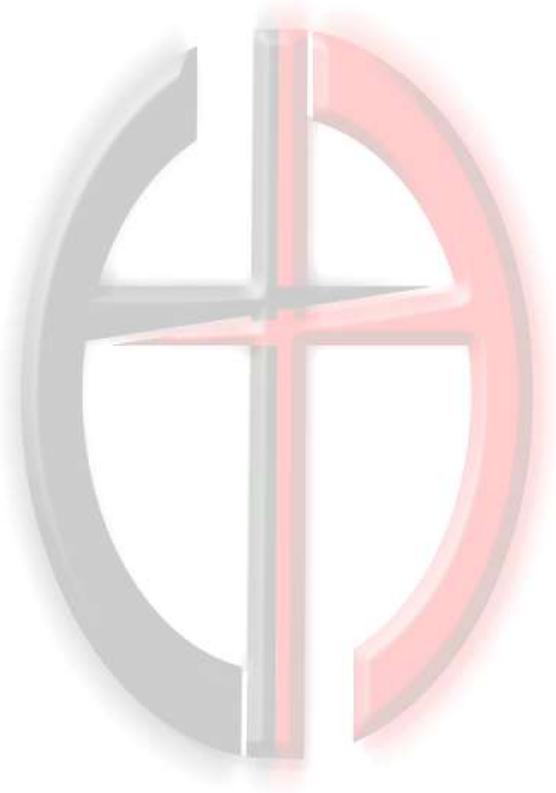
هادی احمدی

شهریور ۱۳۹۴

HadiAhmadi105@gmail.com

HTTP://www.DLP.lxb.ir

فصل اول : شروع کار با جاو اسکریپت



آشنایی با جاوا اسکریپت و مفهوم اسکریپت نویسی

جاوا اسکریپت زبانی است که در صفحات HTML کاربرد دارد و قابلیت های بسیاری از قبیل :افزایش کیفیت طراحی صفحات،کنترل صفحات، کنترل مرورگر بازدید کننده،ساخت و استفاده از کوکی ها و بسیاری چیز های دیگر. جاوا اسکریپت یک زبان برنامه نویسی اسکریپتی است که کد های آن بسیار شبیه به زبان (C) است.زبان جاوا اسکریپت به حروف بزرگ و کوچک حساس است و یک زبان برنامه نویسی شیگرا می باشد. به این معنا که هر عنصر در صفحه وب را به دید یک شی می بیند و با آن رفتار می کند.

از آنجایی که دستورات جاوا اسکریپت مانند دستورات HTML ، نیاز به محیط خاصی برای نوشتن ندارد و برای نوشتن دستورات آن می توان از هر ویرایش گر متنی استفاده کرد. اما نرم افزار های زیادی هم وجود دارد که برای نوشتن کدهای Java Script امکاناتی را در اختیار ما قرار می دهند.مانند رنگی کردن دستورات و پیدا کردن خطاها و ...از جمله این نرم افزارها میتوان dreamweaver نام برد.

آشنایی با چگونگی استفاده از فرامین جاوا اسکریپت

به طور کلی به چهار طریق می توان از جاوا اسکریپت در اسناد HTML استقاده کرد

۱- وارد کردن کدهای جاوا اسکریپت در قسمت <head> صفحه

۲- وارد کردن کدهای جاوا اسکریپت در بخش <body> صفحه

۳- استفاده از کدهای جاوا اسکریپت موجود در یه فایل خارجی

۴- استفاده از کدهای جاوا اسکریپت در درون برچسب های HTML (اسکریپت نویسی Inline)

نکته: شما می توانید از همه روش های بالا به طور همزمان برای وارد کردن جاوا اسکریپت در صفحه استفاده کنید .

وارد کردن کدهای جاوا اسکریپت در قسمت <head> صفحه

برای نوشتن کدهای جاوا اسکریپت در قسمت <head> شما باید کدها را درجای مناسب و میان برچسب شروع <script> و برچسب پایان </script> بنویسد.

نکته : کدهایی که می خواهید با رویداد خاصی اجرا شوند را در این قسمت می نویسید.

```
<html>
  <head>
  <title>untitled document</title>
    <script type=”text/javascript”>
       دستورات جاوا اسکریپت
    </script>
  </head>
  <body>
  </body>
</html>
```

مثال ۱-۱ :

```
<html>
<head>
  <title>untitled document</title>
    <script type=”text/javascript”>
    document.write(“hello world”)
    </script>
  </head>
  <body>
  </body>
</html>
```

نکته : در مورد دستور document.write در فصول بعدی توضیح داده خواهد شد.

وارد کردن کدهای جاوا اسکریپت در بخش <body> صفحه

برای نوشتن کدهای جاوا اسکریپت در قسمت <body> شما باید کدها را درجای مناسب و میان برچسب شروع <body> و برچسب پایان </body> بنویسد.

کدهایی که میخواهید با بارگزاری صفحه اجرا شوند را در این قسمت می نویسید.

```
<html>
  <head>
  <title>my page</title>
  </head>
  <body>
    <script type=”text/javascript”>
     دستورات جاوااسکریپت
    </script>
  </body>
<html>
```

مثال ۲-۱ :

```
<head>
<title>my page</title>
</head>
<body>
<script type=”text/javascript”>
Alert(“hello world”)
</script>
</body>
<html>
```

نکته : در مورد دستور alert در فصول بعد تضیح داده خواهد شد.

استفاده از کدهای جاوا اسکریپت موجود در یه فایل خارجی

برای استفاده از این روش شما می بایست دستورات خود را در یک فایل متنی بنویسید و فایل را با فرمت js . ذخیر کنید و بعد فایل را به وسیله کد زیر که در قسمت <head> می نویسید به سند خود متصل کنید.

مثال ۳-۱:

ابتدا نوت پد خود را باز کنید و سپس دستورات زیر را در آن وارد کنید :

```
Document.write(“hello wold”);
```

سپس فایل را با اسم دلخواه و پسوند js. ذخیره کنید بطور مثال jsfile1.js

سپس با استفاده از دستور زیر آن را به سند html خود اضافه کنید:

```
<html>
  <head>
  <title>my page name</title>
    <script type=”text/javascript” src=”jsfile1.js></script>
  </head>
  <body>
  </body>
</html>
```

نکته ۱ : در مثال بالا فرض براین بوده که فایل html و فایل jsfile1.js در یک فولدر ذخیره شده اند.

نکته ۲ : از src در قطعه کد بالا برای آدرس دهی استفاده می شود.

استفاده از کدهای جاوا اسکریپت در درون برچسب های HTML (اسکریپت نویسی Inline)

مثال ۴-۱:

```
<html>
 <head>
<title>my page</title>
</head>
  <body>
    <input type=”button” value=”ok” onclick=alert”(‘hello world’)”
/>
  </body>
</html>
```

با کلیک بر روی دکمه ok پیام هشداری با مضمون hello world نمایش داده می شود.

نکات مهم در نوشتن کدهای جاوا اسکریپت

۱- جاوا اسکریپت به بزرگی و کوچکی حروف حساس است

۲- نوشتن کارکتر" ; " سیمیکولون در انتهای دستور اجباری نیست اما در مواقعی که شما چند دستور را در یک خط می نویسید نوشتن سمیکالون اجباری است تا پایان هر دستور مشخص شود

۳- برای درج توضیحات یک خطی از// در ابتدای آن خط

۴- برای درج توضیحات چند خطی درابتدای آن /* و درانتهای آن*/ استفاده می شود.

۵- به کاربردن فاصله خالی در کدها از سوی مرورگر نادید گرفته می شود.

توضیحات جاوا اسکریپت

کامنتها می توانند یک توضیح را به جاوا اسکریپت اضافه کنند و یا آن را خوانا تر کنند. کامنتها هیچگاه اجرا نمیشوند و بیشترین استفاده آن برای خود برنامه نویس است چراکه برنامه نویس میتواند با افزودن توضیحات به کد خوانایی و درک آن را برای خود و سایر برنامه نویسات بیشتر کند!

توضیحات یک خطی با // شروع می شوند .

مثال ۵-۱:

این مثال از کامنتهای یک خطی برای توضیح کدها استفاده کرده است.

```
<script type="text/javascript">
    //this will write a header
  document.write("<h1>this is a header</h1>");
    //this will write two paragraphs
  document.write("<h1>this is a paragraph</h1>");
  document.write("<h1>this is a another paragraph </h1>");
</script>
```

توضیحات چند خطی جاوا اسکریپت

توضیحات چند خطی جاوا اسکریپت با /* شروع شده و با */ تمام می شوند.

مثال ۶-۱:

این مثال از کامنتهای چند خطی برای توضیح کدها استفاده کرده.

```
<script type=”text/javascript”>
/*
the code below will write
one header and two paragraphs
*/
document.write(“<h1>this is a header</h1>”);
document.write(“<h1>this is a paragraph</h1>”);
document.write(“<h1>this is a another paragraph </h1>”);
</script>
```

استفاده از کامنت ها برای جلوگیری از اجرای کدهای جاوا اسکریپت

این کار برای اشکال گیری از برنامه به کار می رود علاوه بر این شاید شما می خواهید تغییراتی را در آینده بر روی این قسمت از کدها انجام دهید و برای مدتی کوتاه آن را از دسترس کاربران خارج کردید.

مثال ۷-۱:

```
<script type=”text/javascript”>
  document.write(“<h1>this is a header</h1>”);
  document.write(“<h1>this is a paragraph</h1>”);
    // document.write(“<h1>this is a another paragraph
</h1>”);
</script>
```

مثال ۸-۱:

این مثالی از کامنت است که از اجرای یک کد چند خطی جلوگیری می کند.

```
<script type="text/javascript">
  /*
  document.write("<h1>this is a header</h1>");
  document.write("<h1>this is a paragraph</h1>");
  document.write("<h1>this is a another paragraph </h1>");
  */
</script>
```

استفاده از کامنت در آخر یک خط

این نمونه ای از یک کامنت است که در آخر یک خط قرار دارد , این کامنتها برای توضیحات بیشتر و خواناتر شدن کدهای برنامه بکار می روند.

مثال ۹-۱:

```
<script type="text/javascript">
  document.write("hello") // this will write "hello"
  document.write("dolly") // this will write "dolly"
</script>
```

فصل دوم : متغیرها در جاوا اسکریپت

آشنایی با مفاهیم متغییر ها وداده

متغییرها مکانی در حافظه هستند که برای نگهداری و ذخیره مقادیر انواع مختلف داده ها مورد استفاده قرار می گیرند.

متغییر ها انواع مختلفی دارند و در اکثر زبانهای برنامه نویسی باید هر نوع داده را در متغییر آن داده استفاده کنید و قبل از تعریف متغییر باید نوع متقییر را مشخص کنید اما در جاوا اسکریپت شما نیازی ندارید نوع متقییر را مشخص کنید بعد از ورود داده خود جاوا اسکریپت نوع متغییر را مشخص می کند.

نحوه تعریف متغییر در جاوا اسکریپت

در جاوا اسکریپت برای تعریف متغییر از کلمه کلیدی var که مخفف (variable) به معنی متغییر استفاده می شود و بعد از این کلمه نامی که برای متغییر در نظر می گیرید و کار تشخیص نوع متغییر به عهده مفسر جاوا اسکریپت است.

```
Var variablename;
```

مثال ۱-۲:

```
Var I;
```

نکته : استفاده از کلمه var ضروری نیست و به خوانا تر شدن کدها کمک می کند.

مقداردهی به متغییرها :

```
Var variblename = variablevalue;
```

مثال ۲-۲:

```
Var I = 10;
Var j = 12;
```

انواع داده های اصلی در جاوا اسکریت :

undefined

متغیری است که تعریف می شود ولی مقداردهی اولیه نمی شود.

مثال ۳-۲:

```
var test;
```

Boolean

این نوع می تواند مفادیر شرطی را در خود ذخیره کند مانند: true , false که در عملیاتهای شرطی کاربرد دارد.

null

نول بدین معناست که متغییرمربوطه هچ مقداری ندارد. بدین معنا که صفر و یا فضای خالی نیست , بلکه در حقیقت هیچ است.

مثال ۴-۲:

```
var variablename = null;
```

Character

این نوع فقط یک کارکتر را در خود ذخیره می کنند مانند: "۴","&","x","E"...

number

برای ذخیره مفادیر عدد صحیح مورد استفاده قرار می گیرد مانند : ۱۲۳ و ۱۵ و۸۵-و...

مثال ۵-۲:

```
Var variablename = number;
```

مثال ۶-۲:

```
var paycheck=1200;
var phonebill=29.99;
var savings=0;
var sparetime=-24.5;
```

Float

این نوع برای نگهداری اعداد اعشاری مانند : ۳،۱۴ و ۲،۳۱ و ۰،۸۹ و...

String

این نوع نوع رشته ای را در خود ذخیره می کنند مانند : "سلام دنیا" و "hello world" و ...

مثال ۷-۲:

```
Var variablename = "string text;
```

مثال ۷-۲:

```
var mycar="Corvette";
var oldcar="Big Brown Station Wagon";
var mycomputer="Pentium 3, 500 MHz, 128MB RAM";
var jibberish="what? cool! I am @ home 4 now. (cool, right?)";
```

قواعد نام گذاری متغییر ها

در نام گذاری متغییر ها باید نکات زیر را رعایت کنید:

۱- نام متغییر نمی تواند با رقم شروع شود

۲- نام متغییر نمی تواند شامل فاصله و یا نقطه گذاری باشد

۳- نام متغییر می تواند با حروف و علامت "_" شروع شود

۴- از کلمات رزرو شده نمی توان استفاده کرد

۵- جاوا اسکریپت به بزرگی و کوچکی حروف حساس است به عنوان مثال : url با Url با URL یکی نیست و می توان نام سه متغییر جدا از هم باشد.

مثال ۸-۲:

نمونه هایی نام های غیره مجاز:

```
#paycheck
1paycheck
pay check
pay_check 2
_pay check
```

مثال ۹-۲:

نمونه هایی از نام های مجاز :

```
paycheck
_paycheck
pay2check
pay_check
pay_245
```

آشنایی با کلمات رزرو شده

در تمامی زبان های برنامه نویسی کلماتی وجود دارد که مورد استفاده خود کامپایلر و مفسر آن زبان می باشد و از قبل تعریف شده هستند و این امر در زبان جاوا اسکریپت نبز مستثنی نبوده و شما نمی توانید از این کلمات در نام گزاری متغییر ها و توابع استفاده کنید.

abstract	delete	Goto	Null	Throws
As	Do	If	Pakage	Transient
Boolean	Double	Implements	Pivate	Try
Break	Else	Import	Protected	True
Byte	Enum	In	Public	typeof
Case	Export	Instanceof	Return	Use
Catch	Extends	Int	Short	Var
Char	False	Interface	Static	Void
Class	Final	Is	Super	Volatile
Const	Finally	Long	Switch	While
Continue	Float	Namespace	Synchronized	With
Debugger	For	Native	This	
default	Function	New	Throw	

متغییر های محلی و سراسری

متغییری که در داخل یک تابع جاوا اسکریپت تعریف شود محلی (Local) است و مفدار آن قفط در درون آن تابع در دسترس است در اصطلاح می گوییم تابع scope محلی دارد.

نکته: متغییر های محلی بعد از خروج از تابع ازبین می رود.

متغییری که در بیرون توابع تعریف شود متغییر سراسری (Global) می باشند و تمام اسکریپتها و توابع درون صفحه به آن دسترسی دارند.

نکته: متغییر محلی بعد از بسته شدن صفحه از بین می رود.

زمان حیات متغیرهای جاوااسکریپت

اگر شما متغیری را تعریف کنید و از کلمه "var" در یک تابع استفاده کنید آن متغیر فقط داخل همان تابع در دسترس است. وقتی شما از یک تابع خارج شوید آن متغیر از بین می رود. این متغیرها متغیرهای محلی نامیده میشوند. شما می توانید متغیرهای محلی هم نامی را در توابع مختلف استفاده کنید زیرا این متغیرها فقط در همان توابعی که تعریف شده اند قابل شناسایی هستند.

اگر شما متغیری را خارج یک تابع تعریف کنید آن متغیر در کل صفحه تان در دسترس خواهد بود. این متغیرها زمانی که تعریف می شوند شروع شده و وقتی که صفحه را می بندید پایان می یابند.

چگونگی صدا کردن یک متغیر

در قطعه کد زیر شما نحوه صدا کردن یک متغیر به وسیله تابع document.write را خواهید آموخت.

مثال ۱۰-۲:

```
<script language="JavaScript">
var mycar="Corvette";
document.write(mycar);
</script>
```

شما میوانید قطعه کد بالا را به صورتهایی که در مباحث قبل گفته شد در سند html خود بکار برده و اجرا نمایید.

و اما طرز کار آن:

فارغ از تگهای script در ابتدا ما مقدار corvette را به متغیر mycar نسبت میدهیم , سپس به وسیله تابع document.write() متغیر mycar را فراخوانی می کنیم که در اصل مقدار آن یعنی corvette را برای ما نمایش می دهد.

مثال ۱۱-۲:

```
<script type="text/javascript">
  var mycar="Corvette";
  document.write("I like driving my "+mycar);
</script>
```

نتیجه :

```
I like driving my Corvette.
```

تنها نکته مثال بالا استفاده از عملگر "+" برای پیوند دادن دو قسمت I like driving my و مقدارمتغیر mycar یعنی corvette می باشد که در مباحث بعد در مورد آن توضیح داده خواهد شد.

مثال ۱۲-۲:

```
<script type="text/javascript">
  var mycar="Corvette";
  document.write("I like driving my "+mycar+" every day!");
</script>
```

نتیجه :

```
I like driving my Corvette every day!
```

دقت کنید که چیزی که ببن دوعلامت" کوتیشن" می آید دقیقا به همان صورت نمایش داده می شود. به مثال زیر در رابطه با این موضوع توجه کنید :

مثال ۱۳-۲:

```
<script type="text/javascript">
  var mycar="Corvette";
  document.write("I like driving my +mycar+ every day!");
</script>
```

نتیجه:

```
I like driving my +mycar+ every day!
```

ثابت ها

ثابت ها را با استفاده از کلمه کلیدی Const تعریف می کنند.تفاوت ثابت ها با متغییر ها در این است که ثابت ها می توانند فقط یک مقدار بگیرند (یک مفدار ثابت) و درجایی دیگر نمی توان این مقدار را تغییر

داد .یرای مثال در یک برنامه برای بدست آوردن مساحت دایره به عدد ۳،۱۴ نیاز داریم و این عدد همواره ثابت است و در طول برنامه مقدار p فابل تغییر نیست.

مثال ۱۴-۲:

```
const p = 3.14;
```

تبدیل انواع در جاوا اسکریپت

جاوا اسکریپت نیز مانند اکثر زبانهای برنامه نویسی از قابلیت تبدیل انواع برخوردار است و روشهای ساده ای را برای استفاده از این قابلیت فراهم آورده است.

تبدیل به رشته

سه نوع داده boolean , number & string متدی به نام ( )tostring. برای تبدیل به رشته دارند.

این متد برای متغیرهای از نوع boolean یکی از مقایر رشته ای true یا false را بسته به مقدار متغیر برمی گرداند:

مثال ۱۵-۲:

```
var bFound = true;
alert(bFound.toString());          //outputs "true"
```

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

مثال ۱۶-۲:

```
var iNum1 = 12;
var fNum2 = 17.0;
alert(iNum1.toString());          //outputs "12"
alert(fNum2.toString());          //outputs "17"
```

تبدیل به عدد

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

parseInt( )

parseFloat( )

این متد ها فقط بر روی رشته های حاوی عدد کار می کنند و بر روی بقیه انواع مقدار NaN را برمی 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 میکند تا اینکه به کاراکتری غیر عددی برسد . به عنوان مثال این متد عبارت "123red"را به صورت 123 برمی گرداند.

مثال ۱۷-۲:

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

متد parseFloat()

مثل متد parseInt() عمل کرده و از اولین کاراکتر شروع به جست و جو می کند البته در این متد اواین کاراکتر نقطه حساب نمی شود و آن را به همان صورت برمی گرداند.

اگر دو کاراکتر نقطه در رشته وجود داشته باشند دومین نقطه به عنوان کاراکتر بی ارزش شناخته می شود و عملیات تبدیل متوقف می شود

مثال ۱۸ -۲:

```
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
```

استفاده از Type Casting برای تبدیل انواع

در جاوااسکریپت امکان استفاده از روشی موسوم به Type Casting برای تبدیل انواع وجود دارد. سه نوع type casting در جاوااسکریپت وجود دارد:

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

تابع Boolean() زمانی مقدار true را بر می گرداند که پارامتر دریافتی اش، رشته ای شامل حداقل یک کارکتر، یک عدد غیر از صفر و یا یک شی باشد .مقدارfalse را نیز زمانی بر می گرداند که پارامتر دریافتی اش رشته ای تهی، عدد صفر یا یکی از مقادیر undefined و null باشد:

مثال ۱۹-۲:

```
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 متدهای parseInt() و parseFloat() آرگومان دریافتی را فقط تا اولین کاراکتر بی ارزش بر می گرداند مثلا رشته " 4.5.6 " به 4.5 تبدیل خواهند کرد. اما کاربرد متد Number() برای این رشته مقدار NaN را برمی گرداند زیرا این رشته در کل، از نظر متد Number() امکان تبدیل به یک عدد را ندارد.

اگر رشته ای امکان تبدیل به یک عدد را داشته باشد متد Number() خود، برای استفاده از یکی از توابع parseInt() یا parseFloat() تصمیم می گیرد .در مثال زیر حاصل اجرای تابع Number() برای انواع داده ها را نشان می دهد:

مثال ۲۰-۲:

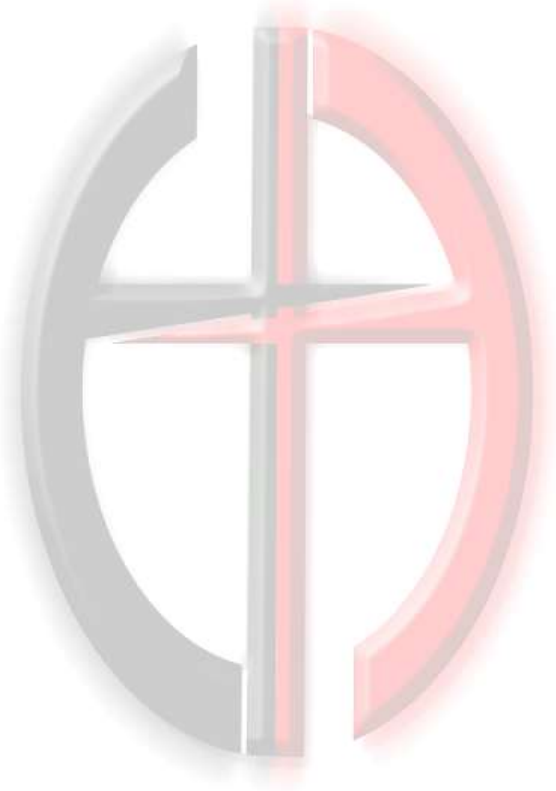
```
Number(false) 0
Number(true) 1
Number(undefined) NaN
Number(null) 0
Number("5.5") 5.5
Number("56") 56
Number("5.6.7") NaN
Number(new Object()) NaN
Number(100) 100
```

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

مثال ۲۱-۲:

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

فصل سوم : توابع



استفاده از توابع در جاوا اسکریپت

یک تابع یک اسکریپت پایه است در درون اسکریپت بزرگتر.که وظیفه آن انجام کار یا مجموعه ای از کارها است که به آن محول می شود برای مثال ممکن است وظیفه نوشتن یک خط در مرورگر یا محاسبه یک فرمول و یا غیره را داشته باشد که در نهایت نتیجه را به اسکریپت اصلی بر می گرداند.

تابع مجموعه دستوراتی است که تنها در صورت رخ دادن یک رویداد و یا از طریق فراخوانی اجرا می شوند. شما می توانید هر تابع را از هر جایی در صفحه فراخوانی کنید (یا حتی از صفحه های دیگر در صورتی که توابع را در یک فایل js. خارجی ذخیره کرده باشید.)

چرا توابع مورد استفاده قرار می گیرند:

اصلی ترین دلیل برای استفاده از توابع, انعطاف پذیری و قدرتی است که در نوشتن کدها به برنامه نویس می دهد تا بتواند کدها و برنامه خود را بهتر مدیریت کند و همچنین خطایابی را بهتر و سریعتر انجام دهد.

ساختار توابع :

در خط اول از ساختار یک تابع شما باید یک اسم برای آن انتخاب کرده و آن را به برنامه خود بشناسانید مانند :

```
function functionname(){
}
```

مثال ۱-۳:

```
function reallycool(){
}
```

قراردادن کدها در درون تابع :

```
function functionname(){
javascript code here
}
```

مثال ۲-۳:

```
function reallycool(){
a = 5; var b =3;
var c = 8;  }
```

مثال ۳-۳:

```
<head>
  <title>Untitled Document</title>
    <script type="text/javascript">
      function dismsg()
      {
      alert("hello");
      }
    </script>
  </head>
  <body>
    <form>
      <input type="button" value="test" onclick="dismsg()" />
    </form>
  </body>
</html>
```

این دستور قبل از اینکه کاربر روی دکمه کلیک کند اجرا نمی شود. تابع dismsg() وقتی اجرا می شود که کاربر روی دکمه کلیک کند.

نام گذاری توابع :

برای نام گذاری توابع باید از دستورات و قوانینی پیروی کنید از جمله اینکه :

- سعی کنید نام تابع با کاری که انجام می دهد همخوانی و هماهنگی داشته باشد.
- نام توابع حتما باید با حروف و یا علامت"_" آغاز شود.
- نام توابع نمی تواند شامل هرگونه فضای خالی باشد.

مثال ۴-۳:

تابع زیر برای چاپ و نمایش یک متن نوشته شده است :

```
function print_strong_text() {
document.write("<strong>This is a strong
statement!</strong>");
}
```

همانگونه که مشاهده می کنید در نوشتن نام تابع از هر سه دستور بالا پیروی و تبعیت شده است.

اضافه کردن پارامتر به توابع :

پارامترها استفاده می شوند تا به توابع این قابلیت را بدهند که یک یا چند مقدار را از جایی از خارج تابع وارد کنند .

```
function functionname(variable1,variable2)
```

مثال ۵-۳:

```
function reallycool(coolcar,coolplace) {
JavaScript code here
}
```

مثال ۶-۳:

```
function reallycool(coolcar,coolplace) {
document.write("My car is a "+coolcar+" and I drive it to
"+coolplace);
}
```

حال اگر مقدار coolcar برابر با Corvette مقدار coolplace برابر Las Vegas باشد آنچه که در مرورگر شما نمایش داده می شود خط زیر خواهد بود:

```
My car is a Corvette and I drive it to Las Vegas
```

مثال ۷-۳

```
<head>
  <title>Untitled Document</title>
    <script type="text/javascript">
      function test(a , b)
      {
      return a*b;
      }
    </script>
  </head>
  <body>
    <script type="text/javascript">
      document.write(test(5,2));
    </script>
  </body>
</html>
```

در مثال بالا تابع تست را قرار دادیم و در داخل پرانتز دو متغیر a,b را به آن اختصاص دادیم و در خط بعد از دستور return استفاده کردیم که منظور ما حاصلضرب a در b می باشد--- در قسمت body به متغیر a مقدار ۸ و به متغیر b مقدار ۵ را اختصاص دادیم که توسط دستور document.write در خروجی نمایش داده می شود.

اضافه کردن دستور return به تابع :

با استفاده از مثال زیر نحوه استفاده از دستور return را شرح می دهیم :

مثال ۸-۳:

```
function get_added_text() {
var textpart1="This is ";
var textpart2="fun!";
var added_text=textpart1+textpart2;
return added_text;
}
```

آنچه که دستور return برمی گرداند را در زیر می بینید :

```
This is fun!
```

شما می توانید با استفاده از دستور return هرچیزی یا هیچ چیزی را برگردانید .

مثال ۹-۳:

```
return "This is cool";
return 42;
return true;
return null;
return;
```

استفاده از یک تابع برای یک هشدار جاوا اسکریپت

مثال ۱۰-۳:

```
function show_message() {
  window.alert("This is an alert!");
  }
    show_message();
```

مثال ۱۱-۳:

```
<head>
  <title>Untitled Document</title>
    <script type="text/javascript">
      function test()
      {
      return ("hello world");
      }
    </script>
  </head>
  <body>
    <script type="text/javascript">
      document.write(test());
    </script>
  </body>
</html>
```

مثال ۱۲-۳:

```
function show_message() {
window.alert("This is an alert!");
}
window.alert("I am first, ha!");
window.alert("I am second, ha ha!");
show_message();
```

صدا زدن یک تابع توسط تابعی دیگر :

با یک مثال این بحث را پی می گیریم :

مثال ۱۳-۳:

```
<script type="text/javascript">
  function update_alert(){
  window.alert("Welcome! This site is updated daily!");}
  function call_alert() {
  update_alert();}
  call_alert();
</script>
```

مثال ۱۴-۳:

```
<script type="text/javascript">
  function update_alert() {
  window.alert("Welcome! This site is updated daily!");
  }
  function section_alert() {
  window.alert("Please visit the picture section!");
  }
  function links_alert() {
  window.alert("Also, check out my links page!");
  }
  function get_messages() {
  update_alert();
  section_alert();
  links_alert();
  }
  get_messages();
</script>
```

مثال ۱۵-۳:

```
<script type="text/javascript">
  var mycar="Honda";
  var paycheck=1200;
  function new_car() {
  mycar="Ferrari";
  paycheck=3500;
  window.alert("You need $"+paycheck+" to get a "+mycar);
  }
  new_car();
  window.alert("You make $"+paycheck+" and have a "+mycar);
</script>
```

مثال ۱۶-۳:

ساخت یک صفحه html با استفاده از توابع

۱.ابتدایک صفحه html ایجاد کنید و آن را ذخیره نمایید.

۲.سپس یک فایل جاوا اسکریپت با نام test.js ایجاد و ذخیره نمایید.

۳.در درون فایل html خود ارجاعی به فایل جاوا اسکریپت خود انجام دهید.در فصول قبل نحوه استفاده از فایل های اسکریپت خارجی در فایل html توضیح داده شده است.

۴.فایل test.js را با استفاده از یک ویرایشگر متن باز کرده و کدهای زیر را در آن وارد کنید.

