

Java Script

تهریه کننده: شرکت داده کاو ۲۱

سال تدوین: ۱۳۹۸

نسخه: دوم

بخش اول(مقدمه ای بر تولید صفحات کلاینت)

- صفحه وب

به مجموعه ای از تصاویر و نوشته ها که به صورت یک صفحه از طریق اینترنت قابل دسترسی می باشند ، صفحه وب می گویند.

صفحات وب بروی شبکه جهانی وب ساخته می شوند. این اسناد با html نوشته و به وسیله ی مرورگرهای شما ترجمه می شوند.

صفحات وب می توانند هم به صورت ایستا و پویا باشند. صفحات ایستا مقادیر ثابت و یکسانی را در هر بار رویت کاربران نمایش می دهد و تغییری در مقادیر آن صفحات مشاهده نخواهد شد، اما صفحات پویا مقادیری دارند که میتوانند در هر زمان تغییر کنند و کاربران به آنها دسترسی پیدا خواهند کرد. این صفحات معمولاً با زبانهای اسکریپتی (متنی) مثل php, perl, asp, jsp نوشته می شوند.

اسکریپت ها در صفحات، توابعی روی سرور اجرامی کند که چیز هایی شبیه به اطلاعات تاریخ و زمان و پایگاه داده را برگشت می دهد.

همه اطلاعات به صورت کد HTML برگشت داده شده است در زمانیکه مرورگر شما صفحه رو دریافت کرده باشد.

همه مرورگرها اطلاعات دریافتی رابه HTML ترجمه می کنند.

توجه داشته باشید که صفحه وب چیزی شبیه به وب سایت نیست. یک وب سایت مجموعه ای از صفحات است.

- چگونگی تولید یک صفحه وب

صفحه وب با کدهای HTML و در اسنادی با پسوند html. تولید و بروی مرورگرهای وب ترجمه و نمایش داده می شوند.

- چگونگی به نمایش قراردادن یک صفحه وب

یک صفحه وب برای نمایش عمومی دربستر اینترنت و در شبکه جهانی وب باید در مسیر زیر قرار بگیرد.

<http://www.yourdomainname.yoursuffixdomain>

HTTP: مخفف شده عبارت Hypertext Transfer Protocol به معنی پروتکل انتقال ابر متنی می باشد. تمامی مسیرها باید از این پروتکل تبعیت کنند تا بتوانند صفحه و طریقه دسترسی به صفحات وب خود به نمایش بگذارند.

WWW: مخفف عبارت World Wide Web شبکه جهانی وب پایگاه قرارگیری و دسترسی به صفحات وب می باشد.

Domain: مجموعه ای از نام دامنه و پسوند آن میباشد مثل ir و یا dadekav21.com و....dadekav21.com

با نوشتن آدرس مربوط در قسمت URL مرورگر و اتصال به شبکه اینترنت جهانی می توانید صفحه وب مورد نظر خود را مشاهده نمایید.

-معرفی مرورگرهای وب

مرورگر ها در حال حاضر بر سر دو ویژگی در حال رقابت هستند: ۱. امنیت ۲. سرعت

مرورگرهای پراستفاده Firefox, IE, Google Chrome, Opera و....

HTML_

Hypertext Markup Language به معنی زبان نشانه گذاری فرamtن میباشد که صفحات وب مبتنی بر این زبان هستند و اسناد آن دارای پسوند .html می باشند.

ساختار کلی یک سند Html :

```
<!Doctype html>  
<html>  
  <head>  
    <title>MY Web Page </title>  
  </head>  
  <body></body>  
</html>
```

تگ های مهم

```
<meta charset="UTF-8">  
<meta name="description" content="my web site">  
<meta name="keywords" content="HTML,CSS,XML,JavaScript">  
<meta name="author" content="Nariman Abyar">  
<meta name="viewport" content="width=device-width,initial-Scale=1.0">  
<style></style>  
<script></script>  
<link/>
```

تگ های مهم ساختاری:

```
<header>  
<div>  
<article>  
<section>  
<aside>  
<nav>  
<footer>
```

دیگر تگ های مهم پر کاربرد:

```
<p>  
<h1...6>  
<span>  
<i>  
<a>  
<table>  
<form>  
<ul>  
<img>
```

CSS-

به معنی صفحات سبک آبشاری به عبارت دیگر صفحاتی به صورت استایل دهی آبشاری حالت طراحی و گرافیکی به خود می گیرند.

سه روش استایل دهی داریم که عبارت است از:

Inline: text

Internal:

```
<style>
```

```
a{  
    color:red;  
}  
</style>
```

نکته مهم این است که فایل ها خارجی استایل دهی با پسوند CSS. می باشند که از طریق تگ link در قسمت head به سند html مورد نظر اتصال پیدا می کند.

```
<link rel="stylesheet" href="style.css"/>
```

استایل مهم و پر کاربرد:

Width,max-width,min-width

Height,max-height,min-height

Background,background-color,-repeat,-size

attachment,position

Text-align,text-shadow,font-family,font-size,color

@font-face, Box-shadow, Postion, z-index , left , right , top

Bottom , float , Display , Visibility , Animation , Transition

Transform , Prespective , @keyframes , Margin , padding

–تولید صفحه وب با کمک HTML/CSS

باتوجه به دوره گذشته و مفاهیم مطرح شده و پادآوری دوباره در آیتم قبل و به کمک استاد مربوطه طرح زیر با کدهای HTML و CSS پیاده سازی کنید.

<p style="text-align: center;">لوگو</p> <p>شرکت داده کاو ۲۱ برنامه نویسی</p> <p style="text-align: center;">in Y T</p> <p style="text-align: center;">درباره ما ارتباط با ما نمومه کار فروشگاه خانه</p>	<p style="text-align: center;">لوگو</p> <p>شرکت داده کاو ۲۱ برنامه نویسی</p> <p style="text-align: center;">in Y T</p>	<p style="text-align: center;">لوگو</p> <p>شرکت داده کاو ۲۱ برنامه نویسی</p> <p style="text-align: center;">in Y T</p>										
<p style="text-align: center;">عکس</p> <p style="text-align: center;">عکس</p>	<p style="text-align: center;">متن</p>	<p style="text-align: center;">عکس</p> <p style="text-align: center;">متن</p>										
<p style="text-align: center;">عکس</p> <p style="text-align: center;">متن</p> <p style="text-align: center;">ادامه مطلب</p>	<p style="text-align: center;">عکس</p> <p style="text-align: center;">متن</p> <p style="text-align: center;">ادامه مطلب</p>	<p style="text-align: center;">عکس</p> <p style="text-align: center;">متن</p> <p style="text-align: center;">ادامه مطلب</p>										
<p style="text-align: center;">ارتباط با ما</p> <table border="1"><tr><td style="width: 50%;">پیام شما :</td><td style="width: 50%;">نام :</td></tr><tr><td></td><td></td></tr><tr><td></td><td>ایمیل :</td></tr><tr><td></td><td>تلفن :</td></tr><tr><td colspan="2" style="text-align: center;">ارسال</td></tr></table>			پیام شما :	نام :				ایمیل :		تلفن :	ارسال	
پیام شما :	نام :											
	ایمیل :											
	تلفن :											
ارسال												
<p>شبکه های اجتماعی</p> <p style="text-align: center;">in Y T</p>	<p>آدرس : تلفن : ایمیل :</p>											
طراحی توسط شرکت داده کاو ۲۱												

- ساختارهای موجود در تولید صفحات وب

بوت استرپ (Bootstrap) بهترین و پراستفاده ترین ساختار در بین طراحان وب می باشد و این ساختار برای توسعه دهنده‌گان فرانت-اند طراحی شده است.

این ساختار به ساختن طراحی مفهومی، برای نمایش در موبایل، شبکه بندی صفحه، تایپوگرافی و چیزهای مهم دیگر کمک کننده بسیار خوبی می‌باشد.

فاندیشن (Foundation): یکی از قدرتمندترین ساختارهای سورس باز در CSS می باشد. این ساختار برای انجام اعمال واکنش گرا بسیار هدفمند می‌باشد.

بلما (Bulma): این ساختار سورس باز می باشد که انرژی توسعه دهنده‌گان طراحی وب را در تولید صفحات ذخیره می‌کند به عبارت دیگر در ساخت صفحات وب انرژی و زمان کمتری را از توسعه دهنده‌گان طراحی وب خواهد گرفت. این ساختار دارای کامپوننت‌های رابط کاربری بسیار زیاد و عالی می باشد و به شکل مازول و دسته بندی شده قابل اضافه شدن به صفحات وب می باشد مثل اضافه کردن مازول نوار ابزارها و...

ساختارهای متنوع و جذاب دیگری مثل UI Ulkit, semantic وجود دارد که به عنوان تحقیق به عهده شما هنرآموزان قرار می‌دهیم.

- تولید یک صفحه وب با استفاده از ساختار Bootstrap

بوت استرپ مجموعه‌ای از استایل‌ها و تگ‌ها و کامپوننت‌ها و کتابخانه و توابع جاوااسکریپتی می‌باشد و به توسعه دهنده‌گان این امکان را می‌دهد تا صفحات کاملاً پویا و واکنش گرا و بسیار جذابی را تولید نمایند.

بوت استرپ رامی توان از آدرس زیر دریافت نمود:

<http://getbootstrap.com/>

پس از دانلود فایل مربوط و قرار دادن آن در پروژه مورد نظر می‌ریم به سراغ نحوه اضافه کردن فایل‌ها به صفحه وب مورد نظر:

```
<link rel="stylesheet" href="/public/css/bootstrap.min.css"/>
```

```
<script src="/public/jquery.min.js"></script>
```

توجه داشته باشید که باید فایل جیکوئری را در سایت jquery.com دانلود نمایید سپس به پروژه اضافه کرده و در صفحه وب قرار دهید.

```
<script src="~/public/js/bootstrap.min.js"></script>
```

مهمترین بحث در این جزو در بخش بوت استرپ بحث شبکه بندی میباشد:

```
<div class="container">  
  <div class="row">  
    <div class="col-sm"> one of three columns </div>  
    <div class="col-sm"> One of three columns</div>  
    <div class="col-sm"> One of three columns </div>  
  </div>  
</div>
```

Col ها بسیار مهم هستند در شبکه بندی کردن درون یک صفحه اگر دقت کرده باشد در هر row چندین Col قرار میگیرد که میپردازیم به توضیحاتی چند در مورد اینها:

هر row شامل ۱۲ col می باشد که میتوان شبکه بندی کرد مثلا:

```
<div class="container">  
  <div class="row">  
    <div class="col">1 of 2 </div>  
    <div class="col">1 of 2</div>  
  </div>  
  <div class="row">  
    <div class="col">1 of 3 </div>  
    <div class="col">2 of 3 </div>  
    <div class="col">3 of 3 </div>  
  </div>  
</div>
```

برای اندازه صفحه کوچک Col-sm

برای اندازه صفحه معمولی Col-md

برای اندازه گیری صفحه بزرگ استفاده می شود. Col-lg

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				
Gutter width	30px (15px on each side of a column)				

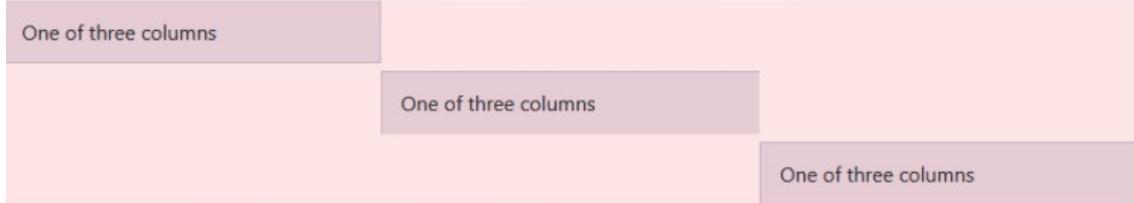
کدهای دیگر درمورد تقسیم بندی یک صفحه:

```
<div class="row">
  <div class="col-sm-8">col-sm-8</div>
  <div class="col-sm-4">col-sm-4</div>
</div>

<div class="row">
  <div class="col-sm">col-sm</div>
  <div class="col-sm">col-sm</div>
  <div class="col-sm">col-sm</div>
</div>
```

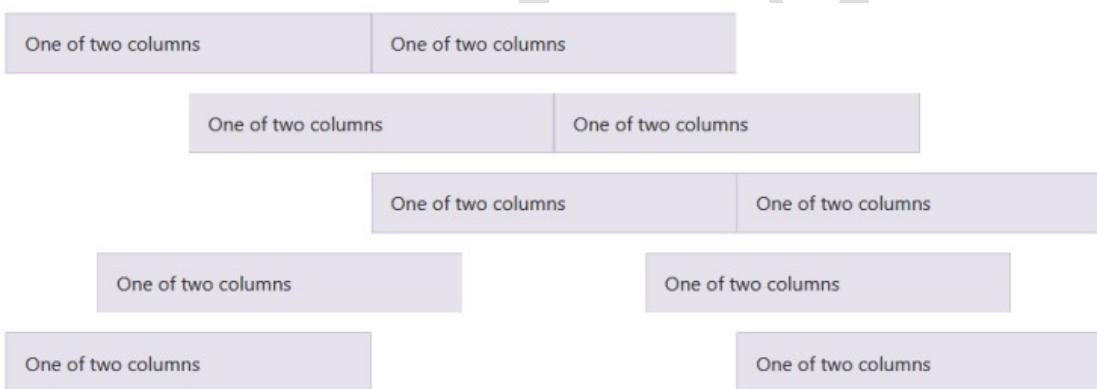


```
<div class="container">  
  
  <div class="row">  
  
    <div class="col align-self-start"> One of three columns </div>  
  
    <div class="col align-self-center"> One of three columns </div>  
  
    <div class="col align-self-end"> One of three columns </div>  
  
  </div>  
  
</div>
```



```
<div class="container">  
  
  <div class="row justify-content-start">  
  
    <div class="col-4"> One of two columns </div>  
  
    <div class="col-4"> One of two columns </div>  
  
  </div>  
  
  <div class="row justify-content-center">  
  
    <div class="col-4"> One of two columns </div>  
  
    <div class="col-4"> One of two columns </div>  
  
  </div>  
  
  <div class="row justify-content-end">  
  
    <div class="col-4"> One of two columns </div>  
  
    <div class="col-4"> One of two columns </div>  
  
  </div>
```

```
<div class="row justify-content-around">  
    <div class="col-4"> One of two columns </div>  
    <div class="col-4">One of two columns </div>  
</div>  
  
<div class="row justify-content-between">  
    <div class="col-4">One of two columns </div>  
    <div class="col-4">One of two columns </div>  
</div>  
</div>
```



به طور کلی دربحث طراحی صفحات وب ساختارها می توانند کمک کننده خوبی باشند برای تولید صفحات بهتر در زمان کمتر اما توصیه می شود توسعه دهندهان و کدرها در تولید صفحات وب از تکنیکها و تجربیات خودشان استفاده کنند و در کنار آن از ساختار مهیج و بسیار کاربردی در حوزه طراحی صفحات وب استفاده نمایند.

صفحه وب زیر را با ساختار بوت استرپ طراحی کنید:

-

لوگو

شرکت داده کاو ۲۱
برنامه نویسی

in Y T

درباره ما ارتباط با ما نموده کار فروشگاه خانه

عکس

عکس

متن

لوگو

شرکت داده کاو ۲۱
برنامه نویسی

in Y T

عکس

متن

ادامه مطلب

عکس

متن

ادامه مطلب

عکس

متن

ادامه مطلب

ارتباط با ما

پیام شما :	نام :
	ایمیل :
	تلفن :
ارسال	

شبکه های اجتماعی

in Y T

آدرس :
تلفن :
ایمیل :

طراحی توسط شرکت داده کاو ۲۱

یکی از فریم ورک های CSS است که از قوانین CSS استفاده می کند، LESS توسط روش هایی مانند متغیرها، حسابگرها، mixin ها و توابع، قابلیت های CSS را توسعه داده و آن را پویا ساخته است. به گونه ای که

می توان کدهای LESS را در یک فایل CSS نوشت. کامپایلر اصلی LESS با استفاده از جاوا اسکریپت نوشته شده است و کدهای نگارش یافته را به فرمت استاندارد CSS تبدیل می کند. نگارش استایل ها با دید ماژولار از قابلیت های این فریم ورک است، LESS خاصیت های پویایی را به CSS اضافه کرده است. به کمک امکانات متعدد آن

می توان از زبان CSS به عنوان یک زبان برنامه نویسی استفاده نمود. استفاده مجدد از مقادیر بالاستفاده از متغیرها، استفاده مجدد از بلاک ها بالاستفاده از mixins، استایل های مختصرتر با قوانین تودرتو، انجام محاسبات با استفاده از توابع و عملگرها از جمله کارهایی است که می توان با این فریم ورک انجام داد.

برنامه های زیادی برای کامپایل فایل less وجود داره اما ما از Node.js استفاده می کنیم.

نصب: LESS

از آنجایی که با جاوا اسکریپت نوشته شده است، روش های مختلفی برای نصب و استفاده از قابلیت های LESS وجود دارد. دو روش پرکاربرد برای نصب و استفاده از LESS وجود دارد.

۱- استفاده از فایل less.js

۲- استفاده از node.js

در روش اول شما به اضافه کردن فایل less.js به صورت HTML به سند realtime فایل هایی با پسوند less رابه CSS کامپایل می کنید. استفاده از این روش راحت است ولی اجرای بسیار کندی دارد به همین دلیل کمتر مورد استقبال قرار می گیرد.

در روش بعدی که آموزش مایه همین اساس پیش می رو داستفاده از less.js است. روند بسیار ساده ایست.

قدم اول: شما ابتدا آخرین نسخه node.js را دانلود کنید و مراحل نصب آن را انجام دهید.

<https://nodejs.org/>

قدم دوم: پس از نصب node.js در خط فرمان ترمینال سیستم عامل با استفاده از دستور npm به راحتی می‌توانید less را توسط دستور زیر نصب کنید.

Npm install -g less

حال برای استفاده از less شما باید فایل‌های html و css خود را در پوشه node.js ایجاد کنید.

برای مثال: یک فایل با نام style.css ایجاد کنید. سپس فایلی با همان نام و با پسورد less یعنی فایلی با نام style.less ایجاد می‌کنیم.

سپس کدهای less را طبق قواعد دستوری less در فایل style.less می‌نویسیم و در نهایت با دستور زیر آن را به فایل style.css کامپایل می‌کنیم.

Less style.less style.css

با استفاده از less شما به راحتی می‌توانید به ویژگی‌های یک کلاس یا id به صورت تودر تو نویسی دسترسی پیدا کنید و آنها را تغییر دهید.

مثلاً فرض کنید که ساختار زیر را در HTML تعریف کردیم:

```
<html>
  <head>
    <title>nested Rules</title>
    <link rel="stylesheet" type="text/css" href="style.css"/>
  </head>
  <body>
    <div class="parent">
      <h1>Less tutorial</h1>
      <p>this is first paragraph</p>
      <div class="class1">
        <h2>second heading</h2>
        <p>this is second paragraph </p>
      </div>
    </div>
  </body>
</html>
```

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

در مرحله بعد فایل style.less را اینگونه می نویسیم:

```
.parent {  
    h1 {  
        Font-size: 25px;  
        Color: #E45456;  
    }  
    P{  
        Font-size: 25px;  
        Color: #3C7949;  
    }  
    .class {  
        h2 {  
            Font-size: 25px;  
            Color: #E45456;  
        }  
        P {  
            Font-size: 25px;  
            Color: #3C7949;  
        }  
    }  
}
```

طبق آموزش قبلی فایل style.less کامپایل می کنیم

Lessc style.less style.css

اگر فایل style.css را باز کنیم متوجه می شویم که به صورت زیر نوشته شده است. با کمک less توانستیم به راحتی و با ساختاری منظم تر به کلاس class1 و دیگر فرزندانش دسترسی پیدا کنیم.

```
.parent h1 {  
    font-size: 25px;  
    color: #E45456;  
}  
  
.parent p {  
    font-size: 25px;  
    color: #3C7949;  
}  
  
.parent .class1 h2 {  
    font-size: 25px;  
    color: #E45456;  
}  
  
.parent .class1 p {  
    font-size: 25px;  
    color: #3C7949;  
}
```

قواعد تودرتو نویسی دایرکتیوها:

دایرکتیوهای LESS در CSS قابل تодرتو نویسی هستند. منظور از دایرکتیوها همان دستوراتی مانند @media، @keyframe، @font-face و ... می باشند فرض کنید برای کلاسی به نام mystyle دایرکتیو media را جهت تغییر رنگ متن در سایزهای مختلف مرورگر می نویسیم.

```
.mystyle {  
    @media screen {  
        color: red;  
        @media (min-width: 1024px) {  
            color: pink;  
        }  
    }  
    @media text {  
        color: green;  
    }  
}
```

طبق روال قبلی فایل style.less را با دستور زیر به فایل style.css کامپایل می کنیم

```
@media screen {  
    .mystyle {  
        Color: red;  
    }  
}  
  
@media screen and (min-width: 1024px) {  
    .myclass {  
        Color: pink;  
    }  
}  
  
@media text {  
    .myclass {  
        Color: green;  
    }  
}
```

عملگرها در LESS

از ویژگی های خوب less پشتیبانی از چهار عمل اصلی مانند ضرب(*)، جمع(+)، تفریق(-) و تقسیم می باشد. ازین عملگرهای اصلی در less بیشتر برای عملیات روی اعداد، رنگ ها و متغیرها استفاده می شود.

استفاده از این عملگرهای less کارت وسیعه CSS را بسیار راحت تر می کند. مثلا فرض کنید متغیری به نام myfont برای ذخیره مقداری پیش فرض برای اندازه سایز فونت تعریف شده است. برای تغییر سایز اندازه فونت در قسمت های مختلف محتوا کافی است با اضافه و کم کردن مقدار myfont تعیین کنیم:

```
<html>
  <head>
    <title>Less operations</title>
    <link rel="stylesheet" type="text/css" href="mystyle.css">
  </head>

  <body>
    <h1>عملیات اصلی در Less</h1>
    <p class="content">
      این یک مثال برای توضیح اجرای عملگرهای اصلی در less می باشد
    </p>
  </body>
</html>
```

در ادامه فایل less را برای این سند اینگونه مینویسیم:

Mystyle.less

```
@fontsize: 15px;
.content {
  font-size: @font-size + 5;
}
```

توسط دستور زیر فایل mystyle.less را به mystyle.css کامپایل می‌کنیم:

Lessc mystyle.less my style.css

اجرای کد بالا فایل mystyle.css را به صورت خودکار می‌سازد.