```
function car_cost(mycar,paycheck) {
  window.alert("You have a "+mycar+" and make $"+paycheck);
}
function get_added_text() {
  var textpart1="This project ";
  var textpart2="is almost fun!";
  var added_text=textpart1+textpart2;
  return added_text;
}
car_cost("Mustang",1500);
var alert_text=get_added_text();
window.alert(alert_text);
```

۵.فایل جاوا اسکریپت خارجی را ذخیره کنید.

۶.بعد از ذخیره کردن فایل و زمانی که شما فایل html خود را باز می کنید دو پیام زیر را مشاهده می نمایید :

```
You have a Mustang and make $1500
This project is almost fun!
```

به عنوان تمرینی دیگر قطعه کد زیر را در یک فایل html ذخیره کنید و نتایج را مورد بررسی قرار دهید :

```
<body>
  <h1>"Welcome to my Function Page," I said.</h1>
    <script type="text/javascript">
      function get_added_text(textpart1,textpart2) {
      var added_text=textpart1+" "+textpart2;
      return added_text;
      }
      function print_text() {
      var myfood=get_added_text("cheese","bread");
      document.write(myfood);
      }
      var alert_text=get_added_text("soup","crackers");
      window.alert(alert_text);
      print_text();
    </script>
<p style="color:red">I'm seeing red!</p>
</body
```

فصل چهارم: عملگرها

معرفی عملگرهای مورد استفاده در JavaScript

در جاوا اسکرپت برای جمع، تفریق، مقایسه و بسیاری از عملیات دیگر از عملگرها استفاده می کنیم. عملگرها به چند دسته تقسیم می شوند : عملگرهای محاسباتی، مقایسه ای، جایگزینی، منطقی، رشته ای، و شرطی. در ادامه به ارائه توضیحاتی برای هر نوع از این عملگرها خواهیم پرداخت.

عملگرهای محاسباتی

از این نوع عملگرها برای انجام عملیات ریاضی نظیر جمع و ضرب استفاده می شود. می توانید این عملگرها را در اینجا مشاهده کنید (در مثال زیر متغیر x را برابر ۸ و متغیر y را برابر ۴ در نظر بگیرید) :

نام	عملگر	مثال	نتیجع
جمع	+	X + y	12
تفریق	-	X - y	4
ضرب	*	X*y	32
تقسیم	/	X / y	2
باقیمانده	%	X % y	1
افزایش	++	X ++	9
کاهش	--	X --	7

از عملگرهای افزایش و کاهش به ترتیب برای افزودن ۱ به مقدار قبلی و کم کردن ۱ از مقدار قبلی استفاده می شود.

عملگرهای جایگزینی

این عملگرها ترکیبی از عملگرهای دیگر هستند و به همین دلیل آنها را عملگرهای ترکیبی هم می نامند. می توانید این عملگرها را در جدول زیر مشاهده کنید (در مثال زیر متغیر x را برابر ۸ و متغیر y را برابر ۴ در نظر بگیرید) :

نام	عملگر	مثال	عبارت معادل
انتساب	=	X = 8	X = 8
انتساب جمع	+=	X += y	X = x+y
انتساب تفریق	-=	X -= y	X = x-y
انتساب ضرب	*=	X *= y	X= x*y
انتساب تقسیم	/=	X /= y	X = x/y
انتساب باقیمانده	%=	X %= y	X = x%y

عملگرهای مقایسه ای

از این عملگرها برای مقایسه دو مقدار استفاده می شود که شامل عملگرهای زیر هستند.

در جدول زیر مقادیر xو y و z را مطابق زیر در نظر بگیرید و توجه داشته باشید که مقدار متغیرهای x و y عددی و مقدار متغیر z از نوع رشته ای است :

x = 4; y = 8; z = "4" ;

نام	عملگر	مثال	نتیجه
تساوی	==	x == z	true
همانی	===	x === z	false
نامساوی	!=	x != z	true
بزرگتر از	>	x > y	false
کوچکتر از	<	x < y	true
بزرگتر یا مساوی	>=	x >= y	true
کوچکتر یا مساوی	<=	x <= y	true

تفاوت عملگر تساوی با عملگر همانی در این است که در عملگر تساوی نوع متغیر ها در نظر گرفته نمی شود و اگر مقدار متغیرها با هم برابر باشد حتی اگر از یک نوع هم نباشند نتیجه True خواهد بود. ولی در عملگر همانی باید متغیرها مقداری مساوی داشته باشند و از یک نوع هم باشند تا نتیجه True باشد.

عملگرهای منطقی

این عملگرها برای انجام عملیات منطقی بر روی دو عبارت به کار می روند (در مثال زیر متغیر x را برابر ۸ و متغیر y را برابر ۴ در نظر بگیرید) :

نام	عملگر	مثال	نتیجه
نقیض	!	!(x == y) !(y == 3)	true false
و	&&	(x=4 && y=8) (x=4 && y=8)	true false
یا	||	(x=4 || y=8) (x>4 || y>8)	true false

عملگر رشته ای

از این عملگر برای چسباندن دو رشته به هم استفاده می شود. مثلاً برای اتصال دو متغیر از نوع رشته از این عملگر استفاده می شود.

نام	عملگر	مثال
عملوند رشته ای	+	a = "design web"; b = "with java script"; c = a + b;

در مثال بالا مقدار متغیر c این است :

```
design webwith java script
```

برای ایجا فاصله بین کلمه وب و کلمه با می توانیم یک فاصله خالی به یکی از متغیرهای a یا b اضافه کنیم یا با استفاده از عملوند رشته ای یک فضای خالی به متغیر c اضافه کنیم :

a = “design web ”; b = “with java script”; c = a+b;	a = “design web”; b = “with java script”; c = a+ “ “+b;

در نتیجه کد بالا مقدار متغیر c به این صورت خواهد بود :

```
design web with java script
```

مثال های متنوع در زمینه بکارگیری عملگرها :

تمامی مثالهایی که در زیر آورده می شوند را می توانید با قرار دادن دربین تگهای زیر امتحان کنید:

```
<script type="text/script>
  محل قرار گیری کدها
</script>
```

The Addition Operator (+)

مثال ۱-۴:

```
var thesum=4+7;
window.alert(thesum);
var num1=4;
```

مثال ۲-۴:

```
var thesum=num1+7;
window.alert(thesum);
```

مثال ۳-۴:

```
var num1=4;
var num2=7;
var thesum=num1+num2;
window.alert(thesum);
```

مثال ۴-۴:

```
var num1=4.73;
var num2=7;
var thesum=num1+num2;
window.alert(thesum);
```

مثال ۵-۴:

```
var num1=4;
var num2="7";
var thesum=num1+num2;
window.alert(thesum);
```

نکته : جواب مثال آخر 47 است زیرا ما 7 را به عنوان یک رشته وارد کرده ایم و در نتیجه عملگر جمع تنها آنها را به هم الحاق می کند.

The Subtraction Operator (–)

مثال ۶-۴:

```
var theresult=10-3;
window.alert(theresult);
```

مثال ۷-۴:

```
var num1=10;
var num2=3;
```

```
var theresult=num1-num2;
window.alert(theresult);
```

The Multiplication Operator (*)

مثال ۸-۴:

```
var num1=4;
var num2=5;
var thetotal=num1*num2;
window.alert(thetotal);
```

The Division Operator (/)

مثال ۹-۴:

```
var num1=10;
var num2=2;
var theresult=num1/num2;
window.alert(theresult);
```

مثال ۱۰-۴:

```
var num1=3;
var num2=4;
var theresult=num1/num2;
window.alert(theresult);
```

مثال ۱۱-۴:

```
var num1=10;
var num2=0;
var theresult=num1/num2;
window.alert(theresult);
```

در مثال بالا جواب بی نهایت می شود چرا که عدد بر صفر تقسیم شده است.

The Modulus Operator (%)

مثال ۱۲-۴:

```
var num1=11;
var num2=2;
var theresult=num1%num2;
window.alert(theresult);
```

The Increment Operator (++)

مثال ۱۳-۴:

```
var num1=2;
var theresult=++num1;
```

The Increment Operator After the Operand

مثال ۱۴-۴:

```
var num1=2;
var theresult=num1++;
```

مثال ۱۵-۴:

```
<script type="text/javascript">
num1=2;
result= ++num1;
alert("num1= "+num1+" result= "+result);
num1=2;
```

```
result= num1++;
alert("num1= "+num1+" result= "+result);
</script>
```

The Decrement Operator (– –)

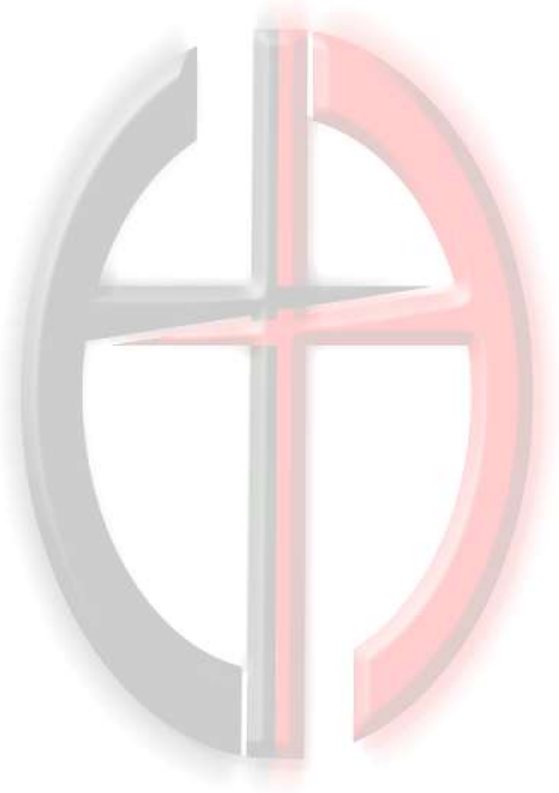
مثال ۱۶-۴:

```
var num1=2;
var theresult=--num1;
```

مثال ۱۷-۴:

```
var num1=2;
var theresult=num1--;
```

فصل پنجم: حلقه ها



معرفی دستور شرطی if...else و کاربردهای مختلف آن

از دستورات شرطی برای انجام دستوراتی به شرط برقراری یک رابطه دیگر استفاده می شود. در طراحی صفحات وب بسیار پیش می آید که می خواهیم در شرایط خاصی صفحه یک رفتار مشخص داشته باشد و در موارد دیگر رفتار صفحه تفاوت داشته باشد. در این مواقع می توانیم از دستور شرطی if استفاده کنیم. با این دستور مشخص می کنیم اگر شرط خاصی برقرار بود یک عمل خاص انجام شود. ساختار این دستور به شکل زیر است :

دستور شرطی if

```
if(condition)
{
دستوری که در صورت درستی شرط اجرا می شود در اینجا قرار می گیرد
}
```

به یک مثال برای دستور if توجه کنید :

مثال ۱-۵:

```
var test = confirm("اگر این پیام را تایید کنید دستور شرطی اجرا می گردد.")
if (test==true){
alert("شما دکمه مورد نظر را کلیک کردید و این پیام به نمایش درآمد")
}
```

برای اینکه کد بالا را آزمایش کنید لینک زیر را کلی کنید و در پیامی که نمایش داده می شود دکمه OK را کلیک کنید. اگر دکمه Cancel را کلیک کنید شرط بالا برقرا نمی شود و بدون اجرا شدن دستور شرطی صفحه مثال باز می شود.

دستور شرطی if...else

برخی مواقع ممکن است بخواهیم دستور شرطی یک شرط را چک کند تا در صورت برقرار بودن آن شرط یک فرمان خاص را اجرا کند و در صورتی که شرط برقار نبود یک فرمان دیگر را به اجرا در آورد. در این موارد می توانیم از دستور if به همراه else استفاده کنیم. ساختار این دستور به شکل زیر است :

```
if(condition){
دستوراتی که در صورت درستی شرط اجرا می شوند}
else{
دستوراتی که در صورت عدم برقراری شرط اجرا می شوند}
```

در کد بالا ابتدا شرط بررسی می شود و در صورت برقراری آن دستورات مربوطه اجرا می شوند ولی در صورتی که شرط برقرار نباشد دستورات مربوط به قسمت else اجرا می شود.

مثال ۲-۵:

```
<script type="text/javascript">
  var yourmark = prompt('enter a number between 0 to 20');
  if (yourmark>10){
  alert('your number great than 10');
  }
  else{
  alert('your number not great than 10');
  }
</script>
```

برای مشاهده نتیجه مثال بالا آن را امتحان کنید و در کادری که نمایش داده می شود یک عدد از ۰ تا ۱۰ وارد کنید و دکمه OK را کلیک کنید.

استفاده از دستور if..else if...else (دستورات شرطی تو در تو)

ممکن است در نظر داشته باشیم تا در صورت برقراری یک شرط یک دستور خاص اجرا شود و در صورتی که شرط برقرار نبود شرط دیگری بررسی شود و دستور مربوط به آن اجرا شود و در صورتی که هیچ یک از شرطهای قبل برقرار نبود دستور دیگری اجرا شود. در این مواقع می توانیم از دستور else if در دستور if...else قبلی استفاده کنید. به صورت زیر :

```
if (condition 1){
دستوراتی که در صورت درستی شرط اول اجرا می شوند
}
else if (condition 2){
دستوراتی که در صورت درستی شرط دوم اجرا می شوند
}
slse{
دستوراتی که زمانی که هیچکدام از شروط درست نباشند اجرا می شوند
}
```

در این روش محدودیتی از نظر تعداد شرطها وجود ندارد و می توانیم به تعداد نامحدود از دستور else if استفاده کنیم ولی در مواردی که تعداد این شرطها خیلی زیاد می شود بهتر است به جای این روش از دستور switch که بعداً توضیح داده خواهد شد استفاده کنیم.

در اینجا مثال قبل را با کمی تغییر برای این روش استفاده می کنیم :

مثال ۳-۵:

```
<script type="text/javascript">
  var yourmark = prompt('enter a number between 0 to 20');
  if (yourmark>0 && yourmark<20){
  alert('شماره وارد شده بین ۰ تا ۲۰ است');
  }
  else if(yourmark>20){
  alert('شماره وارد شده از ۲۰ بزرگتر است');
  }
  else{
  alert('عبارت وارد شده یا ۰ است از به جای عدد از حرف استفاده کرده اید')
  }
</script>
```

دستور switch

اگر بخواهیم برای اسکریپت های خود مسیرهای بیشتری در نظر بگیریم یک راه موجود استفاده از دستورات if تو در تو است. این روش در موارد جزئی کاراییدارد و در صورتی که تعداد شرطها زیاد شود ما را مجبور می کند از تعداد زیادی if و else در اسکریپت خود استفاده کنیم که این مسئله هم باعث طولانی شدن برنامه و از طرفی امکان اشتباه را هم بالا می برد.

راه حل دیگر استفاده از دستور شرطی switch است. این دستور را می توان در هنگامی که می خواهیم یک متغیر را با چندین مقدار مقایسه کنیم و مقدار متناسب با آنرا پیدا کنیم کار برد دارد.

برای درک بهتر به شکل کلی این روش توجه کنید.

```
switch(x)
{
case 1 :       //if x=value(case1) it's run;
break;
case 1 :       //if x=value(case2) it's run;
break;
default;       //if x not equal to value(case1) or value(case2)
```

دستور بالا به این صورت عمل می کند که در ابتدا ما یک عبارت x را داریم که متغیر ما می باشد که یک بار بررسی می شود و سپس مقدار این عبارت با هر کدام از case ها مقایسه می شود تا ببینند که با کدام یک برابر می باشد اگر تطابقی داشته باشد مجموعه کدهای آن case اجراء می شود

:break

از break برای این استفاده می شود که در صورت درستی هر کدام از case ها دستور به case بعدی نرود و از دستور خارج شود

:default

در صورتی که مقدار متغیر مورد نظر با هیچکدام از case ها برابر نبود کد قسمت default اجراء می شود البته استفاده از default اختیاری است شما می توانید از آن به حد نیاز استفاده کرده و یا آن را حذف کنید

نکته:شما در استفاده از دستورات case هیچ محدودیتی ندارید و می توانید به اندازه نیاز از آن ها استفاده کنید

در مثال زیر با استفاده از توابع Date و getDay، عدد متناظر با هر روز (از روزهای هفته) را دریافت و با توجه به آن، یکی از موارد (case) ها را به خروجی ارسال کرده ایم، ملاحظه می کنید که بر اساس روزهای هفته خروجی ما نیز متغیر خواهد بود، به اینصورت می توان به صورت داینامیک از switch و case در جاوا اسکریپت استفاده کرد.

مثال ۴-۵:

```
<script type="text/javascript">
var date = new Date()
var today = date.getDay()
switch(today){
     case 1 :
     document.write("mon");
     break;
     case 2 :
     document.write("tus");
     break;
     case 3 :
     document.write("wen");
     break;
     case 4 :
     document.write("tur");
     break;
     case 5 :
     document.write("fri");
     break;
     case 6 :
     document.write("sat");
     break;
     case 7 :
     document.write("sun");
     break;
     default:
     document.write("error")
}

</script>
```

تابع date و getDay در حالت عادی با روز و زمان میلادی کار می کنند، لذا بر اساس اینکه یکشنبه روز تعطیل در تقویم میلادی محسوب می شود (روز هفتم)، شروع شمارش روزهای هفته از دوشنبه (case 1) خواهد بود.

اعدادی که به عنوان مقادیر به case ها داده شده اند، در واقع حاصل و نتیجه برگردانده شده از قسمت date.getDay و با توجه به تاریخ تنظیم شده سیستم است که بین ۱ تا ۷ در نوسان است.

با توجه به تاریخ سیستم شما، امروز سه شنبه است!

استفاده از حلقه ها :

نحوه استفاده از حلقه های FOR

حلقه ها مجموعه کدهایی را برای تعداد خاصی یا تا زمانی که یک شرطی برقرار شود اجرا می کنند.

معمولا زمانی که شما کدی را می نویسید، نیاز دارید که مجموعه کدهایی پشت سر هم چند بار اجرا شوند. به جای چندین بار اضافه کردن این چندخط پشت سر هم می توانید از دستورات زیر استفاده کنید.

در جاوااسکریپت دو نوع حلقه متفاوت وجود دارد:

for: مجموعه دستورات به تعداد مشخص تکرار می شود.

while: مجموعه دستورات تا تحقق یک شرط ادامه می یابد.

حلقه For

برای شروع ابتدا ساختار کلی حلقه for را نشان می دهیم :

```
for ( مقدار اولیه اندیس حلقه ;شرط حلقه ;گام حرکت)
{
دستورات;
}
```

مثال ۵-۵:

```
<script type="text/javascript">
for (var count=1;count<11;count+=1) {
document.write("I am part of a loop!<br />");
```

```
}
</script>
```

دستور بالا به اندازه 10 بار عبارت I am part of a loop! را چاپ می کند .

مثال زیر حلقه ای را تعریف می کند که از i=0 شروع شده، و تا زمانی که i کوچکتر یا مساوی ۵ باشد ادامه می یابد. هر بار که حلقه اجرا می شود i یک واحد افزایش می یابد.

مثال ۶-۵:

چرخیدن بین تگ های heading در HTML

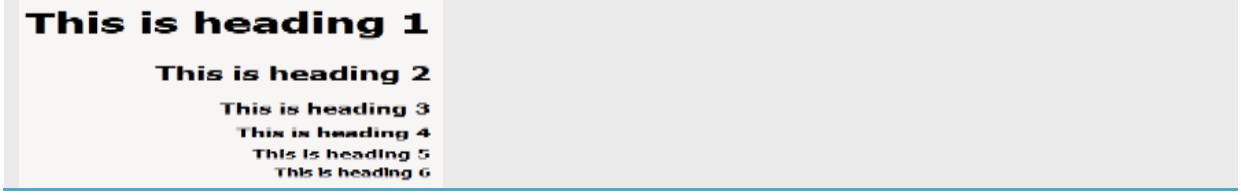
```
<script type="text/javascript">
for(i=1;i<7;i++)
{
     document.write("<h" + i +">this is heading" + i);
     document.write("<h" + i + ">");
     }
</script>
```

حلقه ی for باعث می شود مقدار متغیر i تا کوچکتر و مساوی 6 ادامه پیدا کند و سپس متوقف شود

توسط دستور document.write اول متن خود را بصورت عنوان بوجود آوردیم که باعثش تگ عنوانی است که وارد شده و توسط + i + این دستور هر بار که حلقه تکرار می شود عنوان نیز تغییر پیدا می کند و توسط i+ مقدار متغیر i را در کنار متن خود که به صورت عنوان در آمده تکرار کردیم و در document.write نیز این تگ عنوان را بستیم

نکته:تمامی کدهایی که بین این دو دابل کوت وارد شده اند جدا از i + می باشد

نتیجه:



نحوه استفاده از حلقه های while

حلقه ها مجموعه کدهایی را برای تعداد خاصی یا تا زمانی که یک شرطی برقرار شود اجرا می کنند

این حلقه همان کارایی for را دارد و از همان اجزا تشکیل شده و فقط در ترکیب قرار گرفتن اجزاء فرق دارد.

```
while (مقدار پایانی=>متغیر)
{
دستورات بدنه حلقه
}
```

نکته: به جای => می توان از هر عملگر مقایسه ای دیگر استفاده کرد.

مثال پایین همانند مثال های for می باشد و ترکیب قرار گرفتن اجزاء فرق دارد.

مثال ۷-۵:

```
<script type="text/javascript">
var i = 0;
while(i <= 6)
{
     document.write("the number is :" + i);
     document.write("<br />");
     i++;
}
</script>
```

در مثال بالا مدار متغیر i را ۰ قرار داده ایم و در خط بعد (حلقه ی while) یک متن را به صورت رشته قرار داده ایم و توسط دستور i + حلقه را تکرار کردیم تا در کنار هر متن ما شماره ای به ترتیبی که خودمان قرار داده ایم قرار گیرد و در خط بعد با استفاده از تگ
 HTML جملات خود را در زیر یکدیگر قرار داده ایم و بعد دستور i++ را قرار داده ایم و سپس دستور را بستیم.

حلقه do...while

این حلقه یک تفاوت با while دارد. این حلقه ابتدا مجموعه دستورات را یک بار انجام می دهد، و سپس حلقه را تا زمانی که شرط برقرار شود تکرار می کند.

در بعضی مواقع ما نیاز داریم که حلقه حداقل یکبار اجرا شود، حتی اگر شرط برقرار نباشد! زیرا دستورات قبل از بررسی شرط اجرا می شوند.

دستور:

```
do {
دستورات بدنه حلقه }
while ( متغیر<=مقدار پایانی) ;
```

در مثال زیر از do...while استفاده می کنیم. در این نوع حلقه، مجموعه دستورات حداقل یک بار اجرا می شود حتی اگر شرط نادرست باشد، زیرا دستورات قبل از بررسی شرط اجرا می شوند.

مثال ۸-۵:

```
<script type="text/javascript">
var i = 8;
do
{
     alert("this is a test");
     i++;
}
while(i<=4);
</script>
```

در مثال بالا i=8 است و شرط حلقه این است که i کوچکتر از ۴ باشد، اما با اینکه شرط برقرار نیست ولی اجراء می شود چون این شرط در انتهای حلقه بررسی میشود، پس حلقه یکبار اجرا شده و پیام یک بار نمایش داده میشود.

آموزش کار با دستور for...in

این دستور بین خصوصیت های یک شیء می چرخد.

این دستور به منظور بدست آوردن خصوصیات و یا متد اشیاء استفاده می شود در صورتی که بخواهیم خصوصیت های یک شی ء را بدست آوریم از این روش استفاده می کنیم البته به این نکته توجه کنید که مطلب جلسه پیش در مورد for بسیار کاربردی تر از این مطلب می باشد

دستور:

```
for (variable in object)
{
دستورات بدنه حلقه
}
```

نکته: دستور موجود در بدنه for...in برای هر خصوصیت یک بار اجرا می شود.

مثال ۹-۵:

چرخیدن بین خصوصیت های یک شیء:

```
<script type="text/javascript">
var person={first:"welcome to",second:"your site"};
for(x in person)
{
     document.write(person[x] + " ");
}
</script>
```

خروجی کد بالا به صورت زیر خواهد بود:

```
welcome to your site
```

فصل ششم : رویدادها

رویدادها

رویداد یک خاصیت از پیش تعریف شده در جاوا اسکریپت است که آن را در صفحات وب خود بکار می بریم.

کار را با یک مثال ساده شروع می کنیم :

مثال ۱-۶:

```
<body>
<form>
<input type="button" value="Click Me!"
onclick="window.alert('Hi!';" />
</form>
</body>
```

مثال بالا شامل یک رویداد می باشد که به محض فشردن دکمه Click Me فعال می شود و پیغام Hi! را به نمایش می گذارد.

در ادامه به معرفی انواع رویدادها و مثالهایی در باره هر کدام خواهیم پرداخت .

The Abort Event (onabort)

رویداد onabort زمانی اتفاق می افتد که یک کاربر لود شدن یک تصویر را در نیمه راه متوقف کند.

برای مثال اگر شما بخواهید زمانی که کاربر لود شدن یک عکس را متوقف می کند پیامی را دریافت نماید می توانید از دستور زیر استفاده نمایید:

مثال ۲-۶:

```
<img src="myimage.jpg" alt="my picture"
  onabort="window.alert('Why don\'t you want to see my
picture?');" />
```

رویدادهای مربوط به ماوس

The Click Event (onclick)

این رویداد زمانی رخ می دهد که کاربر روی یکی از المان های صفحه کلیک کند.

مثال ۳-۶:

```
<body>
  <form>
    <input type="button" value="Do not Click Here"
    onclick="window.alert('I told you not to click me!');">
  </form>
</body>
```

مثال ۴-۶:

```
<body>
<a href="http://none" onclick="window.alert('Hey! You clicked
me!');">
Don't Click Me</a>
</body>
```

مشکلی که در کد بالا وجود دارد این است که با وجود هشدار مرورگر تلاش می کند تا به لینک دستیابی پیدا کند و این باعث به نمایش در آمدن پیغام "Server not found" می شود شکل صحیح کد را در زیر مشاهده می کنید

مثال ۷-۶:

```
<body>
<a href="http://none"
onclick="window.alert('Hey! You clicked me!'); return false;">
Don't Click Me</a>
</body>
```

The Doubleclick Event (ondbclick)

این رویداد زمانی رخ می دهد که کاربر روی یکی از المان های صفحه دابل کلیک کند.

مثال ۸-۶:

در مثال زیر با دابل کلیک کردن روی مربع قرمز رنگ این رویداد اتفاق می افتد:

```
<html>
<head>
<title>Mouse Events Example</title>
  <script type="text/javascript">
    function handleEvent(oEvent) {
    var oTextbox = document.getElementById("txt1");
    oTextbox.value += "\n" + oEvent.type;
    }
  </script>
</head>
  <body>
    <p>Use your mouse to click and double click the red
square.</p>
<div style="width: 100px; height: 100px; background-color:
red"
ondblclick="handleEvent(event)" id="div1"></div>
<p><textarea id="txt1" rows="15" cols="50"></textarea></p>
  </body>
</html>
```

The Mouseover Event (onmouseover)

این رویداد زمانی رخ می دهد که کاربر اشاره گر ماوس را روی یکی از المانهای صفحه مانند یک لینک حرکت دهد:

مثال ۹-۶:

```
<a href="http://www.dlp.lxb.ir"
onmouseover="window.alert('I told you not to try to click
me!');">
Don't Try Clicking Me!</a>
```

The Mouseout Event (onmouseout)

این رویداد زمانی رخ می دهد که کاربر اشاره گر ماوس را از محدوده مورد نظر خارج نماید

مثال ۷-۶:

```
:<a href="http://www.dlp.lxb.ir"
onmouseout="window.alert('What, you didn\'t like my link?');">
Click Me!</a>
```

The Mousedown Event (onmousedown)

این رویداد در هنگام فشردن کلید ماوس توسط کابر در محدوده مورد نظر رخ می دهد.

مثال ۸-۶:

در مثال زیر با کلیک بر روی مربع قرمز رنگ و در هنگام پایین رفتن آن به طور کامل آن رویداد به وقوع می پیوندد.

```
<html>
<head>
<title>Mouse Events Example</title>
<script type="text/javascript">
function handleEvent(oEvent) {
var oTextbox = document.getElementById("txt1");
oTextbox.value += "\n" + oEvent.type;
}
</script>
</head>
<body>
<p>Use your mouse to click and double click the red
square.</p>
<div style="width: 100px; height: 100px; background-color:
red"
onmousedown="handleEvent(event)"
id="div1"></div>
<p><textarea id="txt1" rows="15" cols="50"></textarea></p>
</body>
</html>
```

The Mouseup Event (onmouseup)

این رویداد در هنگام رها کردن کلید ماوس فشرده شده توسط کابر در محدوده مورد نظر رخ می دهد.

مثال۹-۶:

در مثال زیر با کلیک بر روی مربع قرمز رنگ و در هنگام بالا رفتن آن به طور کامل رویداد به وقوع می پیوندد.

```
<html>
<head>
<title>Mouse Events Example</title>
<script type="text/javascript">
function handleEvent(oEvent) {
var oTextbox = document.getElementById("txt1");
oTextbox.value += "\n" + oEvent.type;
}
</script>
</head>
<body>
<p>Use your mouse to click and double click the red
square.</p>
<div style="width: 100px; height: 100px; background-color:
red"
onmouseup="handleEvent(event)"
 id="div1"></div>
<p><textarea id="txt1" rows="15" cols="50"></textarea></p>
</body>
</html>
```

The Mousemove Event (onmousemove)

این رویداد در هنگام به حرکت در آوردن نشانگر ماوس در محدوده مورد نظر به وقوع می پیوندد.

در مثال به محض حرکت در محدوده مستطیل قرمز رنگ رویداد به وقوع می پیوندد.

مثال ۱۰-۶:

```
<html>
<head>
```

```
<title>Mouse Events Example</title>
<script type="text/javascript">
function handleEvent(oEvent) {
var oTextbox = document.getElementById("txt1");
oTextbox.value += "\n" + oEvent.type;
}
</script>
</head>
<body>
<p>Use your mouse to click and double click the red
square.</p>
<div style="width:427px; height: 30px; background-color: red"
onmousemove="handleEvent(event)"
 id="div1"></div>
<p><textarea id="txt1" rows="15" cols="50"></textarea></p>
</body>
</html>
```

رویدادهای صفحه کلید

The Keydown Event (onkeydown)

زمانی رخ می دهد که کاربر یکی از کلیدهای صفحه کلید را فشار دهد .

مثال ۱۱-۶:

```
<body>
<input value="onkeydown" type="text"
onkeydown="alert('onkeydown')"  />
</body>
```

The Keypress Event (onkeypress)

زمانی رخ می دهد که کاربر یکی از کلیدهای صفحه کلید را فشار داده و و کاراکنر روی آن را تایپ نماید.

مثال ۱۲-۶:

```
<body>
<input value="onkeypress" type="text" onkeypress="alert('
onkeypress')" />
</body>
```

The Keyup Event (onkeyup)

زمانی رخ می دهد که کاربر کلید فشار داده شده را رها کند.

مثال ۱۳-۶:

```
<body>
<input value="onkeyup" type="text" onkeyup="alert('onkeyup')"
/>
</body>
```

برای درک بهتر رویدادهای مربوط به صفحه کلید فایل htmlی را که در مورد این مبحث است و ضمیمه کتاب می باشد را مورد بررسی قرار دهید.

رویدادهای مربوط به فرم :

این رویدادها مربوط به عملیاتی است که کاربر بر روی اجزای فرم انجام می دهد. البته ممکن است بعضی از این رویدادها بر روی متن های عادی در صفحه هم عکس العمل نشان دهند اما بیشترین کاربرد را در فرم ها دارند.

The Focus Event (onfocus)

زمانی اتفاق می افتد که کاربر بخواهد روی یکی از اجزای فرم یا پنجره متمرکز شود و پیامی را نمایش می دهد یا عکس العملی را نشان می دهد.

منظور از متمرکز شدن این است که بطور مثال برای نوشتن روی یک text box کلیک کند .

مثال ۱۴-۶:

```
<form>
Enter Your Name:
<input type="text" onfocus="window.alert('Don\'t forget to
capitalize!');" />
</form>
```

The Blur Event (onblur)

این رویداد برعکس رویداد focus است و هنگامی رخ می دهد که توسط موس و یا دکمه tab کیبورد بر روی یکی از اجزای فرم تمرکز خود را از دست بدهد.

مثال ۱۵-۶:

```
<form>
Give this box focus:<br />
<input type="text" onblur="window.alert('Hey! Come back!');"
/><br />
then give this box focus to blur the first one:<br />
<input type="text" />
</form>
```

The Change Event (onchange)

این رویداد زمانی رخ می دهد که کاربر یکی از اجزای فرم را تغییر دهد, مانند تغییر متن در کادر متنی.

مثال ۱۶-۶:

```
<form>
Are you cool?<br />
<select onchange="window.alert('Why did you change that?');">
<option selected="selected">Yes</option>
<option>No</option>
<option>Undecided</option>
</select>
</form>
```

The Submit Event (onsubmit)

این رویداد فقط هنگامی رخ می دهد که کاربر در یک فرم دکمه submit را فشار دهد.

مثال ۱۷-۶:

```
<form onsubmit="window.alert('Thank You');">
What's your name?<br />
<input type="text" id="thename" /><br />
<input type="submit" value="Submit Form">
</form>
```

The Reset Event (onreset)

این رویداد زمانی رخ می دهد که کاربر با کلیک بر روی دکمه reset صفحه وب را reset نماید.و به نوعی می توان آن را از رویدادهای مربوط به فرم ها هم در نظر گرفت.

در مثال زیر با کلیک بروی روی دکمه reset صفحه ریست می شود و رویداد به وقوع می پیوندد.

مثال ۱۸-۶:

```
<html >
<head>
<title> Form Events Example </title>
</head>
<body>
```

```
<form onReset="alert('onreset');">

<input type="reset" value="reset"
onReset="alert('onreset');"/><input type="text" />
</form>
</body>
</html>
```

The Select Event (onselect)

وقتی که قسمتی از فرم که این شناسه را در خود دارد انتخاب شود اسکرپت موجود در این المنت اجرا می شود.

در مثال زیر با انتخاب متن موجود در کادر این رویداد اجرا می شود.

مثال ۱۹-۶:

```
<html>
<head>
<title> Form Events Example </title></head>
<body>
<textarea cols="20" rows="10"
onselect="alert('onselect')";>Lerning JavaScript
Hadi Ahmadi
http://dlp.lxb.ir</textarea>
</body>
</html>
```

برای درک بهتر رویدادهای مربوط به فرم فایل htmlای را که در مورد این مبحث است و ضمیمه کتاب می باشد را مورد بررسی قرار دهید.

رویدادهای پنجره مرورگر

The Load Event (onload)

این رویداد هنگامی رخ می دهد که یک صفحه بطور کامل لود شود.

مثال ۲۰-۶:

```
<body onload="window.alert('I\'m done loading now!');">
Text for the body of the page...
</body>
```

اگر می خواهید که آن را به صورت یک فایل خارجی بکار ببریدمی توانید ابتدا قطعه کد زیر را در یک فایل .js مثلا load_alert.js ذخیره کنید .

مثال ۲۱-۶:

```
window.onload = function() {
window.alert('I\'m done loading now!');
};
```

سپس با استفاده از کد زیر آن را درون صفحه html خود بکار برید.

مثال ۲۲-۶:

```
<body>
<script type="text/javascript" src="load_alert.js"></script>
Text for the body of the page...
</body>
```

The Unload Event (onunload)

عکس رویداد بالا عمل می نماید.

مثال ۲۳-۶:

```
<body onunload="window.alert('Be sure to come back, OK?');">
Other HTML code goes here...
</body>
```

برای درک بهتر رویدادهای مربوط به پنجره مرورگر فایل htmlی را که در مورد این مبحث است و ضمیمه کتاب می باشد را مورد بررسی قرار دهید.

The resize event(onresize)

این رویداد در هنگام تغییر سایز پنجره مرورگر توسط کاربر اجرا می شود.

مثال ۲۳-۶:

```
<html>
<head>
<title>OnResize Example</title>
</head>
<body onresize=”alert(‘Resizing’)”>
</body>
</html>
```

The scroll event(onscroll)

این رویداد در هنگام اسکرول کردن یک پنجره به وقوع می پیوندد. چه اسکرول کردن عمودی و چه افقی.

مثال ۲۴-۶:

در مثال زیر به محض اسکرول کردن صفحه یک پیغام برای شما به نمایش در می آید.

```
<html>
<head>
```

```
<title>OnScroll Example</title>
</head>
<body onscroll=”alert(‘Scrolling’)”>
<p>Try scrolling this window.</p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
</body>
</html>
```

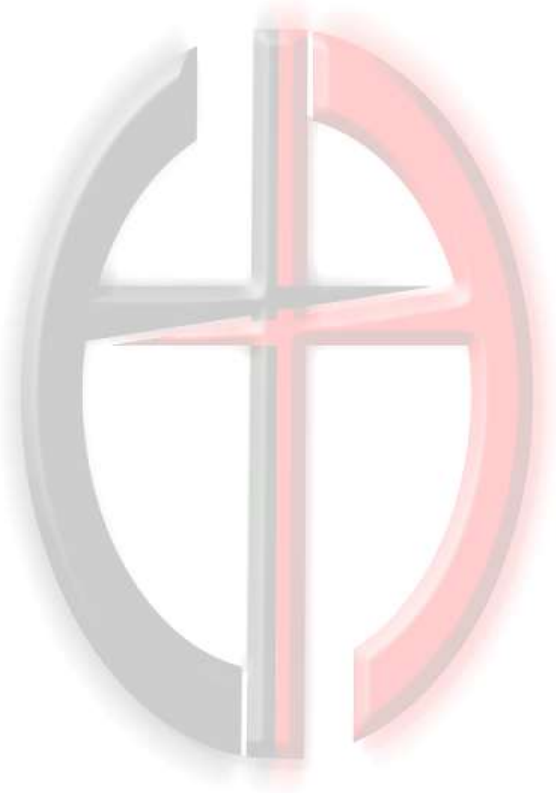
مثال ۲۵-۶:

در مثال زیر در هنگام اسکرول کردن صفحه پیغام هایی در جعبه متنی موجد در آن نمایش داده می شود.

```
<html>
<head>
<title>OnScroll Example</title>
<script type=”text/javascript”>
window.onscroll = function () {
var oTextbox = document.getElementById(“txt1”);
oTextbox.value += “\nscroll is at “ + document.body.scrollLeft
+ “
horizontally and “ + document.body.scrollTop + “ vertically.”;
}
</script>
</head>
<body>
<p>Try scrolling this window.</p>
<p><textarea rows=”15” cols=”50” id=”txt1”></textarea>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
```

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

فصل هفتم : اشیا



برنامه نویسی شیءگرا

جاوااسکریپت یک زبان شیءگراست. نحوه ساخت اشیاء در بخش پیشرفته در فصل های بعدی آموزش داده خواهد شد. در اینجا ما از اشیای داخلی جاوااسکریپت استفاده کرده ونگاهی به آنها می اندازیم و نحوه استفاده از آنها را می آموزیم. در چند جلسه بعدی که مربوط به این بخش است هر یک از این اشیا را بررسی اجمالی کرده و مثال های زیادی از آن ها را خواهیم دید.

هر شیء یک مجموعه خصوصیات (properties) و متدها (methods) دارد.

اشیاء (object)

همانگونه که در اتاق شما اشیاء مختلفی همچون میز , صندلی , تخت و غیره وجود دارد در دنیای مجازی نیز اشیاء وجود دارد به طور مثال دکمه هایی که با html ساخته اید , کادرهایی که برای عکس ساخته اید و غیره نیز مفهوم شیء را به ما می رساند.

اشیاء در جاوا اسکریپت

اشیاء در جاوا اسکریپت به دو دسته کلی تقسیم می شوند: دسته اول اشیایی هستند که به محیط اجرای جاوا اسکریپت تعلق دارند این به معنیست که کدها از قبل در مرورگری که شما از آن استفاده می کنید به صورت پیشفرض بر روی آن ها قرار گرفته شده، دسته دوم اشیایی که از قبل در خود جاوا اسکریپت تعریف شده اند (اشیای درون ساخت).

ساختار اشیا

ساختار اشیا و چگونگی تعریف آنها را با تعری یک مثال پی می گیریم:

مثلا برای ساخت شی car ابتدا آن را به صورت زیر تعریف می کنیم :

مثال ۱-۷:

```
function car(seats,engine,theradio) {
this.seats=seats;
this.engine=engine;
this.theradio=theradio;
}
```

در خط اول از قطعه کد بالا شما نام شی و پارامترهای آن را مشاهده می کنید و در خطوط بعدی و با استفاده از کلمه this انها را به مرورگر می شناسانیم.

حال برای مقدار دهی به پارامترها به طریق زیر عمل می کنیم :

```
var work_car= new car("cloth","V-6","Tape Deck");
```

ابتدا یک متغیر را تعریف کرده و سپس با استفاده از کلید new و نام شی از پیش تعریف شده مقادیر را به شی منتسب کرده و در متغیر ذخیره می نماییم.

و با استفاده از قطعه کد زیر شما می تواید به هر کدام از پارامترها دستیابی داشته باشید :

```
var engine_type= work_car.engine;
```

با استفاده از مثال زیر اشیا و نحوه استفاده از آنها برای شما ملموس تر و قابل فهم تر خواهد شد :

مثال ۲-۷:

```
<script type="text/javascript">
function car(seats,engine,theradio) {
this.seats=seats;
this.engine=engine;
this.theradio=theradio;
}
var work_car= new car("cloth","V-6","Tape Deck");
var fun_car= new car("leather","V-8","CD Player");
var engine_type= work_car.engine;
var seat_type= fun_car.seats;
var radio_type= fun_car.theradio;
document.write("I want a car with "+seat_type+" seats.<br
/>");
document.write("It also needs a "+engine_type+" engine.<br
/>");
document.write("Oh, and I would like a "+radio_type+" also.");
</script>
```

مثال ۳-۷:

```
function car(seats,engine,theradio) {
this.seats=seats;
this.engine=engine;
this.theradio=theradio;
}
var work_car= new car("cloth","V-6","Tape Deck");
var fun_car= new car("leather","V-8","CD Player");
work_car.engine="V-4";
var custom_car= new
car(fun_car.seats,work_car.engine,fun_car.theradio);
document.write("I want a car with "+custom_car.seats+"
seats.<br />");
document.write("It also needs a "+custom_car.engine+"
engine.<br />");
document.write("Oh, and I would like a "+custom_car.theradio+"
also.");
```

در قطعه کد بالا و با استفاده از work_car.engine="V-4"; یک مقدار جدید را شی work_car وارد می کنیم و همانطور که با اجرای کد مشاهده خواهید کرد برنامه مقدار جدید را به جای مقدار تعریف شده نمایش می دهد.

مثال ۴-۷:

```
function car(seats,engine,theradio) {
this.seats=seats;
this.engine=engine;
this.theradio=theradio;
}
var work_car= new car("cloth","V-6","Tape Deck");
var fun_car= new car("leather","V-8","CD Player");
var first_engine=work_car.engine;
work_car.engine="V-4";
var custom_car= new
car(fun_car.seats,work_car.engine,fun_car.theradio);
document.write("At first, I wanted a "+first_engine+"
engine.<br />");
document.write("But after thinking about it a bit:<br />");
document.write("I want a car with "+custom_car.seats+"
seats.<br />");
document.write("It also needs a "+custom_car.engine+"
engine.<br />");
document.write("Oh, and I would like a "+custom_car.theradio+"
also.");
```

اسفاده از روش object initializer

در این روش شما می توانید پارامترها و مقادیر را به صورت زیر در ساختار شی وارد کنید :

مثال ۵-۷:

```
work_car= {seats:"cloth",engine:"V-6",theradio:"Tape Deck"}
fun_car= {seats:"leather",engine:"V-8",theradio:"CD Player"}
```

مثال ۶-۷:

```
work_car= {seats:"cloth",engine:"V-6",theradio:"Tape Deck"}
fun_car= {seats:"leather",engine:"V-8",theradio:"CD Player"}
document.write("I want a car with "+fun_car.seats+"
seats.<BR>");
document.write("It also needs a "+work_car.engine+"
engine.<BR>");
document.write("Oh, and I would like a "+fun_car.theradio+"
also.");
```

مثال عملی در مورد استفاده از اشیا :

در مثال زیر شما نحوه استفاده از اشیا و دیگر ساختارهای جاوااسکریپت برای ساخت و نمایش دادن اطلاعات کاربرپسند در مورد موضوعی خاص (در اینجا آپشن های مختلف خودرو) را خواهید آموخت :

مثال ۷-۷:

```
function get_payment() {
var the_payment=250;
the_payment += (this.seats == "leather") ? 100 : 50;
the_payment += (this.engine == "V-8") ? 150 : 75;
the_payment += (this.theradio == "CD Player") ? 35 : 10;
return the_payment;
}
function car(seats,engine,theradio) {
this.seats=seats;
this.engine=engine;
this.theradio=theradio;
this.payment=get_payment;
}
var work_car= new car("cloth","V-6","Tape Deck");
var fun_car= new car("leather","V-8","CD Player");
var custom_car= new
car(fun_car.seats,work_car.engine,fun_car.theradio);
var work_car_payment= work_car.payment();
var fun_car_payment= fun_car.payment();
var custom_car_payment= custom_car.payment();
document.write("<h2>The information on the cars you
requested:</h2>");
document.write("<strong>Work Car: </strong>");
document.write(work_car.seats+","+work_car.engine+","+work_car
.theradio);
document.write("<br />");
```

```
document.write("<strong>Payments:</strong>
$"+work_car_payment);
document.write("<p>");
document.write("<strong >Fun Car: </strong>");
document.write(fun_car.seats+","+fun_car.engine+","+fun_car.th
eradio);
document.write("<br />");
document.write("<strong>Payments:</strong>
$"+fun_car_payment);
document.write("</p>");
document.write("<p>");
document.write("<strong>Custom Car: </strong>");
document.write(custom_car.seats+","+custom_car.engine+",");
document.write(custom_car.theradio);
document.write("<br />");
document.write("<strong>Payments:</strong>
$"+custom_car_payment);
document.write("</p>");
```

تنها نکته مثال بالا استفاده از تابع زیر می باشد:

```
function get_payment() {
var the_payment=250;
the_payment += (this.seats == "leather") ? 100 : 50;
the_payment += (this.engine == "V-8") ? 150 : 75;
the_payment += (this.theradio == "CD Player") ? 35 : 10;
return the_payment;
}
```

که برای درک بهتر, آن را به صورت ساختار آشنای if else بازنویسی می کنیم :

```
function get_payment() {
var the_payment=250;
if(this.seats == "leather") {
the_payment+=100;
}
else {
the_payment+=50;
if(this.engine == "V-8") {
the_payment+=150;
}
else {
the_payment+=75;
```

```
}
if(this.theradio == "CD Player") {
the_payment+=35;
}
else {
the_payment+=10;
}
return the_payment;
}
```

همانطور که مشاهده می نمایید در تابع بالا شروطی تعیین شده است که در صورت درست بودن مقداری را به متغیر the_payment که مقدار پایه ای و پیش فرضی برای آن در نظر گرفته شده است اضافه گردد مثلا برای موتور v-8 مقدار 150$ و برای سایر انواع موتورها مقدار 75$ به متغییر the_payment اضافه گردد و در آخر با استفاده از دستور return مقدار آن برگشت داده می شود.

دستور For – in Loop

این دستور به شما اجازه دستیابی به پارامترهای یک شی یا نمایش و دستکاری مقادیر آنها را می دهد.

```
for (var variable_name in object_name) {
JavaScript statements
}
```

مثال ۸-۷:

دستور زیر باعث می شود که شما بدون هیچ کار اضافی به پارامترهای شی work_car دسترسی پیدا کنید و آنها را به نمایش در آورید :

```
function car(seats,engine,theradio) {
this.seats=seats;
this.engine=engine;
this.theradio=theradio;
}
var work_car= new car("cloth","V-6","Tape Deck");
for (var prop_name in work_car) {
document.write(work_car[prop_name]+"<br />");
}
```

work_car[prop_name] در واقع یک آرایه است که وظیفه نگهداری ویژگی های شی work_car برای چاپ را برعهده دارد.

دستور The with Statement

این دستور نیز مانند دستور قیل دسترسی به پارامترهای آرایه را راحت تر می کند.

مثال ۹-۷:

```
function car(seats,engine,theradio) {
this.seats=seats;
this.engine=engine;
this.theradio=theradio;
}
var work_car= new car("cloth","V-6","Tape Deck");
with (work_car) {
document.write("Seats: "+seats+"<br />");
document.write("Engine: "+engine+"<br />");
document.write("Radio: "+theradio);
}
```

اشیا از پیش تعریف شده در جاوا اسکریپت

در جاوا اسکریپت اشیا از پیش تعریف شده زیادی وجود دارد که به شما برای دسترسی بهتر و بیشتر به پارامترها و متدها یاری می رسانند.

The Navigator Object(اشیا هدایت گر)

اشیا navigator به شما اجازه دستیابی به اجزا و تنظیمات مرورگر کاربر را می دهند. مانند نام, ورژن و غیره .

Properties

Properties در navigator object به شما قابلیت دسترسی به اطلاعاتی در مورد نوع مرورگر کاربر را می دهد.

در جدول زیر این دستورات را مشاهده می کنید.

Property	Value
appCodeName	The code name of the browser
appMinorVersion	The name of the browser
appMinorVersion	appMinorVersion A string representing the minor version of the browser (Internet Explorer only)
appVersion	The version of the browser and some other information
browserLanguage	The language of the browser being used (Internet Explorer and Opera)
buildID	The build identifier of the browser being used (Firefox only)
cookieEnabled	Specifies whether or not the browser has cookies enabled
cpuClass	A string representing the class of the CPU (Internet Explorer only)
language	The language of the browser being used (Firefox and Opera)
mimeTypes	An array of MIME types supported by the browser
onLine	Specifies whether or not the browser is in “global offline mode”
oscpu	A string representing the operating system of the computer (Firefox only)
platform	The machine type for which the browser was created
plugins	An array of the plugins the browser has installed on it
product	A string representing the product name of the browser being used (Firefox only)
productSub	The build number of the browser being used (Firefox only)
securityPolicy	An empty string—returned a value in Netscape 4.7 (Firefox only)
systemLanguage	The default language used by the operating system (Internet Explorer only)
userLanguage	The natural language of the operating system (Internet Explorer and Opera)
userAgent	The user agent header for the browser
vendor	An empty string—returned a string representing the vendor of the browser being used (Firefox only)
vendorSub	An empty string—returned the vendor version number of the browser being used (Firefox only)

نکته مهم : در هنگام استفاده از Properties در navigator object به نحوه نوشتن حروف بزرگ و کوچک دقت کنید.

مثال ۱۰-۷:

```
<script type="text/javascript">
window.alert("You are using "+navigator.propertyname);
</script>
```

نکته : تمام خاصیت های جدول بالا را می توانید به جای propertyname به کار ببرید.

مثال ۱۱-۷:

کد زیر نوع و اسم مرورگر مورد استفاده کاربر را نمایش می دهد.

```
<body>
<script type="text/javascript">
switch (navigator.appName) {
case "Netscape" : window.alert("Firefox/Netscape is cool.");
break;
case "Microsoft Internet Explorer" : window.alert("Internet
Explorer is Cool.");
break;
case "Opera" : window.alert("Opera is cool."); break;
default : window.alert("What browser is this?");
}
</script>
Hi, and welcome!
</body>
```

متدها

Navigator object دارای تعدادی متد نیز می باشد که می توانید برای انجام کارهای مختلف از آنها استفاده کنید .

در جدول زیر اسامی آنها و توضیحی در باره هر کدام را مشاهده می کنید.

Method	Purpose
javaEnabled()	Used to test whether or not Java is enabled in the browser
mozIsLocallyAvailable()	Checks to see if a file at a certain address is available locally (Firefox only)
preference()	Allows certain browser preferences to be set (requires signed script)
registerContentHandler()	Allows a Web site to set itself as a potential handler of a certain MIME type
registerProtocolHandler()	Allows a Web site to set itself as a potential handler of a certain protocol
taintEnabled()	Returns false— because the method is no longer being in use. It was used to specify whether or not data tainting was enabled in the browser

مثال ۱۲-۷:

کد زیر مشخص می کند که آیا مرورگر شما اجازه نمایش اسکریپت ها را داردیا خیر.

```
var hasJava= navigator.javaEnabled()
if (hasJava==true) {
window.alert("Cool, you have Java!");
}
else {
window.alert("Java disabled? You cannot see my Java Applet!");
}
```

The History Objet

این شی در حقیقت جزء اشیا windows میباشد و اطلاعاتی را در مورد تاریخچه صفحه جاری مرورگر فراهم می کند.

مثال ۱۳-۷:

```
<body>
<script type="text/javascript">
alert("Your current window has viewed "+history.length+"
pages!")
</script>
</body>
```

متدهای شیء تاریخچه

در جدول زیر سه متد برای این شیء آورده شده است:

Method	Purpose
Back()	Sends the browser window back one page in the history list
Forward()	Sends the browser one page forward in the history list
Go()	Sends the browser to a specified page in the history list using an integer value

The back() Method

این متد مرورگر را به آخرین صفحه دیده شده قبل از صفحه جاری در لیست تاریخچه قبل از صفحه جاری می فرستد.

مثال ۱۴-۷:

```
<body>
<form>
<input type="button" value="Back" id="back_button" />
</form>
<script type="text/javascript" >
var bb = document.getElementById("back_button");
bb.onclick = function() {
history.back();
};
</script>
</body>
```

The forward() Method

این متد مرورگر را به صفحه قرار گرفته شده در لیست تاریخچه بعد از صفحه جاری می فرستد.

مثال ۱۵-۷:

```
<body>
<form>
<input type="button" value="Back" id="back_button" /><br />
<input type="button" value="Forward" id="forward_button" />
</form>
<script type="text/javascript" >
var bb = document.getElementById("back_button");
var fb = document.getElementById("forward_button");
bb.onclick = function() {
history.back();
};
fb.onclick = function() {
history.forward();
};
</script>
</body>
```

The go() Method

به اندازه عددی که به عنوان پارامتر دریافت می کند در لیست تاریخچه جابه جا می شود.

مثال ۱۶-۷:

```
history.go(-2);
```

کد بالا دو صفحه در لیست تاریخچه مرورگر به عقب بر میگردد.

فصل هشتم : DOM

مدل شی گرای سند؛DOM

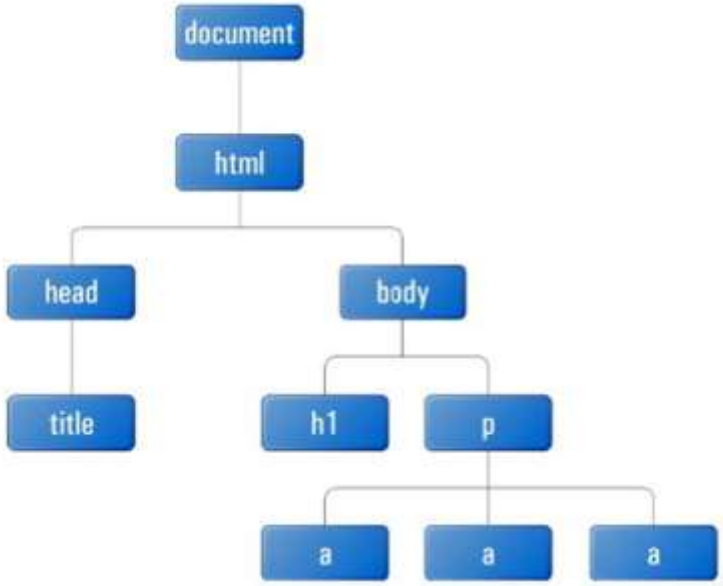
به جرات می توان گفت که DOM مهم ترین جزء برنامه نویسی جاوا اسکریپت است, که به وسیله آن قادر خواهیم بود به عناصر موجود در صفحه دسترسی داشته باشیم , آنها را دستکاری کنیم مانند جابه جا کردن و حذف و اضافه کردن آنها.

با استفاده از قطعه کد زیر ابتدا به تشریح مفاهیم ابتداییDOM خواهیم پرداخت و سپس روش هایی که برای دستکاری عناصر موجود در صفحه را فراهم کرده را توضیح خواهیم داد .

مثال ۱-۸:

```
<html>
<head>
<title>DOM </title>
</head>
<body>
<h1> Learning DOM </h1>
<p>
The document object is an object that is created by the
browser for each new HTML page
(document) that is viewed.
For more information visit :
<a href=http://www.dlp.lxb.ir/ rel="external">Dehloran PC</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 به صورت زیر نمایش داد :



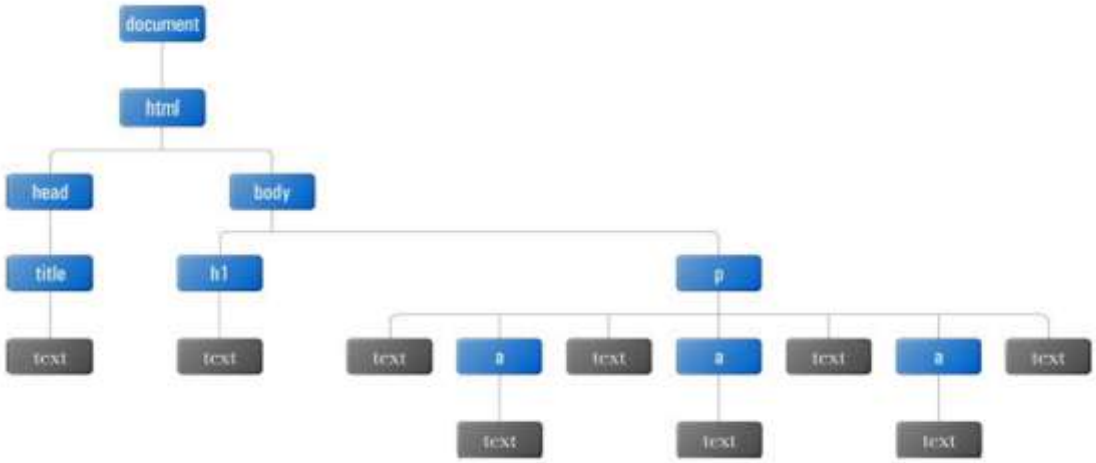
در درخت بالا هر مستطیل به عنوان یک گره در نظر گرفته می شود و در ریشه آن گره document قرار دارد که همیشه در بالاترین سطح درخت قرار می گیرد.

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 ، گره عنصری گفته می شود.

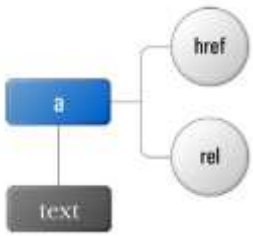
text

این نوع گره ها به متن موجود در داخل یک تگ آغازی و تگ پایانی اشاره دارند .این نوع گره ها هم نمی توانند فرزند داشته باشند. به این نوع گره ها ، 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

گره ای که به یک صفت از یک عنصر اشاره می کند و فاقد فرزند می باشد .به این نوع گره ها ، 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 rel و href هستند که می توان آن ها را به صورت زیر نمایش داد:



comment

به گره های توضیحی اشاره می کند و فاقد فرزند است. غالبا گرهی اصلی به عنوان راس این درخت وجود دارد که همانdocument است.

استفاده از ویژگی های مدل شی گرای سند

DOM این اجازه را به جاوا اسکریپت می دهد تا به ساختار سند در مرورگر دسترسی پیدا کند.و این کار را با استفاده از ویژگی هایی انجام می دهد که در جدل زیر آورده شده اند.

Property	Description
activeElement	Returns a string holding the value of the active element in the document
alinkColor	Returns the hexadecimal value of the active link color of the document
anchors	An array of all the named anchors in the document
async	Used to tell the browser whether to load a document with an asynchronous request or a synchronous request
applets	An array of all the Java applets in a document
bgColor	Returns the hexadecimal value of the background color of the document
body	Returns the body or frameset element of the document
characterSet	Returns a string value that represents the character set used to encode the document
charset	Returns a string value that represents the character set used to encode the document
childNodes	An array of all of the child nodes of the document
compatMode	Returns the string “BackCompat” if the document is rendered in Quirks modor the string “CSS1Compat” if the document is rendered in Strict mode
contentType	Returns a string for the Content-Type from the MIME header of the document
cookie	Used to set JavaScript cookies in a document
defaultCharset	Returns a string value that represents the default character set used to encode the documen
defaultView	References the window object for the document
designMode	Returns a string value that provides information on whether or not the document can be edited
dir	Returns a string value that represents the reading direction of the document
doctype	Returns the doctype declaration associated with the document
documentElement	Returns a string representing the root node of the document
documentURIObject	Returns an object representing the URI of the document (only available to privileged JavaScript code)
domain	Returns the domain name of the server for the document
embeds	An array of all the embed tags in the document
expando	Returns a Boolean value based on whether or not arbitrary variables can be created within the document
fgColor	Returns the hexadecimal value of the default text color of the document
fileCreatedDate	Returns the date the document was created
fileModifiedDate	Returns the date the document was last modified
formName	Not a property itself, but creates a new property with each named form placedin the document
forms	An array of all the form tags in a document
frames	An array of all of the frames used in the document
height	Returns the height, in pixels, of the body element of the document
images	An array of all the image (img) tags in the documen
implementation	Returns a string value representing the implementation object of the document

inputEncoding	Returns a string representing the document's encoding
lastModified	Returns the date of the last modification of the document
layers	An array of all the layer tags on the page (Netscape Navigator 4 only)
all	Allows access to all the objects on a page
linkColor	Returns the hexadecimal value of the default link color for the document
location	Returns the URI of the document
links	An array of all the link (<a>) tags in the document
namespaces	An array of all the namespaces in the document
parentWindow	Returns a reference to the parent window (the parent window's document object)
plugins	An array of all the plugins used in the document
protocol	Returns the protocol portion of the Web address (URL) of the document
readyState	Returns a string value that represents the current state of the document
referrer	Returns the URL of the document that referred the viewer to the current document
scripts	An array of all the script tags used in the document
styleSheets	An array of all the style sheets used in the document
tags	Sets the style of an HTML tag in the document
title	Returns the text used inside the title tags of the document
uniqueID	Returns a string value that represents a unique ID given to the document
URL	Returns the URL of the current document
URLUnencoded	Returns the URL of the document without any encoding
vlinkColor	Returns the hexadecimal value of the visited link color for the document
width	Returns the width, in pixels, of the body element of the document

The Color Properties

این خاصیت برای رنگ پس/پیش زمینه و همچنین رنگ لینکها و غیرع بکار می رود. البته پیشنهاد می شود که این ویژگی ها را در CSS اعمال نمایید و یا تغییر دهید.

The anchors Property (Array)

آرایه ای از تمام نام های anchor موجود در سند مانند (such as <a name="top">) را برمی گرداند.

مثال ۲-۸ :

```
<body>
<h1>My Page</h1>
<a name="sec1"></a>
<h2>Section 1</h2>
```

```
This section is all about section 1 stuff...
<a name="sec2"></a>
<h2>Section 2</h2>
This section talks about all the section 2 issues and ...
<br />
<script type="text/javascript">
document.write("There are "+document.anchors.length+" named
anchors");
</script>
</body>
```

با اجرای کد فوق مشاهیده خواهید کرد که تعداد تگهای anchor به نمایش در خواهد آمد.

The cookie Property[1]

برای ایجاد کوکیها در جاوااسکریپت از خاصیت cookie. شی 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;
```

۱. نکته : مطالب مبحث کوکی عینا از فصلی با همین نام از کتاب آموزش کاربردی جاوا اسکریپت نوشته آقای احمد بادپی آورده شده اند.

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 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 Date شیoGMTString() 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

مثال ۳-۸ :

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 ";
```

مثال ۴-۸ :

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 کاراکترهای نامعتبر 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 کوکی ها

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:

مثال ۵-۸ :

```
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 ودسترسی 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 از جفت های name=value تمامی کوکیهای قابل دسترس برای سند جاری را که با ; از هم جدا شده اند برمی گرداند . به عنوان مثال متغیر x می توانید حاوی رشته ای به صورت زیر باشد:

مثال ۶-۸ :

```
"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 متدهای خاص آرایه به آن ها دسترسی داشته باشیم .به عنوان مثال برای چاپ مقدار کوکی های فوق می توان به صورت زیر عمل کرد:

مثال ۷-۸ :

```
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];
```

The domain Property

اسم سایت ارائه دهنده صفحه را برمی گرداند

مثال ۸-۸ :

```
<body>
<script type="text/javascript">
window.alert("You have reached the "+document.domain+"
domain!");
</script>
</body>
```

اگر این کد در هریک از صفحات یک سایت مثلا : dlp.lxb.ir قرار گرفته باشد هشدار زیر را بر می گرداند:

```
“You have reached the dlp.lxb.ir domain!”
```

The formname Property

The formname Property در واقع به خودی خود یک خاصیت نیست. بلکه زمانی به کار می آید که شما یک فرم و به تبع آن نام یک فرم را داشته باشید برای متوجه شدن مطلب به مثال زیر دقت کرده و آن را اجرا نمایید :

مثال ۹-۸ :

```
<body>
<form name="funform">
<input type="button" name="funb" value="You can click me I
suppose"
onclick="document.funform.funb.value='Thanks, you clicked
me!';" />
</form>
</body>
```

The lastModified Property

این ویژگی تاریخ و ساعت آخرین تغییر در صفحه را برمی گرداند.

مثال ۱۰-۸ :

```
<body>
<h1>My Always Updated Web Page!</H1>
<script type="text/javascript">
document.write("Last Updated: "+document.lastModified);
</script>
</body>
```

The layers Property (Array)

می تواند به شما برای شناسایی مرورگرهای بر پایه نت اسکیپ کمک کند.

مثال ۱۱-۸ :

```
if (document.layers) {
window.alert("You have Netscape Navigator 4!");
}
```

The all Property

می تواند به شما برای شناسایی مرورگر اینترنت اکسپلوررکمک کند.

مثال ۱۲-۸ :

```
if (document.all) {
window.alert("You have Internet Explorer 4 or better!");
}
```

The links Property (Array)

با استفاده از این خاصیت شما می توانید تعداد لینک های موجود در صفحه را شناسایی کنی.

مثال ۱۳-۸ :

```
<body>
<a href="http://www.dlp.lxb.ir" >
<a href="http://www.dlp.loxblog.com">
<script type="text/javascript">
document.write(document.links.length);
</script>
</body>
```

The title Property

این ویژگی عنوان صفحه را به عنوان خروجی برمی گرداند.

مثال ۱۴-۸ :

```
<head>
<title>JavaScript!</title>
</head>
<body>
<script type="text/javascript">
document.write("<h1>" + document.title + "</h1>");
</script>
Learning javascript with us!
</body>
```

The URL Property

آدرس کامل صفحه جاری را برمی گرداند.

مثال ۱۵-۸ :

```
<body>
<h1>Buy Something!</h1>
If you don't buy something I will be really upset so you had
better...
```

```
<br/><br/>
<script type="text/javascript">
document.write("You are at: "+document.URL);
</script>
</body>
```

The URLUnencoded Property

آدرس کامل صفحه جاری را برمی گرداند, بدون هرگونه رمزگشایی(encode کردن)
به طور مثال در URL Property ممکن است که فضای خالی به صورت 20% نمایش داده شود در حالی که در URLUnencoded Property از نمایش آن چشم پوشی می شود.