Mystyle.css

```
.content {  
    font-size: 20px;  
}
```

برگرداندن یک رشته در less یا escaping

توسط Less در escaping شما می‌توانید یک رشته دلخواه را به عنوان یک ویژگی یا مقدار یک متغیر استفاده کنید. شیوه کار نیز به این صورت است که رشته مورد نظر را بین دو کاراکتر " " یا " " نوشته و بعد یک علامت ~ قبل از آن قرار می‌دهیم. فرض کنید رشته مورد نظر ما عبارت "'this is tutorial of escaping in less'"

می‌باشد. پس به شیوه زیر عمل می‌کنیم:

~"this is tutorial of scaping in less"

یا

~'this is tutorial of scaping in less'

به مثال زیر در فایل mystyle.less اینگونه عمل می‌کنیم:

توسط دستور زیر فایل mystyle.css را به mystyle.less کامپایل می‌کنیم.

Lessc my style.less mystyle.css

اجرای کد بالا فایل mystyle.css را به صورت خودکار می‌سازد.

Mystyle.css

```
.myElement {
```

Content: ^/ / * this is tutorial of scaping in less;

درمثالی دیگرفرض کنید قصد داریم مقدار خصوصیت color را برای تگ div برابر red قراردهیم. پس فایل رابه شکل زیر می سازیم mystyle.less

Mystyle.less

```
Div {  
Color: ~"red";  
}
```

توسط دستور زیر فایل mystyle.less را به mystyle.css کامپایل می کنیم.

Less mystyle.less mystyle.css

Mystyle.css

```
Div {  
Color: red;  
}
```

LESS توابع

یکی از قابلیت های مهم در LESS استفاده از توابع می باشد که این توابع به صورت پیش فرض در LESS موجود می باشند با استفاده از این توابع شما می توانید به راحتی رشته ها ورنگ ها را تغییر دهید و محاسبات ریاضی را نیز انجام دهید.

لیست توابع موجود در less در وبسایت توسعه دهنده LESS (<http://lesscss.org/functions>) موجود می باشد.

```
<html>
  <head>
    <title>tutorial Less Functions</title>
    <link rel="stylesheet" type="text/css" href="style.css"/>
  </head>
  <body>
    <h1>Example using Functions in Less</h1>
    <p class="my class">
      Less let you use Functions
    </p>
  </body>
</html>
```

در Less و width راجه تعریف رنگ ها و اندازه عرض عنصر تعریف می کنیم و مقادیر آنها رانیز معین می کنیم.

سپس از تابع percentage جهت تغییر اندازه به درصد، واز تابع saturate جهت خلوص یا اشباع رنگ، واز تابع lighten جهت روشن کردن رنگ واز تابع spin جهت چرخاندن رنگ استفاده می کنیم.

حال فایل style.less رامی سازیم:

```
@color: #f04615;  
@width: 0.5;  
}  
  
.mystyle {  
    width: percentage (0.5);  
    color: saturate (@color, 5%);  
    background-color: spin(lighten(@color, 25%), 8);}
```

حال توسط دستور زیر فایل style.less را به فایل style.css کامپایل می کنیم:

Less style.less style.css

با اجرای کد بالا فایل style.css به شکل زیر ساخته می شود.

```
.myclass {  
    width: 50%;  
    color: #f6430f;  
    background-color: #f8b38d;  
}
```

فضای نام یا namespace چیست؟

قبل از اینکه به آموزش نحوه استفاده از less در namespace بپردازیم به دلیل اینکه شما نمی توانید دو کلاس هم نام را در برنامه داشته باشید توسط namespace ها می توانید کلاس ها را گروه بندی کنید.

فضای نام یا Less در namespace

فضای نام در less به منظور گروه بندی ترکیب ها یا mixin ها تحت عنوان یک نام استفاده می شود . توسط namespace در Less شما می توانید گروهی از mixin ها را از خارج برنامه در داخل برنامه استفاده کنید بدون اینکه تداخل نامی پیش بیايد.

مثال: درمثال زیر نحوه استفاده از فضای نام در Less را توضیح می دهیم.

```
<html>
  <head>
    <title>use Namespaces in Less</title>
    <link rel="stylesheet" type="text/css" href="style.css"/>
  </head>
  <body>
    <h1>How using Namespace in Less</h1>
    <p class="style1">Less is a Css pre-processor,meaning that it extends the Css language</p>
    <p class="style2">Less is a Css pre-processor,meaning that it extends the CSS language</p>
  </body>
</html>
```

حالا فایل style.less را به گونه زیر می سازیم:

```
.myclass1 {  
    .myclass2 {  
  
        Valual(@param) {  
            Background-color: @param;  
            Color:green;  
        }  
  
        .value2(@param) {  
            Background-color: @param;  
            Color:green;  
        }  
    }  
}
```

بسیار خوب در مثال بالا مشاهده می کنید که یک mixin به نام myclass1 موجود است و درون آن دیگری به نام myclass2 نیز قرار دارد. درون myclass2 نیز دو بسته به نام value1 و value2 قرار دارند که هر کدام یک پارامتر که تعیین کننده رنگ زمینه یعنی background-color می باشند می پذیرند. حال پارامترهای هر کدام از value1 و value2 را مقداردهی می کنیم و هر کدام را درون کلاس‌های جدا به نام های style1 و style2 می‌ریزیم. حال برای دسترسی به هر یک از این بسته‌ها به صورت سلسله مراتبی واژ علامت > استفاده می‌کنیم. به صورت زیر:

```
.style1 {  
    Myclass1 > .myclass2 > .value1 (red);  
}  
  
.style2 {  
    Myclass1 > .myclass2 > .value2 (pink);  
}
```

حال فایل style.css را به style.less کامپایل می کنیم.

Lessc style.less style.css

فایل style.less به صورت زیر کامپایل می شود:

```
.style1 {  
    background-color:red;  
    color:green;  
}  
  
.Style2 {  
    background-color: pink;  
    color:green;  
}
```

میدان دید یا Less scope در Less

تعريف میدان دید در less بسیار شبیه به دیگر زبان های برنامه نویسی دیگر است. در زبان های برنامه نویسی میدان دید متغیر فقط در حوزه ای اعتبار دارد که در آن تعریف شده است. برای مثال اگر متغیری را داخل یک تابع تعریف کنید، مقدار متغیر فقط داخل همان تابع اعتبار دارد و خارج از آن تعریف نشده است و به اصطلاح می گویند میدان دید ندارد. میدان دید یا scope در less نیز به همین صورت است. متغیرها یا mixin ها ابتدا مقدار خود را در محل خود جستجو می کنند اگر پیدا نشد والد خود را جستجو می کنند و این روند همین طور ادامه پیدا خواهد کرد. به یک مثال ساده توجه کنید:

```
@color:green;  
  
#mypage {  
    @color:red;  
  
    #my header {  
        color:@color; // red  
    }  
}
```

درمثال بالا مقدار متغیر green ابتدا color تعریف شده است. ولی این متغیر دروالد myHeader که همان است red تعریف شده است. پس مقدار آن همان red می شود.

اگر کدهای بالا را به CSS کامپایل کنیم نتیجه به شکل زیر می شود:

```
#mypage #myheader {  
    color: red;  
}
```

توجه داشته باشید که متغیر ها یا mixin ها را قبل از تعریف فراخوانی نکنید. مثال زیر دقیقا همان نتیجه بالا رامی دهد.

```
@color: green;  
  
#mypage {  
    #myheader {  
        color: @color; // red  
    }  
}
```

```
@color: red;  
}
```

```
#mypage #myheader {  
    color: red;  
}
```

کدهای بالا پس از کامپایل به CSS به این شکل می شود.

درج توضیحات در Less

همانطور که گفته شد توضیحات به دو روش توضیحات خطی و بلاک توضیحات استفاده می شوند.

توضیحات خطی: در این روش توضیحات فقط در یک خط نوشته می شوند و در شروع توضیحات کافیست دو علامت // قرار دهید

```
/* This is inline comment
```

بلاک توضیحات: این نوع توضیحات در دویا چند خط نوشته می شوند و در شروع توضیحات یک علامت /* و در پایان توضیحات یک علامت */ قرار می دهید.

```
/*this is a block comment
```

```
You can write the comments
```

```
In some line*/
```

در مثال زیر در فایل style.less هر دو نوع توضیح را به کار می بیریم.

```
/*it displays the
```

```
green red!*/
```

```
.class1 {
```

```
    Color: red;
```

```
}
```

```
// it displays the white color
```

```
.mystyle1 {
```

```
    Color:white;
```

```
}
```

حال فایل style.less را به style.css کامپایل می کنیم:

```
/*it display the  
green red!*/  
.class1 {  
    Color:red;  
}  
.myclass1 { Color: white;}
```

وارد کردن یا اضافه کردن محتویات یک فایل به فایل دیگر

یکی از قابلیت های بسیار خوب در less این است که میتوانید کل محتویات یک فایل less را در یک فایل less دیگر باز کنید و به آن سند اضافه کنید. برای روشن شدن مطلب به مثال زیر توجه کنید.

فرض کنید یک سند html به شکل زیر داریم که ۳ پاراگراف یا تگ <p> دارای متن دارند.

```
<html>  
  <head>  
    Link rel="stylesheet" type="text/css" href="style.css"/>  
  </head>  
  <body>  
    <h1>Less import Example</h1>  
    <p class="class1">importing works pretty much as expected. </p>  
    <p class="class2">when using an @import(reference),mixins that contain a&  
      Selector get added to the compiled output improperly </p>  
    <p class="class3">ilikephp.ir has many diffrent tutorial programming Language </p>  
  </body>  
</html>
```

در سند بالا ۳ تگ <p> با کلاس های class1 ,class2 ,class3 وجود دارد.فرض کنیم دو کلاس class1 و class2 داخل فایلی به نام myless.less قرار دارند.

myLess.less

```
.class1 {  
    color: red;  
    font-size: 20px;}  
  
.class2 {  
    color: blue;  
    font-size: 30px;}
```

فایل دیگری به نام style.less نیز به شکل زیر وجود دارد.

Style. Less

```
.class3 {  
    color: green;  
    font-size: 40px;}
```

حال برای اضافه کردن یا وارد کردن فایل mystyle.less به فایل style.less به شکل زیر عمل می کنیم.

```
.class3 {  
    color: green;  
    font-size: 40px;}
```

حال فایل style.css را به style.less کامپایل می کنیم.

Less style.less style.css

خروجی به شکل زیر می شود. وقت کنید که فایل mystyle.less وارد (import) فایل style.less شده است.

```
Class1 {  
Color:red;  
font-size:20px;  
{  
Class2 {  
Color:blue;  
font-size:30px;  
{  
Class3 {  
Color:green;  
font-size:40px;}
```

متغیر چیست؟

همانگونه که می دانید CSS یک زبان برنامه نویسی نیست و از همین جهت نمی توان در آن متغیر تعریف کرد. متغیرها مشخصه ای هستند که می توان به آنها ارزشی را نسبت داد که در طول برنامه قابل تغییر نیز باشند که باعث جلوگیری از تکرار کدنویسی می شود. متغیرها از یک نام و مقداری که به این نام اختصاص داده می شود تشکیل شده اند.

تعريف متغير در less

برای تعريف یک متغير در less از نماد (@) درابتدا نام متغیر استفاده می شود و برای تعیین مقدار متغیر بعد از نام متغیر از نماد دو نقطه (:) استفاده می کنیم. به مثال زیر توجه کنید:

```
<html>
  <head>
    <link rel="stylesheet" href="style.css" type="text/css">
    <title>define variable in less</title>
  </head>
  <body>
    <div class="class1">
      <h1>How to Define variable in Less</h1>
      <p>To define a variable in less,the symbol(@) is
        Used at the beginning of the variable name,
        And we use the two-point symbol(:)to determine
        The value of the variable after the name of the variable.</p>
    </div>
  </body>
</html>
```

درفایل style.less یک متغير به نام color تعريف میکنیم و کد رنگ قرمز را به آن نسبت می دهیم.

```
.class {
  @color:#f30bob;
  background-color:@color;
}
```

حال فایل style.less را به style.css کامپایل می کنیم.

Lessc style.less style.css

فایل style.css به صورت زیر ساخته و ویرایش می شود

```
.class {  
background-color:#f30bob;  
}
```

گسترش یا Less extend در

Less Extend به منظور اضافه کردن کلاس به کلاسی گسترش یافته مورد استفاده قرارمی گیرد. به عبارت دیگر Less توسط سلتکتور extend:، کلاس موردنظر را به کلاس توسعه یافته اضافه می کند. برای درک بهتر مطلب به مثال ساده زیر دقت کنید:

فرض کنید کلاسی با نام class1 و class2 با خصوصیات زیر موجود می باشد:

```
.class1 {  
background-color:red;  
font-size: 20px;  
}  
.class2 {  
color:white;  
}
```

حال قصد داریم تا ضمن نگهداری خصوصیات class2، این کلاس را به کلاس class2 نیز اضافه کنیم تا خصوصیات این کلاس را نیز داشته باشد.

دراينجا با کمک less در extend به اين صورت عمل می کنيم:

Style.less

```
.class1 {  
    background-color:red;  
    font-size: 20px;  
}  
  
.class2:extend(.class1) {  
    color:white;  
}
```

حال فایل style.less را به style.css کامپایل می کنیم.

Less style.less style.css

مشاهده می کنید که class2 به class1 اضافه شده است:

Style.css

```
.class,  
.class2 {  
    background-color: red;  
    font-size: 20px;  
}  
  
.class2 {  
    color:white;  
}
```

سلکتور extend می تواند در همان خط یا در داخل عبارت و به صورت تودر تو مورد استفاده قرار گیرد:

Style.less

```
.class1 {  
  background-color: red;  
  font-size: 20px;  
}  
  
.class2 {  
  &:extend(.class1);  
  color:white;  
}
```

حال فایل style.css را به style.less کامپایل می کنیم.

Less style.less style.css

Style.css

```
.class1,  
.class2 {  
  background-color:red;  
  font-size:20px;  
}  
  
.class2 {  
  color:white;  
}
```

درمثال بالا می توانید class2 را به بیش از یک کلاس اضافه کنید. به مثال زیر دقت کنید:

Style.less

```
.class1 {  
    Background-color: red;  
    Font-size: 20px;  
}  
  
P {  
    Border: 1px solid black;  
}  
  
.class2 {  
    &:extend(.class1,p);  
    Color:white;  
}
```

Style.css

```
.class1,  
.class2 {  
    Background-color:red;  
    Font-size: 20px;  
}  
  
P,  
.class2 {  
    Border:1px solid black;  
}  
  
.class2 {  
    Color:white;  
}
```

مقدمه جاوا اسکریپت

یکی از زبان های برنامه نویسی اسکریپتی است ، که اولین بار توسط شرکت عرضه کننده مرورگر معروف **Netscape Communicator** ارائه شد و امروزه متداولترین زبان اسکریپت نویسی صفحات وب است.

خصوصیات مهم **Java Script** :

Java Script یک زبان برنامه نویسی اسکریپتی است . دستور العمل های زبان های اسکریپتی ، در کامپیوتر کاربر و توسط مرورگر اجرا شده و برای اجرا نیازی به برنامه کمکی خاصی ندارند . به این زبان ها در اصطلاح طرف مشتری (Client Side) می گویند . در مقابل زبان های مثل ASP.NET,php ابتدا توسط سرور ارسال کننده وب اجرا شده و سپس نتایج خروجی به زبان HTML برای اجرا در مرورگر فرستاده می شود . به این زبان ها در اصطلاح طرف سرور (Server Side) می گویند.

زبان های اسکریپتی ، جزء زبان های برنامه نویسی سبک هستند . این زبان ها کامپایل آنها در زمان اجرا انجام می شود . و دستورات آن ها به صورت خط به خط اجرا می شوند.

کامپایل : برنامه های نوشته شده به زبان های برنامه نویسی مثل C , VB# یا C# در هنگام اجرا ابتدا توسط کامپایلر به طور کامل خوانده شده و اشکال زدایی می شوند و در صورت عدم وجود اشکال ، اجرا خواهد شد . اما برنامه های نوشته شده به زبان های اسکریپتی ، به صورت خط به خط توسط مرورگر خوانده شده و اجرا می شوند .

برخی از امکانات **Java Script** (جهت آشنایی اولیه)

• **Java Script** به طراحان وب ، یک ابزار برنامه نویسی ساده و کارا می دهد.

• **Java Script** به رویدادهای مختلف در صفحه واکنش نشان می دهد . برای مثال می توان یک تابع

JavaScript تعریف کرده تا در هنگام وقوع یک رویداد مثل کلیک بر روی یک دکمه یا لود شدن صفحه ، اجرا شود.

▪ **Java Script** می تواند اطلاعات ارسالی یک فرم را اعتبار سنجی و کنترل نموده و در

صورتی که صحیح بود ، آنها را به سرور ارسال کند . این کار باعث جلوگیری از ورود

اطلاعات نادرست به سرور و کاهش ترافیک آن می شود.

- توانایی تشخیص نوع و نسخه مرورگر مورد استفاده کاربر را داشته و می‌تواند بر حسب آن نوع مرورگر خاص، تنظیمات و صفحات ویژه‌ای را بارگذاری نماید.
- توانایی خواندن و نوشتן اطلاعات مورد نیاز مرورگر را بر روی کامپیوتر بازدید کننده صفحه را دارد، که در اصطلاح به این کار ایجاد و خواندن **Cookie** می‌گویند.
- می‌تواند انواع کادرهای اخطار، تایید و دریافت ورودی را به کاربر نمایش دهد.

تفاوت‌های جاوا اسکریپت و جاوا

جاوا یک زبان برنامه نویسی کاملاً شی گرا (Object Oriented Programming) است. کاربرد برنامه‌های جاوا در قالب Applet‌ها و صفحات JSP در وب است. در حالی که جاوا اسکریپت به عنوان یک زبان شبه شی گرا که اولین بار توسط شرکت NetScape ارائه شد، تنها یک فایل متنی ساده است که نمی‌توان از آن برای ایجاد برنامه‌های کاملاً مستقل استفاده کرد و برای اجرا می‌باشد در داخل صفحات HTML قرار گرفته و توسط مرورگرها تفسیر و اجرا شوند.

جاوا یک زبان کامپایلی است در حالی که جاوا اسکریپت یک زبان اسکریپتی (تفسیری) است.

زبان‌های کامپایلی به زبان‌هایی گفته می‌شود که قبل از اجرا می‌باشد کامپایل شوند. زبان‌های اسکریپتی نیز به زبان‌هایی گفته می‌شوند که مرحله کامپایل و اجرا آنها جدا نبوده و در واقع کامپایل آنها در زمان اجرا انجام می‌شود. وظیفه تفسیر برنامه‌های جاوا اسکریپت بر عهده مرورگر است.

3. از تفاوت‌های مهم دیگر می‌توان به سبک تعریف متغیرها اشاره کرد. زبان‌های برنامه نویسی از لحاظ تعریف متغیرها به دو دسته زبان‌های Loosely Type و Strongly Type تقسیم می‌شوند.

در زبان‌های با نوع قوی، می‌باشد ابتدا نوع متغیرها را تعیین و سپس در برنامه از آن استفاده نمود. نوع اینگونه متغیرها را نمی‌توان در طول اجرا برنامه تغییر داد و در صورتی که این متغیرها با عملگرهای مناسب خود به کار نروند نتایج نادرست به دست می‌آیند و یا خطایی به وقوع می‌پیوندد. زبان‌های C++ و java از این دست زبان‌ها هستند.

در زبان‌های با نوع ضعیف نیازی به تعریف متغیرها و تعیین نوع داده آنها نمی‌باشد. در این زبان‌ها تعیین نوع های داده به طور خودکار و بر حسب نیاز توسط خود زبان انجام می‌گیرد و بنابراین در طی فرآیند پردازش داده

ها می توان در هر مرحله به راحتی نوع داده ها را بررسی و تغییر داد. زبان هایی همچون javascript و PHP از این دست هستند.

جاوااسکریپت در مرورگرها

کدهای جاوااسکریپت در داخل تگ <script> نوشته می شوند.

وصل کردن یک فایل جاوااسکریپت به صفحه

1) به صورت خارجی

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

2) به صورت داخلی

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

```

<body>
  <input type="button" onclick="sayHi()">
  <!-- body goes here -->
</body>
</html>

```

تفاوت های به کارگیری کدها به صورت داخلی و خارجی در جاوا اسکریپت

- امنیت: هر کسی می تواند با باز کردن source صفحه شما ، کدها را ببیند و چه بسا به حفره های امنیتی آن پی برده و از آنها سواستفاده کند. خارجی امنیت بیشتری دارد.
- ذخیره شدن در حافظه مرورگرها: یکی از مزیت های استفاده از روش خارجی این است که فایل های خارجی جاوا اسکریپت پس از اولین بارگذاری در حافظه نهان مرورگر ذخیره شده و در دفعات بعد ، از حافظه فراخوانی و استفاده خواهند شد.
- نگه داری کدها: چنانچه شما بخواهید از یک کد در چندین صفحه وب استفاده کنید مطمئنا استفاده از روش اول موجب افزایش کدنویسی و در نتیجه حجم صفحه خواهد شد اما می توانیم از روش دوم برای چندین فایل استفاده کنیم.

ویژگی های بنیادی جاوا اسکریپت

۱) جاوا اسکریپت حساس به حروف است. Case Sensitive.

۲) متغیرها بدون نوع هستند.

۳) قرار دادن (;) در انتهای هر دستور اختیاری است

۴) درج توضیحات در جاوا اسکریپت : برای درج توضیحات در میان کدها می توان از روش های زبان های برنامه نویسی همچون C++ و C استفاده نمود یعنی از // برای توضیحات یک خطی یا /* */ برای توضیحات چند خطی

```

//this is a single-line comment
/* this is a multiline
comment */

```

خطایابی در مرورگرها

برای خطایابی به کنسول مرورگرها بروید. کنسول همان مفسری که در مرورگر وجود دارد و کار تحلیل را انجام می‌دهد.

زمانی که مرورگرها قادر به اجرای دستور خاصی از کدهای جاوا اسکریپت نباشند، پیغامی مبنی بر رخداد یک خطا را نمایش میدهند. اغلب برای مشاهده پیغام‌ها می‌بایست به Console جاوا اسکریپت در مرورگرها مراجعه کنید:

- در مرورگر **FireFox** با استفاده از میانبر `ctrl+shift+j` می‌توان به کنسول جاوا اسکریپت دسترسی داشت.
- در مرورگر **opera** `ctrl+shift+o`
- در مرورگر **Google Chrome** `ctrl+shift+j`

ساختار لغوی

در جاوا اسکریپت نیز مثل سایر زبان‌های اسکریپتی دارای قواعدی برای نوشتن کد می‌باشد. به طور کلی تمامی متغیر‌ها و توابع از حروف کوچک شروع می‌شود و مثل کوهان شتر می‌باشد مثل:

`getElementById`

این زبان اسکریپتی شامل متغیر، ارایه، توابع، آجکت و مازول در ساختار برنامه نویسی خود می‌باشد. جاوا اسکریپت زبان پایه نمی‌باشد و یک زبان کاملاً متنی و مفسری هست و توسط مرورگر تفسیر می‌شود. در نظر داشته باشید ساختار کلی جاوا اسکریپت برپایه قواعد EcmaScript می‌باشد.

تحقيق: نوشتن کدهای اسکریپت در تگ `head` با نوشتن اسکریپت‌ها در تگ `body` چه تفاوتی دارد؟

داده ها

فرق **Var** و **Let** در جاوا اسکریپت چیست؟

با توجه به توسعه زبان جاوا اسکریپت در سال های گذشته دیگر این زبان از سال ۲۰۰۹ فقط یک زبان سمت کلاینت نبود بلکه علاوه بر کد نویسی سمت کلاینت با جاوا اسکریپت می شد برنامه نویسی سایت و اپلیکیشن رو هم انجام داد همه این موارد با انتشار node.js اتفاق افتاد.

فرق **let** و **var** در جاوا اسکریپت

برای تعریف متغیر های در JS ما از عبارت **var** استفاده میکردیم که مقداری رو داخلش قرار میدادیم به عنوان مثال:

```
var myText = "dadekav21";
```

فرق **let** و **var** در جاوا اسکریپت چیست؟

یا به عنوان مثال ما این عبارت رو اگر مینوشتیم

```
Var a = 5;  
var b = 10;  
  
if(a == 5){  
    var a = 4;  
    var b = 10;  
  
    console.log(a);  
    console.log(b);  
}  
  
console.log(a);  
console.log(b);
```

در کد ما با استفاده از **var** متغیر تعریف کردیم و بررسی اگر **a** برابر **5** بود **a** رو مساوی **4** قرار بده و **b** هم برابر **10** باشد اما اگر این کد رو خروجی بگیریم خروجی به صورت زیر خواهد بود.

4
10
4
10

به عبارتی مقدار ۴ برای a در همه جای کد نظر گرفته میشه اما در اکما اسکریپت ۶ و با استفاده از let میتوانیم متغیر هایی رو تعریف کنیم که مقدار یک متغیر فقط در یک بلاک قابل استفاده باشه و اگر در کد بالا و در بلاک if به جای var a = 4 را استفاده کنیم عبارت let a = 4 رو بنویسیم و خروج بگیریم متوجه میشویم که خروجی ۱۰ ۵ ۱۰ ۴ خواهد بود چرا که مقدار a = 4 فقط در بلاک if خواهد بود و دیگر سراسری نیست.

استفاده از const به جای let

برای تعریف یک متغیر میتوانیم به جای let و var از const استفاده کنیم که قوانین خاص خودش را دارد.
به عنوان مثال اگر من بیرون از بلاک const a = 5 رو بنویسم و داخل بلاک بنویسیم var a = 4 خطای میگیرد. چرا که a قبلا به صورت یک ثابت تعریف شده و نمی توانیم دوباره آن را تعریف کنیم.
به عبارت دیگر بر روی حرف و یا کلمه ای برای تعریف متغیر استفاده کردیم تمرکز میکند و اگر جای دیگه به کار برده شود باعث ایجاد خطای میشود و نمیتوانیم در جای دیگه کدها از آن عبارت برای تعریف متغیر استفاده کنیم.

انواع داده ها

در جاوا اسکریپت شش نوع داده اصلی به شرح زیر وجود دارد:

undefined
null
boolean
number
string
object

زبان جاوا اسکریپت دارای انواع داده ای مختلفی است که از آنها میتوان استفاده نمود. برای نمونه میتوان از داده نوع رشته برای ذخیره نام یک محصول استفاده کرد. در این فصل با داده نوع های مختلف موجود در زبان جاوا اسکریپت آشنا خواهیم شد.

داده نوع ها

یک متغیر در جاوا اسکریپت میتواند شامل هر داده ای باشد. یک متغیر می تواند در یک لحظه حاوی یک رشته باشد و بعدا یک مقدار عددی بگیرد. عبارت زیر بدون هیچ خطایی اجرا می شود:

```
// no error  
  
let message = "hello";  
  
message = 123456;
```

زبان های برنامه نویسی که اجازه چنین کاری را می دهند دارای نوع های پویا

هستند و این مسئله به این معناست که داده نوع ها در این زبان ها وجود دارند اما متغیرها محدود به هیچ کدام از آنها نیستند.

در زبان جاوا اسکریپت ۶ داده نوع پایه وجود دارد. در این فصل به بررسی این داده نوع ها پرداخته و در فصل های بعدی درباره هر کدام به طور مفصل صحبت خواهیم کرد.

نوع عددی (number)

داده نوع عددی برای هر دو قسم اعداد صحیح و اعشاری استفاده می شود.

یک سری عملیات برای اعداد وجود دارد برای مثال ضرب * ، تقسیم / ، تفریق - ، جمع + و ...

در کنار اعداد معمولی مقادیر عددی خاصی هم وجود دارد که متعلق به نوع عددی هستند:

NaN و Infinity ، Infinity

بیانگر مقدار بی نهایت (∞) در ریاضی است. این مقداری است که بزرگتر از هر عددی می باشد. می توان با تقسیم عددی بر صفر به مقدار بی نهایت دست یافت:

```
alert( 1 / 0 ); // Infinity
```

میتوان مقدار بی نهایت را به صورت زیر صریحا در اسکریپت خود استفاده کنیم :

```
alert( Infinity ); // Infinity
```

بیانگر یک خطای محاسباتی است و نتیجه یک عملیات ریاضی تعریف نشده و یا ناصحیح است. برای

مثال:

```
alert( "not a number" / 2 ); // NaN, such division is erroneous
```

هر عملیاتی که بر روی مقدار NaN انجام شود خود مقدار NaN را تولید خواهد کرد:

```
alert( "not a number" / 2 + 5 ); // NaN
```

بنابراین اگر مقدار NaN در هر جایی از یک عبارت ریاضی باشد، نتیجه NaN خواهد شد.

در زبان جاوا اسکریپت عملیات های ریاضی در اصطلاح امن هستند. برای مثال ما می توانیم هر عملیاتی را انجام دهیم : تقسیم بر صفر ، استفاده از رشته ها به جای اعداد و در این حالات اسکریپت ما هیچ گاه متوقف خواهد شد و تنها نتیجه عملیات NaN خواهد بود.

مقادیر عددی خاصی که در بالا ذکر شدند متعلق به نوع عددی هستند اما آنها عملاً عدد نیستند.

نوع رشته string

رشته ها در جاوا اسکریپت باید در داخل کوتیشن ها قرار بگیرند:

```
let str = "Hello";
```

```
let str2 = 'Single quotes are ok too';
```

```
let phrase = `can embed ${str}`;
```

در زبان جاوا اسکریپت سه نوع کوتیشن وجود دارد:

۱. دابل کوتیشن: "Hello"
۲. سینگل کوتیشن: 'Hello'
۳. بک تیک ها: `Hello`

مورد اول و دوم کوتیشن های ساده هستند و تفاوتی بین آنها در جاوا اسکریپت وجود ندارد. اما مورد سوم یعنی بک تیک ها دارای قابلیت های گسترده‌تری هستند. بک تیک ها به ما اجازه می‌دهند که متغیرها و یا عبارات را در داخل یک رشته با استفاده از دستور ``${...}`` بگنجانیم. برای مثال:

```
let name = "John"; // embed a variable  
alert(`Hello, ${name}!`); // Hello, John!// embed an expression  
alert(`the result is ${1 + 2}`); // the result is 3
```

خروجی نمونه مثال فوق در جلوی هر عبارت به صورت کامنت مشخص شده است. در مثال بالا عبارت داخل دستور ``${...}`` ارزیابی می‌شود و نتیجه آن در رشته قرار می‌گیرد. میتوان هر چیزی را اینجا قرار داد. برای مثال یک متغیر مانند `name` یا یک عبارت ریاضی شبیه $1 + 2$ و یا چیزی پیچیده‌تر.

دقت داشته باشید که ارزیابی فقط در بک تیک ها انجام می‌شود و دیگر کوتیشن ها اجازه چنین ارزیابی را نمی‌دهند:

```
alert("the result is ${1 + 2}"); // the result is ${1 + 2} (double quotes do nothing)
```

در زبان جاوا اسکریپت داده نوع کاراکتری (character) وجود ندارد. در بعضی از زبان‌ها داده نوع خاصی برای کاراکتر‌ها به نام `char` وجود دارد برای مثال در زبان C و `java` برای ذخیره نوع‌های کاراکتری استفاده می‌شود. در جاوا اسکریپت چنین نوعی وجود ندارد. تنها یک نوع وجود دارد و آن هم `string` است. یک رشته می‌تواند شامل یک کاراکتر و یا تعدادی از آنها باشد.

نوع منطقی - **boolean**

داده نوع بولین شامل دو مقدار است: `true` و `false`. این داده نوع به صورت گسترده برای ذخیره مقادیر بله و خیر استفاده می‌شود. `true` به معنای بله و صحیح و `false` به معنای خیر و ناصحیح است. برای مثال:

```
let nameFieldChecked = true; // yes, name field is checked
```

```
let ageFieldChecked = false; // no, age field is not checked
```

مقادیر بولین به عنوان نتیجه مقایسه‌ها نیز استفاده می‌شوند:

```
let isGreater = 4 > 1;  
alert(isGreater); // true (the comparison result is "yes")
```

مقدار null

مقدار خاص `null` به هیچ کدام از داده نوع های شرح داده شده در بالا تعلق ندارد و نوع مختص به خود را دارد که شامل تنها مقدار `null` است :

```
let age = null;
```

در زبان جاوا اسکریپت `null` به شی ای که وجود ندارد و یا اشاره گر `null` همانند دیگر زبان ها اشاره نمی کند . `null` تنها یک مقدار خاص است که به معنای هیچ چیز ، حالی و یا مقدار ناشناخته است. کد نوشته شده در بالا وضعیت متغیر `age` را ناشناخته و یا حالی مشخص میکند.

مقدار undefined

مقدار خاص `undefined` مختص نوع خود است و به هیچ کدام از داده نوع های ذکر شده در بالا تعلق ندارد. معنای `undefined` این است که مقداری تخصیص داده نشده است. اگر متغیری تعریف شود و مقدار دهی اولیه نشود آنگاه مقدار آن برابر با `undefined` خواهد بود:

```
let x;  
alert(x); // shows "undefined"
```

از نظر تکنیکی می توان مقدار `undefined` را به هر متغیری نسبت داد:

```
let x = 123;  
x = undefined;  
alert(x); // "undefined"
```

اما چنین کاری توصیه نمی شود. به طور معمول ما از `undefined` تنها برای اینکه چک کنیم آیا متغیری مقدار دهی شده است و یا خیر استفاده میکنیم.

نوع های object

نوع `object` یک نوع خاص است. تمام دیگر نوع های اولیه نامیده می شوند و این به خاطر این است که آنها تنها می توانند یک چیز را در خود داشته باشند. برای مثال رشته یا عدد و یا چیزهای دیگر. در مقابل

Object ها برای ذخیره مجموعه هایی از داده و موجودیت های پیچیده تر استفاده می شود در فصول آینده مفصل به این داده نوع ها خواهیم پرداخت.

نوع symbol

نوع symbol برای ساخت یک شناسه منحصر به فرد برای اشیا (object) استفاده می شود بعد از بررسی آینده ها به این نوع هم در Object خواهیم پرداخت.

عملگر typeof

عملگر typeof نوع آرگومان خود را بر می گرداند این عملگر برای چک کردن نوع متغیر ها استفاده می شود. به دو صورت می توان از عملگر typeof استفاده کرد:

به عنوان یک عملگر: `typeof x`

به عنوان یک تابع: `typeof(x)`

به عبارتی دیگر از این عملگر با و یا بدون پرانتز می توان استفاده کرد. نتیجه یکسان خواهد بود. فراخوانی typeof رشته را بر می گرداند که نام نوع در داخل آن قرار دارد:

```
typeof undefined // "undefined"
```

```
typeof 0 // "number"
```

```
typeof true // "boolean"
```

```
typeof "foo" // "string"
```

```
typeof Symbol("id") // "symbol"
```

```
typeof Math // "object" (1)
```

```
typeof null // "object" (2)
```

```
typeof alert // "function" (3)
```

در نمونه مثال بالا سه خط آخر نیاز به توضیح اضافه دارد:

یک شی درونی در جاوا اسکریپت است که شامل عملیات های ریاضی است. Math

همانطور که قبل از آن `null` مقداری است مختص نوع خود و این یک خطا در زبان جاوا اسکریپت است که در اینجا نوع آن را `object` نشان میدهد

نتیجه عبارت `typeof alert` برابر با "function" است. زیرا که `alert` یک تابع بوده که مقدار پارامتر خود را در پنجره ای به کاربر نشان می دهد .

تابع در جاوا اسکریپت

با کلمه کلیدی `function` تعریف می شوند. تابع در جاوا اسکریپت یک نوع `object` یا شی هستند.

`Function()` (نام تابع)

{

دستورات تابع

}

تعریف تابع : یک تابع مجموعه ای واحد از یکسری دستورالعمل است که در هر بار فراخوانی ، کل دستورات درون آن یکبار اجرا می شود.

قابلیت های تابع :

- ۱) دستورات یک تابع (حتی در زمانی که اسکریپت آن در درون صفحه قرار دارد) ، تا زمانی که فراخوانی نشود ، اجرا نخواهد شد .
- ۲) از تابع برای تعریف دستورالعمل هایی استفاده می شود که می خواهیم اجرای آنها کنترل شده باشد و در موقع معینی (مثل وقوع یک رویداد یا ...) انجام شود.
- ۳) یک تابع را می توان از هر نقطه ای در صفحه فراخوانی کرد.
- ۴) یک تابع می توان یکسری متغیرها را به عنوان پارامتر ورودی دریافت کرده و همچنین یک مقدار را به عنوان خروجی به نقطه ای که از آن فراخوانی شده است ، باز گرداند.

نحوه تعریف پارامتر برای یک تابع

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

    function ali(num1,num2) {
        document.write(num1*num2); }

    function reza()
    {
        var a=Text1.value;
        var b=Text2.value;
        ali(a,b); }

</script>

<input type="button" id="Button2" onclick="reza()" value="click me !" />
<input type="text" id="Text1" />
<input type="text" id="Text2" />
```

نحوه تعریف مقدار بازگشی برای یک تابع

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

    function ali(num1,num2) {
        return num1*num2;
    }

    function reza() {
        var a=Text1.value;
        var b=Text2.value;
        document.write(ali(a,b))
    }

</script>
```

```

<body>

    <input type="button" id="Button2" onclick="reza( )" value="click me !" />
    <input type="text" id="Text1" />
    <input type="text" id="Text2" />

</body>

```

شیوه نگارش یا syntax توابع در جاوا اسکریپت

```

<script type="text/javascript">

    function username (n,c)
    {
        var name = 'نام شما: ' + n + '<br />';
        var country = 'کشور شما: ' + c + '<br />';
        var output = name + country;
        return output;
    }

    document.write(username('admin','Iran'));

</script>

```

توابع در جاوا اسکریپت باید حتما به صورت رویدادی فراخوانی شوند (مثلا `onload` یا `onclick` و...)، در حالت عادی وجود یک تابع در صفحه به خودی خود، کار خاصی انجام نمی دهد، در مثال بالا فراخوانی تابع با دستور `document.write` انجام شده است.

نحوه فراخوانی توابع در جاوا اسکریپت

در مثال زیر شیوه فراخوانی یک تابع را با رویداد پرکاربرد onclick ملاحظه می کنید.

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

    function add (num1,num2){

        var output = num1 + num2;

        return document.write(output);

    }

</script>

<a href="#" onclick="add(3,6);>کلیک کنید</a>
```

تمرین:

برنامه ای بنویسید که نام و نام کشور کاربر را دریافت کند و سپس نمایش دهد.

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

    function reza()

    {

        var n=Text1.value;

        var c=Text2.value;

        var name ='نام شما: '+ n + '<br />';

        var country ='کشور شما: '+ c + '<br />';

        document.write(name + country);

    }

</script>
```

```

<body>

<label style="margin-left: 41px;">نام:</label>

<input type="text" id="Text1"/><br/>

<label>نام کشور:</label>

<input type="text" id="Text2" /><br/>

<input type="button" id="Button2" onclick="reza()" value="click me !"
style="margin-left: 100px;"/>

</body>

```

تمرین ۱: برنامه ای بنویسید که دو عدد دریافت کند و بگوید کدام عدد بزرگتر است.

تمرین ۲: برنامه ای بنویسید که یک عدد دریافت کند و بگوید اول است یا خیر.

اعلان نوع متغیر

زمانی که متغیر جدیدی را اعلان می کنید، می توانید با استفاده از کلمه کلیدی "new" ، نوع آنرا نیز مشخص نمایید:

```

var carname=new String;
var x= new Number;
var y= new Boolean;
var cars= new Array;
var person= new Object;

```

توجه : تمام متغیرها در JavaScript شیء اند، زمانی که متغیری را اعلان می کنید، در واقع یک شیء ایجاد کرده اید.

تبدیل انواع

تبدیل به رشته: سه نوع داده `string`, `number`, `boolean` متبدی به نام `toString()`. برای تبدیل به رشته دارند. این متد برای متغیرهای از نوع `Boolean` یکی از مقادیر رشته ای `true` و `false` را بسته به مقدار متغیر برمی گرداند:

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

این متد برای متغیرهایی از نوع `number` رشته ای حاوی آن عدد را برمی گرداند:

```
var iNum1 = 10;  
var fNum2 = 10.0;  
var fNum3 = 10.5;  
alert(iNum1.toString()); //outputs "10"  
alert(fNum2.toString()); //outputs "10"  
alert(fNum3.toString()); //outputs "10.5"
```

```
var num=10;  
alert(num.toString(2)); //1010  
alert(num.toString(8)); //12
```

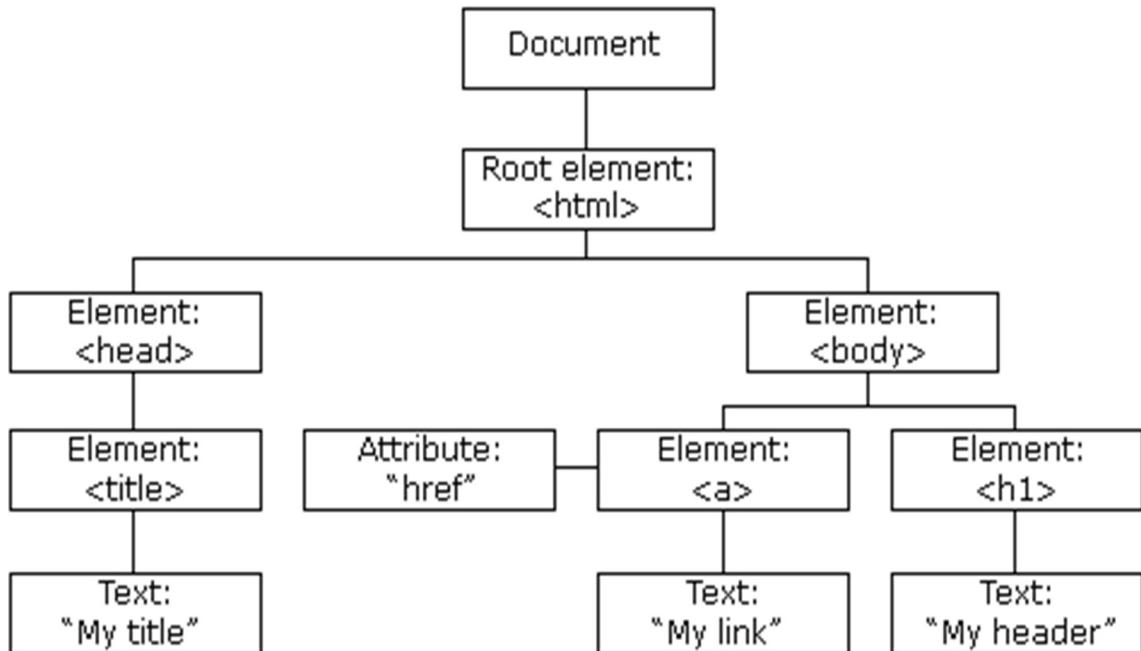
اجزا تشکیل دهنده جاوا اسکریپت

Document Object Model ؛ مدل شی گرای سند **DOM**

یکی از API ها (Application Programming Interface) برای زبان های XML و HTML به شمار می رود.