استفاده از متدهای DOM

در جدول زیر نام هریک از این متدها و متوضیحاتی در مورد آنها را آورده شده است. با استفاده از این متدها شما می توانید ویژگی و قابلیت های جدیدی را به اسناد خود وارد کنید.

Method	Description
attachEvent()	Attaches a function to an event, so that the function runs when the eventoccurs (Internet Explorer only)
createAttribute()	Creates an attribute with a name that is sent to it as a parameter
createAttributeNS()	Creates a new attribute in a particular namespace
createCDATASection()	Creates a new CDATA section
createComment()	Creates a comment with the value that is sent to it as a parameter
createDocumentFragment()	Creates a new document fragment
createElement()	Creates an element of the type sent to it as a parameter
createElementNS()	Creates an element in a particular URI and a particular type sent to it as parameters
createEntityReference()	Creates a new entity reference
createEvent()	Creates an event
createEventObject()	Creates an event object for the purpose of passing event information
createNodeIterator()	Creates a node iterator object
createNSResolver()	Creates a namespace resolver
createProcessingInstruction()	Creates a processing instruction
createRange()	Creates a range object
createStyleSheet()	Creates a style sheet for the document to use (Internet Explorer only)

createTextNode()	Creates a text string from the value sent to it as a parameter
createTreeWalker()	Creates a treewalker object
detachEvent()	Detaches a function from an event (Internet Explorer only)
elementFromPoint()	Returns the element object that appears at the location that is sent to it in two parameter values (pixels from left and pixels from top)
evaluate()	Returns a result based on the parameters sent to it
execCommand()	Executes a command on the document when the document is in designmode
getElementById()	Returns a reference to the object with the ID attribute that is sent to as a parameter
getElementsByClassName()	Returns references to the elements with the class name that is sent to it as a paramete
getElementsByName()	Returns references to the objects with the name attribute that is sent it as a parameter
getElementsByTagName()	Returns references to the elements with the tag name that is sent to it s a parameter
getElementsByTagNameNS()	Returns references to the elements with the tag name and namespace sent to it as parameters
getSelection()	Returns the value of a string of selected text in the document
hasFocus()	Returns a Boolean value based on whether or not the document has focus
load()	Loads an XML document
mergeAttributes()	Copies attributes from an object
open()	Opens a new document that allows you to write its contents using write() or writeln() statements
close()	Closes a new document that has been opened with the open() method
queryCommandEnabled()	Returns a Boolean value based on whether or not a command sent to it as a parameter can be executed
queryCommandIndeterm()	Returns a Boolean value based on whether or not a command sent to it as a parameter is in the indeterminate state
queryCommandState()	Returns a Boolean value based on whether or not a command sent to it as a parameter has executed
queryCommandSupported()	Returns a Boolean value based on whether or not a command sent to it as a parameter is supported
queryCommandValue()	Returns the current value of the document for the command that is sent to it as a parameter
Method	Description
queryCommandValue()	Returns the current value of the document for the command that is sent to it as a parameter
recalc()	Recalculates the dynamic properties in the document
releaseCapture()	Releases the mouse capture from the document
setActive()	Sets an object as active, but does not give it focus
write()	Allows you to write a string of text into an HTML document
writeln()	Allows you to write a string of text into an HTML document,

	but ends the line with a JavaScript newline character

متد getElementById()

این متد به شما اجازه می دهد که به خاصیتهای یک جزء از صفحه به وسیله آیدی آن دسترسی پیدا کنید.

مثال ۱۶-۸ :

```
<div id="some_text">This is some text.</div>
var text_element = document.getElementById("some_text");
```

متد getElementByName()

برای دسترسی به عناصری از صفحه که صفت name آنها برابر مقداری خاص است استفاده می شود.

مثال ۱۷-۸ :

```
<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 می خواهیم فقط مقدارradiobuttonی را که انتخاب شده است را پیدا کنیم.برای این امر از کد زیر استفاده می کنیم:

```
var oRadios = document.getElementsByName("radColor");
```

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

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

متد The getElementsByClassName()

همانند متدهای بالا عمل می کند با این تفاوت که از نام کلاسی که به عناصر تخصیصی یافته است برای دسترسی به آنها استفاده می کند.

مثال ۱۹-۸ :

فرض کنید می خواهیم به تمام اجزایی که کلاس آنها number_one است را دسترسی پیدا کنیم بدین منظور از قطعه کد زیر استفاده می کنیم:

```
var my_class = document.getElementsByClassName("number_one");
```

متد getElementsByTagName()

این متد به شما اجازه دستیابی به آرایه ای از عناصری را می دهد که دارای تگ مشخص هستند.

مثال ۲۰-۸ :

فرض کنید می خواهیم به تمام عناصری که دارای تگ img هستند دسترسی پیدا کنیم بدین منظور از کد زیر استفاده می کنیم:

```
var all_images = document.getElementsByTagName("img");
```

متدهای open() و close()

متد open() به شما اجازه ساخت یک صفحه جدید و نوشتن در آن را می دهد و برای پایان آن از متد close() استفاده می گردد.

مثال ۲۱-۸ :

در مثال زیر کاربر ابتدا نام خود را وارد می کند سپس با submit کردن آن یک صفحه جدید باز شده که ورود کاربر را با ذکر نامی که در ابتدا وارد کرده خوش آمد می گوید.

```
<body>
<strong>Enter your name in the box below, then click
the button to see a personalized page!</strong>
<br />
<form id="newp" onsubmit="newpage();">
Name: <input type="text" id="yourname" size="25">
<br/><br/>
<input type="submit" value="Submit">
</form>
<script type="text/javascript" >
function newpage() {
var thename = document.getElementById("yourname").value;
document.open();
document.write("<h1>Welcome!</h1>");
document.write("Hello, "+thename+", and welcome to my page!");
document.close();
}
</script>
</body>
```

خواص گره ها در DOM

در جدول زیر خاصیتهای گره ها در DOM آورده شده اند.

Property	Description
attributes	An array of all of the attributes in the specified node; the name and value properties of this property can be used to access the attribute name or attribute value for each member of the array
childNodes	An array of all the child nodes of the specified node
className	Returns the value of the class attribute of the specified node
clientHeight	Returns the height, in pixels, of the specified node
clientWidth	Returns the width, in pixels, of the specified node
dir	Returns the value of the direction of the text in the specified node (ltr or rtl)
firstChild	Returns the first child node of the specified node
id	Returns the value of the id of the specified node
innerHTML	Returns the HTML code (text, image code, tags, etc.) within the specified node, such as all of the HTML code within a div element
lang	Returns the language value of the specified node
lastChild	Returns the last child node of the specified node
nextSibling	Returns the node following the specified node
nodeName	Returns the name of the specified node (such as div for a div element)
nodeType	Returns the type of the specified node
nodeValue	Returns the value of the specified node (such as the text within a div element or the value of an attribute)
offsetHeight	Returns the offset height of the specified node
offsetWidth	Returns the offset width of the specified node
ownerDocument	Returns the document object that contains the specified node
parentNode	Returns the parent node of the specified node
previousSibling	Returns the node before the specified node
scrollLeft	Returns the difference between the left edge and the left edge in view of the specified node
scrollTop	Returns the difference between the top edge and the top edge in view of the specified node
scrollHeight	Returns the entire height (including anything hidden and viewable via a scroll bar) of the specified node
scrollWidth	Returns the entire width (including anything hidden and viewable via a scroll bar) of the specified node
style	Returns the style object of the specified node
tabIndex	Returns the tab index of the specified node
tagName	Returns the tag name (in uppercase) of the specified node
title	Returns the value of the title attribute of the specified node

در مثال زیر نحوه استفاده از خاصیت title را تشریح می کنیم.

مثال ۲۲-۸ :

```
<body>
<div id="div1" title="All about me!">
This page is about me, me, and... me!
</div>
<script type="text/javascript">
var me_div = document.getElementById("div1");
var me_title = me_div.title;
window.alert("The title of the div element is" + me_title);
</script>
</body>
```

متدهای مربوط به گره ها در DOM

در جدول زیر لیستی از این متدها را مشاهده می نمایید.

Method	Description
addEventListener()	Adds an event listener to the specified node to run a function on the event sent to it as a parameter
appendChild()	Appends a node as the last child of the specified node
attachEvent()	Attaches an event to the specified node to run a function on the eventsent to it as a parameter
blur()	Removes focus from the specified node
click()	Executes the click event on the specified node
cloneNode()	Creates a clone of the specified node
detachEvent()	Detaches an event from the specified node
dispatchEvent()	Executes an event on the specified node
focus()	Gives focus to the specified node
getAttribute()	Returns the value of the attribute name sent to it as a parameter on the specified node
getAttributeNS()	Returns the value of the attribute name and namespace sent to it as a parameter on the specified node
getAttributeNode()	Returns the attribute node of the attribute name sent to it as a parameter for the specified node
getAttributeNodeNS()	Returns the attribute node of the attribute name and namespace sent to it as parameters for the specified node
getElementsByTagName()	An array of all the child element nodes with the tag name sent to it as a parameter in the specified node

getElementsByTagNameNS()	An array of all the child element nodes with the tag name and namespace sent to it as parameters in the specified node
hasAttribute()	Returns true if the attribute name sent to it as a parameter exists on the specified node, or false if not
hasAttributeNS()	Returns true if the attribute name and namespace sent to it as parameters exist on the specified node, or false if no
hasAttributes()	Returns true if the specified node has any attribute nodes defined, or false if not
hasChildNodes()	Returns true if the specified node has any child nodes. or false if not
insertBefore()	Inserts a node sent to it as a parameter before the node sent to it as a second parameter inside the specified node
normalize()	Normalizes the specified node
removeAttribute()	Removes the attribute node for the attribute name sent to it as a parameter from the specified node
removeAttributeNode()	Removes the attribute node for the attribute node object reference sent to it as a parameter from the specified node
removeAttributeNS()	Removes the attribute node for the attribute name sent to it as a parameter with the namespace sent to it as a parameter from the specified node
removeChild()	Removes the child node sent to it as a parameter from the specified node
removeEventListener()	Removes an event listener from the specified node
replaceChild()	Replaces the child node sent to it as the second parameter with the child node sent to it as the first parameter in the specified node
scrollIntoView()	Scrolls the specified node into view in the browser window
setAttribute()	Sets an attribute node's name (first parameter) and value (second parameter) for the specified node
setAttributeNode()	Sets an attribute node as the attribute node object sent to it as a parameter for the specified node
setAttributeNodeNS()	Sets an attribute node as the attribute node object sent to it as a parameter with the namespace sent to it as a parameter for the specified node
setAttributeNS()	Sets an attribute node's namespace (first parameter), name (secondparameter), and value (third parameter) for the specified node

مثال ۲۳-۸ :

```
<body>
<div id="div1" title="All about me!">
This page is about me, me, and... me!
</div>
</body>
```

این کد دارای یک گره div و یک گره فرزند text است. اگر شما بخواهید گره های دیگری را به آن اضافه کنید باید از document.createElement(),document.createTextNode() , و متد گره DOM, appendChild() استفاده کنید.

ابتدا به کد جاوا اسکریپت رفته و با استفاده از کد زیر به div دسترسی پیدا کنید:

مثال ۲۴-۸ :

```
var me_div = document.getElementById("div1");
```

سپس با استفاده از کد زیر یک عنصر اضافه نمایید :

```
var inner_div = document.createElement("div");
```

بعد از آن یگ گره متنی را به inner_div اضافه نمایید:

```
var inner_div_text = document.createTextNode("More about
me...")
```

سپس با استفاده از متد appendChild() این گره را به عنوان یکی از فرزندان گره inner_div انتخاب نمایید.

```
inner_div.appendChild(inner_div_text);
```

سرانجام با استفاده از کد زیر آن را به عنوان آخرین گره پس از گره me_div اضافه نمایید:

```
me_div.appendChild(inner_div);
```

درآمدی بر تکنیک های پیشرفته در DOM

در این بخش سعی می شود تکنیک ها و روش هایی برای استفاده بهتر و بیشتر از CSS در صفحات خود آموزش داده شود.

امروزه style object شامل تمام خواص و ویژگیهایی است که در css موجود می باشد اگرچه ممکن است کمی در نامگذاری این ویژگی ها تفاوت وجود داشته باشد به طور مثال background-color در CSS به صورت style.backgroundColor به کار می رود.

در جدول زیر ویژگی های موجود در CSS و معادل آنها در DOM را مشاهده می کنید.

CSS Style Attribute	JavaScript Style Property
background-color	style.backgroundColor
color	style.color
font	style.font
font-family	style.fontFamily
font-weight	style.fontWeight

مثال ۲۵-۸ :

در مثال زیر یک border با ویژگیهای زیر و با استفاده object property تعریف می کنیم :

```
var oDiv = document.getElementById("div1");
oDiv.style.border = "1px solid black";
```

مثال ۲۶-۸ :

در کد زیر با فشردن یک دکمه نام رنگ زمینه به نمایش در می آید :

```
<html>
<head>
<title>Style Example</title>
<script type="text/javascript">
function sayStyle() {
var oDiv = document.getElementById("div1");
alert(oDiv.style.backgroundColor);
}
```

```
</script>
</head>
<body>
<div id=”div1” style=”background-color: red; height: 50px;
width:
50px”></div><br />
<input type=”button” value=”Get Background Color”
onclick=”sayStyle()” />
</body>
</html>
```

مثال ۸-۲۷ :

با استفاده از کد زیر یک مربع قرمز به نمایش در می آید که با ورود نشانگر ماوس به محدوده آن به آبی تغییر رنگ می دهد و با خروج نشانگر ماوس از محدوده مربع به رنگ قبلی خود باز می گردد.

```
<html>
<head>
<title>Style Example</title>
</head>
<body>
<div id=”div1”
style=”background-color: red; height: 50px; width: 50px”
onmouseover=”this.style.backgroundColor = ‘blue’”
onmouseout=”this.style.backgroundColor = ‘red’”></div>
</body>
</html>
```

مثال ۸-۲۸ :

در مثال زیر هنگامی که روی مربع قرمز رنگ کلیک می کنید با استفاده از قطعه کد this.style.cssText ویژگیهای مربع به نمایش در می آیند.

```
<html>
<head>
<title>Style Example</title>
</head>
<body>
<div id=”div1”
style=”background-color: red; height: 50px; width: 50px”
```

```
onclick="alert(this.style.cssText)"></div>
</body>
</html>
```

DOM style methods

DOM style methods در مورد متدهایی است که عمل تبدیل متقابل را بین DOM و Css انجام می دهند .با استفاده از جدول و مثال های زیر و این مبحث را روشن تر می کنیم.

getPropertyValue(*propertyName*)	Returns the string value of the CSS property *propertyName*. The property must be specified in CSS style, such as "background-color" instead of "backgroundColor"
getPropertyPriority()	Returns the string "important" if the CSS property "!important" is specified in the rule; otherwise it returns an empty string
item(*index*)	Returns the name of the CSS property at the given *index*, such as "background-color"
removeProperty(*propertyName*)	Removes *propertyName* from the CSS definition
setProperty(*propertyName*, *value*, *priority*)	Sets the CSS property *propertyName* to *value* with the given *priority* (either "important" or an empty string)

مثال ۲۹-۸ :

```
<html>
<head>
<title>Style Example</title>
<script type="text/javascript">
function useMethods() {
var oDiv = document.getElementById("div1");
alert(oDiv.style.item(0));            //outputs "background-
color"
alert(oDiv.style.getPropertyValue("background-color"));
}
</script>
</head>
<body>
```

```
<div id=”div1” style=”background-color: red; height: 50px;
width:
50px”></div><br />
<input type=”button” value=”Use Methods”
onclick=”useMethods()” />
</body>
</html>
```

زمانی که دکمه موجود در صفحه فشرده می شود ابتدا یک پیام هشدار با مضمون background-color نمایش داده می شود و سپس با تایید آن پیام هشدار دیگری با مضمون نام رنگ زمینه ی div که در اینجا red است نمایش داده می شود.

Custom tooltips

یکی دیگر از موارد استفاده جذاب style object استفاده از tooltipهاست این ابزار در واقع توضیحاتی هستند که زمانی که نشانگر ماوس روی یکی از اشیای صفحه قرار می گیرد نمایش داده می شوند.

مثال ۳۰-۸ :

```
<html>
<head>
<title>Style Example</title>
<script type=”text/javascript”>
function showTip(oEvent) {
var oDiv = document.getElementById(“divTip1”);
oDiv.style.visibility = “visible”;
oDiv.style.left = oEvent.clientX + 5;
oDiv.style.top = oEvent.clientY + 5;
}
function hideTip(oEvent) {
var oDiv = document.getElementById(“divTip1”);
oDiv.style.visibility = “hidden”;
}
</script>
</head>
<body>
<p>Move your mouse over the red square.</p>
<div id=”div1”
style=”background-color: red; height: 50px; width: 50px”
onmouseover=”showTip(event)”
onmouseout=”hideTip(event)”></div>
<div id=”divTip1”
style=”background-color: yellow; position: absolute;
visibility:
hidden; padding: 5px”>
<span style=”font-weight: bold”>Custom Tooltip</span><br />
More details can go here.
</div>
</body>
</html>
```

با اجرا شدن کد بالا یک مربع قرمز رنگ به نمایش در می آید که زمانیکه نشانگر ماوس روی آن قرار می گیرد در قسمت پایین و سمت چپ نشانگر ماوس پیامی با پس زمینه ی زرد رنگ و مضمون More details can go here. به نمایش در می آید.

Collapsible sections

یکی دیگر از ابزاری که برای جذاب شدن صفحات وب بکار می رود استفاده از Collapsible section ها یا منوهای کرکره ای می باشد.ایده ی اصلی در این مورد این است که به کاربر این اجازه را بدهید تا قسمت هایی از صفحه را که تمایل به دیدن آنها ندارد را مخفی نماید.

مثال ۳۱-۸ :

```
<html>
<head>
<title>Style Example</title>
<script type=”text/javascript”>
function toggle(sDivId) {
var oDiv = document.getElementById(sDivId);
oDiv.style.display = (oDiv.style.display == “none”) ? “block”
:
“none”;
}
</script>
</head>
<body>
<div style=”background-color: blue; color: white; font-weight:
bold;
padding: 10px; cursor: pointer”
onclick=”toggle(‘divContent1’)”>Click Here</div>
<div style=”border: 3px solid blue; height: 100px; padding:
10px”
id=”divContent1”>This is some content
to show and hide.</div>
<p> </p>
<div style=”background-color: blue; color: white; font-weight:
bold;
padding: 10px; cursor: pointer”
onclick=”toggle(‘divContent2’)”>Click Here</div>
<div style=”border: 3px solid blue; height: 100px; padding:
10px”
id=”divContent2”>This is some content
to show and hide.</div>
</body>
</html>
```

innerText and innerHTML

innerText property برای تغییر یا اصلاح متن بین دو تگ طراحی شده است.

مثال ۳۲-۸ :

فرض کنید شما یک تگ خالی دارید که می خواهید آن را به New text for thediv. تغیر دهید با استفاده از DOM این عمل به صورت زیر انجام می گیرد :

```
oDiv.appendChild(document.createTextNode("New text for the
div."));
```

این کد زیاد مشکل نیست ولی طولانی و در مواقعی خسته کننده می باشدبا استفاده از innerText می توانید آن را به صورت زیر انجام دید :

```
oDiv.innerText = "New text for the div.";
```

مثال ۳۳-۸ :

اگر بخواهیم Hello World را با استفاده از DOM بنویسیم به صورت زیر باید بازنویسی شود :

```
var oStrong = document.createElement("strong");
oStrong.appendChild(document.createTextNode("Hello"));
var oEm = document.createElement("em");
oEm.appendChild(document.createTextNode("World"));
oDiv.appendChild(oStrong);
oDiv.appendChild(document.createTextNode(" ")); //space between
"Hello" and "World"
oDiv.appendChild(oEm);
```

که با استفاده از innerHTML به صورت زیر در می آید:

```
oDiv.innerHTML = "<strong>Hello</strong> <em>World</em>";
```

در جدول زیر و با استفاده از یک کد مشخص تفاوت innerText و innerHTML نشان داده شده است.

Code	innerText	innerHTML
Hello world	"Hello world"	"Hello world"
Hello world	"Hello world"	"Hello world"
	""	""

در واقع innerText فقط بخش های نوشتاری را بر می گرداند در حالیکه innerHTML کدهای HTML مربوط به تمام اجزا, و متن ها را برمی گرداند.

outerText and outerHTML

همانند innerText است با ایم تاوت که گره را با یک گره متنی جابه جا می کند.

مثال ۳۴-۸ :

```
oDiv.outerText = "Hello world!";
```

کد بالا در dom به صورت زیر در می آید:

```
var oText = document.createTextNode("Hello world! ");
var oDivParent = oDiv.parentNode;
oDivParent.replaceChild(oText, oDiv);
```

مثال ۳۵-۸ :

```
oDiv.outerHTML = "<p>This is a paragraph.</p>";
```

کد بالا نیز در dom به صورت زیر نوشته می شود:

```
var oP = document.createElement("p");
oP.appendChild(document.createTextNode("This is a paragraph.
"));
var oDivParent = oDiv.parentNode;
oDivParent.replaceChild(oP, oDiv);
```

در جدول زیر و با استفاده از یک کد مشخص تفاوت outerText و outerHTML نشان داده شده است.

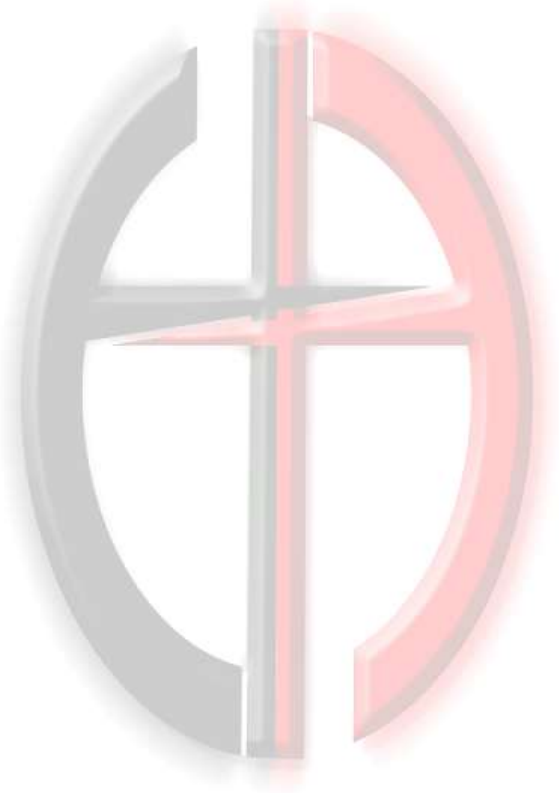
Code	**outerText**	**outerHTML**
Hello world	"Hello world"	" Hello world "
div>Hello world	"Hello world"	" Hello world "
	""	" "

مثال ۳۶-۸ :

This is my با یک گره متنی شامل زمانی که شما دکمه موجود در مثال زیر را فشار می دهید original text. جایگزین می شود

```
<html>
<head>
<title>OuterText Example</title>
<style type=”text/css”>
div.special {
background-color: red;
padding: 10px;
}
</style>
<script type=”text/javascript”>
function useOuterText() {
var oDiv = document.getElementById(“div1”);
oDiv.outerText = oDiv.outerText;
alert(document.getElementById(“div1”));
}
</script>
</head>
<body>
<div id=”div1” class=”special”>This is my original text</div>
<input type=”button” value=”Use OuterText”
onclick=”useOuterText()” />
</body>
</html>
```

فصل نهم : اشیاء پنجره



اشیای پنجره

در جاوا اسکریپت پنجره ها, فریم ها و غیره به عنوان شی در نظر گرفته می شوند. و همانطور که در فصول قبل دیدیم, برای کار با اشیا به متدها و خصوصیاتی نیاز داریم. در ین فصل به تفصیل در مورد متدها و خصوصیاتی سخن خواهیم گفت که جاوااسکریپت به طور اختصاصی برای کار با پنجره ها آماده کرده است.

خصوصیات_Properties

برای شروع کار با متدهای شی پنجره اجازه دهید تا ابتدا با خصوصیات این شی آشنا شویم.

جدول زیر شامل لیستی از این خصوصیات به همراه توضیحاتی در مورد هر کدام است.

Property	Description
closed	Holds the value based on whether or not a window has been closed
defaultStatus	Defines the default message displayed in the status bar
document	A reference to the document object of the window
frames	An array that represents all of the frames in a given window
history	Provides information on the browser history of the current window
innerHeight	Returns the height, in pixels, of the viewable area within the window
innerWidth	Returns the width, in pixels, of the viewable area within the window
length	Holds a value equal to the number of frames in a window
location	Holds the value of the current URL of the window
name	Enables a window to be named
opener	Refers to the window that opened another window
outerHeight	Returns the width, in pixels, of the entire browser window
outerWidth	Returns the width, in pixels, of the entire browser window
parent	Refers to the frame set that contains the current frame
screen.availWidth	Returns the available width of the browser window in pixels
screen.availHeight	Returns the available height of the browser window in pixels
screen.colorDepth	Returns the color depth of the screen in bits per pixel
screen.height	Returns the total height of the screen in pixels
screen.pixelDepth	Returns the bit depth of the screen in bits per pixel
screen.width	Returns the total width of the screen in pixels
self	Provides another way to reference the current window
status	Enables a message to be placed in the status bar; overrides defaultStatus
top	A reference to the top window containing a frame, frame set, or nested frame set
window	A reference to the current window

The closed Property

وظیفه این خاصیت اینست که چک می کند که آیا پنجره ای با نام مشخص بسته شده است یا خیر؟, و در صورت بسته بودن کدهای داخل آن اجرا می شوند. این خاصیت عمل مرد نظرش را بر اساس نام پنجره ها انجام می دهد.

```
if (windowname.closed) {
JavaScript Statements
}
```

innerHeight Property

این خاصیت مقدار ارتفاع محدوده ای از پنجره که کاربر مشاهده می کند را نگهداری می کند, این خاصیت شامل اسکرول بار, منوبار, تولبار و دیگر خصوصیات مرورگر نمی شود. این خصوصیت در مرورگرهای اپرا و فایرفاکس کار می کند ولی نتیجه ای را در اینترنت اکسپلورر بر نمی گرداند.

innerWidth Property

این خاصیت مقدار عرض محدوده ای از پنجره که کاربر مشاهده می کند را نگهداری می کند, این خاصیت شامل اسکرول بار, منوبار, تولبار و دیگر خصوصیات مرورگر نمی شود. این خصوصیت در مرورگرهای اپرا و فایرفاکس کار می کند ولی نتیجه ای را در اینترنت اکسپلورر بر نمی گرداند.

مثال ۱-۹ :

```
var mywin_width = 450;       // Sets a low default value
if (window.innerWidth || document.body.clientWidth) {
mywin_width = (window.innerWidth) ? window.innerWidth :
document.body.clientWidth;
}
var div_width = (mywin_width >= 800) ? "750px" : "400px";
document.write('<div style="width:'+div_width+'; background-
color:#CCC;">');
document.write('Some text for the new div element.');
document.write('</div>');
```

The length Property

این خاصیت به شما می گوید که چه تعداد فریم در پنجره موجد می باشد. درست همانند متد window.frames.length .

The location Property

این متد URL صفحه جاری را نگه می دار دشما می تونید از این ویژگی برای ایجاد دکمه لینک استفاده نمایید.

مثال ۲-۹ :

```
<head>
<title>Page has moved</title>
<script type="text/javascript">
window.location="page2.html";
</script>
</head>
<body>
Lacking JavaScript? Click the link below for the new page
then!
<br />
<a href="page2.html">New Page</a>
</body>
```

The name Property

این متد برای نگهداری نام صفحه جاری به کار می رود و به شما این امکان می دهد تا یک نام را به صفحه ای اختصاص دهید.

مثال ۳-۹ :

```
<body>
<script type="text/javascript">
window.name="cool_window";
document.write("This window is named "+window.name);
</script>
</body>
```

این مثال یک نام را به صفحه اختصاص می دهد سپس آن را در ود صفحه به نمایش می گذارد.

The opener Property

این متد برای مراجعه به پنجره ای است که پنجره ای دیگر را باز کرده است. و زمانی اتفاق می افتد که یک پنجره جدید به وسیله متد open() باز شود.

The parent Property

این متد فقط هنگامی به کار می رود که چند فریم در صفحه وجود داشته باشند و به شما این امکان را می دهد تا به parent frame set دسترسی پیدا کنید.

The self Property

این متدی راهی است برای صدا کردن صفحه جاری در جاواسکریپت. این متد مانند یک شی پنجره بکار برده می شود و می تواند از تمام خاصیت های شی پنجره جاری بهره مند گردد.

متدها

حال که شما با تعدادی از ویژگی های شی پنجره آشنا شده اید نوبت آن رسیده تا با متدهای این شی نیز آشنا شوید.

شما می توانید لیستی از این متدها را در زیر مشاهده نمایید.

Method	Description
alert()	Pops up an alert to the viewer, who must then click OK to proceed
back()	Takes the window back one item in its history list
blur()	Removes the focus from a window
clearInterval()	Cancels the action of a setInterval() method call
clearTimeout()	Cancels the action of a setTimeout() method call
close()	Closes a browser window
confirm()	Displays a confirmation dialog box to the viewer, who must then click OK or Cancel to proceed
escape()	Converts special characters in a string to hexadecimal characters
find()	Enables the viewer to launch the Find utility in the browser to find text on a page
focus()	Gives the focus to a window
forward()	Takes the window one item forward in its history list
home()	Sends the viewer to the home page the viewer has set in the Web browser settings

moveBy()	Moves a window by certain pixel values that are sent as parameters
moveTo()	Moves the top-left corner of the window to the coordinates sent as parameters
open()	Opens a new browser window
print()	Prints the contents of the window
prompt()	Pops up a prompt dialog box asking the viewer to input information
resizeBy()	Resizes a window by moving the bottom-right corner by certain pixel values that are sent as parameters
resizeTo()	Resizes an entire window to the height and width that are sent as parameters
scrollBy()	Scrolls the viewing area of a window by certain pixel values that are sent as parameters
scrollTo()	Scrolls the viewing area of the window to the specified coordinates that are sent as parameters
setInterval()	Calls a function each time a certain amount of time passes
setTimeout()	Calls a function once after a certain amount of time has passed
stop()	Stops the window from loading its content
unescape()	Converts an escaped string back to its normal characters

The alert() Method

طریقه استفاده از این متد را در فصول قبل دیده اید. این متد یک پاپ آپ به وجود می آورد که به وسیله آن پیامی را به کاربر گوشزد می کنیم.

مثال ۴-۹ :

```
window.alert("Hi there!");
```

The confirm() Method

این متد نیز یک پنجره pop up باز می کند که در آن علاوه بر پرسیدن سوالی از کارب یا دادن هشدار و یا غیره دو گزینه ok و cancel وجود دارد.

```
var varname = window.confirm("Your Message");
```

مثال ۵-۹ :

```
var is_sure = window.confirm("Are you sure?");
```

مثال ۶-۹ :

```
<body>
<a href="http://www.dlp.lxb.ir" id="myweb">My Weblog</a>
<script type="text/javascript">
var s_link = document.getElementById("myweb");
s_link.onclick = function() {
var is_sure = window.confirm("Are you sure you want to
leave?");
if (!is_sure) {
window.alert("OK. You can stay here.");
return false;
}
};
</script>
</body>
```

در مثال بالا به محض کلیک کردن روی لینک پیغامی به کاربر نمایش داده می شود مبنی بر اینکه آیا از ترک صفحه اطمینان دارید ؟

در صورت کلیک کردن روی ok از صفحه خارج می شوید ولی اگر روی cancel کلی کنید در صفحه باقی می مانید.

The find() Method

شما می توانید با بکار بردن این متد به کاربر اجاز جست و جوی لغت یا جمله ای خاص در صفحات خود را بدهید.

مثال:

در مثال زیر با کلیک کردن بر روی دکمه Click to Find Text یک پنجره جست و جو برای کاربر باز می شود که می تواند لغت مورد نظر خود را در آن تایپ نماید و آپشن های مختلف آن را انتخاب نماید.

مثال ۷-۹ :

```
<form>
```

```
<input type="button" value="Click to Find Text"
onclick="window.find();" />
</form>
```

The home() Method

این متد برای این استفاده می شود تا کاربر را به صفحه اصلی که در تنظیمات وب مرورگر_ Web browser settings_ خود تایپ کرده برگرداند. این متد در مرورگرهای اپرا و فایرفاکس کار می کند و در مرورگر اینترنت اکسپلورر بدون نتیجه.

مثال ۸-۹ :

```
<form>
<input type="button" value="Go Home!"
onclick="window.home();">
</form>
```

The print() Method

همانطور که از نام آن بر می آید کاربر را قادر می سازد برای گرفتن پرینت از یک سند .

مثال ۹-۹ :

```
<form>
<input type="button" value="Click to Print Page"
onclick="window.print();">
</form>
```

The prompt() Method

این متد نیز یک پنجره پاپ آپ ایجاد می کند که به کاربر توانایی وارد کردن اطلاعات در آن را می دهد.

این اطلاعات می توانند کارکتر, لغات یا هر اطلاعات متنی دیگری باشند.