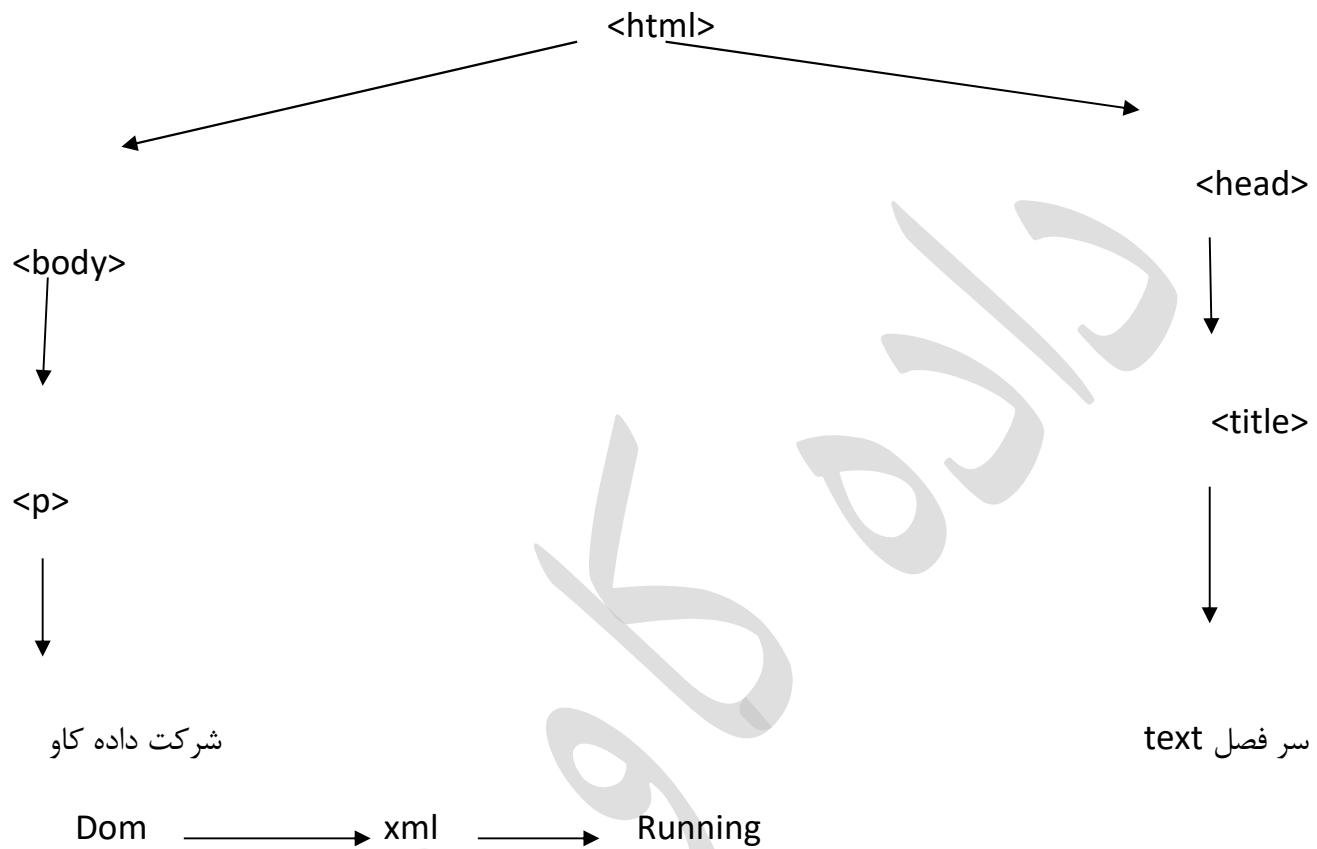
تمام عناصر موجود در یک صفحه وب را به صورت درختی از گره ها نمایش می دهد و امکان کنترل آنها برای توسعه دهندگان وب را فراهم می آورد.

با استفاده از DOM می توان گره ها را به راحتی حذف ، اضافه ، جابه جا و یا جایگزین کرد. تغییر استایل ها و متن موجود در سند.....



Dom مفهوم

یک ساختار درختی است و در طراحی از آن استفاده می شود.



Model شی گرا متنی Document Object Model

Dom دسترسی گسترده به صفحات Html می دهند که آنها را قادر می سازد.

بزرگترین پیشرفت در زمینه وب از Html بوده است و برای اتصال چند سند از طریق اینترنت مورد استفاده قرار می گیرد.

Dom به برنامه نویسان وب دسترسی بی سابقه ای را به Html می دهد. که آنان را قادر می سازد که به طور دقیق با Html به عنوان یک سند xml برخورد کنند.

Xml از زبان قدیمی SGML گرفته شده است.

(Standard Generalized Markup Language) SGML

SGML یا زبان نشانه‌گذاری تعمیم یافته استاندارد نام استانداردی برای ساختار زبان‌های نشانه‌گذاری است. این زبان به تنها‌ی برای نشانه‌گذاری در اسناد کاربردی ندارد بلکه به عنوان مبنای استاندارد سازی زبان‌های نشانه‌گذاری دیگر استفاده می‌شود و به همین خاطر به آن فرازبان می‌گویند. برای نمونه HTML یک زبان نشانه‌گذاری برپایه SGML است و یک فرازبان دیگر است که از XML الگوبرداری کرده‌است.

معرفی شی document

جاوا اسکریپت یک زبان برنامه نویسی مبتنی بر شی است.

این زبان به برنامه نویس، قابلیت استفاده از اشیای پیش ساخته و یا تعریف و ایجاد اشیای جدید مورد نیاز خود را می‌دهد.

```
document.write("hello");  
document.getElementById("demo");  
document.getElementsByTagName("a")  
document.getElementsByClassName("a")[0].children.length-1  
document.querySelector(); "#id" / ".class" / "p"  
document.querySelectorAll(); "#id" / ".class" / "p"
```

در کدنویسی با جاوا اسکریپت، دو متده کاربرد `innerHTML` و `getElementById` جزء برجسته ترین این موارد هستند.

متده `innerHTML`

از `innerHTML` در جاوا اسکریپت برای دریافت یک مقدار از درون تگ‌های HTML یا اختصاص دادن یک مقدار به تگ‌ها و تغییر محتوای آنها استفاده می‌شود.

۱. دریافت محتوا عنصر

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

function SetValue(){

    var element = document.getElementById('result');

    var value = element.innerHTML; // دریافت مقادیر از عنصر مورد نظر

    alert(value +"لطفا حل شود");

}

</script>

<input type="button" onclick="SetValue()" value="کلیک کنید">

<div id="result">این یک تست است</div>
```

۲. اختصاص دادن محتوا به عنصر

کاربرد innerHTML محدود به دریافت یک مقدار نیست و می توان برای اختصاص مقادیر به عناصر HTML از آن استفاده کرد، به مثال زیر توجه کنید.

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

function SetValue(id, value){

    var element = document.getElementById(id);

    element.innerHTML = value; // اختصاص مقدار به عنصر مورد نظر

}

</script>

<input type="button" onclick="SetValue('result', 'این یک تست است');" value="کلیک کنید">

<div id="result"></div>
```

نکته ۱: متد innerHTML وابسته به متد و آبجکت پیش از خود (document و getElementById) است.

نکته ۲: استفاده از متد innerHTML برای تغییر محتوای یک عنصر والد (parent) باعث حذف تگ های فرزند

(child) آن نیز می شود، به فرض اگر درون تگ div از چند تگ تو در تو استفاده شده باشد، با به کار بردن

innerHTML برای تگ والد، محتوای آن هر چه باشد (شامل تگ های div تو در تو) حذف و با مقادیر جدید

جاگزین می شود.

نکته ۳: برای دریافت مقادیر از تگ input یا تغییر محتوای آن، باید از متد value به جای innerHTML استفاده شود.

متدد getElementById

از getElementById برای تعیین یک مرجع (reference) به یک عنصر (element) با توجه به ID آن

استفاده می شود، به زبان ساده یعنی انتخاب یک عنصر HTML (به فرض تگ div) با توجه به ID آن،

مثال اگر بخواهیم مقادیر value یک input را با توجه به ID آن استخراج کنیم

```
<input id="test" type="text" >  
  
<script type="text/javascript">  
  
    var input = document.getElementById('test').value;  
  
    alert(input);  
  
</script>
```

نکته ۱: متد getElementById خود وابسته به شی (object) دیگری به نام document است، لذا الزاما باید به همراه آن آورده شود.

نکته ۲: متد getElementById نسبت به بزرگ یا کوچک بودن مقادیر ID حساس است، به طور مثال Id با id در این متد دو مقدار متفاوت هستند.

متده استفاده از querySelector()

با استفاده از querySelector() می‌توان به عناصر موجود در صفحه وب دسترسی پیدا نماییم، در واقع همانند یک انتخابگر (selector) عمل می‌کند و می‌شود به راحتی تگ‌ها، id‌ها، کلاس‌ها و ... را از DOM انتخاب نمود.

اگر با CSS آشنایی داشته باشید، روش‌های مختلفی برای گرفتن عناصر از صفحه وب وجود دارد که بتوان به آن‌ها استایل داد.

انواع استفاده از عناصر در CSS:

```
body {}  
.class-test {}  
#id-test {}
```

حال اگر بخواهیم توسط متده استفاده از querySelector() عناصر را انتخاب کنیم:

```
let body = document.querySelector("body");  
let classTest = document.querySelector(".class-test");  
let idTest = document.querySelector("#id-test");
```

انتخاب تگ div که درونش تگ p قرار دارد و درون تگ p کلاس .class-test است:

```
let body = document.querySelector("div p.class-test");
```

انتخاب input با type="text":

```
let body = document.querySelector("input[type='test']);
```

متده **querySelectorAll()**

توجه داشته باشید که متده **querySelector()** همیشه اولین عنصر را از صفحه انتخاب میکند، یعنی اگر ۱. کلاس **class-test**. داشته باشیم، اولین کلاس را انتخاب میکند و با بقیه کاری ندارد.

حال اگر بخواهیم تمام عناصر را انتخاب کنیم باید از متده **querySelectorAll()** استفاده کنیم.

انتخاب عنصر با : index

```
let classTest = document.querySelectorAll(".class-test")[5];
```

انتخاب عناصر با : for loop

```
let classTest = document.querySelectorAll(".class-test");
for (let i = 0; i < classTest.length; i++) {
    classTest[i].style.color = "red";
}
```

رویداد ها (events) در جاوا اسکریپت (JavaScript)

رویداد (event) به معنی اتفاقی در یک صفحه وب است که می تواند عامل آن، کاربر یا به فرض بارگذاری کامل یک صفحه باشد، بعد از بروز این اتفاق (رویداد)، مفسر جاوا اسکریپت مرورگر، آن را شناسایی کرده و مناسب با تابع تعریف شده، آن را اجرا می کند.

رویدادها در واقع حکم استارت(شروع)، برای موتور توابع را بازی می کنند و یک تابع بدون رویداد معمولاً قابل اجرا نیست.

```
<script type="text/javascript">  
    function hello()  
    {  
        alert ("۲۱ داده کاو");  
    }  
</script>  
<input type="button" name="button" value="کلیک کنید" onclick="hello();"/>  
نحوه فراخوانی تابع بالا با رویداد
```

رویدادهای HTML و تعامل آنها با جاوا اسکریپت

این رویدادها، اتفاق هایی هستند که روی المان های HTML رخ می دهند.

- بارگذاری یک صفحه HTML به اتمام برسد.
- دسته ای از ورودی های یک صفحه HTML تغییر کند.
- روی دکمه ای از یک صفحه HTML کلیک شود.

معمولاً بعد از اینکه هر یک از این رویدادها انجام می گیرند، می خواهیم که کاری صورت بپذیرد. این امر با استفاده از رویدادهای جاوا اسکریپت انجام می گیرد.

مهمترین رویدادهای HTML

مهمترین رویدادهای مهم **Input** عبارتند از:

Onchange، این رویداد در حالتی اجرا می شود که کار ما با مقادیر یک فیلد فرم در صفحات وب تمام شود و خارج از آن فیلد کلیک کرده یا Enter را بزنیم، اعمال می گردد.

Oninput، این رویداد پس از افزودن هر کارکتر در فیلد فرم اعمال می گردد.

Onsubmit، این رویداد در حالتی کاربرد دارد که کاربر یک فرم را ارسال می کند.

رویدادهای موس (**Mouse Events**):

Onclick، این رویداد پس از یکبار کلیک بر روی عنصری اعمال می گردد.

Ondblclick، این رویداد پس از دوبار کلیک بر روی عنصری اعمال می گردد.

Onmouseover، این رویداد رویداد زمانی رخ می دهد که mouse از روی عنصر می گذرد.

اشیاء (Object JavaScript)

برای تعریف یک شیء، از آکولاد استفاده می شود. داخل آکولاد، خصوصیات شیء بصورت (مقدار=نام خصوصیت) تعریف می شود. خصوصیت ها با کاما از هم جدا می شوند:

```
var person={firstname:"ali", lastname:"karimi", id:5566};
```

شی **person** در مثال بالا، سه خصوصیت یا (property) دارد **firstname** و **lastname** و **id**.

اعلان متغیر می تواند در چند خط باشد: (فاصله ها و خطوط اضافه مهم نیستند)

```
var person={  
firstname : "ali",  
lastname : "karimi",  
id: 5566  
};
```

برای دسترسی به خصوصیت های یک شیء، دو روش وجود دارد:

1. let name=person.lastname;
2. let name=person["lastname"];

جاوا اسکریپت یک زبان برنامه نویسی مبتنی بر شی است.

این زبان به برنامه نویس، قابلیت استفاده از اشیای پیش ساخته و یا تعریف و ایجاد اشیای جدید مورد نیاز خود را می دهد.

- جاوا اسکریپت بسیاری از قابلیت های شی گرایی، زبان برنامه نویسی شی گرایی را ندارد.

- جاوا اسکریپت از قابلیت های اساسی زبان برنامه نویسی شی گرا یعنی بسته بندی و پنهان سازی اطلاعات پشتیبانی نمی کند.

چون جاوا اسکریپت یک زبان اسکریپتی است نه یک زبان برنامه نویسی، جاوا اسکریپت از ایجاد انواع شی و نمونه سازی آنها برای ایجاد نمونه های شی پشتیبانی می کند.

از ترکیب اشیاء مولفه ای و استفاده مجدد از آنها را پشتیبانی نمی کند.

جاوا اسکریپت از قابلیت وراثت هم پشتیبانی نمی کند. اشیاء تعریف شده در جاوا اسکریپت شامل اشیاء مرورگر و اشیاء خاص جاوا اسکریپت می باشد.

اشیاء مرورگر وقتی صفحه وب توسط مرورگر بارگذاری می شود. مرورگر چندین شی جاوا اسکریپت ایجاد می کند که امکان دستیابی به صفحه وب و عناصر HTML موجود در آن را فراهم می کند.

به جزء اشیاء مرورگر اشیاء اختصاصی دیگری هم داریم مثل

Global, String, RegExp, Array, Date, Math, boolean, Number, Event, Error, function

شی Global

متدها و خواص این شیء بالاترین سطح و متدهای خاص هستند و والدی ندارند.

متدها و خواص این شیء در تمام اسکریپت‌های جاوا اسکریپت به طور عمومی قابل استفاده است. مثلاً چند نمونه از متغیرهای Global عبارت اند از:

`parseFloat()`, `parseInt()`

نکته: از روی شی Global نمی‌توان نمونه سازی کرد.

ایجاد نمونه‌های از اشیاء ایجاد اشیاء در جاوا اسکریپت

`variable=new objecttype(parameters)`

متغیر / Objecttype سازنده شی Parameters / لیست پارامترهایی که سازنده ارسال می‌کند.

تبدیل به عدد

متد `parseInt()`. از اولین کاراکتر رشته شروع می‌کند اگر عدد بود آن را برمی‌گرداند در غیر این صورت مقدار NaN را برمی‌گرداند. این روند تا آخرین کاراکتر ادامه پیدا می‌کند تا اینکه به کاراکتری غیر عددی برسد. به عنوان مثال این متد عبارت "RED ۱۲۳" را به صورت 123 برمی‌گرداند

```
var iNum1 = parseInt("1234blue"); //returns 1234  
var iNum2 = parseInt("22.5"); //returns 22  
var iNum3 = parseInt("blue"); //returns NaN  
var iNum4 = parseInt("123blue34"); //returns 123
```

متدهای `parseInt()` و `parseFloat()` نیز مثل عمل کرده و از اولین کاراکتر شروع به جستجو

می‌کند. البته در این متدهای اولین نقطه حساب نمی‌شود و آن را به همان صورت بر می‌گرداند.

اگر دو کاراکتر نقطه در رشته وجود داشته باشند دومین نقطه به عنوان کاراکتر بی ارزش شناخته می‌شود و عملیات تبدیل متوقف می‌شود. مثال‌ها:

```
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
```

متدهای `eval()`. این متدهای مقدار از نوع رشته می‌گیرد و آن را ارزیابی یا اجرا می‌کند. این متدهای شبیه مفسر جاوا

اسکریپت عمل می‌کنند.

```
var x = 5;  
var y = 10;  
  
console.log(eval("x * y")); => 50  
console.log(eval("2 + 2")); => 4  
  
console.log(eval("x + 13")); => 18
```

عملگر (+) unary plus، قبل عملوند خود قرار می‌گیرد و عملوند را ارزیابی می‌کن، که آیا عدد هست یا خیر. در صورت نبود عدد آن را تبدیل به عدد می‌کند.

```
let x = "1";  
console.log(+x); => 1  
console.log(+ ""); => 0  
console.log(+true); => 1  
console.log(+false); => 0
```

❖ اگر از عملگر (-) استفاده شود خروجی ما به صورت منفی باز می‌گردد.

isFinite() متد چک کردن یک عدد است یا خیر(شامل اعداد صحیح و اعشاری)، خروجی `true` و `false` بر می‌گرداند.

```
alert(isFinite(num))  
alert(isFinite("hello")) //false  
alert(isFinite("123")) //true  
alert(isFinite("1224hello")) //false  
alert(isFinite("123.55")) //true  
Alert(NaN==NaN) //false
```

Rest Parameter

به یک تابع اجازه می‌دهد تا تعداد نامحدودی از parameterها را به عنوان یک آرایه بپذیرد و راهی برای نمایش توابع متغیر فراهم می‌کند.

```
function sumMethod(...nums) {  
    let sum = 0;  
  
    for (let i of nums) {  
        sum += i;  
    }  
  
    console.log("Sum = " + sum);  
}  
  
sumMethod(1, 2, 3);      => 6  
sumMethod(1, 2, 3, 4);  => 10
```

شی string

شی string یکی از اشیای پیش ساخته در جاوا اسکریپت است . این شی برای دستکاری و انجام عملیات بر روی داده های متنی استفاده می شود . هر متغیر از نوع متنی ، نمونه ای از شی string است .

خاصیت length

این خاصیت ، تعداد کاراکترها (حرف) عبارت متنی را بر می گرداند . فاصله بین حروف نیز یک کاراکتر محسوب می شود . شکل کلی استفاده از این خاصیت به شرح زیر است :

Syntax	object.length
--------	---------------

مثال : در مثال زیر یک متغیر رشته ای را تعریف و سپس به وسیله دستور `document.write` ، طول آن را نشان داده ایم :مثال :

<code><script type="text/javascript"> var name = "Developer Studio" ; document.write (name.length) ; </script></code>	کد
16	خروجی

متدهای شی `string`

متدهای شی `string` در جدول زیر معرفی شده اند .

نام خاصیت	شرح
<code>charAt ()</code>	برای نمایش مقدار یک حرف (کاراکتر) مورد نظر در یک متغیر رشته ای استفاده می شود .
<code>charCodeAt ()</code>	برای نمایش کد اسکی یک حرف (کاراکتر) مورد نظر در یک متغیر رشته ای استفاده می شود .
<code>concat ()</code>	از این متده برای چسباندن و اضافه کردن دو یا چند متغیر رشته ای به هم استفاده می شود .
<code>indexOf ()</code>	این متده ، شماره مکان قرار گیری اولین نمونه یک حرف یا کلمه را در یک متغیر متنی را بر می گرداند .
<code>lastIndexOf ()</code>	این متده ، شماره مکان قرار گیری آخرین نمونه یک حرف یا کلمه را در یک متغیر متنی را بر می گرداند .
<code>match ()</code>	از این متده ، برای جستجوی یک حرف یا کلمه در یک متغیر متنی استفاده می شود .

از این متدهای جایگزینی یک حرف یا کلمه خاص در یک متغیر متنی و جایگزینی آن با یک مقدار جدید استفاده می‌شود.	replace ()
از این متدهای جستجو یک حرف یا کلمه خاص در یک متغیر متنی استفاده می‌شود.	search ()
از این متدهای جستجو یک حرف یا کلمه خاص در یک متغیر متنی استفاده می‌شود.	slice ()
از این متدهای تقسیم کردن یک متغیر متنی به آرایه‌ای از کاراکترها استفاده می‌شود.	split ()
از این متدهای برش تعداد معینی از کاراکترهای یک متغیر متنی بین دو نقطه مشخص استفاده می‌شود.	substring ()
از این متدهای نمایش متن یک متغیر رشته‌ای با حروف کوچک استفاده می‌شود.	toLowerCase ()
از این متدهای نمایش متن یک متغیر رشته‌ای با حروف بزرگ استفاده می‌شود.	toUpperCase ()
این متدهای رشته‌ای به اندازه دلخواه را به ابتدای رشته مورد نظر ما می‌چسبانند.	padStart ()
این متدهای رشته‌ای به اندازه دلخواه را به انتهای رشته مورد نظر ما می‌چسبانند.	padEnd ()

متدهای شی string

متدهای شی string : این متدهای شی string مقدار یک حرف (کاراکتر) در یک متغیر متنی، که شماره آن را توسط خاصیت `index` تعیین می‌کنیم، را بر می‌گرداند.

نکته : شماره گذاری حروف یک عبارت رشته‌ای در جاوا اسکریپت، از سمت چپ بوده و شماره گذاری از عدد صفر شروع می‌شود. بنابراین در کلمه ای مثل "Java Script" حرف شماره ۲، حرف U و شماره ۶ حرف C خواهد بود.

نکته ۲ : فاصله خالی بین حروف نیز یک کاراکتر حساب شده و دارای شماره خواهد بود.

Syntax	<code>stringobject.charAt (Index)</code> * <code>Index</code> = شماره حرف مورد نظر در متغیر
--------	--

مثال : در مثال زیر یک متغیر رشته ای به نام matn را ایجاد و مقدار دهی کرده ایم . سپس حروف شماره ۳ و ۹ آن را به کمک متده charAt نمایش داده ایم: مثال:

<pre><script type="text/javascript"> var matn = "Developer Studio" ; document.write (matn.charAt (3) + "
"); document.write (matn.charAt (10)); </script></pre>	کد
E,S	خروجی

متده **charCodeAt()** : این متده ، کد اسکی یک حرف (کاراکتر) در یک متغیر متنی ، که شماره آن را توسط خاصیت index تعیین می کنیم ، را بر می گرداند .

نکته : شماره گذاری حروف یک عبارت رشته ای در جاوا اسکریپت ، از سمت چپ بوده و از شماره گذاری از عدد صفر شروع می شود . بنابراین در کلمه ای مثل "Java Script" حرف شماره ۲ ، حرف ۷ و شماره ۷ حرف خواهد بود .

نکته ۲ : فاصله خالی بین حروف نیز یک کاراکتر حساب شده و دارای شماره خواهد بود .

Syntax	stringobject.charCodeAt (Index) * Index = شماره حرف مورد نظر در متغیر
--------	---

مثال : در مثال زیر یک متغیر رشته ای به نام matn را ایجاد و مقدار دهی کرده ایم . سپس کد اسکی حروف شماره ۳ و ۹ آن را به کمک متده charCodeAt نمایش داده ایم: مثال:

<pre><script type="text/javascript"> var matn = "Developer Studio" ; document.write (matn.charCodeAt (3) + "
"); document.write (matn.charCodeAt (10)); </script></pre>	کد
۱۰۱ , ۸۳	خروجی

متده concat() : از این متده برای چسباندن و اضافه کردن دو یا چند متغیر رشته ای به هم استفاده می شود . خروجی ، عبارت رشته ای جدیدی است شامل تمام متغیرهای رشته ای به هم اضافه شده .

راهنمایی : در syntax این متده stringobject نمایانگر اولین متغیر رشته ای بوده و string1 و string2 و ... سایر متغیرهایی هستند ، که می خواهیم به ترتیب به متغیر اول اضافه شوند . در این متده باید حداقل دو متغیر برای اضافه شدن به هم تعیین شوند .

Syntax	stringobject.concat (String1 , String2 , ...) * stringobject = متغیر رشته ای اول * String1 = متغیر رشته ای دوم * String2 = متغیر رشته ای شماره سوم که تعیین آن اختیاری است و ...
--------	---

مثال : در مثال زیر یک متغیر رشته ای به نام matn را ایجاد و مقدار دهی کرده ایم . سپس به وسیله متده concat دو متغیر دیگر را به آن اضافه کرده و متن جدید را در خروجی چاپ کرده ایم : مثال :

<script type ="text/javascript"> var matn = "Developer Studio " ; var str1 = "a Website " ; var str2 = "for Developers" ; document.write (matn.concat (str1 , str2)) ; </script>	کد
Developer Studio a Website for Developers	خروجی

متده indexOf()

این متده شماره مکان قرار گیری اولین نمونه یک حرف یا کلمه خاص مورد نظر در یک متغیر رشته ای ، را بر می گرداند . در این متده حرف یا کلمه مورد نظر توسط خاصیت searchvalue تعیین می شود . همچنین می توان مکان شروع جستجو در متغیر را نیز به وسیله خاصیت fromindex تعیین کرد . در این صورت محل آغاز جستجو به جای اول متغیر ، از کاراکتر تعیین شده خواهد بود .

نکته ۱ : شماره گذاری حروف یک عبارت رشته ای در جاوا اسکریپت ، از سمت چپ بوده و از شماره گذاری از عدد صفر شروع می شود . بنابراین در کلمه ای مثل "Java Script" حرف شماره ۲ ، حرف ۷ و شماره ۷ حرف ۸ خواهد بود .

نکته ۲ : فاصله خالی بین حروف نیز یک کاراکتر حساب شده و دارای شماره خواهد بود.

نکته ۳ : متدها `indexOf` به بزرگ یا کوچک بودن حروف حساس است.

نکته ۴ : چنانچه حرف یا کلمه مورد جستجو در متغیر رشته ای وجود نداشته باشد ، مقدار بازگشتی ۱ - خواهد بود.

Syntax	<code>stringobject.indexOf (searchvalue , fromindex)</code> * <code>searchvalue</code> = حرف یا کلمه مورد جستجو در متغیر رشته ای را تعیین می کند * <code>fromindex</code> = شماره مکان کاراکتری که می خواهیم عمل جستجو از آن آغاز شود را تعیین می کند
--------	---

مثال : در مثال به وسیله متدها `indexOf` به جستجوی یک حرف و دو عبارت در متغیر `matn` پرداخته و نتایج را در خروجی نشان داده ایم . به دلیل عدم وجود کلمه "studio" در متغیر رشته ای مثال ، نتیجه خروجی ۱ بوده است . در آخر هم به جستجو حرف e در شرایطی که شروع جستجو از کاراکتر شماره ۵ تعیین شده ، پرداخته ایم :

<code><script type = "text/javascript"> var matn = "Developer Studio" ; document.write (matn.indexOf ("e") + "
") ; document.write (matn.indexOf ("Studio") + "
") ; document.write (matn.indexOf ("studio") + "
") ; document.write (matn.indexOf ("e" , 5)) ; </script></code>	خروجی
	1
	10
	-1
	7

متدهای `lastIndexOf`: این متدهای شماره مکان قرار گیری آخرین نمونه یک حرف یا کلمه خاص مورد نظر در یک متغیر رشته ای ، را بر می گرداند . در این متدها حرف یا کلمه مورد نظر توسط خاصیت `searchvalue` تعیین می شود . همچنین می توان شماره مکان یک کاراکتر را نیز به وسیله خاصیت `fromindex` به متدهای `lastIndexOf` اعلام کرد ، که عملیات جستجو از سمت راست به چپ (بر عکس) ، براساس مکان آن کاراکتر صورت بگیرد .

نکته ۱ : شماره گذاری حروف یک عبارت رشته ای در جاوا اسکریپت ، از سمت چپ بوده و از شماره گذاری از عدد صفر شروع می شود . بنابراین در کلمه ای مثل "Java Script" حرف شماره ۲ ، حرف L و شماره ۷ حرف C خواهد بود .

نکته ۲ : فاصله خالی بین حروف نیز یک کاراکتر حساب شده و دارای شماره خواهد بود.

نکته ۳ : متدهای `lastIndexOf` به بزرگ یا کوچک بودن حروف حساس است.

نکته ۴ : چنانچه حرف یا کلمه مورد جستجو در متغیر رشته ای وجود نداشته باشد ، مقدار بازگشتی ۱ - خواهد بود

Syntax	stringobject.lastIndexOf (searchvalue , fromindex) * searchvalue = حرف یا کلمه مورد جستجو در متغیر رشته ای را تعیین می کند * fromindex = شماره مکان کاراکتری که می خواهیم عمل جستجو از آن به صورت بر عکس از راست به چپ آغاز شود را تعیین می کند
--------	--

مثال : در مثال به وسیله متده lastIndexOf به جستجوی یک حرف و دو عبارت در متغیر matn پرداخته و نتایج را در خروجی نشان داده ایم . به دلیل عدم وجود کلمه "studio" در متغیر رشته ای مثال ، نتیجه خروجی ۱ - بوده است . در آخر هم به جستجو حرف e در شرایطی که شروع جستجو از کاراکتر شماره ۵ تعیین شده ، پرداخته ایم: مثال :

<pre><script type="text/javascript"> var matn = "Developer Studio" ; document.write (matn.lastIndexOf ("e") + "
"); document.write (matn.lastIndexOf ("Studio") + "
"); document.write (matn.lastIndexOf ("studio") + "
"); document.write (matn.lastIndexOf ("e" , 5)); </script></pre>	کد
7 10 -1 3	خروجی

متده **match ()**: این متده ، جهت جستجو برای وجود یا عدم وجود یک حرف یا کلمه خاص در یک متغیر رشته ای استفاده می شود . عملکرد این متده بسیار شبیه متده indexOf است ، با این تفاوت که به جای شماره مکان قرا گیری حرف یا کلمه مورد جستجو ، خود آن را بر می گرداند .
 کلمه یا حرف مورد نظر توسط خاصیت **searchvalue** تعیین می شود .
 نکته ۱ : متده match به بزرگ یا کوچک بودن حروف حساس است.

نکته ۲ : چنانچه حرف یا کلمه مورد جستجو در متغیر رشته ای وجود نداشته باشد ، مقدار بازگشتی null خواهد بود.

Syntax	stringobject.match (searchvalue) * searchvalue = کلمه مورد جستجو در متغیر رشته ای را تعیین می کند
--------	--

مثال : در مثال به وسیله متده است match به جستجوی یک حرف و دو عبارت در متغیر matn پرداخته و نتایج را در خروجی نشان داده ایم . به دلیل عدم وجود کلمه "studio" در متغیر رشته ای مثال ، نتیجه خروجی null بوده است. مثال :

<script type ="text/javascript"> var matn = "Developer Studio" ; document.write (matn.match ("e") + " ") ; document.write (matn.match ("Studio") + " ") ; document.write (matn.match ("studio") + " ") ; </script>	کد
e Studio null	خروجی

متده replace() : از این متده ، برای جستجو یک حرف یا کلمه خاص مورد نظر در یک متغیر رشته ای و جایگزینی آن با یک مقدار جدید استفاده می شود . حرف یا کلمه مورد جستجو توسط خاصیت findstring و مقدار جایگزین توسط خاصیت newstring تعیین می شود.

Syntax	stringobject.replace (findatring , newstring) * findstring = کلمه یا حرف مورد جستجو * newstring = مقدار جدیدی که می خواهیم جایگزین مقدار قبلی شود
--------	--

مثال : در مثال زیر یک متغیر رشته ای به نام ex را ایجاد و مقدار دهی کرده ایم . سپس با استفاده از متده replace کلمه Java را با #C عوض کرده و خروجی جدید را بر روی صفحه نشان داده ایم . به دلیل عدم به کار بردن پارامتر g فقط اولین مورد کلمه Java با #C عوض شده و مورد دوم بدون تغییر باقی مانده است. مثال :

```

<script type ="text/javascript">
    var ex = "Java is a powerful programing language . Java support Object
Oriented Programming completely ";
    document.write ( ex.replace ( "Java" , "C#" ) );

//document.write ( ex.replace ( /Java/g , "C#" ) );/* حروف بزرگ با جایگزینی می کند*/
//document.write ( ex.replace ( /Java/i , "C#" ) );/* حروف کوچک را جایگزینی می کند*/
</script>

```

کد

متده است (): از این متده برای برش و جدا کردن بخشی از یک متغیر رشته ای ، بین دو نقطه مشخص استفاده می شود .

در این متده ، نقطه شروع را توسط خاصیت **start** و بر حسب شماره یک کاراکتر و نقطه پایان را نیز با خاصیت **stop** آن هم بر حسب شماره مکان یک کاراکتر دیگر ، در طول متغیر متنی تعیین می کنیم .

Syntax	stringobject.substr (start , stop)
	* start = شماره کاراکتر آغاز نقطه برش در طول متغیر
	* length = شماره کاراکتر نقطه پایان عملیات برش

نکته : تعیین نقطه پایان برای عملیات برش اختیاری بوده و می تواند تعیین نشود . در صورت عدم تعیین آن ، انتهای متغیر به عنوان نقطه پایان برش در نظر گرفته می شود .

مثال : در مثال زیر در ۲ حالت به برش متغیر متن پرداخته ایم . در حالت اول عملیات برش را بین کاراکترهای ۴ و ۹ تعیین کرده ایم . در حالت دوم عملیات برش را از کاراکتر ۴ آغاز کرده ایم ، ولی نقطه پایان آن را در نظر نگرفته ایم . در این حالت انتهای متغیر به عنوان نقطه پایان عملیات برش در نظر گرفته شده است :

<pre><script type="text/javascript"> var matn = "Developer Studio" ; document.write (matn.substrnig (4 , 6) + "
") ; document.write (matn.substrnig (4)) ; </script></pre>	کد
lo lопер studio	خروجی

متدها (slice): از این متدهای برای برش و جدا کردن بخشی از متن یک متغیر رشته‌ای و سپس ذخیره آن در یک متغیر جدید یا چاپ در خروجی استفاده می‌شود.

شماره کاراکتری که می‌خواهیم عمل برش از آن آغاز شود را توسط خاصیت start و شماره کاراکتری که می‌خواهیم عملیات برش در آنجا پایان پذیرد، را توسط خاصیت end تعیین می‌کنیم. مجموعه کاراکترهایی که بین این دو مقدار باشند، به عنوان خروجی نمایش داده می‌شود.

نکته ۱: تعیین نقطه پایانی اختیاری است و می‌تواند تعیین نشود. در صورت عدم تعیین آن، نقطه پایان برش متغیر، انتهای آن در نظر گرفته می‌شود.

نکته ۲: شماره گذاری حروف یک عبارت رشته‌ای در جاوا اسکریپت، از سمت چپ بوده و از شماره گذاری از عدد صفر شروع می‌شود. بنابراین در کلمه‌ای مثل "Java Script" حرف شماره ۲، حرف l و شماره ۷ حرف L خواهد بود.

نکته ۳: فاصله خالی بین حروف نیز یک کاراکتر حساب شده و دارای شماره خواهد بود.

Syntax	<pre>stringobject.slice (start , end)</pre> <p>* start = شماره مکان کاراکتر آغاز قسمت برش</p> <p>* end = شماره مکان کاراکتر انتهای قسمت برش</p>
--------	---

مثال: در مثال زیر در دو حالت به برش متغیر matn پرداخته ایم. در حالت اول نقطه شروع کاراکتر ۳ و نقطه پایان کاراکتر ۱۱ در نظر گرفته شده است. در حالت دوم نقطه شروع کاراکتر ۳ تعیین شده و نقطه پایان مشخص نشده است و در حالت دوم عمل برش تا انتهای متغیر انجام شده است:

<pre><script type="text/javascript"> var matn = "Developer Studio" ; document.write (matn.slice (3 , 11) + "
"); document.write (matn.slice (3)); </script></pre>	کد
eloper S eloper Studio	خروجی

متده است **split()** از این متده برای تقسیم کردن یک متغیر رشته ای به آرایه ای از متغیرهای رشته ای استفاده می شود . نتیجه خروجی ، شامل قطعات تقسیم شده متغیر رشته ای است که با کاما از هم جدا شده اند . در این متده توسط خاصیت **seprator** ، کاراکتر یا کلمه ای که می خواهیم عمل تقسیم شدن بر مبنای آن صورت بگیرد را تعیین کرده و توسط خاصیت **number** ، تعداد دفعات تقسیم متغیر رشته ای به قطعات کوچکتر را تعیین می کنیم . برای مثال اگر عدد ۳ وارد شود ، متغیر متنه به ۳ قطعه تقسیم می شود . تعیین این مقدار اختیاری است و می تواند تعیین نشود .

نکته ۱ : اگر خاصیت **seprator** ، "" تعیین شود ، عمل تقسیم بر حسب کلمات موجود در یک متغیر صورت می گیرد .

نکته ۲ : اگر خاصیت **seprator** ، """" تعیین شود ، عمل تقسیم بر حسب کلمات حروف در یک متغیر صورت می گیرد .

Syntax	<pre>stringobject.split (seprator , number)</pre> <p>* seprator = کاراکتر یا کلمه ای که می خواهیم عمل تقسیم بر حسب آن انجام شود .</p> <p>* end = تعداد دفعات تقسیم متغیر رشته ای به قطعات کوچکتر را تعیین می کند</p>
--------	--

مثال : در مثال زیر در چهار حالت به برش متغیر **matn** پرداخته ایم . در حالت اول ، تقسیم بر مبنای حرف **e** ، در حالت دوم تقسیم بر مبنای "" ، در حالت سوم تقسیم بر مبنای "" و تعداد دفعات تقسیم هم ۴ بار در نظر گرفته شده است . حالت اول در آخر هم تقسیم بر حسب حروف متغیر تعیین شده است .

<pre><script type="text/javascript"> document.write (matn.split ("e") + "
"); document.write (matn.split ("") + ">br /<"); document.write (matn.split ("", 4) + "
"); document.write (matn.split (" ")); </script></pre>	کد
<p>D,v,lop,r Studio D,e,v,e,l,o,p,e,r ,S,t,u,d,i,o D,e,v,e Developer,Studio</p>	خروجی

متدهای **search()**: این متدهای جستجو برای وجود یا عدم وجود یک حرف یا کلمه خاص در یک مغایر رشته ای استفاده می‌شود. عملکرد این متدهای بسیار شبیه متدهای **match** است، با این تفاوت که این متدهای **search** را می‌توان با پارامتر آ به کار برد. به کار بردن پارامتر آ با این متدهای باعث عدم حساسیت آن، به بزرگ یا کوچک بودن حروف می‌شود.

نکته ۱: کلمه یا حرف مورد نظر توسط خاصیت **searchstring** تعیین می‌شود.

نکته ۲: چنانچه حرف یا کلمه مورد جستجو در متغیر رشته ای وجود داشته باشد، این متدهای شماره اولین کاراکتر آن در طول متغیر رشته ای را برمی‌گردانند و در صورت عدم وجود حرف یا کلمه مقدار بازگشتی ۱- خواهد بود.

Syntax	stringobject.search (searchstring) * searchstring = کلمه یا حرف مورد جستجو در متغیر رشته ای را تعیین می‌کند
--------	--

مثال: در مثال به وسیله متدهای **match** و **search** به جستجوی یک حرف و دو عبارت در متغیر **matn** پرداخته و نتایج را در خروجی نشان داده ایم. توضیحات مثال:

۱. در دستور اول شماره مکان قرار گیری اولین مورد حرف e در متغیر برگشت داده شده است.
۲. در دستور دوم شماره مکان قرار گیری اولین کاراکتر کلمه Studio در متغیر برگشت داده شده است.
۳. در دستور سوم به دلیل حساسیت متدهای **search** کوچک یا بزرگ بودن حروف، کلمه studio در متغیر پیدا نشده و مقدار خروجی ۱- بوده است.
۴. در دستور چهارم به دلیل به کار بردن متدهای **search** با پارامتر آ، حساسیت آن نسبت بمتدهای **match** کوچک بودن حروف از بین رفته و شماره مکان قرار گیری اولین کاراکتر کلمه Studio برگشت داده شده است.

<pre><script type="text/javascript"> var matn = "Developer Studio" ; document.write (matn.search ("e") + "
"); document.write (matn.search ("Studio") + "
"); document.write (matn.search ("studio") + "
"); document.write (matn.search (/studio/i)); </script></pre>	کد
1 10 -1 10	خروجی

متدهای **toLowerCase()** و **toUpperCase()** از این متدها استفاده می‌شود. چنانچه در متغیر رشته ای مورد استفاده، حرفی به شکل بزرگ باشد، به صورت اتوماتیک به شکل کوچک آن تبدیل می‌شود.

Syntax	stringobject.toLowerCase ()
--------	------------------------------

مثال: در مثال زیر یک متغیر رشته ای را با متدهای **toLowerCase()** و **toUpperCase()** نمایش داده ایم:

<pre><script type="text/javascript"> var matn = "Developer Studio" ; document.write (matn.toLowerCase ()); </script></pre>	کد
developer studio	خروجی

متدهای **toLowerCase()** و **toUpperCase()** از این متدها استفاده می‌شود. چنانچه در متغیر رشته ای مورد استفاده، حرفی به شکل کوچک باشد، به صورت اتوماتیک به شکل بزرگ آن تبدیل می‌شود.

Syntax	stringobject.toLowerCase ()
--------	------------------------------

مثال: در مثال زیر یک متغیر رشته ای را با متدهای **toLowerCase()** و **toUpperCase()** نمایش داده ایم:

```
<script type ="text/javascript">
var matn = "Developer Studio" ;
document.write ( matn.toUpperCase ( ) ) ;
</script>
```

کد

DEVELOPER STUDIO

خروجی

:padStart() متد

این متد رشته‌ای به اندازه دلخواه را به ابتداء رشته‌ی مورد نظر ما می‌چسباند.

```
str.padStart(targetLength, padStarting);
```

طول رشته که مد نظر ما است را معین می‌کند و padStarting نیز رشته‌ی جدیدی است که به انتهای رشته‌ی اصلی ما می‌چسبد و به اندازه طول معین شده در انتهای قرار خواهد گرفت.

```
let str = "5";
str .padStart(2, "0"); => "05"
```

:padEnd() متد

این متد رشته‌ای به اندازه دلخواه را به ابتداء رشته‌ی مورد نظر ما می‌چسباند.

```
str.padEnd(targetLength, padEnding);
```

طول رشته که مد نظر ما است را معین می‌کند و padEnding نیز رشته‌ی جدیدی است که به انتهای رشته‌ی اصلی ما می‌چسبد و به اندازه طول معین شده در انتهای قرار خواهد گرفت.

```
let str = "5";
str .padEnd(2, "0"); => "50"
```

اعتبارسنجی فرم ها در JavaScript

از JavaScript می توان برای اعتبارسنجی فرم ها، قبل از اینکه به سرور ارسال شود استفاده نمود.

فرم های داده ای که توسط JavaScript اعتبارسنجی می شوند به صورت معمول می توانند شامل موارد زیر باشد:

آیا فیلد های الزامی (ستاره دار) پر شده است؟

آیا کاربر آدرس ایمیل را به فرمت صحیح وارد کرده است؟

آیا کاربر، داده صحیح را وارد کرده است؟

آیا کاربر در یک فیلد عددی، فقط کاراکترهای عددی را وارد کرده است؟

فیلد های الزامی (Required)

تابع زیر، خالی بودن فیلد را چک می کند. اگر خالی باشد، پیغام

"First name must be filled out" ظاهر می شود و فرم به سرور ارسال نمی شود:

```
function validateForm()
{
var x=document.forms["myForm"]["fname"].value;
if (x==null || x=="")
{
alert("First name must be filled out");
return false;
}
}
```

تابع بالا، زمانی فراخوانی می شود که، رویداد "onsubmit" فرم اتفاق بیافتد:

اعتبار سنجی ایمیل (E-mail Validation)

تابع زیر، فرمت معمول آدرس ایمیل را چک می کند.