```
var varname = window.prompt("Your Text","Default Entry");
```

varname به جای آن یک نام به پنجره پاپ آپ خود اختصاص دهید.

Your Text پیامی که مایل هستید تا کاربر مشاهده نماید را در اینجا می نویسید.

Default Entry شما را قادر می شازد تا پیامی را در مدخل ورودی و به صورت پیش فرض به کاربر نشان دهید.

مثال ۱۰-۹ :

```
<script type="text/javascript">
var thename = window.prompt("What's your name?","");
if (thename.length < 1) {
thename = "Anonymous Visitor";
}
document.write("Hello "+thename+"!");
</script>
```

در مثال بالا ابتدا از شما درخواست می شود نامی را وارد نمایید در صورت وارد کردن نام و زدن کلید اینتر نام شما در صفحه نمایش داده می شود ولی اگر چیزی را در کادر تایپ ننمایید و اینتر را فشار دهید پیام Anonymous Visitor در صفحه چاپ می شود.

مثال ۱۱-۹ :

```
<body>
<div id="greeting">
<h1>Hello! Welcome!</h1>
</div>
<div id="content">
This page talks about what I think about...
</div>
<script type="text/javascript">
var greet = document.getElementById("greeting");
var thename = window.prompt("What's your name?","");
if (thename.length < 1) {
thename = "Anonymous Visitor";
}
greet.innerHTML = "<h1>Hello " + thename + "! Welcome!</h1>";
vf
</script>
```

```
</body>
```

The open() Method

این متد شما را قادر می سازد تا به وسیله جاوااسکریپت پنجره جدیدی باز نمایید.این متد سه پارامتر دارد که در زیر مشاهده می نمایید.

```
window.open("URL","name","attribute1=value,attribute2=value");
```

URL

به جای این پارامتر آدرس صفحه یا سایت مورد نظر قرار می گیرد.

name

به جای این پارامتر نامی را که می خواهید به صفحه جدید اطلاق شود را می نویسید.

attribute1=value,attribute2=value

به جای این پارامترها مقادیر عددی و یا گزینه های ,"no", "yes" قرار می گیرد. اگر از این پارامترها استفاده ننمایید از گزینه های پیش فرض پنجره جاری استفاده می شود.

مثال ۱۲-۹ :

```
window.open("http://www.dlp.lxb.ir","my_blog");
```

مثال ۱۳-۹ :

```
window.open("http://www.dlp.lxb.ir","my_blog","width=400,heigh
t=300");
```

Standard Attributes

اگر می خواهید که از ویژگی های متد open() استفاده نمایید باید با برخی از خواص این متد آشنا شوید که به آنها خواص استاندارد می گویند نامگذاری آنها به خواص استاندارد از آن جهت اهمیت دارد که دسته ای دیگر از خواص مربوط به پنجره وجود دارند که در بخش بعد در مورد آنها صحبت خواهیم کرد. لیستی از خواص استاندارد را در جدول زیر می بینید.

Attribute Name	Possible Values	Function
width	number	Defines the width of the new window in pixels
height	number	Defines the height of the new window in pixels
directories	yes, no, 1, 0	Defines whether or not the new window has directory buttons (like the What's New or Link buttons near the top of the browser)
location	yes, no, 1, 0	Defines whether or not the new window has a location box to type in a new URL
menubar	yes, no, 1, 0	Defines whether or not the window has a menu bar (File menu, Edit menu, and so on)
resizable	yes, no, 1, 0	Defines whether or not the viewer is allowed to resize the new window
scrollbars	yes, no, 1, 0	Defines whether or not the new window has scroll bars if the contents of the window are larger than the window's size
status	yes, no, 1, 0	Defines whether or not the new window has a status bar at the bottom
toolbar	yes, no, 1, 0	Defines whether or not the new window has a toolbar (Forward and Back buttons, Stop button, and so on)

مثال ۹-۱۴ :

```
window.open("http://dlp.lxb.ir","dehloran
pc","width=300,height=200,menubar=yes");
```

مثال ۹-۱۵ :

```
window.open("http://dlp.lxb.ir","dehloran pc
","width=300,height=200,directories=yes,
location=yes,menubar=yes,resizable=yes,scrollbars=yes,status=y
es,
toolbar=yes");
```

Extended Attributes

دسته ای دیگر از خواص مربوط به متد ایجاد پنجره جدید وجود دارد که در زیر به معرفی آنها می پردازیم. این ویژگی ها که برای توسعه متد new window ایجاد شدهاند ممکن است که در همه مرورگرها به خوبی عمل نکنند به همین دلیل است که آنها را جزء خواص استاندارد قرار نداده اند.

لیستی از این خواص را در جدول زیر مشاهده می نمایید.

Attribute	Possible Values	Function
fullscreen	yes, no, 1, 0	Defines whether or not the window should open in a full screen
left	number	Defines the distance from the left of the screen for the new window
personalbar	yes, no, 1, 0	Defines whether or not the new window has a personal toolbar
screenX	number	Defines the distance from the left of the screen for the new window
screenY	number	Defines the distance from the top of the screen for the new window
top	number	Defines the distance from the top of the screen for the new window

مثال ۱۶-۹ :

```
window.open("http://dlp.lxb.ir","dehloran pc"
,"width=400,height=300,status=yes,screenX=0,left=0,screenY=0,t
op=0");
```

در مثال بالا یک پنجره با عرض ۴۰۰ پیکسل , ارتفاع ۳۰۰ پیکسل, در گوشه بالا و سمت چپ پنجره جاری ایجاد می شود.

The close() Method

این متد برای بستن پنجره ای که با جاوااسکریپت باز کرده اید به کار می رود. این متد ممکن است روی بعضی از مرورگرها کار نکند.

```
<body>
I am a new window! I am newer than that old window
that opened me, so I am special. Ha, ha!
<form>
<input type="button" value="Close Window"
onclick="window.close();" />
</form>
</body>
```

The moveBy() Method

این متد می تواند برای جا به جایی یک پنجره جدید به یک مکان از صفحه نمایشگر بکار رود.و این عمل را با استفاده از پارامترهای عددی انجام می دهد که بر حسب پیکسل به آن می دهیم. شکل کلی آن به صورت زیر است:

```
window.moveBy(x-pixels,y-pixels);
```

مثال ۱۷-۹ :

```
<body>
I am a new window! I am newer than that old window
that opened me, so I am special. Ha, ha!
<form>
<input type="button" value="Move Window"
onclick="window.moveBy(50,50);" />
<br /><br />
<input type="button" value="Close Window"
onclick="window.close();" />
</form>
</body>
```

مثال بالا شما دو دکمه می باشد یکی از آنها با نام Move Window که وظیفه جابه جایی به اندازه ۵۰ پیکسل در راستای محورهای X_Y را برعهده دارد و دیگری با نام Close Window وظیفه بستن پنجره. این متد ممکن است روی بعضی از مرورگرها کار نکند.

The moveTo() Method

این متد می تواند برای انتقال یک پنجره جدید به یک مکان از صفحه نمایشگر بکار رود.و این عمل را با استفاده از پارامترهای عددی انجام می دهد که بر حسب پیکسل به آن می دهیم. فرق این متد با متد بالا در این است که این متد تنها یک بار اجرا می شود شکل کلی آن به صورت زیر است:

```
window.moveTo(x-value,y-value);
```

برای درک بهتر به مثال زیر دقت نمایید:

مثال ۱۸-۹ :

```
<body>
I am a new window! I am newer than that old window
that opened me, so I am special. Ha, ha!
<form>
<input type="button" value="Move Window"
onclick="window.moveTo(50,50);" />
<br /><br />
<input type="button" value="Close Window"
onclick="window.close();" />
</form>
</body>
```

این متد ممکن است روی بعضی از مرورگرها کار نکند.

The resizeBy() Method

این متد برای تغییر اندازه یک پنجره بکار می رود و نحوه عملکرد و دریافت پارامتر آن همانند متد moveBy() می باشد.شکل کلی آن بصورت زیر می باشد.

```
window.resizeBy(x-value,y-value);
```

مثال ۱۹-۹ :

```
<body>
I am a new window! I am newer than that old window
that opened me, so I am special. Ha, ha!
<form>
<input type="button" value="Resize Window"
onclick="window.resizeBy(50,50);" />
<br /><br />
<input type="button" value="Close Window"
onclick="window.close();" />
</form>
</body>
```

در مثال بالا با هربار کلیک بر روی دکمه Resize Window به اندازه ۵۰ پیکسل به طول و عرض آن افزوده می شود.

The resizeTo() Method

این متد برای تغییر اندازه یک پنجره بکار می رود و نحوه عملکرد و دریافت پارامتر آن همانند متد moveTo() می باشد.شکل کلی آن بصورت زیر می باشد.

```
window.resizeTo(x-value,y-value);
```

The setInterval() Method

این متد در حقیقت کار یک حلقه را انجام می دهد که با توجه به پارامتر زمانی که بر حسب میلی ثانیه به آن میدهیم به صورت مکرر اجرا می شود تا زمانی که از متد clearInterval() استفاده نماییم.شکل کلی این متد به صورت زیر است:

```
window.setInterval("function()",time);
```

مثال ۲۰-۹ :

```
<body>
<script type="text/javascript">
function annoy_alert() {
window.alert("Am I bothering you yet?");
}
window.setInterval("annoy_alert()",10000);
</script>
</body>
```

در مثال بالا هر ده ثانیه یک بار پیام هشداری برای کاربر به نمایش در می آید.

The clearInterval() Method

از این متد برای از کار انداختن متد setInterval() استفاده می شود.شکل کلی آن به صورت زیر می باشد:

```
window.clearInterval(name);
```

شما باید به جای name نام متد یا متغییر را وارد کنید که به setInterval() نسبت داده شده است.

مثال ۲۱-۹ :

```
<body>
<script type="text/javascript">
function annoy_alert() {
window.alert("Am I bothering you yet?");
}
var madness = window.setInterval("annoy_alert()",10000);
</script>
Click the button below to end the endless barrage of
alerts.<br />
<form>
<input type="button" value="Stop the Madness!"
onclick="window.clearInterval(madness);" />
</form>
</body>
```

در مثال بالا به محض زدن دکمه Stop the Madness متد setInterval() متوقف می شود.

The setTimeout() Method

همانند متد setInterval() با این تفاوت که تنها یک بار اجرا می شود.

مثال ۹-۲۲ :

```
<body>
<script type="text/javascript">
function annoy_alert() {
window.alert("Sign my guest book NOW!");
}
var theguest = window.setTimeout("annoy_alert()",10000);
</script>
</body>
```

The clearTimeout() Method

از این متد برای از کار انداختن متد setTimeout() استفاده می شود.

مثال ۹-۲۳ :

```
<body>
<script type="text/javascript">
function annoy_alert() {
window.alert("Sign my guest book NOW!");
}
var theguest = window.setTimeout("annoy_alert()",10000);
</script>
Click the button below within 10 seconds to avoid an alert
message.<br />
<form>
<input type="button" value="No Alert for Me!"
onclick="window.clearTimeout(theguest);" />
</form>
</body>
```

فصل دهم : آرایه

آرایه

یک راه ذخیره سازی عناصر از یک نوع برای دسترسی سریع و آسان در مواقع مختلف آرایه است. در جاوا اسکریپت روش های مختلفی برای ایجاد آرایه ها و همچنین دسترسی به آنها وجود دارد. ساده ترین نوع یک آرایه , آرایه ی یک بعدی می باشد که دسترسی به عناصر آن به راحتی و با استفاده از ایندکس آنها صورت می گیرد.

نامگذاری یک آرایه

نامگذاری آرایه ها همانند نامگذاری متغیرها , اشیا و توابع می باشد و از قوانین نامگذاری گفته شده در مورد آنها نیز پیروی می کند.مانند اجتناب از آوردن اعداد در ابتدای نام آرایه یا استفاده از فضای خالی و کلمات رزرو شده.

تعریف یک آرایه

تعریف یک آرایه به سادگی تعریف هر متغیر دیگری صورت می گیرد. مانند :

```
var arrayname = new Array(element0, element1);
```

مثال ۱-۱۰:

```
var s_list = new Array("Thomas","Roger","Amber","Jennifer");
```

در مثال بالا یک آرایه به نام s_list و با چهار عنصر تعریف کرده ایم.

دسترسی به اجزای آرایه

برای دسترسی به عناصر یک آرایه یک بعدی می توان از ایندکس آنها استفاده کرد مانند:

```
var varname = arrayname[0];
```

مثال ۲-۱۰:

```
<body>
<script type="text/javascript">
var s_list = new Array("Thomas","Roger","Amber","Jennifer");
var tall_student = s_list[0];
document.write("The tallest student in class is
"+tall_student);
</script>
</body>
```

در مثال بالا و با استفاده از کد s_list[0] به خانه اول آرایه دسترسی پیدا می کنیم و سپس آن را در tall_student ذخیره می نماییم و در نهایت آن را در خروجی نمایش می دهیم.

مثال ۳-۱۰:

```
<body>
<script type="text/javascript">
var s_list = new Array("Thomas","Roger","Amber","Jennifer");
document.write("The tallest student in class is "+s_list[0]);
</script>
</body>
```

در مثال بالا بدون ذخیره کردن s_list[0] در یک متغییر خارجی آن را مستقیما در خروجی نمایش می دهیم, البته مقدار آن را که در اینجا Thomas است.

دیگر راه های تعریف آرایه

یکی دیگر از راه های تعریف آرایه این است که ابتدا فضای مورد نیاز را به آن اختصاص دهیم سپس در موقع نیاز مقادیر را به آن مربوط نماییم.

مثال ۴-۱۰:

```
var s_list = new Array(4);
```

در مثال بالا یک آرایه به نام s_list و چهار خانه برای آن ایجاد نموده ایم.

حال می توانیم به هریک از خانه های آن مقداری را منتصب نماییم و حتما لازم نیست که این اتصاب ها به ترتیب باشند.

```
s_list[2] = "Amber";
s_list[4] = "Pat";
```

راه سوم برای تعریف آرایه اینست که ابتدا فضای مناسب را به آن اختصاص دهیم و بلافاصله مقادیر را به آن اتصاص دهیم.

مثال ۵-۱۰:

```
var s_list = new Array(4);
s_list[0]="Thomas";
s_list[1]="Roger";
s_list[2]="Amber";
s_list[3]="Jennifer";
```

ایجاد یک آرایه فقط با استفاده از اختصاص یک نام و بدون در نظر گرفتن فضا برای ذخیره سازی عناصر آن نیز یکی از راه های تعریف آرایه می باشد.

مثال ۶-۱۰:

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

در این روش می توانیم بعدا و در شرایط مختلف مقادیر متفاوتی را به آن اختصاص دهیم.

مثال ۷-۱۰:

```
var s_list= new Array();
var x = 17;
var y = x+2;
var my_message = "Hi!";
s_list[0]="Thomas";
```

مثال ۸-۱۰:

```
var s_list = new Array();
var x = 17;
var y = x+2;
var my_message = "Hi!";
s_list[29]="Thomas";
```

ویژگی ها و متدهای آرایه

در جدول زیر ویژگی های مربوط به آرایه را مشاهده می نمایید.

Property	Description
constructor	Refers to the constructor function used to create an instance of an object
index	Used when an array is created by a regular expression match
input	Used when an array is created by a regular expression match
length	Contains a numeric value equal to the number of elements in an array
prototype	Allows the addition of properties and methods to objects such as the JavaScript Array object

The constructor

این خاصیت شامل تمام کدهایی است که یک آرایه را می سازند.

```
function Array() { [native code] }
```

برای دسترسی به این خاصیت شما باید از نام آرایه و سپس نقطه و بعد از آن کلمه کلیدی constructor استفاده نمایید.

مثال ۹-۱۰:

```
var s_list = new Array(4);
window.alert(s_list.constructor);
```

The length

این خاصیت تعداد اجزای آرایه را به عنوان مقدار بر می گرداند.

مثال ۱۰-۱۰:

```
var s_list = new Array(4)
s_list[0]="Thomas";
s_list[1]="Roger";
s_list[2]="Amber";
s_list[3]="Jennifer";
window.alert("The array has "+s_list.length+" elements");
```

The prototype

یکی دیگر از خاصیت های آرایه , خاصیت prototype می باشد.این خاصیت به شما اجازه می دهد تا ویژگی و متدهای مختلفی را به اجزای موجود در صفحه از جمله آرایه ها اضافه نمایید.

```
Array.prototype.new_property=default_value;
```

مثال ۱۱-۱۰:

```
Array.prototype.attitude = "cool";
var s_list = new Array();
window.alert("This place is "+s_list.attitude);
```

مثال ۱۲-۱۰:

```
Array.prototype.attitude = "cool";
var s_list = new Array();
window.alert("This place is "+s_list.attitude);
```

مثال ۱۳-۱۰:

```
var fish = new Array();
fish.attitude = "wide-eyed";
window.alert("Fish are "+fish.attitude);
```

متدهای آرایه

حالا که تا حدودی با ویژگی ها و خاصیت های آرایه آشنا شدید وقت آن رسیده تا متدهای آرایه ها و موارد استفاده آن را مورد بررسی قرار دهیم.

Method	Description
concat()	Combines the elements of two or more arrays into one new array
join()	Combines the elements of an array into a single string with a separator character
pop()	Removes the last element from an array and then returns the removed element if needed
push()	Adds elements to the end of an array and then returns the numeric value of the new length of the array if needed
reverse()	Reverses the direction of the elements in an array: the first ele
shift()	Removes the first element from an array and then returns that element if needed
unshift()	Adds elements to the beginning of an array and returns the numeric value of the length of the new array if needed
slice()	Pulls out a specified section of an array and returns the section as a new array
splice()	Removes elements from an array or replaces elements in an array
sort()	Sorts the elements of an array into alphabetical order based on the string values of the elements
toString()	Combines the elements of an array into a single string with a comma as a separator character

The concat() Method

این متد برای چسباندن دو یا چند آرایه به کار می رود.

مثال ۱۴-۱۰:

```
var fruits = new Array("oranges","apples");
var veggies = new Array("corn","peas");
var fruits_n_veggies = fruits.concat(veggies);
```

برای اینکه بخواهید اجزای یک آرایه در ابتدا آورده شوند باید نام آن در ابتدا نوشته شود.

مثال ۱۵-۱۰:

```
var fruits = new Array("oranges","apples");
var veggies = new Array("corn","peas");
var fruits_n_veggies = veggies.concat(fruits);
```

در مثال بالا ابتدا اجزای آرایه ی veggies آورده می شوند.

برای اتصال سه آرایه هم به روش زیر عمل می کنیم .

مثال ۱۶-۱۰:

```
var fruits = new Array("oranges","apples");
var veggies = new Array("corn","peas");
var meats = new Array("fish","chicken");
var three_groups = fruits.concat(veggies,meats);
```

نتیجه مثال بالا oranges, apples, corn, peas, fish, chicken می باشد.

مثال ۱۷-۱۰:

```
var fruits = new Array("oranges","apples");
var veggies = new Array("corn","peas");
var meats = new Array("fish","chicken");
var three_groups = meats.concat(veggies,fruits);
```

در مثال بالا نتیجه fish, chicken, corn, peas, oranges, apples می شود.

The join() Method

این متد برای ترکیب اجزای یک آرایه در یک رشته به کار می رود. همچنین به وسیله این متد می توانیم جداکننده های بین اجزای آرایه را هم تغییر دهیم.

مثال ۱۸-۱۰:

```
<body>
<script type="text/javascript">
var fruits = new Array("oranges","apples","pears");
var fruit_string = fruits.join();
document.write("The new string is "+fruit_string);
</script>
</body>
```

در مثال بالا آرایه ی fruits با رشته The new string is ترکیب می شود.

مثال ۱۹-۱۰:

```
<body>
<script type="text/javascript">
var fruits = new Array("oranges","apples","pears");
var fruit_string = fruits.join(":");
document.write("The new string is "+fruit_string);
</script>
</body>
```

در مثال بالا جداکننده های اجزای آرایه به ":" تغییر پیدا می کند.

The pop() Method

این متد برای حذف کردن آخرین جزء یک آرایه به کار می رود.

مثال ۲۰-۱۰:

```
var fruits = new Array("oranges","apples","pears");
fruits.pop();
```

در این مثال pears از آرایه ی fruits حذف می شود.

مثال ۲۱-۱۰:

```
var fruits = new Array("oranges","apples","pears");
var picked_fruit = fruits.pop();
window.alert("You picked my "+picked_fruit);
```

The push() Method

این متد برای اضافه کردن یک جزء به انتهای یک آرایه بکار می رود.

مثال ۲۲-۱۰:

```
var fruits = new Array("oranges","apples");
fruits.push("pears");
```

مثال ۲۳-۱۰:

```
var fruits = new Array("oranges","apples");
fruits.push("pears","grapes");
```

مثال ۲۴-۱۰:

```
var fruits = new Array("oranges","apples");
var who_knows = fruits.push("pears","grapes");
```

```
window.alert("The method returned "+who_knows);
```

The reverse() Method

این متد ترتیب اجزای یک آرایه را معکوس می کند.

مثال ۱۰-۲۵:

```
var fruits = new Array("oranges","apples","pears");
fruits.reverse();
```

The shift() Method

این متد همانند متد pop عمل می کند با این تفاوت که عنصر ابتدایی آرایه را حذف می کند.

مثال ۱۰-۲۶:

```
var fruits = new Array("oranges","apples","pears");
fruits.shift();
```

مثال ۱۰-۲۷:

```
var fruits = new Array("oranges","apples","pears");
var picked_fruit=fruits.shift();
window.alert("You picked my "+picked_fruit);
```

نتیجه

```
You picked my orang
```

The slice() Method

این متد هنگامی مفید است که بخواهیم قسمتی از یک آرایه را جدا کرده و به عنوان آرایه ای دیگر مجزا از آرایه اول ذخیره نماییم.

```
arrayname.slice(start,stop)
```

مثال ۲۸-۱۰:

```
var fruits = new Array("oranges","apples","pears","grapes");
var somefruits = fruits.slice(1,3);
```

نتیجه

```
apples , pears
```

The splice() Method

این متد به شما اجازه حذف یا جایگزینی عناصر یک آرایه را می دهد. پارامترهایی که می توانید به این متد ارسال کنید عبارتند از شماره ایندکس عنصری که می خواهید عملیات از آنجا شروع شود, تعداد اجزایی که می خواهید حذف شوند و در نهایت ویژگی های عنصر جدیدی که می خواهید به آرایه اضافه نمایید.

مثال ۲۹-۱۰:

```
var fruits = new Array("oranges","apples","pears","grapes");
var somefruits = fruits.splice(2,1);
```

در مثال بالا splice کردن از از عنصر سوم "pears" شروع شده و در همانجا هم خاتمه می یابد و در نتیجه باعث حذف آن می شود.

نکته : ایندکس اجزای آرایه از صفر شروع می شود.

مثال ۳۰-۱۰:

```
var fruits = new Array("oranges","apples","pears","grapes");
```

```
var somefruits = fruits.splice(2,2);
```

همانند مثل بالا از عنصر سوم عملیات شروع می شود با این تفاوت که علاوه بر عنصر سوم"pears" عنصر چهارم یعنی "grapes" نیز حذف می شود.

مثال ۳۱-۱۰:

```
var fruits = new Array("oranges","apples","pears","grapes");
var somefruits = fruits.splice(2,1,"watermelons");
```

در مرحله اول عنصر pears از آرایه حذف می شود.سپس در مرحله بعد عنصر watermelons به جای آن و در همان مکان جایگزین می گردد.

مثال ۳۲-۱۰:

```
var fruits = new Array("oranges","apples","pears","grapes");
var somefruits = fruits.splice(2,0,"watermelons","plums");
alert(somefruits);
document.write(fruits);
```

هیچ عنصری از آرایه حذف نمی شود بلکه عناصر watermelons و plums نیز به آن اضافه می گردند.

The sort() Method

برای مرتب کردن اجزای یک آرایه بکار می رود.

مثال ۳۳-۱۰:

```
var fruits = new Array("oranges","apples","pears","grapes");
fruits.sort();
```

The toString Method

تبدیل اجزای آرایه به یک رشته

مثال ۳۴-۱۰:

```
var fruits = new Array("oranges","apples","pears","grapes");
var fruit_list = fruits.toString();
document.write(fruit_list);
```

متدهای بیشتر

در جاوااسکریپت ۱.۶ و بالاتر یک سری متد وجود دارد که در جدول زیر مشاهده می نمایید. در هر صورت این متدها ممکن است در تمام مرورگرها به درستی کار نکنند و یا نتایج یکسانی نداشته باشند در نتیجه در استفاده از آنها با احتیاط رفتار کنید.

Method	Description
filter()	Returns a new array containing elements from an array that returned true based on the function used to filter it
forEach()	Calls a specified function for each element in the array
every()	Returns true if all elements in the array return true for the specified function used to test them
indexOf()	Returns the lowest index number for an element that has a value equal to the specified value sent as a parameter
lastIndexOf()	Returns the highest index number for an element that has a value equal to the specified value sent as a parameter
map()	Returns a new array that results from calling a specified function on every element in the array
reduce()	Runs a function on two values in the array at a time, from left to right, until only a single value is left
reduceRight()	Runs a function on two values in the array at a time, from right to left, until only a single value is left
some()	Returns true if one or more elements return true for the specified function used to test them

ساخت اجزای آرایه با استفاده از حلقه ها

استفاده از حلقه ها برای ساخت آرایه زمانی مفید می شود که مثلا شما بخواهید داده هایی را به طور پیاپی از کاربر سوال کنید.

مثال ۳۵-۱۰:

```
var s_list = new Array(4);
for(var count=0;count<4;count++) {
s_list[count] = window.prompt("Enter a name","");
}
document.write(s_list);
```

در این مثال برنامه از کاربر درخواست وارد کردن یک اسم را می دهد و این عمل چهار با تکرار می گردد و در نهایت اسامی وارد شده در آرایه s_list ذخیره می شوند.

مثال ۳۶-۱۰:

برنامه برای شناسایی و نمایش اعداد زوج بزرگ تر از صفر و کوچکترمساوی بیست.

```
var even_nums = new Array(10);
var a_count = 0;
for (var count=0;count<20;count+=2) {
even_nums[a_count] = count+2;
a_count ++;
}
document.write(even_nums);
```

مثال ۳۷-۱۰:

نمایش اسم دانش آموزان بصورت ستونی

```
<body>
<h1>Student Names</h1>
<script type="text/javascript" >
```

```
var s_list = new Array(4);
s_list[0]="Thomas";
s_list[1]="Roger";
s_list[2]="Amber";
s_list[3]="Jennifer";
for(var count=0;count<4;count++) {
document.write(s_list[count]+ "<br />");
}
</script>
</body
```

در برنامه بالا می توانیم از قطعه کد زیر برای جایگزینی حلقه for استفاده نماییم.

```
for(var count=0;count<s_list.length;count++)
```

مثال ۳۸-۱۰:

مرتب کردن اسم دانش آموزان

```
<script type="text/javascript" >
var s_list = new Array();
s_list[0]="Thomas";
s_list[1]="Roger";
s_list[2]="Amber";
s_list[3]="Jennifer";
s_list[4]="Pat";
s_list[5]="Kelly";
s_list[6]="Jerry";
s_list.sort();
for(count=0;count<s_list.length;count++) {
document.write(s_list[count]+ "<br />");
}
</script>
```

مثال ۳۹-۱۰:

```
var s_list = new Array();
s_list[0]="Thomas";
s_list[1]="Roger";
s_list[2]="Amber";
```

```
s_list[3]="Jennifer";
for (student in s_list) {
document.write(s_list[student] + "<br />");
}
```

بازنویسی کد بالا به وسیله for each

```
var s_list = new Array();
s_list[0]="Thomas";
s_list[1]="Roger";
s_list[2]="Amber";
s_list[3]="Jennifer";
for each (student in s_list) {
document.write(student + "<br />");
}
```

آرایه های شرکت پذیر(Associative Arrays)

تعریف

به عنوان آخرین مبحث در مورد آرایه, آرایه های شرکت پذیر یا انجمنی را معرفی می کنیم. این آرایه ها دارای این خاصیت می باشند که می توان به هر یک از اجزای آن نام و مقدار مجزایی را اختصاص داد.

مثال ۴۰-۱۰:

```
var s_list= new Array();
s_list["tall"]="Thomas";
s_list["cool"]="Roger";
s_list["clever"]="Amber";
s_list["attentive"]="Jennifer";
```

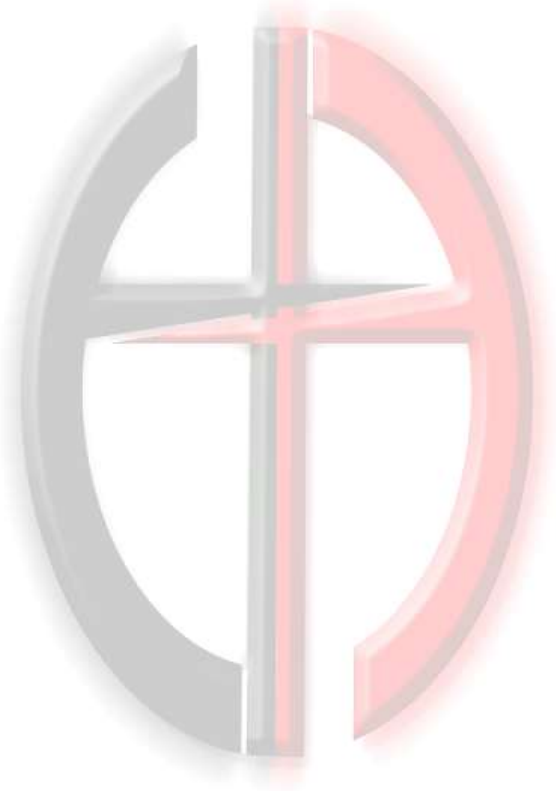
دسترسی به آرایه های انجمنی

دسترسی به اجزای این آرایه ها هم مانند سایر راههای دسترسی به آرایه های معمولی می باشد.

مثال ۴۱-۱۰:

```
<body>
<h1>Student Names</h1>
<script type="text/javascript">
var s_list= new Array();
s_list["tall"]="Thomas";
s_list["cool"]="Roger";
s_list["clever"]="Amber";
s_list["attentive"]="Jennifer";
document.write("The tall one is " +s_list["tall"]+ "<br />");
document.write("The cool one is " +s_list["cool"]+ "<br />");
document.write("The clever one is " +s_list["clever"]+ "<br
/>");
document.write("The attentive one is " +s_list["attentive"]+
"<br />");
</script>
</body>
```

فصل یازدهم :اشیاء ریاضی



اشیا ریاضی

در جاوااسکریپت اشیا و توابع از پیش ساخته ای تدارک دیده شده است که می توانند شما را در محاسبات خود یاری نمایند.

Properties

اشیا ریاضی ویژگی ها و خاصیت های زیادی را در اختیار شما قرار می دهند که لیستی از آنها را در زیر مشاهده می نمایید.

Property	Value
E	Value of Euler's constant (E), which is about 2.71828…
LN10	Value of the natural logarithm of 10, which is about 2.302585…
LN2	Value of the natural logarithm of 2, which is about 0.693147…
LOG10E	Value of the base 10 logarithm of E, which is about 0.43429…
LOG2E	Value of the base 2 logarithm of E, which is about 1.442695…
PI	Value of pi, often used with circles, which is about 3.14159…
SQRT2	Value of the square root of 2, which is about 1.4142…
SQRT1_2	Value of the square root of one half, which is about 0.7071…

استفاده از ویژگی ها

با استفاده از ویژگی های گفته شده در جدول بالا می خواهیم اسکریپتی را برنامه نویسی کنیم که بعد از وارد کردن شعاع یک دایره محیط آن را محاسبه کند.

مثال ۱-۱۱:

```
<body>
  <form>
    To find the area of a circle, input a radius:<br />
    <input type="text" id="radius" />
    <input type="button" value="Get the Area!" id="getarea" />
  </form>
  <script type="text/javascript">
    var area_button = document.getElementById("getarea");
    area_button.onclick = function() {
    var rad = document.getElementById("radius").value;
    if (rad.length < 1) {
```

```
    window.alert("Please enter a radius!");
    return false;
    }
    else if (rad != (rad*1)) {
    window.alert("Radius must be numeric!");
    return false;
    }
    else {
    var the_area = Math.PI * (rad * rad);
    window.alert("The area is " + the_area + " square units.
");
    return false;
    }
    };
</script>
</body>
```

در برنامه بالا ابتدا کادری باز می شود که از شما درخواست یک عدد می نماید و در صورت وارد کردن آن در کادر دیگری محیط آن به نمایش در می آید. قابل ذکر است که در صورت وارد نشدن عدد صحیح پیام متناسب با آن نمایش داده می شود.

Methods

متدهای موجود در شی ریاضی به شما کمک می کنند تا محاسباتی را که قبلا قادر به انجام آنها نبودید را در اسکریپت های خود انجام دهید.

در جدول زیر لیستی از این متدها به همراه توضیحاتی در مورد هریک از آنها را مشاهده می نمایید.

Method	Purpose
abs()	Returns the absolute value of the number sent as a parameter
acos()	Returns the arccosine of the number sent as a parameter, in radians
asin()	Returns the arcsine of the number sent as a parameter, in radians
atan()	Returns the arctangent of the number sent as a parameter, in radians
atan2()	Returns the arctangent of the quotient of two numbers sent as parameters, in radians
ceil()	Returns the smallest integer greater than or equal to the number sent as a parameter
cos()	Returns the cosine of the number sent as a parameter, in radians
exp()	Returns the value of E to the power of the number sent to the method as a parameter
floor()	Returns the largest integer less than or equal to the number sent as a parameter
log()	Returns the natural logarithm of the number sent as a parameter

max()	Returns the larger of the two numbers that are sent as parameters
min()	Returns the smaller of the two numbers that are sent as parameters
pow()	Returns the numeric value of the first parameter raised to the power of the second parameter
random()	Returns a random number between 0 and 1; does not require a parameter
round()	Returns the value of the number sent as a parameter rounded to the nearest integer
sin()	Returns the sine of the number sent as a parameter, in radians
sqrt()	Returns the square root of the number sent as a parameter
tan()	Returns the tangent of the number sent as a parameter, in radians

متدهای اصلی

منظور از متدهای اصلی در این کتاب, متدهایی هستند که که تنها با وارد کردن یک عدد و بدون نیاز به محاسبه خاصی برای برنامه نویس نتیجه را برمی گردانند.لیستی از این متدها را در زیر مشاهده می کنید.

- abs()
- acos()
- asin()
- atan()
- cos()
- exp()
- log()
- sin()
- sqrt()
- tan()

مثال ۲-۱۱:

اسکریپتی کاربرپسند برای محاسبه مجذور اعداد با استفاده از تابع sqrt()

```
<body>
  <form>
    Enter a (positive) number or zero: <br />
    <input type="text" id="sr_num" />
    <input type="button" value="Get a Square Root" id="getroot" />
  </form>
  <script type="text/javascript">
    var root_button = document.getElementById("getroot");
    root_button.onclick = function() {
    var thenum = document.getElementById("sr_num").value
    if (thenum < 0) {
    window.alert("Hey! I said to enter a positive number! Try
again.");
    return false;
```

```
    }
    else if (thenum != (thenum*1)) {
    window.alert("Input must be numeric!");
    return false;
    }
    else {
    var theroot = Math.sqrt(thenum);
    window.alert("The square root of "+thenum+" is "+theroot);
    return false;
    }
    };
  </script>
</body
```

متدهای دو پارامتری

این مبحث در مورد متدهایی است که برای صحیح اجرا شدن نیاز به دریافت دو پارامتر دارند, که از جمله این متدها می توان به متدهای مقایسه ای بزرگ تر و کوچک تر و یا توان اشاره کرد.

- atan2()
- max()
- min()
- pow()

مثال ۳-۱۱:

اسکریپتی برای مقایسه دو عدد و برگرداندن عدد بزرگ تر

```
<body>
  <form>
    <input type="button" value="Which Number is Bigger?" id="getmax"
/>
  </form>
  <script type="text/javascript">
    var max_button = document.getElementById("getmax");
    max_button.onclick = function() {
    var num1 = window.prompt("Enter a number.","");
    var num2 = window.prompt("Enter another number","");
    var largenum = Math.max(num1,num2);
    var smallnum = Math.min(num1,num2);
    if (largenum == smallnum) {
    window.alert("Those two numbers are equal!");
```

```
    }
    else {
    window.alert(largenum+" is larger than "+smallnum);
    }
    };
  </script>
</body>
```

اساس کار تابع بالا استفاده از متدهای max() و min() و می باشد و همچنین برای کاربر پسند شدن ظاهر برنامه از کادرهای prompt جهت دریافت اعداد استفاده کرده ایم که شما می توانید به جای آنها از کادرهای متنی استفاده نمایید.

توان pow()

این متد دو مقدار را به عنوان پارامتر می گیرد و سپس مقدار اول را به توان مقدار دوم می رساند.
مانند : Math.pow(2,3);

مثال ۴-۱۱:

```
<body>
  <form>
    <input type="button" value="Find a Power" id="getpow" />
  </form>
  <script type="text/javascript">
    var pow_button = document.getElementById("getpow");
    pow_button.onclick = function() {
    var num1 = window.prompt("Enter a base number.","");
    var num2 = window.prompt("What power should we set it to (a
    number)?","");
    var theresult = Math.pow(num1,num2);
    window.alert(num1+" to the power of "+num2+" is "+theresult);
    };
  </script>
</body>
```

دیگر متدها

- ceil()
- floor()
- round()

The ceil() Method

این متد بزرگ ترین عدد صحیح نزدیک به عدد مربوطه را نشان می دهد. به طور مثال اگر شما عدد ۱۰.۱_ده ممیز یک دهم_ را وارد نمایید این تابه ۱۱ را برمی گرداند.

مثال ۵-۱۱:

```
<body>
  <form>
    <input type="button" value="Find a Ceiling" id="getceiling" />
  </form>
  <script type="text/javascript>
    var ceil_button = document.getElementById("getceiling");
    ceil button.onclick = function() {
    var num1= window.prompt("Enter a number.","");
    var theceil= Math.ceil(num1);
    window.alert("The ceiling of "+num1+ " is "+theceil);
    };
  </script>
</body>
```

The floor() Method

این متد کوچک ترین عدد صحیح نزدیک به عدد مربوطه را نشان می دهد. به طور مثال اگر شما عدد ۱۰.۱_ده ممیز یک دهم_ را وارد نمایید این تابه ۱۰ را برمی گرداند.

The round() Method

این متد وظیفه گردکردن اعداد اعشاری به نزدیکترین عدد صحیح به آن ها را دارد. در این تابع اعدادی که مقدار اعشار آن ها کوچکتر از ۵._نیم_باشد با کوچک ترین عدد صحیح نزدیک به آنها و اعدادی که مقدار اعشاری آنها بزرگ تر یا مساوی ۵._نیم_ باشد با بزرگ ترین عدد صحیح نزدیک به آنها جمع می شوند.

مثال ۶-۱۱:

10.2 ➔ 10
10.5 ➔ 11
10.6 ➔ 11

The random() Method

یکی از مفیدترین متدهای شی ریاضی متد راندوم می باشد.چرا که همواره یک عدد اتفاقی بین ۰ و ۱ را برمیگرداندو برای مواقعی که بخواهیم متن عکس یا هرچیز دبگری را بصورت اتفاقی در وب خود انتشار دهیم می تواند بکار رود.

مثال ۷-۱۱:

در مثال زیر با هربار رفرش شدن صفحه یک سوال اتفاقی از کاربر پرسیده می شود.

```
<body>
<h1>My Random Quote for You:</h1>
  <div id="my_quote">
    Look in the mirror. Are you looking at me?
  </div>
  <script type="text/javascript" >
    var quotes= new Array(10);
    quotes[0]="Look in the mirror. Are you looking at me?";
    quotes[1]="It is time for a rhyme, I guess.";
    quotes[2]="Where is my JavaScript book?";
    quotes[3]="If I had a buck for every dollar I spent--Oops, never
    mind.";
    quotes[4]="I suppose you were expecting a real quote here.";
    quotes[5]="Quotes are great, but don't quote me on that.";
    quotes[6]="What should I write here?";
    quotes[7]="Wut hapns iff eye miss spel ohn purpas?";
    quotes[8]="Mark my words, I will mark my words.";
    quotes[9]="This spot reserved for a better quote.";
    var q_div = document.getElementById("my_quote");
    var rand_int = Math.floor(Math.random()*10);
    q_div.innerHTML = quotes[rand_int];
  </script>
</body>
```

اشیای عددی

اشیای عددی در جاوااسکریپت شامل چندین ویژگی و متد مفید برای دستکاری اعداد می باشند در این بخش سعی می کنیم تعدادی از آنها را مورد بررسی قرار دهیم.

ویژگی ها

در جدول زیر لیستی از ویژگی های مربوط به اشیا عددی را مشاهده می کنید.

Property	Purpose
constructor	Holds the value of the constructor function that created the object
MAX_VALUE	Holds a constant number value, representing the largest value before JavaScript interprets a number as infinity
MIN_VALUE	Holds a constant number value, representing the smallest value before JavaScript interprets a number as negative infinity
NaN	Represents the value of “Not a Number”
NEGATIVE_INFINITY	Represents the value of negative infinity
POSITIVE_INFINITY	Represents the value of infinity
prototype	Enables you to add properties to the object if you wish

متدها

در جدول زیر لیستی از این متدها را مشاهده می نمایید.

Method	Purpose
toExponential()	Returns a string value that represents the number in exponential notation
toFixed()	Returns a string value that represents the number rounded to the specified number of digits after the decimal
toPrecision()	Returns a string value that represents the number rounded to the specified number of significant digits
toSource()	Returns a string value that represents the source code of the object
toString()	Returns a string value for a Number object
valueOf()	Used by JavaScript internally most often

The toExponential() Method

این متد عدد دریافتی را به صورت نماد علمی نشان می دهد.

مثال ۸-۱۱:

```
var the_num = 100;
document.write(the_num.toExponential());
```

نتیجه

```
1e+2
```

The toFixed() Method

این متد برای حذف تعداد ارقام اعشار بعد از ممیز به کار می رود.در واقع به اندازه پارامتری که به آن می دهیم از بعد از ممیز و به سمت راست شروع به شمارمی نماید و به اندازه پارامتر جلو می رود, سپس اعدادی که شمارش نمده را برمی گرداند و بقیه را حذف می نماید.

مثال ۹-۱۱:

در مثال زیر تقریب حذف اعداد اعشاری را ۲ در نظر گرفته ایم بدین معنا که تا دو رقم از اعداد اعشاری را حفظ نماید و بقیه را حذف کند.

```
var mymoney = 2000;
var mykids = 7;
var one_share = mymoney/mykids;
document.write("One share of my money is $"+
one_share.toFixed(2));
```

نتیجه:

```
One share of my money is $285.71
```

در صورت استفاده نکردن از متد toFixed() در مثال بالا نتیجه به صورت زیر می شود:

```
One share of my money is $285.7142857142857
```

The toPrecision() Method

این متد نیز یکی دیگر از متدهای حذف ارقام اعشاری می باشد و مانند متد toFixed() با این تفاوت که شمارش ارقام را از سمت چپ ترین عدد شروع می نماید و به اندازه پارامتر داده شده به سمت راست حرکت می کند و بقیه اعشار باقیمانده را با آخرین عدد اعشاری گرد می کند.

مثال ۱۰-۱۱:

```
var the_num = 45.57689349;
document.write(the_num.toPrecision(5));
```

نتیجه

```
45.57
```

The toString() Method

همانطور که از نام آن پیدا است برای تبدیل اعداد به رشته های متنی به کار می رود.

اشیای زمان

این اشیا نیز یکی دیگر از اشیای از پیش تعریف شده در جاوااسکریپت هستند که به شما توانایی دادن/گرفتن یک زمان مشخص به/از اسکریپت خود می نمایند.

ویژگی ها

در جدول زیر این ویژگی ها را مشاهده می نمایید

Property	Purpose
constructor	Holds the value of the constructor function that created the object
prototype	Enables you to add properties to the object if you wish

The constructor Property

به مبحثی با همین نام در آرایه ها مراجعه نمایید.

The prototype Property

به شما این قابلیت را می دهد که ویژگی یا متد جدیدی را به شی Date اضافه کنید.

مثال ۱۱-۱۱:

```
Date.prototype.morning="a.m.";
var rightnow= new Date();
window.alert("This date is "+rightnow.morning);
```

متدها

شی زمان نمی تواند قابلیت های زیادی را در اختیار شما قرار دهد اما این شی به وسیله تعداد زیادی متد پشتیبانی می شود که این متدها ویژگی ها و قابلیتهای زیادی را در اختیار شما قرار می دهند.

لیستی از این متدها را در جدول زیر مشاهده می نمایید.

Method	Purpose
getDate()	Returns the day of the month based on the viewer's local time
getDay()	Returns the number of days into the week based on the viewer's local time (0–6)
getHours()	Returns the number of hours into the day based on the viewer's local time (0–23)
getMilliseconds()	Returns the number of milliseconds into the second based on the viewer's local time (0-999)
getMinutes()	Returns the number of minutes into the hour based on the viewer's local time (0–59)
getMonth()	Returns the number of months into the year based on the viewer's local time (0–11)
getSeconds()	Returns the seconds into the minute based on the viewer's local time (0–59)
getTime()	Returns the number of milliseconds since 1/1/1970 for the Date object
getTimezoneOffset()	Returns the time-zone offset (from Greenwich Mean Time) in minutes based on the viewer's local time zone
getYear()	Returns the year based on the viewer's local time (two digits)
getFullYear()	Returns the full year based on the viewer's local time (four digits)
getUTCDate()	Returns the day of the month in Coordinated Universal Time
getUTCDay()	Returns the number of days into the week in Coordinated Universal Time (0-6)
getUTCFullYear()	Returns the full year in Coordinated Universal Time (four digits)
getUTCHours()	Returns the number of hours into the day in Coordinated Universal Time (0–23)
getUTCMilliseconds()	Returns the number of milliseconds into the current second in Coordinated Universal Time (0-999)
getUTCMinutes()	Returns the number of minutes into the hours in Coordinated Universal Time (0–59)

getUTCMonth()	Returns the number of months into the current year in Coordinated Universal Time (0–11)
getUTCSeconds()	Returns the number of seconds into the current minute in Coordinated Universal Time (0-59)
parse()	Returns the number of milliseconds since 1/1/1970 of a date sent as a parameter based on the viewer's local time
setDate()	Sets the day of the month for an instance of the Date object
setHours()	Sets the hours for an instance of the Date object
setMilliseconds()	Sets the milliseconds for an instance of the Date object
setMinutes()	Sets the minutes for an instance of the Date object
setMonth()	Sets the month for an instance of the Date object
setSeconds()	Sets the seconds for an instance of the Date object
setTime()	Sets the time (in milliseconds since January 1, 1970, at midnight) for an instance of the Date object
setYear()	Sets the year for an instance of the Date object (two digits)
setFullYear()	Sets the full year for an instance of the Date object (four digits)
setUTCDate()	Sets the day of the month in Coordinated Universal Time
setUTCFullYear()	Sets the full year in Coordinated Universal Time (four digits)
setUTCHours()	Sets the number of hours into the day in Coordinated Universal Time (0–23)
setUTCMilliseconds()	Sets the number of milliseconds into the current second in Coordinated Universal Time (0-999)
setUTCMinutes()	Sets the number of minutes into the hours in Coordinated Universal Time (0–59)
setUTCMonth()	Sets the number of months into the current year in Coordinated Universal Time (0–11)
setUTCSeconds()	Sets the number of seconds into the current minute in Coordinated Universal Time (0-59)
toDateString()	Returns the date portion of the Date object as a string in American English
toGMTString()	Returns a string that is the date in Greenwich Mean Time (GMT) format (toUTCString() is now used instead)
toLocaleString()	Returns a string that is the date in a format based on the locale
toLocaleDateString()	Returns the date portion of the Date object as a string based on the locale
toLocaleTimeString()	Returns the time portion of the Date object as a string based on the locale
toString()	Returns a string that is the date in American English
toTimeString()	Returns the time portion of the Date object as a string in American English

متدهای برگرداننده ی مقدار

همانطور که از نام این متدها پیداست این متدها مقداری را برمی گردانند که شما را قادر می سازند تا زمان دلخواه یا جاری را در اسکریپت خود بکار ببرید. نام این متدها را در زیر مشاهده می نمایید.

- getDate()
- getDay()
- getHours()
- getMilliseconds()
- getMinutes()
- getMonth()
- getSeconds()
- getTime()
- getTimezoneOffset()
- getYear()
- getFullYear()
- getUTCDate()
- getUTCDay()
- getUTCFullYear()
- getUTCHours()
- getUTCMilliseconds()
- getUTCMinutes()
- getUTCMonth()
- getUTCSeconds()

برای استفاده از این کتدها شما باید در ابتدا یک شی زمانی ایجاد کنید مانند:

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

The getDate() Method

این متد شماره روز جاری براساس تقویم سیستم شما را برمی گرداند. مثلا اگر امروز پنجمین روز ماه باشد این متد مقدار عددی ۵ را برمی گرداند.

مثال ۱۲-۱۱:

```
var rightnow= new Date();
var theday= rightnow.getDate();
document.write(theday);
```

The getDay() Method

این متد همانند متد بالا عمل می کند با این تفاوت که به جای شماره اسم روز جاری را برمی گرداند.مانند friday.

مثال ۱۳-۱۱:

```
var rightnow= new Date();
var theday= rightnow.getDay();
document.write(theday);
```

The getHours() Method

برگرداندن ساعت روز جاری به صورت بیست و چهار ساعته.

`range 0-23`

مثال ۱۴-۱۱:

```
var rightnow= new Date();
var theday= rightnow.getHours();
document.write(theday);
```

The getMilliseconds() Method

این متد میلی سکند ذخیره شده در شی زمان را برمی گرداند.

`range 0-999`

مثال ۱۵-۱۱:

```
var rightnow= new Date();
var theday= rightnow.getMilliseconds();
document.write(theday);
```

The getMinutes() Method

برگرداندن دقیقه بر اساس سیستم شما .

`range 0-59`

مثال ۱۶-۱۱:

```
var rightnow= new Date();
var theday= rightnow.getMinutes();
document.write(theday);
```

The getMonth() Method

برگرداندن نام ماه جاری بر اساس سیستم شما.

مانند : january

مثال ۱۷-۱۱:

```
var rightnow= new Date();
var theday= rightnow.getMonth();
document.write(theday);
```

The getSeconds() Method

برگرداندن ثانیه بر اساس سیستم شما .

`range 0-59`

مثال ۱۸-۱۱:

```
var rightnow= new Date();
var theday= rightnow.getSeconds();
document.write(theday);
```

The getTime() Method

برگرداندن تعداد میلی ثانیه ها از تاریخ ۱ ژانویه ۱۹۷۰ تا تاریخ موجود در شی زمان.

مثال ۱۹-۱۱:

```
var rightnow= new Date();
var theday= rightnow.getTime();
document.write(theday);
```

The getTimezoneOffset() Method

تعداد دقایقی که شما با منطقه زمانی GMT اختلاف دارید را برمی گرداند.مثلا اختلاف زمانی ما با GMT ۷ ساعت است که در صورت استفاده کردن از این تابع ۴۲۰ را بر می گرداند.

مثال ۲۰-۱۱:

```
var rightnow= new Date();
var theday= rightnow.getTimezoneOffest();
document.write(theday);
```

The getFullYear() Method

برگرداندن سال میلادی بر اساس تاریخ سیستم کاربر.

مثال ۲۱-۱۱:

```

```

```
var rightnow= new Date();
var theyear= rightnow.getFullYear();
document.write(theday);
```

متدهای نشاننده ی مقدار

این متدها برای نشاندن یک مقدار زمانی در اسکریپت شما بکار می روند.لیست این متدها را در زیر مشاهده می کنید.

- setDate()
- setHours()
- setMilliseconds
- setMinutes()
- setMonth()
- setSeconds()
- setTime()
- setYear()
- setFullYear()
- setUTCDate()
- setUTCFullYear()
- setUTCHours()
- setUTCMilliseconds()
- setUTCMinutes()
- setUTCMonth()
- setUTCSeconds()

برای بکار بردن این متدها شما باید یک پارامتر عددی را به آنها ارسال نمایید.

مثال ۲۲-۱۱:

```
var rightnow= new Date();
rightnow.setDate(22);
```

دیگر متدها

این قسمت شامل متدهایی است که در هیچ یک از دسته های بالا قرار نمی گیرند.

The parse() Method

این متد تعداد میلی ثانیه های بین زمان مبدا _۱ ژانویه ۱۹۷۰_و زمانی را که به عنوان پارامتر به آن وارد می کنیم را برمی گرداند.

مثال ۲۳-۱۱:

```
<script type="text/javascript">
var rightnow= new Date();
var thenum= Date.parse("Dec 12, 1999")

var theday = rightnow.setTime(thenum);
document.write(theday);
</script>
```

در مثال بالا زمان Dec 12, 1999 را به عنوان پارامتر وارد کرده ایم.

toLocaleTimeString()
toDateString()
toTimeString()
toLocaleDateString()
toString()
toGMTString()
oLocaleString()

این متدها کار متدهای بالا را انجام می دهند با این تفاوت که خروجی آنها از نوع رشته های متنی است.در و طریقه بکارگیری آنها نیز مانند دیگر متدها می باشد.

مثال ۲۴-۱۱:

به عنوان مثال پایانی ساخت یک ساعت دینامیک را در اینجا به شما آموزش می دهیم. لازم بذکر است که این ساعت در پروژه های ضمیمه همراه این کتاب نیز گنجانده شده است.

```
<body>
<h1>Current Time:</h1>
<div id="my_clock">
<!-- call to a server-side script for backup could go here -->
</div>
<script type="text/javascript" >
function startclock() {
var thetime = new Date();
var hours = thetime.getHours();
var mins = thetime.getMinutes();
var secn = thetime.getSeconds();
var ap = (hours >= 12) ? "p.m." : "a.m.";
if (hours >= 13) {
hours -= 12;
}
if (hours < 1) {
hours = 12;
}
if (mins < 10) {
mins = "0" + mins;
}
if (secn < 10) {
secn = "0" + secn;
}
var clock_div = document.getElementById("my_clock");
clock_div.innerHTML = hours + ":" + mins + ":" + secn + " " +
ap;
setTimeout("startclock()", 1000);
}
startclock();
</script>
</body>
```

نکته ای که در ساخت این ساعت بکار رفته است و باعث پویایی آن شده است استفاده از innerHTML می باشد که باعث می شود تا ساعت متحرک بنظر بیاید.

فصل دوازدهم : رشته های متنی

رشته های متنی

برای کار با رشته های متنی در جاوا اسکریپت شما باید با چگونگی بکاربردن ویژگی ها و متدهای مربوط به آنها آشنایی داشته باشید.در این فصل اول با ساختار و تعریف رشته ها در جاوااسکریپت آشنا می شوید سپس بکارگیری متدها ورشته ها را خواهید آموخت.

The String Object

در گام اول شما باید قادر به ساختن یک شی متنی باشید برای این کار از کلمه کلیدی new و به دنبال آن string استفاده می کنید به شکل زیر :

```
var instance_name = new String("string value here");
```

توضیحات :

var : برای ایجاد یک متغییر برای نگهداری رشته متنی

instance_name : نامی که شما به رشته متنی اختصاص خواهید داد.

new : یک لغت کلیدی در جاوااسکریپت برای ایجاد هر شی جدید.

String : لغت کلیدی برای ایجاد رشته متنی.

("string value here"): محتوای رشته متنی شما در اینجا و بین ("") قرار می گیرد.

مثال ۱-۱۲:

```
var guitar_string = new String("G");
```

کد بالا یک رشته متنی با نام guitar_string و با محتوای G ایجاد می کند.

ساخت لیترال متنی

شما می توانید یک لیترال متنی را تنها با استفاده از نسبت دادن آن به یک متغییر ایجاد کنید, این تکنیک مقدار فضای کمتری را نسب به شی نوشتاری مصرف می کند مانند زیر :

مثال ۲-۱۲:

```
var guitar_string = "G";
```

اما اگر از روش قبلی استفاده نمایید اجازه دسترسی به تعداد زیادی متد و ویژگی مفید را خواهید داشت که می توانند شما را در بهتر کردن اسکریپت خود یاری نمایند, امری که در صورت استفاده کردن از روش دوم تقریبا منتفی می شود.

ویژگی های شی متنی

شی متنی فقط سه ویژگی را دارا می باشد که در جدول زیر مشاهد می نمایید .

Property	Purpose
constructor	Holds the value of the constructor function for an instance of the object
length	Holds the numeric value of the length of the string (its number of characters)
prototype	Allows you to add properties to the object

The constructor Property

این ویژگی همان کاری را می کند که در اشیا دیگر جاوااسکریپت. یعنی برگرداندن سازنده کدهای محلی شی متنی.

مثال ۳-۱۲:

```
<body>
<script type="text/javascript">
var guitar_string = new String("G");
document.write(guitar_string.constructor);
```

```
</script>
</body>
```

var guitar_string = new String("G");
ایجاد یک شی متنی جدید با نام و مقدار مشخص.

document.write(guitar_string.constructor);
چاپ کردن مقدار ویژگی constructor در صفحه نمایش.

The length Property

این ویژگی طول رشته متنی را برمی گرداند, یعنی تعداد کارکترهای موجود در رشته متنی. شما مینوانید هم از شی متنی و هم از رشته لیترالی استفاده نمایید.

مثال ۴-۱۲:

```
<body>
<script type="text/javascript">
var myname="John";
document.write("The name has "+myname.length+" characters.");
</script>
</body>
```

var myname="John";
ساخت یک لیترال متنی

document.write("The name has "+myname.length+" characters.");
عدد مربوط به تعداد کارکترها در صفحه نمایش نشان داده می شود.

The prototype Property

شما می توانید با استفاده از این ویژگی, متدها و ویژگی های دگری را به شی متنی اختصاص دهید.

مثال ۵-۱۲:

```
String.prototype.attitude="cool";
var rightnow= new String("Joe");
window.alert("This string is "+rightnow.attitude);
```

String.prototype.attitude="cool";
ویژگی prototype یک خاصیت جدید را به شی متنی اضافه میکند.

var rightnow= new String("Joe");
ساخت یک شی متنی

window.alert("This string is "+rightnow.attitude);
جایگزینی cool به جای Joe و چاپ آن در نمایشگر.

متدهای شی متنی

شی متنی متدهای بسیار زیادی دارد که آنها را در جدول زیر مشاهده می نمایید. استفاده از این متدها در موقعیت های مختلف و برای ساخت یک صفحه پویا و زیبا می تواند بسیار کمک کنند باشد.

Method	Purpose
anchor()	Creates an HTML anchor tag with a target on a page
big()	Adds <big> and </big> tags around a string value
blink()	Adds <blink> and </blink> tags around a string value
bold()	Adds and tags around a string value
charAt()	Finds out which character is at a given position in a string
charCodeAt()	Finds the character code of a character at a given position in a string
concat()	Adds two or more strings together and returns the new combined string value
fixed()	Adds <tt> and </tt> tags around a string value
fontcolor()	Adds and tags around a string value, which change the color of the string to a specified color

fontsize()	Adds and tags around a string value, which change the size of the string to a specified size given as a number
fromCharCode()	Uses character codes sent as parameters to create a new string
indexOf()	Searches for a character sent as a parameter in a string: if it's found, the position of the first instance of the character is returned; otherwise, it returns –1
italics()	Adds and tags around a string value
lastIndexOf()	Searches for a character sent as a parameter in a string: if it's found, the position of the last instance of the character is returned; otherwise, it returns –1
link()	Creates HTML links using the string as the link text and linking to the URL sent as a parameter
match()	Compares a regular expression and a string to see if they match
replace()	Finds out if a regular expression matches a string and then replaces a matched string with a new string
search()	Executes the search for a match between a regular expression and a specified string
slice()	Pulls out a specified section of a string value and returns a new string
small()	Adds and tags around a string value
split()	Separates a string into an array of strings based on a character sent as a parameter to the method
strike()	Adds <strike> and </strike> tags around a string value
sub()	Adds _{and} tags around a string value
substr()	Allows a portion of the string specified with a starting position and ending after a certain number of characters to be returned
substring()	Allows a portion of the string specified with a starting position and an ending position to be returned
sup()	Adds ^{and} tags around a string value
toString()	Returns the string literal value of a String object
toLowerCase()	Converts a string to all lowercase letters and returns the result
toUpperCase()	Converts a string to all uppercase letters and returns the result

The big() Method

این متد همان کار تگ های <big></big> در صفحات html انجام می دهند. یعنی درشت تر کردن جمله, نسبت به سایر متون.

مثال ۵-۱۲:

```
var little_bit = "I only want a little bit of cake.";
var tagged_phrase = little_bit.small();
document.write(tagged_phrase);
```

معادل :

```
<small>I only want a little bit of cake.</small>
```

The blink() Method

این متد همان کار تگ های < blink ></ blink > در صفحات html ا انجام می دهند. یعنی متنی که در بین تگ های آن قرار میگیرد را به صورت چشمک زن در می آورد.الیته این متد در همه مرورگرها اجرا نمی شود.

مثال ۶-۱۲:

```
var little_bit = "I only want a little bit of cake.";
var tagged_phrase = little_bit.blink();
document.write(tagged_phrase);
```

معادل :

```
<blink>I only want a little bit of cake.</blink>
```

از دیگر متدهای این گروه می توان به,bold(), fixed(), italics(), small(), strike(), sub() و sup() اشاره کرد. که طرز کار آنها همان مثالهای قبلی می باشد و هرکدام سبک و استالی خاصی را به متن می بخشند.

The anchor() method

حتما با تگ های آنچور(لنگر) در اچ تی ام ال آشنایی دارید, تگ هایی که از آنها برای ارتباط قسمت های مختلف متن به یک دیگر استفاده می شود, بدین صورت که قسمتی از یک متن به قسمت دیگر لینک

شده و با کلیک برروی لینک مبدا به لینک مقصد, که می تواند متن, عکس و یا غیره باشد دسترسی پیدا می کنیم. در این قسمت می خواهیم متدی را معرفی کنیم که این کار را برای ما و این با به وسیله کدهای جاوااسکریپت انجام می دهد : یعنی : متد anchor()

مثال ۷-۱۲:

می خواهیم یک تگ آنچور برای دسترسی به part 1 ایجاد کنیم . ابتدا باید part 1 را به به درستی و به وسیله زیر تعریف کنیم :

```
<a name="part1">Part 1</a>
```

سپس نوبت به کدنویسی متد anchor می شود :

```
var anchor_text = "Part 1";
var full_anchor = anchor_text.anchor("part1");
document.write(full_anchor);
```

The fontcolor() Method

اگر بخواهیم به یک متن در html رنگ خاصی ببخشیم باید چیزی همانند کد زیر را بکار ببریم, البته در اینجا منظور تنها استفاده از کدهای html و بدون کمک گرفتن از Css می باشد :

```
<font color="color_value">text_string</font>
```

اما اگر بخواهیم همین عمل را به وسیله جاوااسکریپت شبیه سازی کنیم می توانیم از متد fontcolor() بهره ببریم همانند مثال زیر.

مثال ۸-۱۲:

```
<body>
<script type="text/javascript">
var the_text = "I am so mad I am red!";
document.write(the_text.fontcolor("red"));
```

```
</script>
</body>
```

معادل :

```
<font color="red">I am so mad I am red!</font>
```

در هنگام استفاده از متد بالا می توانیم از معادل هگزادسیمال رنگ ها هم استفاده نماییم.

مثال ۹-۱۲:

```
<body>
<script type="text/javascript">
var the_text = "I am so mad I am red!";
document.write(the_text.fontcolor("#FF0000"));
</font>
```

معادل:

```
<font color= "#FF0000">I am so mad I am red!</font>
```

The fontsize() Method

یکی دیگر از متدهای موجود در جاوااسکریپت fontsize() می باشد, که همانطور که از نام آن برمی آید برای تنظیم کردن اندازه متن به کار می رود.

مثال ۱۰-۱۲:

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

```
var the_text = "I am pretty small!";
document.write(the_text.fontsize(2));
</script>
</body>
```

معادل :

```
<font size="2">I am pretty small!</font>
```

The link() Method

برای قرار دادن لینک های مختلف در صفحه از این متد استفاده می نماییم.

مثال ۱۱-۱۲:

```
<body>
<script type="text/javascript">
var link_text = "A Web Site";
var full_link = link_text.link("http://www.dlp.lxb.ir");
document.write(full_link);
</script>
</body>
```

معادل :

```
<a href="http://www.www.dlp.lxb.ir">A Web Site</a>
```

مثال ۱۲-۱۲:

در مثال زیر نحوه کار متدهای لینک و آنچور را خواهید آموخت :

```
<body>
<p>
<script type="text/javascript">
var anchor_text = "Part 1";
var full_anchor = anchor_text.anchor("part1");
document.write(full_anchor);
</script>
```

```
</p>
<p>
Part 1 is about this, that and the other thing.<br />
<br /><br /><br /><br />
This is irrelevant text in this case used for filler.
</p>
<p>
<p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br />
<p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br />
<p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
```

```
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<br /><p>This is irrelevant text in this case used for
filler.</p><br />
<br />
<script type="text/javascript">
var link_text="Back to Beginning of Part 1";
var full_link= link_text.link("#part1");
document.write(full_link);
</script>
</p>
</body>
```

The charAt() Method

این متد برای پیدا کردن حرف متناظر با یک موقعیت در رشته بکار می رود. بدین معنا که شما با دادن پارامتر عددی به حروفی را که در رشته متناظر با آن عدد هستند را دریافت خواهید کرد. در استفاده از این متد باید دقت داشته باشید که حروف یک رشته از موقعیت صفر شذوع می شوند نه یک.

مثال ۱۳-۱۲:

```
var the_text = "Character";
var first_char = the_text.charAt(0);
window.alert("The first character is "+first_char);
```

پیدا کردن حرف آخر یک کلمه با استفاده از خاصیت length و به کمک متد charAt().

شما همیشه می توانید با استفاده از متد charAt() به حرف اول یک رشته دسترسی داشته باشید اما پیدا کردن حرف آخر با استفاده از خاصیت length بسیا راحت تر می باشد.

مثال ۱۴-۱۲:

```
var the_text = "Character";
var position = the_text.length-1;
var last_char = the_text.charAt(position);
window.alert("The last character is "+last_char);
```

The concat() Method

این متد همان کاری را انجام میدهد که در فصل مربوط به آرایه ها آموختیم. یعنی به چسباندن دو یا چند رشته.

مثال ۱۵-۱۲:

```
var string1 = "I went to the store ";
var string2 = "then ";
var string3 = "I played a video game";
window.alert(string1.concat(string2,string3));
```

نتیجه :

```
I went to the store then I played a video game
```

مثال ۱۶-۱۲:

```
var string1 = "I went to the store ";
var string2 = " then ";
var string3 = "I played a video game";
window.alert(string3.concat(string2,string1));
```

نتیجه :

```
I played a video game then I went to the store
```

The fromCharCode() Method

این متد با استفاده از کد مربوط به هر کارکتر عملی را انجام میدهد.

مثال ۱۶-۱۲:

```
window.alert(String.fromCharCode(72,73));
```

این کد ابتدا اولین پارامتریعنی ۷۲ را گرفته و آن را به معادل آن یعنی H تبدیل نموده سپس همین کار را با پرامتر دوم یعنی ۷۳ هم انجام داده و آن را به I ترجمه می نماید و در نهایت HI را به عنوان خروجی برمی گرداند.

The indexOf() Method

این متد موقعیت اولین حرف یک کلمه یا رشته را برمی گرداند.واگر حرف مورد نظر پیدا نشود مقدار ۱- را برمی گرداند.

مثال ۱۷-۱۲:

```
var the_text = "Cool";
var position = the_text.indexOf("C");
window.alert("Your character is at position "+position);
```

در مثال بالا 0 را برمی گرداند زیرا حرف C اولین حرف کلمه Cool می باشد.

مثال ۱۸-۱۲:

```
var the_text = "Cool";
var position = the_text.indexOf("c");
window.alert("Your character is at position "+position);
```

در مثال بالا متد مقدار 1- را برمی گرداند زیرا "c" در کلمه مورد نظر یافت نمی شود.دقت نمایید که Cool با C بزرگ شروع شده است نه c کوچک.

مثال ۱۹-۱۲:

```
var the_text = "Cool";
var position = the_text.indexOf("c");
if (position == -1) {
window.alert("Your character is not in the string!");
}
else {
window.alert("Your character is at position "+position);
}
```

در مثال بالا جمله Your character is not in the string برگردانده می شود زیرا حرف مورد نظر در کلمه یافت نمی شود.

مثال ۲۰-۱۲:

```
var the_text = "Cool";
var position = the_text.indexOf("o");
if (position == -1) {
window.alert("Your character is not in the string!");
}
else {
window.alert("Your character is at position "+position);
}
```

در مثال بالا جمله Your character is at position 1 را برمی گرداند. زیرا اولین ("o") در موقعیت ۱ کلمه Cool یافت می شود.

مثال ۲۱-۱۲:

```
var the_text = "I like fruit!";
if ((the_text.indexOf("fruit") != -1) &&
(the_text.indexOf("candy") ==
-1)) {
window.alert("Yes, fruit is good for you!");
```

```
}
else {
window.alert("Please consider fruit rather than candy.");
}
```

در مثال بالا جمله ("Yes, fruit is good for you!") برگشت داده می شود زیرا متد indexOf کلمه fruit را در رشته مورد نظر شناسایی کرده و با تطبیق دادن آن جمله اول را برمی گرداند.

The lastIndexOf() Method

این متد عکس متد ()indexOf عمل می کند, بدین معنا که ابتدا تمام جمله را پایش کره سپس آخرین موقعیتی که مورد نظر است را برمی گرداند.

مثال ۲۲-۱۲:

```
var the_text = "Cool Cruising Car";
var position = the_text.indexOf("C");
window.alert("Your character is at position "+position);
```

در مثال بالا از متد indexOf استفاده کرده ایم در نتیجه مقدار صفر برگشته داده می شود.

در حالی که در کد زیر :

```
var the_text = "Cool Cruising Car";
var position = the_text. lastIndexOf("C");
window.alert("Your character is at position "+position);
```

که از متد ()lastIndexOf استفاده کرده ایم مقدار ۱۴ برگردانده می شود.

The replace() Method

این متد دو پارامتر دریافت می کند. و طریقه عمل کردن آن بدین صورت است که ابتدا پارامتر اول را با متن یا رشته تطبیق می دهد و در صورت یافتن آن در متن یا رشته, پارامتر دوم را جایگزین آن می کند.

مثال ۱۲-۲۳:

```
<body>
<script type="text/javascript">
var the_text = "Benz Cruising Car";
var the_text_replace = the_text.replace("Benz","BMW");
window.alert(the_text_replace);
</script>
</body>
```

The slice() Method

این متد برای جدا کردن قسمتی از متن یا رشته و تبدیل آن به رشته ای جداگانه به کار می رود. این متد دو پارامتر دریافت می کند, که پارامتر اول موقعیت کارکتری را نشان می دهد که عملیات از آن شروع می شود و پارامتر دوم موقعیت کارکتری را نشان می دهد که عملیات برش تا آنجا ادامه می یابد.

مثال ۱۲-۲۴:

```
var the_text = "Do not cut this short!";
var shorter_string = the_text.slice(0,7);
window.alert(shorter_string);
```

در مثال بالا ” Do not“ به عنوان خروجی نهایی برگردانده می شود.

The split() Method

این متد یک آرایه از اجزایی که در رشته متنی قرار دارند ایجاد می کند.این متد پارامتری را به عنوان جداکننده دریافت می نماید و بر اساس آن عمل split کردن را انجام می دهد.

مثال ۱۲-۲۶:

```
var the_text = "orange:apple:pear:grape";
```

```
var split_text = the_text.split(":");
var end_count = split_text.length;
for (var count=0; count<end_count; count++) {
document.write(split_text[count]+"<br />");
}
```

The substr() Method

این متد قسمتی از متن را حذف کرده و آن را به عنوان خروجی میدهد. از دو پارامتر برای این عمل استفاده می کند که پارامتر اول شروع عملیات و پارامتر دوم توقف آن را اعلام می کند.

مثال ۱۲-۲۷:

```
var the_text = "Do not cut this short!";
var shorter_string = the_text.substr(0,7);
window.alert(shorter_string);
```

The substring() Method

این متد همانند متد substr() عمل می کند اما این متد به شما اجازه می دهد تا پارامترها را برای اولین و آخرین موقعیت در جمله ارسال کنید و درنهایت متن حذف شده را به عنوان خروجی برمی گرداند.

مثال ۱۲-۲۸:

```
var the_text = "Do not cut this short!";
var shorter_string = the_text.substring(3,7);
window.alert(shorter_string);
```

The toString() Method

این متد برای تبدیل یک String object به string literal به کار می رود.

مثال ۱۲-۲۹:

```
var string_obj = new String("Cool");
var string_lit = string_obj.toString();
```

The toLowerCase() Method

گرفتن حروف بزرگ موجود در متن و تبدیل آنها به حروف کوچک کاری است که این متد انجام می دهد.

مثال ۱۲-۳۰:

```
<body>
<script type="text/javascript">
var the_text = "I FEEL CALM, REALLY.";
document.write(the_text.toLowerCase());
</script>
</body>
```

نتیجه:

```
i feel calm, really.
```

The toUpperCase() Method

عملی عکس متد بالا انجام می دهد. یعنی تبدیل حروف کوچک به حروف بزرگ.

مثال ۱۲-۳۱:

```
<body>
<script type="text/javascript">
var the_text = "I am yelling!";
document.write(the_text.toUpperCase());
</script>
</body>
```

نتیجه :

```
I AM YELLING!
```

حالا شما با متدهای موجود در شی رشته ای آشنا شده اید, در مثال زیر سعی شده است چند تا از این متدها را با هم به کار ببریم.

مثال ۳۲-۱۲:

```
<body>
<h1>Welcome!</h1>
<script type="text/javascript">
function getname() {
var the_text=window.prompt("Enter your first and last
name","");
if (the_text.indexOf(" ") == -1) {
window.alert("Put a space between your first and last name.
Try again.");
getname();
}
var split_text= the_text.split(" ");
if ((split_text[0].charAt(0) != "Z") ||
(split_text[0].charAt(0) != "z")) {
var shorter_fn_string =
split_text[0].substring(1,split_text[0].length);
new_fn_name = "Z"+shorter_fn_string;
}
else {
var shorter_fn_string =
split_text[0].substring(1,split_text[0].length);
new_fn_name = "W"+shorter_fn_string;
}
if ((split_text[1].charAt(0)!= "Z") ||
(split_text[1].charAt(0)!= "z")) {
var shorter_ln_string=
split_text[1].substring(1,split_text[1].length);
new_ln_name="Z"+shorter_ln_string;
}
else {
var shorter_ln_string=
split_text[1].substring(1,split_text[1].length);
new_ln_name="W"+shorter_ln_string;
}
window.alert("Now your name is "+new_fn_name+"
"+new_ln_name+"!");
}
getname();
</script>
</body>
```

در این مثال ابتدا اسم کاربر دریافت می شود و سپس حروف اول با حرف "Z" جایگزین می شود. هر چند مثال بالا یک مثال کاربردی نمی باشد اما شما را با استفاده از متدهای موجود در شی رشته متنی بیشتر آشنا می کند.

شیء عبارات منظم (RegExp Object)

RegExp چیست

یک عبارت منظم (Regular Expression) شی ئی است که الگوی کاراکترها را مشخص می کند. وقتی که شما در حال جستجو درباره عبارتی در یک متن هستید نشان دهنده این است که شما به دنبال چه می گردید. یک الگوی ساده می تواند شامل فقط یک کاراکترباشد.

الگوهای پیچیده تر کاراکترهای بیشتری دارند و برای تجزیه (Parse)، بررسی ساختار یا فرمت، جانشینی و یا کارهای دیگر مورد استفاده قرار گیرد.

عبارات منظم ابزاری قدرتمند برای اجرای "الگو سنجی و جستجو و جابجایی" در متن ها به کار می رود.

```
var varname = /your_pattern/flags
```

مثال ۳۳-۱۲:

```
var thename = window.prompt("Enter your name","");
var tomatch = /John/;
var is_a_match = tomatch.test(thename);
if (is_a_match) {
window.alert("Wow, we have the same name!");
```

```
}
else {
window.alert("Not my name, but it will work!");
}
```

در مثال بالا اگر نامی که کاربر وارد می کند John باشد پیام Wow, we have the same name به نمایش در می آید در غیر اینصورت پیام Not my name, but it will work نمایش داده می شود.

Adding Flags

با استفاده از پرچم ها (Flags) شما می توانید مشخص کنید که عملیات تطبیق دادن در این متن یا در هر متن دیگری انجام گیرد.

لیست پرچم ها را در زیر مشاهده می نمایید.

Flag(s)	Purpose
i	Makes the match case insensitive
g	Makes the match global
m	Makes the match work in multiline mode

Flag i :

خاصیت این Flag اینست که مشخص می کند عملیات تطبیق, صرفنظر از بزرگی یا کوچکی حروف انجام گیرد در نتیجه حساسیت به بزرگی و کوچکی حروف از بین می رود.

مثال ۳۴-۱۲:

```
var thename= window.prompt("Enter your name","");
var tomatch=/John/i;
if (tomatch.test(thename)) {
window.alert("Wow, we have the same name!");
}
else {
window.alert("Not my name, but it will work!");
}
```

Flag g :

از (global) g برای اینکه بگوییم تمام همخوانی ها را بعد از پیدا کردن اولین همخوانی نیز به دست آورد به کار می رود. این مدیفایر به حروف کوچک و بزرگ حساس است و آن ها را قبول نمی کند همانند همین مثال که فقط دو is را شناسایی کرد و از Is اول اجتناب کرد این مدیفایر به دنبال کلمه ای که شما در قسمت pattern وارد کرده اید می گردد و همانند آن را از داخل متغیر str استخراج می کند.

مثال ۱۲-۳۵:

```
<body>
<h1>Welcome!</h1>
<script type="text/javascript">
var str = "Is this all there is?";
var patt1 = /is/g;
document.write(str.match(patt1));
</script>
</body>
```

نتیجه :

```
is,is
```

در مثال زیر از هر دو Flag استفاده شده است و خروجی تلفیقی از آن ها می باشد

مثال ۱۲-۳۶:

```
<body>
<h1>Welcome!</h1>
<script type="text/javascript">
var str = "Is this all there is?";
var patt1 = /is/gi;
document.write(str.match(patt1));
</script>
</body>
```

نتیجه:

```
Is,is,is
```

متد test()

این متد در داخل یک متن دنبال مقدار مورد نظر شما می گردد و در صورتی که در آن وجود داشته باشد مقدار true و در غیر این صورت مقدار false را بر می گرداند.

مثال ۱۲-۳۷:

```
<body>
<script type="text/javascript">
var patt1 = new RegExp("e");
document.write(patt1.test("the best things in life are
free"));
</script>
</body>
```

نتیجه:

```
true
```

متد exec()

این متد در داخل متن دنبال مقدار مورد نظر شما گشته و در صورت پیدا کردن آن متن پیدا شده را بر می گرداند و در صورتی که پیدا نکند مقدار null را بر می گرداند.

مثال ۱۲-۳۸:

```
<body>
<script type="text/javascript">
var patt1 = new RegExp("e");
document.write(patt1.exec("the best things in life are
free"));
</script>
</body>
```

نتیجه:

```
e
```

در مثال بالا اگر شما ده e نیز داشته باشید فقط یک e در خروجی چاپ می شود که مفهوم موجود است را می رساند.

در زیر لیستی از کدهای RegExp و توضیحاتی مختصر در مورد آنها را مشاهده می نمایید.

Character	Character	Example
\	Used to escape special characters or to make a normal character special	\@ escapes the *@* character \n represents a newline character
[\b]	Matches a BACKSPACE keystroke	/[\b]/ matches a backspace
\b	Matches when the character before or after it is located at a word boundary, such as before or after a space character; to match the beginning of a word, place the character to the right of the symbol (\b*c*); to match the end of a word, place the character to the left (*c*\b)	/\bc/ matches *c* in *my car* /\bm/ matches *m* in *my car* /\bc/ does not match *c* in *ace* /\bm/ does not match *m* in *Sam* /m\b/ matches *m* in *Sam* /c\b/ matches *c* in *Mac W* /m\b/ does not match *m* in *emu* /c\b/ does not match *c* in *my car*
\B	Matches a character that is not located at a word boundary	/\Ba/ matches *a* in *car* /\Bc/ does not match *c* in *car*
\c*X*	Using a letter character to replace *X*, matches when the user presses the CTRL key followed by typing the letter *X*	/\cX/ matches *CTRL-X* /\cV/ matches *CTRL-V* /\cS/ does not match *CTRL-Z*
\D	Matches a single character if it is *not* a numeric character	/\D/ matches *s* /\D/ does not match *4*
\f	Matches if there is a form feed	/\f/ matches a form feed
\n	Matches if there is a new line	/\n/ matches a new line
\r	Matches if there is a carriage return	/\r/ matches a carriage return

\s	Matches a single character if it represents white space (such as a space or a new line)	/\s/ matches the space in *b c* /\s/ matches the tab in *b c* /\s/ does not match *bc*
\S	Matches a single character if it does *not* represent white space	/\S/ matches *d* /\S/ does not match a blank space
\t	Matches if there is a tab	/\t/ matches the tab in *b c*
\v	Matches if there is a vertical tab	/\v/ matches a vertical tab
\w	Matches any single character that is a letter, number, or underscore	/\w/ matches *4* /\w/ does not match *@*
\W	Matches any single character that is *not* a letter, number, or underscore	/\W/ matches *@* /\W/ does not match *g*
^	Matches only from the beginning of a line	/^c/ matches *c* in *corn* /^c/ does not match *c* in *acorn*
$	Matches only at the end of the line	/r$/ matches *r* in *Car* /r$/ does not match *t* in *Cat*
*	Matches the character preceding it if the character occurs zero or more times	/co*/ matches *co* or *c* /co*/ does not match *pi*
+	Matches the character preceding it if it occurs one or more times	/co+/ matches *co* or *cooooo* /co+/ does not match *ca*
?	Matches the character preceding it if it occurs zero or one time	/o?l/ matches *style* or *column* /co?l/ does not match *cool*
.	Matches any individual character, excluding the newline character	/.l/ matches *al* or *@l* /.l/ does not match *lnl* or *l*
(*x*)	By replacing *x* with characters, matches that sequence and keeps it in memory to be used later; used for grouping of expressions	/(a)/ matches *a* /(cool)/ matches *cool* /(cool)/ does not match *coal*
|	Used as a logical OR symbol to allow a match of what is on the left of the symbol OR what is on its right	/cool|bad/ matches *cool* /cool|bad/ matches *bad* /cool|bad/ does not match *car*
{*x*}	Using a number to replace *x*, matches when there are exactly *x* occurrences of the character preceding it	/n{1}/ matches *n* /nn{2}/ matches *nnn* /nn{1}/ does not match *nnn*
{*x*,}	Using numbers to replace *x* and *y*, matches when there are at least *x* occurrences of the character preceding it but no more than *y* occurrences of it	/n{1,2}/ matches *n* /n{1,2}/ matches *nn* /n{2,3}/ does not match *n* /n{4,7}/ does not match *nnn*
[]	Matches a character set of your choice; will match when any one of the characters in the brackets (such as [abc]) or any one of a range of characters (such as [a-k]) is present	/[abc]/ matches *a* /[abc]/ matches *b* /[abc]/ matches *c* /[a-k]/ matches *j* /[a-k]/ does not match *n*

[^]	Matches when the characters in your character set are *not* present; may be a set (such as [abc]) or a range (such as [a-k])	/[^abc]/ matches *d* /[^abc]/ does not match *b* /[^a-k]/ matches *n* /[^a-k]/ does not match *j*

The replace() Method

شما می توانید از متد replace بهمراه RegExp در شی رشته ای استفاده کنید. طریقه استفاده از آن به صورت زیر می باشد.

```
varname= stringname.replace(regex,newstring);
```

در مثال زیر نحوه استفاده از این متد را نشان می دهیم.

مثال ۱۲-۳۹:

```
var mystring= "I like the way a new car smells, and cars are
fun.";
var toreplace=/car/;
var newstring= mystring.replace(toreplace,"skunk");
window.alert(newstring);
```

در این مثال متد replace اولین کلمه car که در جمله ببیند را با کلمه skunk عوض می کند.

مثال ۱۲-۴۰:

```
var mystring= "I like the way a new car smells, and cars are
fun.";
var toreplace=/car/g;
var newstring= mystring.replace(toreplace,"skunk");
window.alert(newstring);
```

مثال بالا همانند مثال قبل می باشد با این تفاوت که از g به همراه آن استفاده کرده ایم RegExp درنتیجه متد replace هر کلمه car که در جمله ببیند را با کلمه skunk عوض می کند.

The match() Method

وظیفه این متد پیدا کردن یک کلمه مطابق با کلمه مورد نظر است. اگر این لغت پیدا شود متد آن را برمی گرداند و در غیر اینصورت مقدار ۱- را برمی گرداند.

مثال ۴۱-۱۲:

```
var mystring = "I am Ironman!";
var tomatch = /Iron/;
if (mystring.match(tomatch)) {
window.alert("Your string contains Iron!");
}
else {
window.alert("Sorry, no Iron in your string.");
}
```

The search() Method

این متد وظیفه پیدا کردن و برگرداندن لغت مورد نظر در جمله را برعهده دارد. البته این متد نه خود لغت را که موقعیت عددی اولین حرف آن را برمی گرداند.

مثال ۴۲-۱۲:

```
var mystring = "I am Ironman!";
var tomatch = /Iron/;
if (mystring.search(tomatch)) {
window.alert("Iron found at position
"+mystring.search(tomatch)+"!");
}
else {
window.alert("Sorry, no Iron in your string.");
}
```

مثال ۴۳-۱۲:

در مثال زیر نحوه استفاده از چند متد و RegExp را مشاهده می کنید. وظیفه اسکریپت زیر اینست که مشخص کند آیا در متن مورد نظر عددی وجود دارد یا خیر.

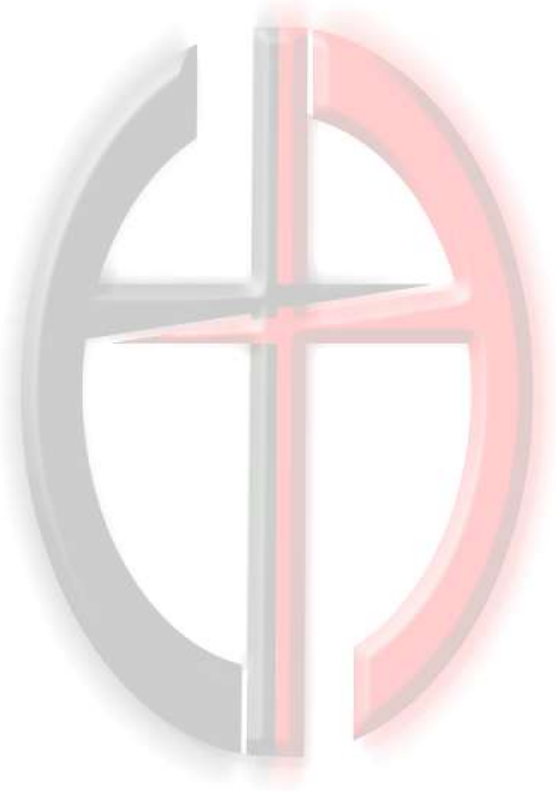
```
<body>
<form>
Enter some text: <input type="text" id="has_digits" />
<input type="button" id="t_btn" value="Test" />
</form>
<script type="text/javascript">
var t_button = document.getElementById("t_btn");
t_button.onclick = function() {
var has_num = document.getElementById("has_digits").value;
var tomatch = /\d+/;
if (tomatch.test(has_num)) {
window.alert("Your entry contained one or more numbers!");
}
else {
window.alert("Your entry did not contain any numbers!");
}
}
</script>
</body>
```

اطلاعات بیشتر

چنانچه توضیحات موجود در این بخش را ناکافی می دانید می توانید برای توضیحات بیشتر به یکی از وبسایتهای زیر مراجعه نمایید.

- www.regular-expressions.info/
- www.regular-expressions.info/javascript.html (includes specifics on the JavaScript engine)
- https://developer.mozilla.org/en/Core_JavaScript_1.5_Guide/Regular_Expressions

فصل سیزدهم : فرم در جاوا اسکریپت



فرم ها

زمانی که شما دو تگ <form></form> را در کد html خود وارد می کنید یک شی فرم ایجاد کرده اید فرم ها نقشی حیاتی و اساسی در ارتباط کاربر با صفحات وب ایفا می کنند. در نتیجه متدها و خاصیت های فراوانی برای کار با آنها در جاوا اسکریپت فراهم شده است.

دسترسی به فرم ها

برای دسترسی به فرم ها سه را پیش رو دارید:

۱- استفاده از form array شی سند.
۲- تخصیص یک نام به فرم و استفاده از آن نام برای دسترسی.
۳- اختصاص یک id به فرم و استفاده از متد document.getElementById() برای دسترسی به فرم.

استفاده از form array

آرایه فرم به شما این اجازه را می دهد تا با دادن یک ایندکس عددی به فرم ها دسترسی داده باشید. در واقع این ایندکس عددی مکان عددی فرم را نشان می دهد.

مثال ۱-۱۳:

```
document.forms[0]
```

به اولین فرم در صفحه اشاره می کند.

نکته : همانطور که در مثال بالا مشاهده می کنید شروع آرایه از عدد صفر است در نتیجه مثلا document.forms[1] به دومین فرم موجود در صفحه اشاره می کند.

length Property

این خاصیت تعداد اجزای موجود در یک فرم را برمی گرداند.

مثال ۲-۱۳:

```
<body>
<form>
Name: <input type="text" /> <br />
E-mail: <input type="text" /> <br />
<input type="submit" value="Submit" />
</form>
<p>
<script type="text/javascript">
document.write("The form has "+document.forms[0].length+"
elements.");
</script>
</p>
</body>
```

نتیجه :

```
The form has 3 elements.
```

زیرا در فرم مثال بالا سه جزء قرار گرفته است دو ورودی متنی و یک دکمه submit

نکته :

document.forms.length : تعدا اجزای موجود در تمام فرم های صفحه را برمی گرداند.

document.forms[x].length : تعداد اجزای موجود در فرم قرار گرفته در موقعیت *x* را برمی گرداند.

مثال ۳-۱۳:

```
<body>
<h1>Form Lengths</h1>
<h2>Form 1</h2>
<form>
Name: <input type="text" /><br />
E-mail: <input type="text" /><br />
<input type="submit" value="Submit" />
```

```
</form>
<h2>Form 2</h2>
<form>
Favorite Color: <input type="text" /><br />
Favorite Food: <input type="text" /><br />
<input type="reset" value="Reset" /> 
<input type="submit" value="Submit" />
</form>
<h2>Results</h2>
<script type="text/javascript" >
for(var count=0;count<document.forms.length;count++) {
var formnum = count+1;
document.write("Form "+formnum+" has
"+document.forms[count].length);
document.write(" elements.<br />");
}
</script>
</body>
```

نتیجه:

```
Form 1 has 3 elements.
Form 2 has 4 elements.
```

استفاده از Form Names

همانطور که قبل تر بیان کردیم می توان به هر فرم در صفحه یک نام اختصاص داد تا بعدا به وسیله آن نام بتوان به فرم دسترسی پیدا کرد .مزیت این امر نسبت به استفاده از آرایه فرم در این است که شما نیازی به دانست موقعیت فرم در صفحه نخواهید داشت. شما باید این نام را در تگ فرم وارد نمایید.

```
name="yourname"
```

مثال ۴-۱۳:

در مثال زیر یک نام info_form_ به فرم اختصاص داده ایم و بعد با استفاده از این نام و با کمک خاصیت length به آن دسترسی پیدا کرده ایم.

```
<body>
```

```
<form name="info_form">
Name: <input type="text" /><br />
<input type="submit" />
</form>
<p>
<script type="text/javascript">
document.write("The form has "+document.info_form.length+"
elements.");
</script>
</p>
</body>
```

نتیجه :

```
The form has 2 elements.
```

استفاده از ID

استفاده از ID نیز همانند دادن یک نام به فرم است و مزیتی که دارد اینست که با دادن یک ID منحصر به فرد به هر فرم در صفحه آنها را از یکدیگر متمایز کنیم.

مثال ۵-۱۳:

```
<body>
<form id="info_form">
Name: <input type="text" /><br />
<input type="submit" />
</form>
<p>
<script type="text/javascript">
var f_length = document.getElementById("info_form").length;
document.write("The form has "+f_length+" elements.");
</script>
</p>
</body>
```

Properties

هر شی در جاوا اسکریپت دارای خاصیت هایی می باشد و چون فرم ها نیز شی محسوب می شوند در نتیجه از این قاعده مستثنی نیستند. تعدادی از خئاص شی فرم را در جدول زیر مشاهده می نمایید.

Property	Value
action	The value of the action attribute in the HTML form tag
elements	An array that includes an array element for each form element in an HTML form
encoding	The value of the enctype attribute, which varies with different browsers
length	The value of the total number of elements in an HTML form
method	The value of the method attribute in an HTML form tag
name	The value of the name attribute in an HTML form tag
target	The value of the target attribute in an HTML form tag

The action Property

فرم ها به خودی خود کاری به جز جمع آوری اطلاعات از کاربر انجام نمی دهند. کاری که بعد از جمع آوری اطلاعات اهمیت دارد پردازش این اطلاعات است که به وسیله اسکریپت هایی موجود در سرور صورت می گیرد. این متد به شما اجازه وارد کردن آدرس صفحه ای را می دهد که قرار است فرم شما را پردازش نماید.

مثال ۶-۱۳:

```
<body>
<form name="info_form"
action="http://someplace.com/php/form.php">
Name: <input type="text" /><br />
<input type="submit" />
</form>
<p>
<script type="text/javascript">
document.write("The form goes to "+document.info_form.action);
</script>
</p>
</body>
```

The elements Property (Array)

یک روش دسترسی به اجزای فرم می باشد. که این عمل را با استفاده از پارمترهای عددی و متناظر کردن آنها با موقعیت عددی عناصر فرم انجام می دهد.

مثال ۷-۱۳:

```
<form name="info_form">
Name: <input type="text" /><br />
<input type="submit" />
</form>
```

document.info_form.elements[0] : این کد باعث دسترسی به ورودی متنی می شود.

document.info_form.elements[1] : این کد باعث دسترسی به دکمه submit می شود.

شما همچنین می توانید از ID نیز برای دسترسی به اجزای آنها کمک بگیرید.

مثال ۸-۱۳:

```
<form>
Name: <input type="text" id="yourname"><br />
<input type="submit">
</form>
```

document.getElementByID("yourname"); : دسترسی به تمام اجزای آرایه با استفاده از ID فرم.

The checked Property

برای چک کردن علامت زدن/ نزدن رادیو دکمه ها_radio buttons_ بکار می رود. و یک مقدار بولین را برمیگرداند که می توان از این مقدار بولین برای فعال کردن یک اسکریپت دیگر استفاده کرد.

مثال ۹-۱۳:

```
<body>
<form>
Check box to say Yes: <input type="checkbox" id="yes_no">
<br /><br />
<input type="button" value= "See the Answer"
onclick="is_it_checked();" />
</form>
<script type="text/javascript">
function is_it_checked() {
var y_n = document.getElementById("yes_no");
if (y_n.checked) {
window.alert("Yes! The box is checked!");
}
else {
window.alert("No, the box is not checked!");
}
}
</script>
</body>
```

با کلیک کردن روی دکمه See the Answer پیغامی مبنی بر علامت زدن / نزدن دکمه رادیویی به نمایش در می آید.

در جدول زیر اجزای یک فرم بهمراه متدها و ویژگی های هرکدام را مشاهئه می نمایید.

Element Type	Object Name	Properties	Methods
Button	button	form, name, type, value	blur(), click(), focus()
Check box	checkbox	checked, defaultChecked, form, name, type, value	blur(), click(), focus()
Hidden field	hidden	form, name, type, value	None
Radio button	radio	checked, defaultChecked, form, name, type, value	blur(), click(), focus()
Reset button	reset	form, name, type, value	blur(), click(), focus()
Select box	select	form, name, options, selectedIndex, type	blur(), focus()
Submit button	submit	form, name, type, value	blur(), click(), focus()
Text box	text	defaultValue, form, name, type, value	blur(), focus(), select()
Text area	textarea	defaultValue, form, name, type, value	blur(), focus(), select()

The defaultChecked Property

این متد در واقع مقادیر بولین را برمی گرداند. و وظیفه آن مشخص کردن مقدار پیش فرض checkbox یا radiobuttonها است.

مثال ۱۰-۱۳:

```
<form>
Do you want us to send you e-mail updates and offers?<br />
Yes <input type="checkbox" id="yes" checked="checked" />
No <input type="checkbox" id="no" />
</form>
```

در مثال بالا ok به صورت پیش فرض تیک خورده است.

The defaultValue Property

وظیفه متد بالا دادن مقدار پیش فرض به text box و text area می باشد.

مثال ۱۱-۱۳:

در مثال زیر اگر کاربر سایت مورد نظر خود را در جعبه متنی تایپ نماید و یا مقدار آن را پاک کند و حال به هر دلیلی بخواهد از مقدار پیش فرض استفاده نماید با زدن دکمه Reset Default به این امر دست پیدا می کند.