فرمت صحیح آدرس ایمیل شامل موارد زیر است:

1. باید شامل یک علامت "@" باشد.

2. باید حداقل یک نقطه(.) داشته باشد.

3. علامت "@" نباید اولین کاراکتر باشد.

4. آخرین نقطه، باید بعد از علامت "@" ظاهر شود.

5. بعد از آخرین نقطه باید حداقل ۲ کاراکتر بیاید.

```
function validateForm()
{
var x=document.forms["myForm"]["email"].value;
var atpos=x.indexOf("@");
var dotpos=x.lastIndexOf(".");
if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)
{
alert("Not a valid e-mail address");
return false;
}
}
```

تابع بالا، زمانی فراخوانی می شود که، رویداد "onsubmit" فرم اتفاق بیافتد:

```
<form name="myForm" action="js_validation.php" onsubmit="return
validateForm();" method="post">
Email: <input type="text" name="email">
<input type="submit" value="Submit">
</form>
```

آرایه ها در JavaScript

شیء آرایه یک نوع خاص از متغیر هاست که می تواند چندین داده را در قالب یک نام در خود ذخیره کند.

آرایه چیست؟

آرایه یک متغیر خاص است که می توانید بیشتر از یک مقدار را در یک زمان در آن ذخیره نمایید.

اگر لیستی از آیتم ها داشته باشید (برای مثال، یک لیست از نام ماشین ها)، و بخواهید هر آیتم را در یک متغیر تنها ذخیره نمایید، می توان مانند زیر عمل نمود:

```
var car1="pride";
```

```
var car2="206";
```

```
var car3="BMW";
```

حالا:

اگر لیست شما بیشتر از ۳ آیتم باشد مثلاً ۳۰۰ تا چه کار می کنید؟

اگر در این لیست به دنبال یک ماشین خاص باشید چه کار می کنید؟

در اینجا بهترین راه حل استفاده از آرایه ها است.

یک آرایه می تواند مقادیر متغیرها را تحت یک نام برای شما نگه دارد. و شما از طریق ایندکس آرایه می توانید به مقادیر دسترسی داشته باشید.

هر آیتم در آرایه ایندکس منحصر به فردی برای خود دارد که به راحتی از طریق ایندکس می توانید به مقادیر دسترسی پیدا کنید.

ایجاد آرایه

آرایه ها به سه روش ایجاد می شوند:

۱: با قاعده

```
var myCars=new Array();  
myCars[0]="pride";  
myCars[1]="206";  
myCars[2]="BMW";
```

۲: خلاصه شده

```
var myCars=new Array("pride","206","BMW");
```

۳: تحت الفظی

```
var myCars=["pride","206","BMW"];
```

در کدهای بالا، یک شیء آرایه با نام `myCars` ایجاد می شود.

دسترسی به عناصر آرایه

هر آیتم در آرایه ایندکس منحصر به فردی برای خود دارد که به راحتی از طریق ایندکس می توانید به مقادیر دسترسی پیدا کنید.

کد زیر، مقدار اولین عنصر آرایه `myCars` را در متغیر `name` قرار می دهد:

```
var name=myCars[0];
```

کد زیر، اولین عنصر آرایه `myCars` را تعریف می کند:

```
myCars[0]="Opel";
```

توجه: ایندکس آرایه از صفر شروع می شود، یعنی اولین آیتم [۰] است، دومین آیتم [۱] و ...

ذخیره اشیاء مختلف در یک آرایه

تمام متغیرها، عناصر آرایه و توابع در JavaScript شیء محسوب می‌شوند.

شما می‌توانید، انواع مختلف متغیرها، خروجی یک تابع و یا یک آرایه را در آرایه‌ای دیگر ذخیره نمایید:

```
myArray[0]=Date.now;  
myArray[1]=myFunction;  
myArray[2]=myCars;
```

خصوصیت **length** و متدهای **indexof**

در مثال زیر، خصوصیت **length**، تعداد عناصر آرایه myCars را در متغیر x قرار می‌دهد و تابع **indexof()**، ایندکس عنصر مشخص شده را برابر می‌گرداند:

```
var myCars=new Array("Saab","Volvo","BMW");  
  
var x=myCars.length // the number of elements in myCars  
var y=myCars.indexOf("Volvo") // the index position of "Volvo"
```

چاپ آرایه

```
var i;  
  
var mycars = new Array();  
  
mycars[0] = "Saab";  
mycars[1] = "Volvo";  
mycars[2] = "BMW";  
  
for (i=0;i<mycars.length;i++)  
{  
    document.write(mycars[i] + "<br>");  
}
```

ایجاد متدهای جدید

یک سازنده (constructor) عمومی در JavaScript است. از این طریق می‌توان، برای هر شیءی در JavaScript یک خصوصیت یا متدهای جدید ساخت.

```
Array.prototype.ucase=function()  
{  
    for (i=0;i<this.length;i++)  
    {this[i]=this[i].toUpperCase();}  
}
```

در مثال بالا، یک متدهای جدید با نام `ucase` ساخته شده است که مقادیر عناصر آرایه را به حروف بزرگ تبدیل می‌کند.

حلقه `for...in`

یک دستور محدود است، این دستور برای شمردن تک تک خصلت‌های یک شی به کار می‌رود.

به تعداد دفعات معین خاصیت‌های (property) یک شی را تکرار می‌کند، برای حرکت درون اعضای یک آرایه یا مجموعه property‌های یک شی استفاده می‌شود. به ازای خواندن هر یک از اعضا آرایه یا یکی از property‌های شی مورد نظر، یکبار statement داخل حلقه اجرا خواهد شد.

حلقه `for...in` در جاوا اسکریپت :

از حلقه `for ... in` در جاوا اسکریپت، برای حرکت در درون اعضای یک آرایه یا مجموعه خواص یک شی استفاده می‌شود. به ازای خواندن هر یک از اعضا آرایه یا یکی از خواص شی مورد نظر، یکبار دستورات درون حلقه اجرا خواهد شد.

تعداد دفعات تکرار دستورات حلقه ، برابر با تعداد اعضای آرایه و یا تعداد خواص شی مورد نظر است . در این حلقه معمولا از یک متغیر به عنوان شمارنده یا اندیس آرایه استفاده می شود .
شکل کلی تعریف یک حلقه `for...in` به صورت زیر است :

```
(نام یک آرایه / مجموعه خواص یک شی in متغیر)
{
    دستورات بدن حلقه
}

var person = {fname:"John", lname:"Doe", age:25};

var text = "";
var x;
for (x in person) {
    text += person[x];
}
```

```
for(sprop in window)
    document.write(sprop+"<br>");

var myCars=new Array("pride","206","BMW");

var text = "";
var x;
for (x in myCars) {
    text += myCars[x];
}
document.write(text);
```

حلقه `:for ...of`

حلقه `for...of` یکی دیگر از ساختارهای تکرار در جاوااسکریپت می باشد که مخصوصاً برای آرایه ها، رشته ها و مجموعه ها طراحی شده است. حلقة `for...of` با هر بار گردش در بین اجزاء، مقادیر هر یک از آنها را در داخل یک متغیر موقتی قرار می دهد و شما می توانید بواسطه این متغیر به مقادیر دسترسی پیدا کنید.

```
for (var temporaryVar of array/string) {  
    code to execute;  
}
```

متغیری است که مقادیر اجزای آرایه را در خود نگهداری می‌کند. سپس کلمه کلیدی **of** و بعد از آن نام آرایه، رشته و یا مجموعه را می‌نویسیم.

```
var numbers = [ 1, 2, 3, 4, 5 ];  
for (var n of numbers) {  
    console.log("Number ", n);  
}
```

متدهای آرایه:

اتصال دو یا چند آرایه - **concat()**

```
<body>  
    <p id="demo">Click the button to join three arrays.</p>  
    <button onclick="myFunction()">Try it</button>  
  
<script>  
function myFunction(){  
    var name1= ["ali", "reza"];  
    var name2 = ["hossein", "nima", "ashkan"];  
    var name3 = ["amir"];  
    var name = name1.concat(name2,name3);  
    var x=document.getElementById("demo");  
    x.innerHTML=name;  
}  
</script>  
</body>
```

تمام عناصر آرایه در یک متغیر - join()

```
<body>

<p id="demo">Click the button to join the array elements into a string.</p>

<button onclick="myFunction()">Try it</button>

<script>

function myFunction()

{

var fruits = ["Banana", "Orange", "Apple", "Mango"];

var x=document.getElementById("demo");

x.innerHTML=fruits.join();

}

</script>

</body>
```

حذف آخرین عنصر آرایه - pop()

```
<body>

<p id="demo">Click the button to remove the last array element.</p>

<button onclick="myFunction()">Try it</button>

<script>

var fruits = ["Banana", "Orange", "Apple", "Mango"];

function myFunction() {

fruits.pop();

var x=document.getElementById("demo");
```

```
x.innerHTML=fruits; }

</script>

</body>
```

اضافه کردن یک عنصر جدید به انتهای آرایه - `push()`

```
<body>

<p id="demo">Click the button to add a new element to the array.</p>

<button onclick="myFunction()">Try it</button>

<script>

var fruits = ["Banana", "Orange", "Apple", "Mango"];

function myFunction(){

fruits.push("Kiwi")

var x=document.getElementById("demo");

x.innerHTML=fruits;}

</script>

</body>
```

معکوس کردن ترتیب عناصر آرایه - `reverse()`

```
<body>

<p id="demo">Click the button to reverse the order of the elements in the array.</p>

<button onclick="myFunction()">Try it</button>

<script>

var fruits = ["Banana", "Orange", "Apple", "Mango"];

function myFunction(){

fruits.reverse();

var x=document.getElementById("demo");
```

```
x.innerHTML=fruits;}  
  
</script>  
  
</body>
```

حذف اولین عنصر آرایه - shift()

```
<body>  
  
    <p id="demo">Click the button to remove the first element of the array.</p>  
  
    <button onclick="myFunction()">Try it</button>  
  
<script>  
  
    var fruits = ["Banana", "Orange", "Apple", "Mango"];  
  
    function myFunction(){  
  
        fruits.shift();  
  
        var x=document.getElementById("demo");  
  
        x.innerHTML=fruits;}  
  
</script>  
  
</body>
```

برش و انتخاب یک قسمت از آرایه - slice()

```
<body>  
  
    <p id="demo">Click the button to extract the second and the third elements from the </p>  
  
    <button onclick="myFunction()">Try it</button>  
  
<script>  
  
    function myFunction(){  
  
        var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];  
  
        var citrus = fruits.slice(1,3);  
  
        var x=document.getElementById("demo");
```

```
x.innerHTML=citrus; }

</script>

</body>
```

مرتب کردن آرایه بر اساس حروف الفبا و صعودي - sort()

```
<body>

<p id="demo">Click the button to sort the array.</p>

<button onclick="myFunction()">Try it</button>

<script>

function myFunction(){

    var fruits = ["Banana", "Orange", "Apple", "Mango"];

    fruits.sort();

    var x=document.getElementById("demo");

    x.innerHTML=fruits;}

</script>

</body>
```

مرتب کردن آرایه عددی و صعودي - sort()

```
<body>

<p id="demo">Click the button to sort the array.</p>

<button onclick="myFunction()">Try it</button>

<script>

function myFunction(){

    var points = [40,100,1,5,25,10];

    points.sort(function(a,b){return a-b});

    var x=document.getElementById("demo");
```

```
x.innerHTML=points;}\n</script>\n</body>
```

مرتب کردن آرایه عددی و نزولی - sort()

```
<body>\n  <p id="demo">Click the button to sort the array.</p>\n  <button onclick="myFunction()">Try it</button>\n<script>\n  function myFunction(){\n    var points = [40,100,1,5,25,10];\n    points.sort(function(a,b){return b-a});\n    var x=document.getElementById("demo");\n    x.innerHTML=points;}\n</script>\n</body>
```

اضافه کردن یک عنصر جدید در مکان ۲ یک آرایه - splice()

```
<body>\n  <p id="demo">Click the button to add elements to the array.</p>\n  <button onclick="myFunction()">Try it</button>\n<script>\n  function myFunction(){\n    var fruits = ["Banana", "Orange", "Apple", "Mango"];\n    fruits.splice(2,0,"Lemon","Kiwi");\n    var x=document.getElementById("demo");\n  }
```

```
x.innerHTML=fruits;}\n</script>\n</body>
```

تبدیل یک آرایه به یک رشته - `toString()`

```
<body>\n<p id="demo">Click the button to convert the array into a String.</p>\n<button onclick="myFunction()">Try it</button>\n<script>\nfunction myFunction(){\n    var fruits = ["Banana", "Orange", "Apple", "Mango"];\n    fruits.toString();\n    var x=document.getElementById("demo");\n    x.innerHTML=fruits;}\n</script>\n</body>
```

اضافه کردن یک عنصر جدید به ابتدای یک آرایه - `unshift()`

```
<body>\n<p id="demo">Click the button to add elements to the array.</p>\n<button onclick="myFunction()">Try it</button>\n<script>\nfunction myFunction(){\n    var fruits = ["Banana", "Orange", "Apple", "Mango"];\n    fruits.unshift("Lemon","Pineapple");\n    var x=document.getElementById("demo");\n}
```

```

x.innerHTML=fruits;}

</script>

<p><b>Note:</b> The unshift() method does not work properly in Internet Explorer 8 and earlier, the values will be inserted, but the return value will be <em>undefined</em>.</p>

</body>

```

:map() متدها

عملکرد این متدها را با یک مثال توضیح خواهیم داد. فرض کنید یک آرایه شامل چند شی دریافت کرده‌اید که هر یک از آن‌ها نماینده یک شخص هستند. با این حال شما در انتها به یک آرایه شامل id هر شخص نیاز دارید:

```

let pilots = [
  { id: 1, name: "Ali Kazemi", faction: "Rookie" },
  { id: 6, name: "Mehran Talebi", faction: "Veteran" },
  { id: 23, name: "Mina Salimi", faction: "Rookie" },
];

```

چندین روش برای رسیدن به آرایه وجود دارد. ممکن است با استفاده از `for ... of` و `forEach` ساده به مقصود برسیم.

یکی از روش‌ها استفاده از `forEach`

```

let pilotsId = [];
pilots.forEach(function (pilot) {
  pilotsId.push(pilot.id);
}

```

حالا استفاده از `:map()`

```
let pilotsId = pilots.map(function (pilot) { return pilot.id });
```

طريق کار متدها map() را ديديم. با آرگومانی به نام call back هر مقدار از آرایه را می گيرد و همه مقادير جديده را در آرایه حاصل باز می گرداند.

متدها :filter()

همانطورکه از نام اين متدها آشكار است. با استفاده از آن می توان به بخشی از عناصر دلخواه در آرایه دسترسی يافت.
فرض کنيم، آرایه‌اي داريم و می خواهيم خلبانان تازه کار و کهنه کار را برای ما نمایش دهد. با متدها filter() به راحتی می توان اين کار را انجام داد.

```
let rookie = pilots.filter(function (pilot) {  
    return pilot.faction === "Rookie";  
});  
  
let veteran = pilots.filter(function (pilot) {  
    return pilot.faction === "Veteran";  
});
```

متدها :reduce()

این متدها مختص آرایه است و می توانيم با کمک آن آرایه‌ها را ساده کنیم.
ساده کردن آرایه یعنی چی؟ یعنی آرایه آن قدر کوچک و ساده شود که به یک مقدار واحد برسد. مثلا در آرایه‌اي از اعداد مجموع تمام عناصر را می خواهيم که طبیعتا خروجي ما عددی تک مقداري است.

```
array.reduce(callBack, initialValue);
```

تابع دلخواه ما است که برای ساده کردن به کار می رود و دارای چهار آرگومان است که اصلی‌ترین آرگومان آن callBack است. currentValue و accumulator

خروجی callBack را با هر پیمايش روی اعضای آرایه در خود ذخیره می کند و نهايتا آن را به خروجي می فرستد.
عضوی از آرایه است که در حال اجرا و پردازش است.

اين آرگومان اختياری است و مقداری که به آن می دهیم به عنوان مقدار اولیه برای accumulator در نظر گرفته می شود. اگر از آن صرف نظر کنيم مقدار اولیه برابر با اولین عنصر آرایه ما می شود.

```

let numbers = [1, 2, 3, 4];

let sum = numbers.reduce((pre, current) => {

    return pre + current;

});

console.log(sum);

```

شی (Math) در جاوا اسکریپت:

از شی Math در جاوا اسکریپت برای انجام امور ریاضی استفاده می شود . این شی شامل تعداد زیادی متدها و تابع برای انجام کارهای ریاضی می باشد . همچنین این شی دارای تعدادی ثابت عددی مثل عدد π است که میتوانید از آنها در محاسبات خود استفاده نمایید .

شكل کلی تعریف و استفاده از یک ثابت عددی یا متدهای Math در جاوا اسکریپت به صورت زیر است :

Syntax	$\text{Math.} \text{نام ثابت عددی.}$ $\text{Math.}(\text{نام متدها})$ یا عدد پی (3.14) را به متغیر نسبت می دهد // : مثال $\text{var MyNum = Math.PI;}$ جذر عدد داخل پرانتز را حساب کرده و به متغیر نسبت می دهد // : مثال $\text{var MyNum = Math.sqrt(9);}$
--------	--

ثابت های عددی شی:

همانطور که اشاره شد شی Math ، دارای تعدادی ثابت عددی مثل عدد π است که از آنها می توانید در انجام امور ریاضی و محاسبات خود استفاده نمایید . در لیست زیر این ثابت ها قرار داده شده اند .

ثابت عددی	مقدار حدودی	کاربرد
E	2.718	این ثابت عددی مقدار عدد حقیقی e را برابر می گرداند .
LN2	0.693	این ثابت عددی ، مقدار لگاریتم طبیعی ۲ را برابر می گرداند .
LN10	2.302	این ثابت عددی ، مقدار لگاریتم طبیعی ۱۰ را برابر می گرداند .
LOG2E	1.442	این ثابت عددی ، مقدار لگاریتم عدد e را بر مبنای ۲ برابر می گرداند .
LOG10E	۰.۴۳۴	این ثابت عددی ، مقدار لگاریتم عدد e را بر مبنای ۱۰ برابر می گرداند .

متدها و تابع های عددی شی : Math

پس از آشنایی با ثابت های عددی شی Math ، در این قسمت به معرفی و توضیح تابع ها و متدهای ریاضی این شی می پردازیم که می توانید از آنها در انجام امور ریاضی و محاسبات خود استفاده نمایید . در لیست زیر این تابع ها قرار داده شده اند . برای دریافت توضیحات بیشتر و مثال های عملی بر روی نام هر تابع کلیک نمایید

نام تابع	کاربرد
$\text{abs}(x)$	این تابع قدر مطلق عددی x را برابر می‌گرداند.
$\text{acos}(x)$	این تابع آرک کوسینوس عدد x را برابر می‌گرداند.
$\text{asin}(x)$	این تابع آرک سینوس عدد x را برابر می‌گرداند.
$\text{atan}(x)$	این تابع آرک تانژانت عدد x را برابر می‌گرداند.
$\text{ceil}(x)$	این تابع عدد x را به بالا گرد کرده و نزدیک ترین عدد صحیح بدان را برابر می‌گرداند.
$\cos(x)$	این تابع کوسینوس x را برابر می‌گرداند.
$\exp(x)$	این تابع مقدار عدد e به توان x را برابر می‌گرداند.
$\text{floor}(x)$	این تابع عدد x را به پایین گرد کرده و نزدیک ترین عدد صحیح بدان را برابر می‌گرداند.
$\log(x)$	این تابع لگاریتم x را برمبنای عدد e برابر می‌گرداند.
$\text{max}(a,b, \dots, x,y,z)$	این تابع بزرگترین عددی را که به عنوان پارامتر به آن ارسال شده است را برابر می‌گرداند.
$\text{min}(a,b, \dots, x,y,z)$	این تابع کوچکترین عددی را که به عنوان پارامتر به آن ارسال شده است را برابر می‌گرداند.
$\text{pow}(a,b)$	این تابع عدد a را به توان b رسانده و نتیجه را برابر می‌گرداند.
$\text{random}()$	این تابع یک عدد تصادفی بین ۰ و ۱ را انتخاب کرده و به عنوان خروجی بر می‌گرداند.
$\text{round}(x)$	این تابع عدد x را گرد کرده و به نزدیکترین عدد صحیح بدان تبدیل می‌کند.
$\text{trunc}(x)$	این متدها قسمت صحیح عدد اعشاری را برمی‌گردانند.
$\sin(x)$	این تابع مقدار سینوس x را برابر می‌گرداند.
$\tan(x)$	این تابع مقدار تانژانت x را برابر می‌گرداند.

تمام متدهای شیء Math

برگرداندن قدر مطلق یک عدد: **abs(x)**

`Math.abs(-7.25);`

نتیجه کد بالا:

7.25

متد **ceil()**: گرد کردن یک عدد به بالا، به سمت نزدیک ترین عدد صحیح:

`Math.ceil(1.4)`

نتیجه کد بالا:

تعریف و کاربرد

متد **ceil()**، یک عدد را به بالا به سمت نزدیک ترین عدد صحیح گرد می کند و نتیجه را برمی گرداند.

در صورتی که آرگومان یک عدد صحیح باشد، عمل گرد شدن اتفاق نمی افتد.

استفاده از متد **ceil()** بر روی اعداد مختلف:

```
var a = Math.ceil(0.60);
var b = Math.ceil(0.40);
var c = Math.ceil(5);
var d = Math.ceil(5.1);
var e = Math.ceil(-5.1);
var f = Math.ceil(-5.9);
```

مقدار های a و b و c و d و e و f

1

1

5

6

-5

-5

متد floor: گرد کردن یک عدد به سمت پایین، به نزدیک ترین عدد صحیح:

```
Math.floor(1.6);
```

خروجی کد بالا: ۱

تعریف و کاربرد

متد **floor()**، یک عدد را به سمت پایین به نزدیک ترین عدد صحیح گرد می کند و نتیجه را برمی گرداند.