```
<body>
<form>
Favorite URL:<br />
<input type="text" id="favurl" value="http://www.yahoo.com">
<br /><br />
<input type="button" value="Reset Default"
onclick="back_to_default();">
</form>
<script type="text/javascript">
function back_to_default() {
var url_box = document.getElementById("favurl");
url_box.value = url_box.defaultValue;
}
</script>
```

```
</body>
```

The form Property

این متد زمانی رخ می دهد که شما از کلمه کلیدی this برای ارجاع به فرم یا یکی از اجزای آن استفاده نمایید.در مثال زیر مشاهده خواهید کرد که اگر بخواهیم مقدار پیش فرض یک جعبه متنی را تغییر دهیم باید ابتدا به فرمی که جعبه در آن قرار دارد با استفاده از کلمه کلیدی this اشاره کنیم سپس به ترتیب ابتدا کلمه form سپس نام textbox و بعد از آن خاصیت مورد نظر که می خواهیم تغییر دهیم مانند value و در نهایت مقدار جدید را وارد کنیم.

مثال ۱۲-۱۳:

```
<form>
Favorite URL:<br />
<input type="text" name="favurl" value=http://www.yahoo.com />
<br /><br />
<input type="button" value="Change"
onclick="this.form.favurl.value='http://www.lycos.com';" />
</form>
```

The options Property (Array)

این متد در واقع یک آرایه است که گزینه های جز option در فرم را نگهداری می کند. عدد ایندکس این متد از ۰ صفر شروع می شود.

مثال ۱۳-۱۳:

```
<body>
<form>
Fruits:
<select id="optlist">
<option selected="selected" value="orange">Orange</option>
<option value="apple">Apple</option>
<option value="pear">Pear</option>
</select>
</form>
<p>
```

```
<script type="text/javascript">
var fbox = document.getElementById("optlist");
document.write("The second option is ");
document.write(fbox.options[1].value);
</script>
</body>
```

در مثال بالا با استفاده از کد fbox.options[1].value به مقدار دومین عنصر المنت option دسترسی پیدا کرده ایم.

The type Property

برای مشخص کردن نوع عنصر بکار رفته در فرم استفاده می شود.

مثال ۱۴-۱۳:

```
type="text"
type="button"
```

The value Property

یک نام به عنصر موجود در فرم اختصاص می دهد. این نام همان نامی است که کاربر مشاهده می کند.

The focus() Method

هنگامی رخ می دهد که کاربر بخواهد کار بروی یکی از اجزای فرم متمرکز شود.

مثال ۱۵-۱۳:

```
<body>
<form>
Your Favorite Food
<input type="text" id="fav_food" /><br />
Drink <input type="text" />
</form>
<script type="text/javascript">
```

```
var f_box = document.getElementById("fav_food");
f_box.focus();
</script>
</body>
```

The blur() Method

عملی برعکس خاصیت بالا انجام می دهد ترک تمرکز بر روی اجزای فرم.

مثال ۱۶-۱۳:

```
<body>
<form>
Your Favorite Food
<input type="text" name="fav_food"
onblur="this.form.annoy.click();"
/><br />
Drink <input type="text" />
<br /><br />
<input type="reset" name="annoy" value="Reset Form">
</form>
</body>
```

در مثال بالا هنگامی که روی جعبه متنی دوم کلیک می کنید در واقع تمرکز خود را از جعبه متنی اول به دوم انتقال داده اید و در این لحظه متد به وقوع می پیوندد و پیامی برای شما به نمایش در می آید.

The method Property

این خاصیت برای مشخص کردن طریقه ارسال اطلاعات به سرور است و دارای دو مقدار get و post است.

مثال ۱۷-۱۳:

```
<form name="f1" method="post" action= "http://site.com/cgi-
bin/form.cgi">
<!-- form contents here -->
</form>
```

The name Property

برای اختصاص دادن یک نام به فرم بکار می رود.

مثال ۱۸-۱۳:

```
<form name="cool_form">
<!-- form contents here -->
</form>
```

The target Property

این خاصیت محل ظاهر شدن پنجره جدید در صورت نیاز را نشان می دهد.و شامل مقادیر زیر می باشد:

- blank
- new
- parent
- top
- self

توصیه می شود برای فمیدن مطالب این بخش متون آموزش html را مطالعه نمایید.

مثال ۱۹-۱۳:

```
<form name="cool_form" target="place" action="program.cgi">
<!-- form contents here -->
</form>
```

Methods

The reset() Method

از این متد برای ریست کردن فرم استفاده می شود همچنین به شما اجازه ریست کردن رویدادها را نیز می دهد.

مثال ۲۰-۱۳:

```
<body>
<form>
Your Favorite Food
<input type="text" name="fav_food" onblur="this.form.reset();"
/><br />
Drink <input type="text" />
<br /><br />
<input type="reset" value="Reset Form" />
<input type="submit" value="Submit Form" />
</form>
</body>
```

The submit() Method

این متد به شما اجازه submit کردن فرم را به شما می دهد فارغ از اینکه کاربر کلید submit را فشار دهد یا نه.

مثال ۲۱-۱۳:

```
<body>
<form action="http://site.com/php/form.php">
Your Favorite Food
<input type="text" name="fav_food"
onblur="this.form.submit();" /><br />
Drink <input type="text" />
<br /><br />
<input type="submit" value="Submit Form" />
```

```
</form>
</body>
```

در مثال بالا بعد از وارد کردن اطلاعات در دو جعبه متنی عملیات Submit انجام می گیرد.

اطمینان از قابلیت دسترسی در فرم ها

استفاده از اجزا و برچسب های آماده

استفاده از ابزار و برچسب ها باعث می شود تا کاربر راحت تر با فرم ارتباط برقرار کرده و اطلاعات را در جای مناسب وارد کند .

مثال ۲۲-۱۳:

```
<input type="text" name="yourname" id="yourname" /> Name<br />
<input type="text" name="zip_code" id="zip_code" /> Zip
Code<br />
```

یا

```
Name <input type="text" name="yourname" id="yourname" /><br />
Zip Code <input type="text" name="zip_code" id="zip_code"
/><br />
```

Using <label></label> Tags

استفاده از تگ هایlabel باعث می شود تا وابستگی برچسب ها به اجزای فرم مشخص شود.

مثال ۲۳-۱۳:

```
<label for="yourname">Name</label>
<input type="text" name="yourname" id="yourname" /><br />
```

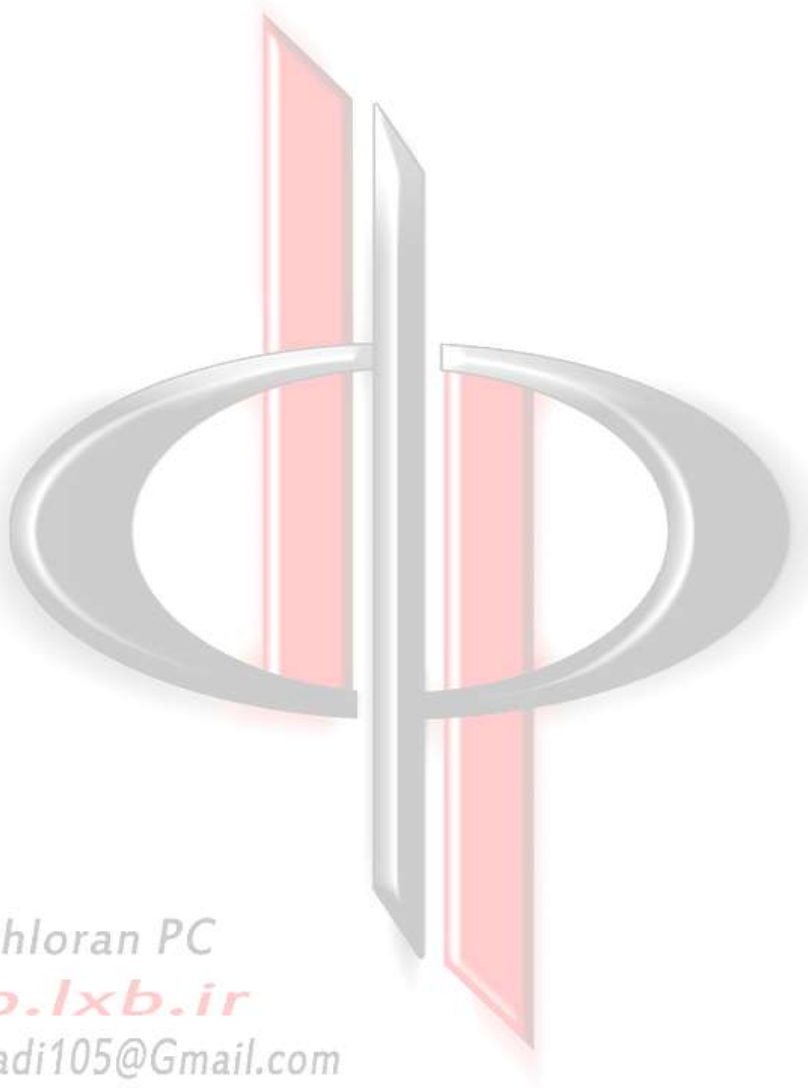
Using <fieldset></fieldset> Tags

استفاده از fieldsetها می تواند به شما در ایجاد radio buttonها و check boxهایی که یک وظیفه مشخص دارند کمک کند و آنها را در یک گروه قرار دهد.

مثال ۲۴-۱۳:

```
<fieldset>
<legend>Select a Fruit:</legend>
<input type="radio" name="fruits" id="fruits1" value="Orange"
/>
<label for="fruits1">Orange</label>
<input type="radio" name="fruits" id="fruits2" value="Banana"
/>
<label for="fruits2">Banana</label>
<input type="radio" name="fruits" id="fruits3" value="Apple"
/>
<label for="fruits3">Apple</label>
</fieldset>
```

فصل چهاردهم : فریم در جاوا اسکریپت



فریم ها در جاوا اسکریپت

یکی از امکاناتی که html برای طراحان فراهم آورده است اینست که می توانیم به وسیله یکسری عناصر خاص به نمایش چندین صفحه در یک پنجره مرورگر بپردازیم, در اصل با استفاده از این عناصر می توانیم که پنچره مرورگر را به چند بخش که به هرکدام یک فریم (frame) گفته می شود تقسیم کرد و در هر قسمت یک صفحه را نمایش دهیم.

این بخش مربوط به متدها و روش هایی است که جاوااسکریپت برای کار با فریم ها آماده کرده است که در ادامه به معرفی آنها خواهیم پرداخت.

برای شروع آموزش در ابتدا نیاز است که دو صفحه html با نام های frame1.html و frame2.html ایجاد نموده و ذخیره نمایید.

frame1.html

```
<html>
<body>
I am frame1.html, and I am on the left side!
</body>
</html>
```

frame2.html

```
<html>
<body>
I am frame2.html, and I am on the right side!
</body>
</html>
```

در ابتدا با استفاده از کد زیر یک پنچره با دو فریم ایجاد می نماییم:

```
<html>
<frameset cols="20%,80%">
<frame src="frame1.html"></frame>
<frame src="frame2.html"></frame>
</frameset>
<noframes>
Sorry, your browser does not support frames. Use the link
below to go to the frameless version of the site.<br />
<a href="noframes.html">Frameless Site</a>
</noframes>
</html>
```

نکته: در صدهایی که در تگ <frameset> بکار رفته اند برای تقسیم کردن پنجره مرورگر بکار می روند بدین صورت که ۲۰٪ پنجره را به فریم۱ و ۸۰٪ فضای پنجره را به فریم ۲ اختصاص می دهند.

با استفاده از سه کد بالا سه صفحه html ایجاد نمایید و آنها را در پوشه ذخیره نمایید.

Frame Options

همانطور که مشاهده می کنید تگ frameset می تواند صفحه را به صورت افقی برای نمایش فریم ها تقسیم کند مانند مثال زیر :

مثال ۱-۱۴:

```
<html>
<frameset rows="20%,80%">
<frame src="frame1.html"></frame>
<frame src="frame2.html"></frame>
</frameset>
<noframes>
Use the link below to go to the frameless version of the
site.<br />
<a href="noframes.html">Frameless Site</a>
</noframes>
</html>
```

در مثال بالا صفحه به سطرهایی برابر با ۲۰٪ اندازه اش برای نمایش یک فریم و ۸۰٪ اندازه اش برای نمایش دیگر فریم تقسیم شده است.

اگر بخواهید که فقط اندازه یکی از پنجره ها را مشخص کنید بطوری که مقدار باقی مانده به فریم دیگر اختصاص یابد می توانید از علامت "*" استفاده نمایید.

مثال ۲-۱۴:

```
<html>
<frameset rows="150,*">
<frame src="frame1.html"></frame>
<frame src="frame2.html"></frame>
</frameset>
<noframes>
Use the link below to go to the frameless version of the
site.<br />
<a href="noframes.html">Frameless Site</a>
</noframes>
</html>
```

برای نمایش سطرها به صورت ستونی نیز می توانید همانند مثال زیر عمل نمایید.

در مثال زیر مقدار ۱۵۰ پیکسل برای فریم سمت چپ و ۱۵۰ پیکسل برای فریم سمت راست در نظر گرفته شده است و دیگر فضای باقی مانده بین دو فریم به فریم وسطی اختصاص می یابد.

مثال ۳-۱۴:

```
<html>
<frameset cols="150,*,150">
<frame src="frame1.html"></frame>
<frame src="frame2.html"></frame>
<frame src="frame3.html"></frame>
</frameset>
<noframes>
Use the link below to go to the frameless version of the
site.<br />
<a href="noframes.html">Frameless Site</a>
```

```
</noframes>
</html>
```

مثال ۴-۱۴:

در زیر نحوه تقسیم کردن صفحه به فریم های افقی و عمودی به طور همزمان را مشاهده می نمایید.

```
<frameset rows="100,*">
<frame src="frame1.html"></frame>
<frameset cols="150,*">
<frame src="frame2.html"></frame>
<frame src="frame3.html"></frame>
</frameset>
</frameset>
<noframes>
Use the link below to go to the frameless version of the
site.<br />
<a href="noframes.html">Frameless Site</a>
</noframes>
```

در مثال بالا ۱۰۰ پیکسل از کل فضای افقی صفحه به فریم ۱ اختصاص می یابد, سپس فضای باقی مانده صفحه نمایش برای نمایش فریم های عمودی استفاده می گردد.

دسترسی به فریم ها

آیا می دانید که چگونه می توان به فریم ها در جاوا اسکریپت دسترسی پیدا کرد؟ در ادامه این فصل به صورت مفصل به این امر خواهیم پرداخت.

The frames Array

شما می توانید از آرایه فریم برای دسترسی به فریم ها استفاده نمایید و آن هم با ارسال پارامترهای عددی که شروع آنها از صفر است.

مثال ۵-۱۴:

```
<html>
<frameset cols="60%,40%">
<frame src="frame1.html"></frame>
<frame src="frame2.html"></frame>
</frameset>
<noframes>
Use the link below to go to the frameless version of the
site.<br />
<a href="noframes.html">Frameless Site</a>
</noframes>
</html>
```

برای تکمیل شدن این مثال شما باید کد زیر را در frame2.html نمایید.

```
<body>
The first frame is from: <br />
<script type="text/javascript">
document.write(top.frames[0].location);
</script>
</body>
```

حال شما با باز کردن پنجره اصلی نتیجه را مشاهده می نمایید که در آن فریم سمت راست مکان ذخیره سازی فریم سمت چپ را می نویسد.

مثال ۶-۱۴:

```
<html>
<frameset cols="20%,80%">
<frame src="frame1.html"></frame>
<frame src="frame2.html"></frame>
</frameset>
<noframes>
Use the link below to go to the frameless version of the
site.<br />
<a href="noframes.html">Frameless Site</a>
</noframes>
</html>
```

کد زیر را در frame1 ذخیره نمایید.

```
<html>
<body>
I am frame 1!
</body>
</html>
```

کد زیر را در frame2 ذخیره نمایید.

```
<body>
<script type="text/javascript">
for (var count=0; count<top.frames.length; count+=1) {
var framenum = count+1;
document.write("Frame "+framenum+" is from
"+top.frames[count].location);
document.write("<br />");
}
</script>
</body>
```

حال با باز کزدن پنجره اصلی در فریم ۱ متن زیر به نمایش در می آید:

```
I am frame 1!
```

و در فریم ۲ مکان ذخیره سازی فریم های ۱ و ۲.

استفاده از نام فریم ها

یک راه دیگر برای دسترسی به فریم ها استفاده از نام آنها می باشد.به طور مثال در زیر دو فریم با نام های left_side و right_side مشاهده می کنید. که ما با استفاده از نام آنها به آنها دسترسی پیدا کرده و مکانشان را نمایش می دهیم.

مثال ۷-۱۴:

سه صفحه جداگانه ایجاد نمایید و در هر کدام از صفخات کدهای زیر را وارد کنید:

main farme

```
<html>
<frameset cols="50%,50%">
<frame src="frame1.html" name="left_side"></frame>
<frame src="frame2.html" name="right_side"></frame>
</frameset>
<noframes>
Use the link below to go to the frameless version of the
site.<br />
<a href="noframes.html">Frameless Site</a>
</noframes>
</html>
```

frame1.html

```
<body>
The second (right) frame is from: <br />
<script type="text/javascript">
document.write(top.right_side.location);
</script>
</body>
```

frame2.html

```
<body>
The first (left) frame is from: <br />
<script type="text/javascript">
document.write(top.left_side.location);
</script>
</body>
```

عوض کردن فریم تکی

برای عوض کردن یک فریم در HTML به طور مثال از کد زیر استفاده می نماییم.

```
<a href= "nextpage.html" target="right_side">Next Page</a>
```

حال اگر بخواهیم این عمل را با استفاده از جاوااسکریپت انجام دهیم باید از طریق کدهای زیر به آنها دسترسی پیدا کرده سپس آنها را عوض کنیم.

مثال ۸-۱۴:

فرض کنید یک صفحه جدید به اسم nextpage ایجاد کرده ایم و می خواهیم آن را جایگزین right_side در مثال قبل نماییم برای این عمل به صورت زیر عمل می کنیم.

```
top.right_side.location="nextpage.html";
```

حال برای تکمیل عمل می توانیم از کد زیر کمک بگیریم:

```
<a href="n" onclick= "top.right_side.location='newpage.html';
return false;">
New Page</a>
```

و یا

```
<a href="javascript:top.right_side.location='newpage.html'">
New Page</a>
```

عوض کردن فریم های چندگانه

برای عوض بیش از یک فریم در زمان شما باید از جاوا اسکریپت استفاده نمایید.

مثال ۹-۱۴:

ابتدا یک فایل با نام html ایجاد نمایید سپس کد زیر را در آن وارد کرده و آن را با نام frameset4.html ذخیره نمایید.

```
<html>
<frameset cols="20%,80%">
<frame src="frame1.html" name="left_side"></frame>
<frame src="frame2.html" name="right_side"></frame>
```

```
</frameset>
<noframes>
Use the link below to go to the frameless version of the
site.<br />
<a href="noframes.html">Frameless Site</a>
</noframes>
</html>
```

حال چهار فایل html دیگر با نام های frame1.html تا frame4.html ایجاد و پس از وارد کردن کدهای زیر آنها را ذخیره نمایید.

frame1.html

```
<body>
<script type="text/javascript">
function twoframes() {
top.right_side.location="frame4.html";
self.location="frame3.html";
}
</script>
<a href="javascript:twoframes();">Change Both Frames</a>
</body>
```

frame2.html

```
<body>
I am frame 2!
</body>
```

frame3.html

```
<body>
I am frame 3!
</body>
```

frame4.html

```
<body>
```

```
I am frame 4!
</body>
```

حال فایل frameset4.html را باز نمایید مشاهده می نمایید که با کلیک بر روی Change Both Frames فریم های ۱ و ۲ عوض می شوند و فریم های ۳ و ۴ جایگزین آنها می شوند.

نکته : توجه نمایید که تمام فایل را باید در یک پوشه ذخیره نمایید.

ایجاد یک جعبه انتخاب با فریم

اگر شما بخواهید که یک جعبه انتخاب با استفاده از فریم ها ایجاد نمایید در مورد select box navigation در جاوا اسکریپت اطلاعاتی داشته باشید.

برای آشنایی با این مبحث مثال زیر را مطرح می نماییم.

مثال ۱۰-۱۴:

می خواهیم یک پنجره شامل دو فریم ایجاد نماییم بطوریکه در فریم بالا یک جعبه انتخاب وجود داشته باشد که با انتخاب گزینه های آن اعمال مناسبی در فریم زیرین صورت پذیرد.از جمله عوض کردن فریم زیرین با فریم های دیگر.

همانند مثال قبل ابتدا فایل زیر را ایجاد و ذخیره نمایید.

frameset5.html

```
<html>
<frameset rows="120,*">
<frame src="frame1.html" name="t_frame"></frame>
<frame src="frame2.html" name="b_frame"></frame>
</frameset>
<noframes>
Use the link below to go to the frameless version of the
site.<br />
<a href="noframes.html">Frameless Site</a>
</noframes>
</html>
```

frame1.html

```
<body>
<form onsubmit="return go_there();">
<label for="s1">Change the lower frame:</label>
<select id="s1">
<option selected="selected" value="#">Choose
Destination</option>
<option value="frame3.html">Frame 3</option>
<option value="frame4.html">Frame 4</option>
<option value="frame2.html">Back to Frame 2</option>
</select>
<input type="submit" id="submit" value="Go!" />
</form>
<script type="text/javascript">
function go_there() {
s = document.getElementById("s1");
top.b_frame.location = s.options[s.selectedIndex].value;
return false;
}
</script>
</body>
```

frame2.html

```
<body>
I am frame 2!
</body>
```

frame3.html

```
<body>
I am frame 3!
</body>
```

frame4.html

```
<body>
I am frame 4!
</body>
```

حال با اجرای frameset5

یک جعبه انتخاب در فریم ۱ به نمایش در می آید که با انتخاب گزینه های آن و فشردن دکمه Go! تغییرات در فریم ۲ ایجاد می شوند.

متغیرهای سراسری در فریم ها

یکی از موارد مفید استفاده از فریم اینست که شما می توانید متغیری را در یک فریم ذخیره کرده و در دیگر فریم ها به آن دسترسی پیدا نمایید.برای مثال شما می توانید اطلاعات یک فرم را در یک فریم ذخیره کرده و سپس آن را در فریم دیگری در هنگام لود شدن بارگزاری نمایید.

مثال ۱۱-۱۴:

اسناد زیر را مرحله به مرحله ایجاد کرده و در یک مکان ذخیره نمایید.

frameset6.html

```
<html>
<frameset cols="150,*">
<frame src="frame1.html" name="left_side"></frame>
<frame src="frame2.html" name="right_side"></frame>
</frameset>
<noframes>
Use the link below to go to the frameless version of the
site.<br />
<a href="noframes.html">Frameless Site</a>
</noframes>
</html>
```

این فریم ست ۱۵۰ پیکسل برای فریم سمت چپ و بقیه اندازه صفحه نمایشگر را به فریم سمت راست اختصاص می دهد.

frame1.html

```
<head>
<script type="text/javascript">
var thename="";
var thefood="";
</script>
</head>
<body>
This is frame1.html; it holds the variable values.
You can put any content you like here.
</body>
```

در این فریم که در سمت چپ قرار می گیرد متغییرهایی بدون مقدار ایجاد و ذخیره می گردند.

frame2.html

```
<body>
I'd like to get your name. Please enter it below.
<br />
<form onsubmit="return store_info();">
<label for="yourname">Your Name:</label>
<input type="text" id="yourname" size="25" />
<br /><br />
<input type="submit" value="Continue" />
</form>
<script type="text/javascript">
function store_info() {
var yn = document.getElementById("yourname").value;
top.left_side.thename= yn;
self.location="frame3.html";
return false;
}
</script>
</body>
```

در این فریم از شما درخواست می شود تا یک نام وارد نمایید, این نام در فریم سمت چپ و در متغیر thename ذخیره می شود.

تابع store_info() زمانی صداد زده می شود که کاربر روی دکمه کلیک نماید و با استفاده از top.left_side به فریم سمت چپ دسترسی پیدا می شود.و با استفاده از top.left_side.thename به متغیر thename نسبت داده می شود.

frame3.html

```
<body>
<script type="text/javascript">
document.write("Hi, "+top.left_side.thename+"!<br />");
</script>
Now I'd like to get your favorite food. Please enter it below:
<br />
<form onsubmit="return more_info();">
<label for="yourname">Your Name:</label>
<input type="text" id="yourname" size="25" />
<br />
<label for="yourfood">Favorite Food:</label>
<input type="text" id="yourfood" size="25" />
<br /><br />
<input type="submit" value="Continue" />
</form>
<script type="text/javascript">
var tn = document.getElementById("yourname");
tn.value = top.left_side.thename;
function more_info() {
var fd = document.getElementById("yourfood");
top.left_side.thefood= fd.value;
self.location="frame4.html";
return false;
}
</script>
</body>
```

بعد از وارد کردن نام و زدن دکمه ادامه اینبار نوبت فریم سوم تا به نمایش در آید و نام شما را نشان داده و از شما درخواست وارد کردن غذای مورد علاقه تان را می نماید.

frame4.html

```
<body>
<script type="text/javascript">
function print_info() {
```

```
document.write("Thank you, "+top.left_side.thename+"!");
document.write("<br /><br />");
document.write("You must really like
"+top.left_side.thefood+"!");
}
print_info();
</script>
</body>
```

در انتها و زمان که شما دکمه ادامه را در فریم سوم فشار می دهید مقادیر متغیرها به فریم چهارم منتقل شده و در آنجا برای شما به نمایش در می آیند.

```
<body>
<script type="text/javascript">
function print_info() {
document.write("Thank you, "+top.left_side.thename+"!");
document.write("<br /><br />");
document.write("You must really like
"+top.left_side.thefood+"!");
}
print_info();
</script>
</body>
```

سخن پایانی

در این آموزش فرض بر این گذاشته شد که شما می دانید که چگونه جاوااسکریپت را برای پویاتر شدن و جذاب تر شدن وب سایتتان به صفحات HTML خود اضافه کنید. شما یاد گرفتید چگونه به متدها پاسخ دهید، چگونه فرم ها را اعتبارسنجی کنید و چگونه اسکریپت های مختلفی برای پاسخ به سناریوهای مختلف بسازید.

این آموزش ها آموزش برنامه نویسی به زبان جاوااسکریپت بود و نه آموزش ترفندهای جاوااسکریپت و بیشتر مناسب کسانی است که می خواهند این زبان را به عنوان یک زبان برنامه نویسی سمت کلاینت فرابگیرند و نه کسانی که بخواهند قالب وبلاگ خود را با کدهای جاوااسکریپت زیبا تر کنند. البته با این آموزش ها و تلفیق آن ها به سادگی تمام این کارهایی که به عنوان کدهای جاوااسکریپت برای وبلاگ ها ارائه می شود را در صورتی که کمی با علم برنامه نویسی آشنا باشید انجام دهید.

خوب حالا از این به بعد چه کار باید کرد؟

گام بعدی یادگیری HTML DOM ، jQuery و AJAX است. اگر می خواهید یک زبان سمت سرور را فرابگیرید PHP و ASP.NET گزینه های پیش روی شماست

HTML DOM

HTML DOM استانداردی برای دسترسی و کار کردن و ایجاد تغییر روی عناصر HTML است.
یک پلتفرم و مستقل از زبان برنامه نویسی است که شما می توانید در زبان های برنامه نویسی مختلف اعم از Java، JavaScript، VBScript و ... از آن استفاده کنید.

حتماً این پلتفرم را بیاموزید زیرا که بسیار به جاوااسکریپت و جاوااسکریپت به آن وابسته است (خصوصاً از سمت جاوااسکریپت) شاید بعد از جی کئری و یا شاید همزمان با هم آموزش این بخش را هم شروع کنیم.

jQuery

یک کتابخانه جاوااسکریپت است.

برنامه نویسی با جاوااسکریپت را بسیار بسیار ساده می کند.

AJAX

AJAX = Asynchronous JavaScript and XML جاوااسکریپت و XML ناهمگام

یک زبان برنامه نویسی نیست ولی راهی جدید برای استفاده از استاندارهای موجود است.

درباره جابجا کردن داده ها با یک سرور و به روزرسانی بخشی از صفحه بدون بارگزاری مجدد صفحه است.

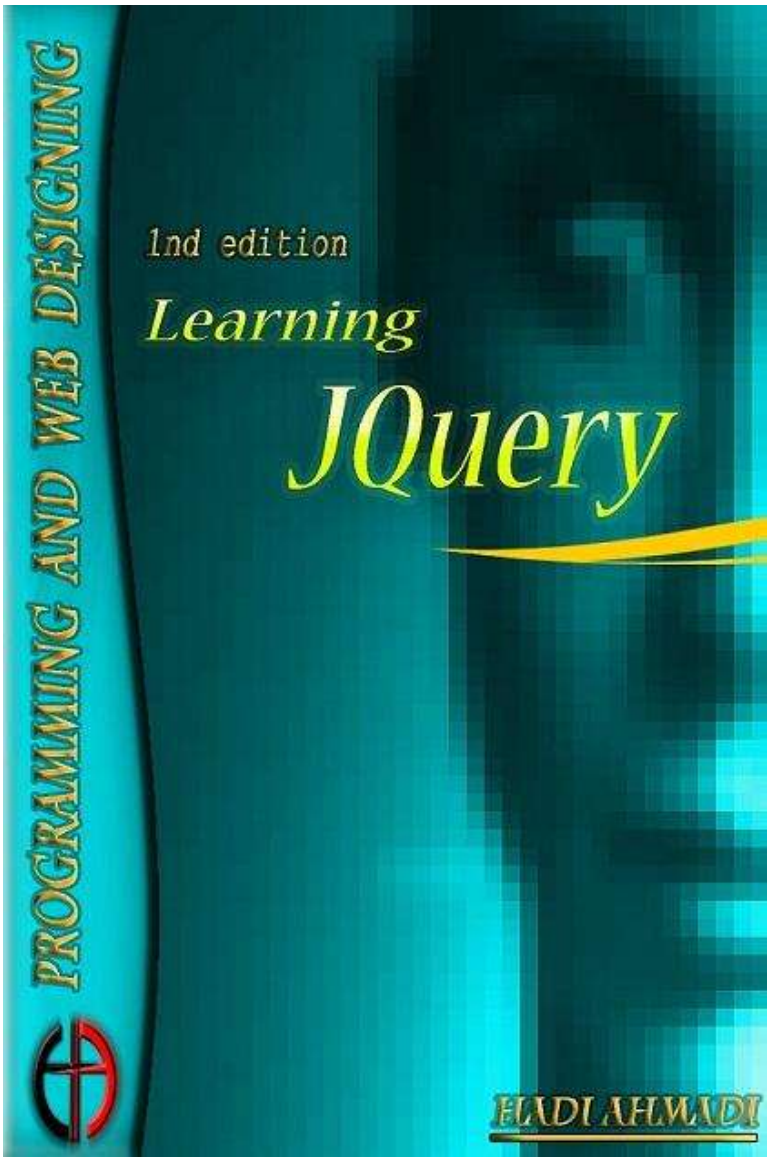
مثال هایی از آژاکس مثال گوگل مپ، جیمیل، یوتیوب و تب های فیس بوک است.

ASP.NET / PHP

همانطور که یک فایل HTML در سمت کلاین (در داخل مرورگر) اجرا می شود یک فایل ASP.NET/PHP در سمت سرور اجرا می شود.

با استفاده از این زبان ها شما می توانید به صورت پویا هر محتوایی را به وب سایتتان اضافه، از آن کم و یا آن را ویرایش کنید. می توانید از دیتابیس ها برای ذخیره اطلاعات استفاده نمایید.

به زودی...


PROGRAMMING AND WEB DESIGNING
1nd edition
Learning
JQuery
HADI AHMADI