در صورتی که آرگومان یک عدد صحیح باشد، گرد کردن صورت نمی پذیرد.

استفاده از متد **floor()** بر روی اعداد مختلف:

```
var a = Math.floor(0.60);
var b = Math.floor(0.40);
var c = Math.floor(5);
var d = Math.floor(5.1);
var e = Math.floor(-5.1);
var f = Math.floor(-5.9);
```

خروجی a و b و c و d و e و f:

0
0
5
5
-6
-6

متد max: برگرداندن بزرگترین عدد:

```
Math.max(5, 10);
```

نتیجه کد بالا: ۱۰

تعريف و کاربرد

متده است، بزرگترین عدد داده شده به آن را برمی گرداند.

برگرداندن بزرگترین عدد:

```
var a = Math.max(5, 10);
var b = Math.max(0, 150, 30, 20, 38);
var c = Math.max(-5, 10);
var d = Math.max(-5, -10);
var e = Math.max(1.5, 2.5);
```

خروجی a و b و c و d و e

10
150
10
-5
2.5

متده است: برگرداندن کوچکترین عدد:

```
Math.min(5, 10);
```

نتیجه کد بالا : ۵

تعريف و کاربرد

متده است، کوچکترین عدد داده شده به آن را برمی گرداند.

برگرداندن کوچکترین عدد:

```
var a = Math.min(5, 10);
var b = Math.min(0, 150, 30, 20, 38);
var c = Math.min(-5, 10);
var d = Math.min(-5, -10);
var e = Math.min(1.5, 2.5);
```

خروجی a و b و c و d و e

```
5  
0  
-5  
-10  
1.5
```

متد pow: برگرداندن عدد ۴ به توان ۳: $(4^4)^4$

```
Math.pow(4, 3);
```

نتیجه کد بالا: 64

متد random: برگرداندن یک عدد تصادفی بین ۰ و ۱ (خود ۱ شامل نمی شود):

```
Math.random();
```

نتیجه برابر است با: 0.07487111231511756

متد round: گرد کردن یک عدد به نزدیک ترین عدد صحیح:

```
Math.round(2.5);
```

نتیجه کد بالا: ۳

متد trunc: این متاد تنها قسمت صحیح عدد اعشاری را برمی گرداند.

```
Math.trunc(2.5);
```

نتیجه کد بالا: ۲

شی Date

از این شی برای دستکاری و کار با تاریخ و زمان استفاده می شود . به طور کلی هر متغیری از نوع تاریخ و زمان نمونه ای از شی Date خواهد بود .

نحوه تعریف یک متغیر جدید از نوع Date

از شیء Date برای کار کردن بر روی تاریخ و زمان استفاده می شود.

اشیاء Date به وسیلهٔ دستور new Date() ایجاد می شوند.

چهار راه برای تعریف یک شیء Date وجود دارد:

```
var d = new Date();  
  
var d = new Date(milliseconds);  
var d = new Date(dateString);  
var d = new Date(year, month, day, hours, minutes, seconds, milliseconds);
```

برای تعریف یک متغیر جدید از نوع تاریخ و زمان از تابع Date ، به صورت کلی زیر استفاده می شود :

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

نکته ۱ : هر متغیری از نوع Date ، که به روش فوق ایجاد شود ، زمان و تاریخ جاری سیستم در لحظه ایجاد خود را ، به عنوان مقدار پیش فرض در درون خود نگهداری می کند . این مقدار شامل مخفف نام روز جاری، مخفف نام ماه جاری ، شماره روز جاری در ماه ، سال جاری ، ساعت دقیق که به صورت ساعت ، دقیقه و ثانیه است و فرمات ساعت خواهد بود .

مثال : در مثال زیر یک متغیر به نام NowTime را به روش اشاره شده ایجاد و مقدار دهی کرده ایم . سپس به وسیلهٔ دستور document.write مقدار آن را بر روی صفحه نمایش داده ایم . این متغیر هر بار که صفحه مجدداً بار گذاری شده و یا Refresh می شود ، مقدار آن دوباره به تاریخ و ساعت جاری سیستم Set شده و تغییر می کند . بنابراین مقدار آن ثابت نیست و با هر بار لود شدن صفحه و اجرای مجدد اسکریپت ، ساعت و تاریخ جدید جایگزین مقدار قبلی می شود . برای درک بهتر هر چند لحظه یکبار صفحه را Refresh کرده و به مقدار خروجی دقت کنید :

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

کد

Mon Jul 01 2019 03:00:33 GMT-0700 (Pacific Daylight Time)

خروجی

نکته ۲ : همچنین می توان در هنگام تعریف یک متغیر جدید از نوع `Date` ، یک تاریخ مورد نظر را به عنوان تاریخ ذخیره شده در آن متغیر را در پرانتز جلوی تابع `Date` تعریف کرد . در این صورت سیستم به صورت اتوماتیک ، تاریخ تعیین شده را به فرمت مورد استفاده در جاوا اسکریپت تبدیل کرده و در متغیر ذخیره خواهد کرد.

مثال : در مثال زیر ابتدا یک متغیر به نام `MyDate` را ایجاد کرده و یک تاریخ را نیز به عنوان مقدار برای آن در نظر گرفته ایم . سپس آن متغیر را بر روی صفحه در خروجی چاپ کرده ایم . همانطور که می بینید ، مقدار خروجی برابر با مقدار تعیین شده است و ارتباطی با تاریخ و ساعت جاری سیستم ندارد:

```
< script type="text/javascript" >
var MyDate = new Date ( "12/05/2004" ) ;
document.write ( MyDate ) ;
</script >
```

کد

Sun Dec 05 2004 00:00:00 GMT-0800 (Pacific Standard Time)

خروجی

متدهای `Date()`

متد	توضیحات
<code>getDate()</code>	از این متد در Javascript ، برای برگرداندن روز نسبت به ماه استفاده می شود(از ۱ تا ۳۱).
<code>getDay()</code>	از این متد در Javascript ، برای برگرداندن روز نسبت به هفته استفاده می شود(از ۰ تا ۶).
<code>getFullYear()</code>	از این متد در Javascript ، برای برگرداندن سال استفاده می شود.
<code>getHours()</code>	از این متد در Javascript ، برای برگرداندن ساعت استفاده می شود(از ۰ تا ۲۳).
<code>getMilliseconds()</code>	از این متد در Javascript ، برای برگرداندن میلی ثانیه ها استفاده می شود(از ۰ تا ۹۹۹).

getMinutes()	از این متده در Javascript، برای برگرداندن دقیقه ها استفاده می شود(از ۰ تا ۵۹).
getMonth()	از این متده در Javascript، برای برگرداندن ماه استفاده می شود(از ۰ تا ۱۱).
getSeconds()	از این متده در Javascript، برای برگرداندن ثانیه ها استفاده می شود(از ۰ تا ۵۹).
toLocaleDateString()	از این متده در Javascript، برای برگرداندن قسمت مربوط به تاریخ از یک تاریخ کامل که فرمات نوشت آن بصورت معمولی نوشتن تاریخ است، استفاده می شود.
toLocaleTimeString()	از این متده در Javascript، برای برگرداندن قسمت مربوط به زمان از یک تاریخ کامل که فرمات نوشت آن بصورت معمولی نوشتن زمان است، استفاده می شود.
toString()	از این متده در Javascript، برای تبدیل قسمت زمان از یک شیء تاریخ، به یک رشته استفاده می شود.

getDate() متده

این متده، شماره روز جاری را در ماه تاریخ متغیر زمانی مورد استفاده را بر می گرداند. برای مثال اگر امروز روز هفدهم ماه جاری باشد، عدد بازگشته ۱۷ خواهد بود.

Syntax	DateObject = DateObject.getDate () *متغیر زمانی مورد نظر
--------	--

مثال : در مثال زیر تاریخ جاری سیستم را توسط تابع () در متغیر MyDate ذخیره کرده ایم . سپس توسط متده getDate ، شماره روز جاری را بر روی صفحه نمایش داده ایم :

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

27

مثال ۲ : همانطور که می دانید ، می توان به یک متغیر زمانی مقداری دلخواه و بدون ارتباط با تاریخ جاری سیستم داد . در صورت مقدار دهی و اعمال متده getDate به یک متغیر ، این متده شماره روز تاریخ ذخیره

شده را برمی گرداند . برای مثال در کد زیر ، متغیر SomeDate را ابتدا مقدار دهی کرده و سپس شماره روز تاریخ آن را باز گردانده ایم :

```
<script type ="text/javascript">
var SomeDate = new Date ( "12/09/2008" );
document.write ( SomeDate.getDate ( ) );
</script>
```

9

متدهای getDay

متدهای getDay() از یک تاریخ مشخص روز را نسبت به هفته برمی گرداند(از ۰ تا ۶).
نکته: یکشنبه(sunday) هم ارز با ۰ است و دوشنبه(monday) هم ارز با ۱ و به همین ترتیب الی آخر.

```
<script type ="text/javascript">
var d = new Date();
var n = d.getDay();

document.write(n);
</script>
```

0

مثال (متدهای getDay)

روز را از هفته برمی گرداند(نه فقط منحصرأ یک عدد):

```
var d = new Date();
var weekday = new Array(7);
weekday[0]= "Sunday";
weekday[1] = "Monday";
weekday[2] = "Tuesday";
weekday[3] = "Wednesday";
weekday[4] = "Thursday";
weekday[5] = "Friday";
weekday[6] = "Saturday";
var n = weekday[d.getDay()];
```

Sunday

: خروجی n در کد بالا

متدهای Date

```
<script type ="text/javascript">  
var d = new Date();  
var n = d.getFullYear();  
</script>
```

2014

مثال (متدهای Date)

سال را از یک تاریخ مشخص برمی گرداند:

```
var d = new Date("July 21, 1983 01:15:00");  
var n = d.getFullYear();
```

1983 در کد بالا:

n خروجی

متدهای Date

ساعت را نسبت به زمان محلی بر می گرداند:

```
<script type ="text/javascript">  
var d = new Date();  
var n = d.getHours();  
</script>
```

2014

مثال (متدهای Date)

ساعت را از یک تاریخ و زمان مشخص برمی گرداند:

```
var d = new Date("July 21, 1983 01:15:00");  
var n = d.getHours();
```

1 در کد بالا:

n خروجی

(getHours متد)

استفاده از () getHours() و () getMinutes() و () getSeconds() برای نمایش دادن زمان:

متدهای Date

```
<script type ="text/javascript">  
var d = new Date();  
var n = d.getMilliseconds();  
</script>
```

2014

(مثال (متدهای Date

برگرداندن میلی ثانیه ها از یک تاریخ و زمان مشخص:

```
var d = new Date("July 21, 1983 01:15:00:526");  
var n = d.getMilliseconds();
```

خروجی n در کد بالا: 526

getMinutes

دقیقه ها را نسبت به زمان محلی برمی گرداند:

متدهای getMinutes() از یک تاریخ مشخص، دقیقه ها را (از ۰ تا ۵۹) برمی گرداند.

```
<script type ="text/javascript">  
var d = new Date();  
var n = d.getMinutes();  
</script>
```

2014

(مثال (متدهای Date

دقیقه ها را از یک تاریخ و زمان مشخص برمی گرداند:

```
var d = new Date("July 21, 1983 01:15:00");  
var n = d.getMinutes();
```

خروجی n در کد بالا: 15

مثال (متدهای getSeconds(), getMinutes() و getHours())

استفاده از (getMilliseconds() برای نمایش دادن زمان(علاوه میلی ثانیه ها):

```
function addZero(x,n) {  
    if (x.toString().length < n) {  
        x = "0" + x;  
    }  
    return x;  
}  
  
function myFunction() {  
    var d = new Date();  
    var x = document.getElementById("demo");  
    var h = addZero(d.getHours(), 2);  
    var m = addZero(d.getMinutes(), 2);  
    var s = addZero(d.getSeconds(), 2);  
    var ms = addZero(d.getMilliseconds(), 3);  
    x.innerHTML = h + ":" + m + ":" + s + ":" + ms;}
```

متدهای getMonth()

متدهای getMonth() از ۰ تا ۱۱ یک تاریخ مشخص، نسبت به زمان محلی برمی گرداند.

نکته: (ثانیه) هم ارز با ۰ است و February (فوریه) هم ارز با ۱ است و همینطور الی آخر.

```
<script type ="text/javascript">  
var d = new Date();  
var n = d.getMonth();  
</script>
```

2014

مثال (متده **getMonth**

برگرداندن نام ماه (نه فقط یک عدد):

```
var d = new Date();
var month = new Array();
month[0] = "January";
month[1] = "February";
month[2] = "March";
month[3] = "April";
month[4] = "May";
month[5] = "June";
month[6] = "July";
month[7] = "August";
month[8] = "September";
month[9] = "October";
month[10] = "November";
month[11] = "December";
var n = month[d.getMonth()];
```

خروجی کد بالا:

December

متده **getSeconds** ثانیه ها را (از ۰ تا ۵۹) از یک تاریخ و زمان مشخص برمی گرداند.

```
<script type ="text/javascript">
var d = new Date();
var n = d.getSeconds();
</script>
```

2014

مثال (متده **getSeconds**

استفاده از () getHours() و () getMinutes() و () getSeconds() برای نمایش دادن زمان:

```

function addZero(i) {
    if (i < 10) {
        i = "0" + i;
    }
    return i;
}
function myFunction() {
    var d = new Date();
    var x = document.getElementById("demo");
    var h = addZero(d.getHours());
    var m = addZero(d.getMinutes());
    var s = addZero(d.getSeconds());
    x.innerHTML = h + ":" + m + ":" + s;
}

```

متدها

toLocaleDateString()

متدها `toLocaleDateString()`، قسمت تاریخ از یک شیء تاریخ را تبدیل به یک رشته معمولی و قابل خواندن می کند.

`Date.toLocaleDateString()`

مثال (متدها `toLocaleDateString()`) برگرداندن تاریخ از شیء تاریخ مثل یک رشته معمولی:

```

var d = new Date();
var n = d.toLocaleDateString();

```

خروجی `n` در کد بالا

toLocaleTimeString()

متدها `toLocaleTimeString()`، قسمت زمان از یک شیء تاریخ را، تبدیل به یک رشته معمولی و قابل خواندن می کند.

`Date.toLocaleTimeString()`

مثال (متدها `toLocaleTimeString()`) برگرداندن زمان از شیء تاریخ، مثل یک رشته معمولی:

```

var d = new Date();
var n = d.toLocaleTimeString();

```

خروجی `n` در کد بالا :

متدها

toTimeString()

متدهایی که زمان را به شیوه تاریخ تبدیل می‌کنند.

Date.toTimeString()

مثال (متدهایی که زمان را به شیوه تاریخ تبدیل می‌کنند):

```
var d = new Date();
var n = d.toTimeString();
```

خروجی n در کد بالا (toTimeString()): 20:22:15 GMT+0330 (Iran Standard Time):

زمانبندی رویدادها در JavaScript

در JavaScript، این امکان وجود دارد که یک قطعه کد را در فاصله های زمانی معین time-intervals اجرا کنید. به این امکان "زمانبندی رویدادها" یا (Timing Events) می گویند.

دو متدهای (method) ساده برای زمانبندی رویدادها وجود دارد:

• **setInterval()**: در فاصله های زمانی معین، یک تابع را بارها و بارها اجرا می کند.

• **setTimeout()**: بعد از گذشت یک فاصله زمانی معین، یک تابع را یکبار اجرا می کند.

توجه: متدهای setTimeout() و setInterval() هر دو جزء متدهای شیء window در مدل DOM هستند.

setInterval()

متدهای setInterval()، در فاصله های زمانی معین، یک تابع مشخص را بارها و بارها اجرا می کند.

نحوه استفاده:

```
window.setInterval("javascript function",milliseconds);
```

متدهای setInterval() window را می توان بدون پیشوند window نیز نوشت.

پارامتر اول متدهای setInterval() باید یک تابع باشد.

پارامتر دوم، فاصله های زمانی بین هر بار اجرای تابع را مشخص می کند. (بر حسب میلی ثانیه)

توجه: هر ۱۰۰۰ میلی ثانیه معادل ۱ ثانیه است.

هر ۳ ثانیه یکبار، پیغام "Hello" نمایش داده می شود:

```
setInterval(function(){alert("Hello")},3000);
```

این مثال، فقط طرز کار متدهای setInterval() را نشان می دهد و اینکه بخواهید هر ۳ ثانیه یک پیغام را نمایش دهید خیلی واقعی نیست.

در مثال زیر، متدهای setInterval() برای نمایش یک ساعت دیجیتال، تابع myTimer() را هر ۱ ثانیه یکبار اجرا می‌کند:

نمایش زمان جاری

```
var myVar=setInterval(function(){myTimer()},1000);
function myTimer()
{
var d=new Date();
var t=d.toLocaleTimeString();
document.getElementById("demo").innerHTML=t;
}
```

چگونه اجرا را متوقف کنیم؟

متدهای clearInterval()، برای توقف اجرای تابع مشخص شده در setInterval() استفاده می‌شود.

متدهای clearTimeout()، برای توقف اجرای تابع مشخص شده در setTimeout() استفاده می‌شود.

نحوه استفاده:

```
window.clearTimeout(timeoutVariable)
```

متدهای window.clearTimeout() را می‌توان بدون پیشوند window نیز نوشت.

برای اینکه بتوان از متدهای clearTimeout() استفاده کرد، باید زمان ایجاد متدهای setTimeout()، از یک متغیر عمومی استفاده کرد:

```
myVar=setTimeout("javascript function",milliseconds);
```

سپس، اگر تابع قبل اجرا نشده باشد، قادر خواهد بود برای توقف اجرای تابع، از متدهای setTimeout() استفاده کنید.

مثال زیر مانند قبل است، اما یک دکمه "Stop time" به آن اضافه شده است:

```
var myVar;

function myFunction()
{
myVar=setTimeout(function(){alert("Hello")},3000);
}

function myStopFunction()
{
clearTimeout(myVar);
}
```

مثال ها

یک مثال ساده

کد

```
<!DOCTYPE html>

<html>
<head>
<script>

function timedText()

{
var x=document.getElementById('txt');

var t1=setTimeout(function(){x.value="2 seconds"},2000);

var t2=setTimeout(function(){x.value="4 seconds"},4000);

var t3=setTimeout(function(){x.value="6 seconds"},6000);

}

</script>

</head>
```

```
<body>  
  
<form>  
  
  <input type="button" value="Display timed text!" onclick="timedText()" />  
  
  <input type="text" id="txt" />  
  
</form>  
  
<p>Click on the button above. The input field will tell you when two, four, and six seconds have passed.</p>  
  
</body>  
  
</html>
```

ایجاد یک ساعت دیجیتال

کد

```
<!DOCTYPE html>  
  
<html>  
  
<head>  
  
<script>  
  
function startTime()  
  
{  
  
var today=new Date();  
  
var h=today.getHours();  
  
var m=today.getMinutes();  
  
var s=today.getSeconds();  
  
// add a zero in front of numbers<10  
  
m=checkTime(m);
```

```
s=checkTime(s);

document.getElementById('txt').innerHTML=h+":"+m+":"+s;

t=setTimeout(function(){startTime()},500);

}

function checkTime(i)

{

if (i<10)

{

i="0" + i;

}

return i;

}

</script>

</head>

<body onload="startTime()">

<div id="txt"></div>

</body>

</html>
```

ایجاد کردن یک عنصر در JavaScript

ایجاد کردن یک عنصر جدید

برای اضافه کردن یک عنصر جدید در مدل DOM ، ابتدا باید عنصر را ایجاد کنید و سپس آنرا به یک عنصر موجود، اضافه نمایید. مثال:

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>
<script>
  var para=document.createElement("p");
  var node=document.createTextNode("This is new.");
  para.appendChild(node);
  var element=document.getElementById("div1");
  element.appendChild(para);
</script>
```

توضیح مثال: کد زیر، یک عنصر جدید `<p>` ایجاد می کند:

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

برای اضافه کردن متن به عنصر `<p>` ابتدا باید یک نود متنی ایجاد نمایید. کد زیر، این کار را انجام می دهد:

```
var node=document.createTextNode("This is a new paragraph.");
```

سپس باید نود متنی را به عنصر `<P>` اضافه نمایید:

```
para.appendChild(node);
```

در آخر باید، عنصر جدید را به یک عنصر موجود اضافه نمایید.

کد زیر، یک عنصر موجود در صفحه را پیدا می کند:

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

کد زیر، عنصر جدید را به عنصر موجود، اضافه می کند:

```
element.appendChild(para);
```

حذف کردن یک عنصر موجود

برای حذف کردن یک عنصر HTML، ابتدا باید والد (parent) آنرا مشخص نمایید: مثال:

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>
<script>
  var parent=document.getElementById("div1");
  var child=document.getElementById("p1");
  parent.removeChild(child);
</script>
```

توضیح مثال: کد HTML زیر، شامل یک عنصر `<div>` است دو عنصر (`<p>`):

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>
```

ابتدا عنصر با شناسه "div1" را به عنوان والد، شناسایی می کنیم: متغیر یا شیء (parent)

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

سپس عنصر با شناسه "p1" را به عنوان فرزند، شناسایی می کنیم: متغیر یا شیء (child)

```
var child=document.getElementById("p1");
```

در آخر، از شی parent به عنوان والد، شی child به عنوان فرزند، حذف می شود:

```
parent.removeChild(child);
```

توجه: برای حذف کردن یک عنصر در مدل DOM ، باید حتماً والد آنرا مشخص نمایید.

راه حل معمول برای مشخص کردن والد، استفاده از خصوصیت `parentNode` است:

```
var child=document.getElementById("p1");
child.parentNode.removeChild(child);
```

ماژول ها

به مجموعه ای از شی ها و توابع و متغیر های درون شی را ماژول می گویند که به طور کلی یک فعالیت مهم و موجودیت خاصی را در صفحه نمایش به اجرا می گذارند.

شیوه کد نویسی آن همان به صورت شی گرایی می باشد . ماژول در اینجا تنها یک مفهوم تلقی می شود.

- عبارات منظم

regular expression شیء می باشد که یک الگوی شخصیت را توصیف می کند.
کلاس جاوا اسکریپت RegExp بیانگر عبارات منظم است و RegExp و String متدهایی را تعریف می کنند که از عبارات منظم برای انجام تطبیق الگویی قوی ، جستجو و جایگزینی در متن استفاده می کنند.

دستور نگارش (syntax)

regular expression می تواند با سازنده () RegExp تعریف شود.

var pattern = new RegExp(pattern, attributes);

or simply

var pattern = /pattern/attributes;

در زیر شرح پارامترها قرار دارد:

- Pattern : رشته ای که الگوی عبارات منظم یا یکی دیگر از عبارات منظم را مشخص می کند.
- Attributes : یک رشته اختیاری حاوی هر کدام از ویژگی های "g" ، "a" و "m" که به ترتیب جهانی، حساس به حروف بزرگ و کوچک و چند خطی را مشخص می کند.

براکت ها

براکت ها ([]) در زمان استفاده در عبارات منظم استفاده کنید. آنها برای یافتن طیف وسیعی از کاراکتر ها استفاده می شوند.

Expression	شرح
[...]	هر کاراکتر در بین براکت ها می باشد.
[^...]	هر کاراکتر در بین براکت ها نمی باشد.
[0-9]	با تمام ارقام میان ۰ تا ۹ مطابقت دارد.
[a-z]	با تمام حروف کوچک میان a تا z مطابقت دارد.
[A-Z]	با تمام حروف بزرگ میان A تا Z مطابقت دارد.
[a-Z]	با تمام حروف میان a تا Z مطابقت دارد

محدوده نشان داده شده در بالا به طور کلی می باشد؛ شما همچنین می توانید محدوده [۰-۳] را برای مطابقت با هر رقم دسیمال از ۰ تا ۳ یا محدوده [b-v] برای هر کاراکتر کوچک با دامنه b از طریق v استفاده کنید.

Quantifier

فرکانس یا موقعیت توالی کاراکترها و کاراکترهای تک با یک کاراکتر خاص ممکن است نشان داده شود. هر یک از کاراکترهای خاص دارای خاصیت خاصی می باشند. تمامی پرچم های + ، * ، ? و \$ توالی کاراکتر هارا دنبال می کنند.

Expression	شرح
p ⁺	هر رشته دارای یک یا چند p را مطابقت می دهد.
p [*]	هر رشته دارای صفر یا چند p را مطابقت می دهد.
p [?]	هر رشته دارای یک p را مطابقت می دهد
p{N}	هر رشته دارای توالی N تا p را مطابقت می دهد.
p{2,3}	هر رشته دارای دو یا سه p را مطابقت می دهد.
p{2,3}	هر رشته با حداقل دو p را مطابقت می دهد.
p\$	هر رشته ای را که با p پایان یابد را مطابقت می دهد.
^p	هر رشته ای را که با p آغاز شود را مطابقت می دهد

مثال ها

مثال های زیر بیش تر رابطه کاراکترها را مطابقت می دهد.

Expression	شرح
[^a-zA-Z]	هر رشته ای را که دارای کاراکتر های رنج a تا Z و A تا Z نمی باشد مطابقت می دهد.
p.p	رشته شامل p که توسط کاراکتر های p دیگری دنبال می شود را مطابقت می دهد.
^.{2}\$	هر رشته که دقیقاً دارای دو کاراکتر می باشد را مطابقت می دهد.
(.*)	هر رشته ار را که درون و قرار دارد را مطابقت می دهد.
p(hp)*	رشته شامل p که توسط صفر یا نمونه بیشتری از توالی hp دنبال می شود را مطابقت می دهد.

کاراکترهای Meta یا Metacharacters

به سادگی یک کاراکتر الفبایی است که پیش از بک اسلش قرار دارد و به ترکیب یک معنای خاص می دهد.

به عنوان مثال، می توانید جمع کل مبلغ پول را با استفاده از 'd' metacharacter جستجو کنید : /([d]+)000/. در اینجا d هر رشته با کاراکتر عددی (numerical character) را جستجو می کند. در جدول زیر مجموعه ای از metacharacter ها لیست شده که می توانند در PERL Style Regular Expressions استفاده شوند.

کاراکتر	شرح
.	یک کاراکتر
s	کاراکتر فضای خالی (فضا، تب، خط جدید)
S	بدون کاراکتر فضای خالی
d	رقم (۰ - ۹)
D	بدون رقم
w	(_ ، a-z ، A-Z ، ۰-۹) کلمه کاراکتر
W	بدون کاراکتر کلمه
[b]	یک عقبگرد (literal مورد خاص)
[aeiou]	یک کاراکتر تک در مجموعه داده شده را مطابقت می دهد.
[^aeiou]	یک کاراکتر خارج از مجموعه داده شده را مطابقت می دهد.
(foo bar baz)	هر یک از گزینه های مشخص را مطابقت می دهد.

Modifier ها

چند (modifier) اصلاح کننده در دسترس هستند که می توانند روش کار با `regexp` را ساده تر کنند، مانند حساسیت به حروف کوچک و بزرگ، جستجو در چند خط و غیره.

Modifier	شرح
i	تطبیق حساسیت به حروف کوچک و بزرگ (case-insensitive) را انجام دهد.
m	مشخص می کند که اگر رشته دارای رشته های جدید یا حروف بازگشتی می باشد، عملگرهای ^ و \$ با مرز خط جدید به جای مرز رشته مطابقت می کند
g	یک تطابق جهانی انجام می شود، تمامی مطابقت ها را پیدا می کند و بعد از یافتن اولین تطابق توقف نمی کند.

RegExp متدهای

در زیر لیستی از متدهای `RegExp` شرح داده شده است.

متدها	شرح
<code>test()</code>	تست برای یک تطابق در پارامتر رشته آن است.
<code>exec()</code>	جستجو برای یک تطابق را در پارامتر رشته انجام می دهد
<code>toSource()</code>	شی <code>literal</code> را که نشان دهنده شی مشخص شده است، بازگرداند؛ شما می توانید از این مقدار برای ایجاد یک شی جدید استفاده کنید.
<code>toString()</code>	رشته ای را نشان می دهد که شی مشخص شده را بر می گرداند

Regular expression

یک زبان برای توصیف یک الگو در یک رشته است

```
var name="mohammadreza esmaeeli";  
  
var email="esmaeeli1923@yahoo.com";  
  
var pattern1=/mohmmadreza/i;  
  
alert(pattern1.test(name));  
  
var pattern1=/mohmmadreza/g;  
  
var pattern=/gmail.com$/g;
```

```
alert(pattern.test(email));
```

به حروف بزرگ و کوچک حساس نباشد:

```
var pattern1=/mohmmadreza/i;
```

اول رشته با mohammadreza شروع شود:

```
var pattern1=/^mohmmadreza/g;
```

آخر رشته با mohammadreza تمام شود:

```
var pattern1=/mohmmadreza$/g;
```

```
var pattern=/gmail.com$/g;
```

بررسی رقم بعد از رشته:

```
var email="esmaeeli1235@gmail.com";
```

```
var pattern=/^esmaeeli+\d/;
```

```
alert(pattern.test(email));
```

// \d means followed by numbers

// \D means not followed by numbers

بررسی فضای خالی بعد از رشته:

```
var email="esmaeeli hello@gmail.com";
```

// \s means followed by white space

// \S means not followed by white space

الگوی ما با یک رشته خاص ادامه پیدا کند:

```
/* var email="esmaeelihello@gmail.com";
```

```
var pattern=/^esmaeeli+h/;
```

الگو با یک h یا بیشتر ادامه پیدا کند:

```
var pattern=/^esmaeeli+h+/;
```

الگو با صفر h یا بیشتر ادامه پیدا کند:

```
var pattern=/^esmaeeli+h*/;
```

حداقل یکی از کاراکتر ها باشد:

```
var pattern=/[abcd]/;
```

```
var pattern=/[0123]/;
```

```
var pattern=/[a-d]/;
```

```
var pattern=/[0-5]/;
```

```
var pattern=/[a-z1-9]/;
```

```
var pattern=/[a-z]+\.[0-9]/;
```

الگوی ایمیل:

```
var pattern=/[a-z1-9]+\@[a-z]+\.[a-z]/;
```

جستجوی یک رشته:

```
Var name="esmaeili";
```

```
var email="esmaeelihello@gmail.com";
```

```
var pattern=new RegExp("e");
```

```
var pattern=new RegExp("gmail");
```

```
alert(pattern.exec(name));
```

BOM

مدل شی گرای مرورگر Browser Object Model

یکی دیگر از API های ساخته شده برای HTML

از BOM برای دسترسی و دستکاری ویژگی های پنجره یک مرورگر می توان استفاده کرد.

توسعه دهندهان وب با استفاده از BOM می توانند کارهایی همچون جا به جایی و تغییر متن موجود در نوار وضعیت مرورگر و دیگر کارهایی که ارتباط مستقیمی با محتوای تشکیل دهنده صفحه (سند) ندارند انجام دهند.(روکش کار)

BOM با پنجره ها و فریم ها سرو کار داشته و می توان از طریق آن کارهای زیر را انجام داد:

۱. باز کردن پنجره های **popup**
۲. توانایی باز کردن پنجره های جدید و تغییر اندازه و جایه جایی و یا بستن آنها
۳. بدست آوردن اطلاعاتی از مرورگر و سیستم عامل کاربران همچون نوع، نسخه و ...
۴. بدست آوردن اطلاعاتی در مورد سند و موقعیت صفحه ای که در مرورگر باز شده است
۵. بدست آوردن اطلاعاتی در مورد وضوح صفحه نمایش کاربر
۶. پشتیبانی از **cookie** ها

پنجره pop up

pop up در برنامه نویسی وب به پنجره های کوچکی گفته می شود که به صورت خودکار یا با دخالت کاربر در مرورگر نمایش داده می شوند،

```
<script>  
var color="blue ";  
var num=10;  
alert(color+num);  
alert("Hello")  
</script>
```

دستور confirm

دستور **confirm** که به معنی تایید کردن و تصدیق کردن است، بعد از اجرای این دستور در پنجره ای به صورت **pop up** به کاربر یک هشدار نشان داده می شود، اگر او دکمه **ok** را انتخاب کند، مقادیر ۱ یا **true** برگردانده می شود و اگر دکمه **cancel** را کلیک کند، عدد صفر یا **false** بازگشت داده شده و عملیات لغو می شود.

```
<script>  
    var r=confirm("کلیک کنید");  
    // var r>window.confirm("");  
    if(r==true)  
        document.write("OK");  
    else  
        document.write("cancel");  
</script>
```

مثال کاربردی با دستور confirm

در مثال زیر با استفاده از دستور **confirm** از مطمئن بودن کاربر از ارسال یک فرم **html** اطمینان حاصل می کنیم.

```
<form action="#" method="post" onsubmit="return confirm('');">  
    <input type="submit" value=" ارسال "/>  
</form>
```

دستور prompt

با اجرا شدن آن، یک پنجره به کاربر نشان داده می شود که یک سوال و یک فیلد متنی در آن وجود دارد، سپس عبارتی که کاربر در فیلد مورد نظر وارد می کند. در ادامه تابع قابل استفاده است،

```
<script>  
var a=prompt('خواندن عدد,' عدد را وارد کنید');  
if((a%2)==0)  
    alert("زوج");  
else  
    alert("فرد");  
</script>
```

آموزش مدل BOM

شیء JavaScript در window

مدل BOM یا (Browser Object Model) به این اجازه را می دهد تا با مرورگر صحبت کند.

(Browser Object Model) یا BOM

استاندارد رسمی برای مدل BOM وجود ندارد.

از آنجایی که مرورگرهای امروزی برای تعامل بیشتر با JavaScript، تقریباً به خصوصیت‌ها و متدهای یکسانی مجهز هستند، اغلب خصوصیت‌ها و متدهای BOM را نیز دارا هستند.

شیء window

شیء window در تمام مرورگرها پشتیبانی می شود و به پنجره مرورگر اشاره دارد.

تمام اشیاء، توابع و متغیرهای عمومی در JavaScript، عضوی از شیء window هستند.

متغیرهای عمومی، خصوصیت‌های (property) شیء window هستند.

توابع عمومی، متدهای (method) شیء window هستند.

حتی شیء document در مدل DOM نیز یکی از خصوصیت‌های شیء window است.

```
window.document.getElementById("header");
```

در کد بالا، قسمت window را می توان حذف کرد:

```
document.getElementById("header");
```

اندازه پنجره مرورگر

برای معین کردن اندازه پنجره مرورگر، می توان سه خصوصیت مختلف را بکار برد (اندازه مرورگر شامل scrollbars و toolbars نمی شود)

مرورگرهای Internet Explorer, Chrome, Firefox, Opera, Safari:

window.innerHeight •

window.innerWidth •

مرورگر Internet Explorer 8, 7, 6, 5:

document.documentElement.clientHeight •

document.documentElement.clientWidth •

یا

document.body.clientHeight •

document.body.clientWidth •

یک راه حل عملی در JavaScript (پوشش دهنده تمام مرورگرهای):

```
var w=window.innerWidth  
|| document.documentElement.clientWidth  
|| document.body.clientWidth;
```

```
var h=window.innerHeight  
|| document.documentElement.clientHeight  
|| document.body.clientHeight;
```

مثال بالا، عرض و ارتفاع مرورگر را نمایش می دهد (بدون scrollbars/toolbars)

شیء JavaScript در screen

شیء `window.screen`، شامل اطلاعاتی دربارهٔ صفحهٔ مانیتور بازدیدکننده است.

بعاد صفحهٔ مونیتور

شیء `window.screen` را می‌توان بدون پیشوند `window` نوشت.

بعضی از خصوصیت‌های شیء `window.screen`:

عرض قابل دسترس صفحهٔ مونیتور • `screen.availWidth`

ارتفاع قابل دسترس صفحهٔ مونیتور • `screen.availHeight`

عرض قابل دسترس صفحهٔ مونیتور

خصوصیت `screen.availWidth`، عرض قابل دسترس صفحهٔ مونیتور را به پیکسل برمی‌گرداند (این اندازه شامل taskbar نمی‌شود)

مثال (شیء JavaScript در screen)

مثال زیر، عرض صفحهٔ مونیتورتان را برمی‌گرداند:

```
document.write("Available Width: " + screen.availWidth);
```

ارتفاع قابل دسترس صفحهٔ مونیتور

خصوصیت `screen.availHeight`، ارتفاع قابل دسترس صفحهٔ مونیتور را به پیکسل برمی‌گرداند (این اندازه شامل taskbar نمی‌شود)

مثال شیء JavaScript در screen

مثال زیر، ارتفاع صفحهٔ مونیتورتان را برمی‌گرداند:

```
document.write("Available Height: " + screen.availHeight);
```

تمام خصوصیت های شیء screen در یک مثال

```
<body>

<h3>Your Screen:</h3>

<script>

document.write("Total width/height: ");

document.write(screen.width + "*" + screen.height);

document.write("<br>");

document.write("Available width/height: ");

document.write(screen.availWidth + "*" + screen.availHeight);

document.write("<br>");

document.write("Color depth: ");

document.write(screen.colorDepth);

document.write("<br>");

document.write("Color resolution: ");

document.write(screen.pixelDepth);

</script>

</body>
```

شیء JavaScript در location

شیء `window.location`، می تواند برای برگرداندن آدرس صفحه جاری (URL) و یا هدایت کاربر به یک صفحه جدید استفاده شود.

شیء window.location

شیء `window.location` را می توان بدون پیشوند `window` نیز نوشت.

بعضی از خصوصیت های شیء **:location**

- نام دامین جاری را برمی گرداند. : **location.hostname**
- مسیر و نام صفحه جاری را برمی گرداند. : **location.pathname**
- شماره Port وب هاست را برمی گرداند. (۴۴۳ یا ۸۰) : **location.port**
- پروتکل مورد استفاده را برمی گرداند (**https** یا **http**) : **location.protocol**

location.href خصوصیت

خصوصیت **location.href**، آدرس صفحه (URL) جاری را برمی گرداند.

مثال زیر، کل URL صفحه جاری را برمی گرداند: مثال:

```
document.write(location.href);
```

location.pathname خصوصیت

خصوصیت **location.pathname**، مسیر و نام صفحه جاری، برروی سرور را برمی گرداند.

مثال زیر، مسیر و نام صفحه جاری را برمی گرداند: مثال:

```
document.write(location.pathname);
```

location.reload خصوصیت

خصوصیت **location.reload**، صفحه جاری را مجدد بارگذاری می کند.

```
location.reload();
```

location.assign() متده

متده **location.assign()**، یک صفحه جدید را بارگزاری می کند.

```
<head>
<script>
    function newDoc()
    {
        window.location.assign("http://www.beyamooz.com")
    }
</script>
</head>
<body>
    <input type="button" value="Load new document" onclick="newDoc()">
</body>
```

localStorage

با توجه به اهمیت ذخیره سازی اطلاعات در وب سایت‌ها و اپلیکیشن‌های تحت وب، فضاهای مختلفی برای ذخیره اطلاعات در سمت سرور و در سمت کلاینت وجود دارند که بعضی از این فضاهای مانند localStorage و sessionDatabase در سمت سرور و برخی دیگر مانند cookie و Database در سمت کلاینت قرار دارند.

برخی اطلاعات باید در مرورگر کاربر ذخیره شوند که برای این کار می‌توان از localStorage استفاده نمود. localStorage به صورت محلی در مرورگر کاربر وجود دارد و با سرور تعاملی ندارد، در نتیجه زحمت سرور کاهش می‌یابد که این امر سبب افزایش سرعت سیستم و همچنین کاهش استفاده از پهنای باند می‌شود.

نحوه تعریف داده در localStorage:

```
localStorage.setItem("Key", "Value");
```

نحوه گرفتن داده از :localStorage

```
localStorage.getItem("Key");
```

نحوه حذف داده از :localStorage

```
localStorage.removeItem("Key");
```

مثال:

```
<script>  
  localStorage.setItem("email", "example@gmail.com");  
  
  let email = localStorage.getItem("email");  
  console.log(email);  
  
  localStorage.removeItem("email");  
  console.log(email);  
</script>
```

Cookie

Cookie ها این امکان را به ما می‌دهند تا اطلاعات کاربران را در صفحه وب ذخیره کنیم. Cookie داده‌هایی هستند که در فایل‌های متغیر کوچک بر روی کامپیوتر ذخیره می‌شود. هنگامی که وب سرور یک صفحه وب را به یک مرورگر ارسال می‌کند، ارتباط قطع می‌شود و سرور هر داده‌ای در مورد کاربر را فراموش می‌کند. می‌توان گفت cookie ها برای حل این مشکل اختراع شده‌اند.

وقتی کاربر از یک صفحه وب بازدید می‌کند، اطلاعاتی مانند: username، email، password و دیگر موارد را می‌تواند در cookie ذخیره کند. و کاربر دفعه بعد که خواست از صفحه وب دیدن کند، اطلاعات مربوط به ورود یا تکمیل اطلاعاتش را نمایش خواهد داد.

شکل ایجاد :cookie

```
document.cookie = "Cookie name=Cookie value; expires=";
```

یا تاریخ انقضا برای پاک شدن cookie در زمان معین یا دلخواه است.

```
<script>  
  document.cookie = "user-name=johndoe";  
  
  document.cookie = "user-name=johndoe; expires=Thu, 18 Dec 2013 12:00:00  
UTC";  
</script>
```

jQuery

jQuery (جی کوئری) یکی از کتابخانه های محبوب JavaScript است.

jQuery (جی کوئری) چیست؟

jQuery یک کتابخانه کم وزن، به معنی: "کارهای زیاد با نوشتن کد کمتر" است.

هدف از این آموزش، بکارگیری آسانتر JavaScript در وب سایتتان است.

jQuery بسیاری از کارهای پیچیده در JavaScript مانند: AJAX، دستکاری عناصر و ... را ساده کرده است.

توجه: jQuery تقریباً برای هر کاری که تصور کنید یک Plugin دارد.

آموزش یکی از کتابخانه های JavaScript چرا

کتابخانه های زیادی برای JavaScript وجود دارد، اما بنظر می رسد که jQuery در بین آنها محبوب تر و همچنین توسعه پذیرتر است.

بعضی از شرکت های بزرگی که از jQuery استفاده کرده اند:

Google •

Microsoft •

IBM •

Netflix •

اضافه کردن jQuery به صفحه

چندین راه برای استفاده از jQuery وجود دارد:

دانلود کتابخانه jQuery از سایت [jQuery.com](http://jquery.com) و لینک کردن آن به صفحه وب توسط تگ

<script>

استفاده از یک CDN مانند Google و اضافه کردن URL آن به صفحه وب توسط تگ <script>

توجه : CDN مخفف کلمات Content Delivery Networks است (تحویل محتوی، روی شبکه). یک شبکه از چندین کتابخانه عمومی JavaScript است.

دانلود jQuery

دو نسخه از jQuery برای دانلود در دسترس است:

- نسخه **Production**: این نسخه، یک فایل فشرده از jQuery است و برای وب سایت های در حال کار مناسب است.

- نسخه **Development**: این نسخه، یک فایل بدون فشرده سازی و قابل خواندن است و برای زمانی که در حال آماده سازی وب سایت هستید مناسب است.

هر دو نسخه را می توان از سایت jQuery.com دانلود نمود.

کتابخانه jQuery، تنها یک فایل JavaScript است و همانطور که گفته شد می توانید توسط تگ `<script>` آنرا به صفحه و بتان لینک کنید تگ `<head><script>` باید در قسمت `<head>` صفحه باشد.

```
<head>
  <script src="/jquery-1.9.1.min.js"></script>
</head>
```

توجه : خصوصیت `src` باید با آدرس فایل دانلود شده تنظیم شود، در مثال بالا، فایل jQuery دقیقاً در همان مسیر صفحه جاری قرار دارد. اگر فایل jQuery در آدرسی غیر از آدرس صفحه جاری وجود دارد باید آنرا در نظر بگیرید.

شاید بپرسید که چرا در تگ `<script>` خصوصیت `type` را با مقدار "text/javascript" تنظیم نکرده اید؟ در HTML5 نیازی به این کار نیست. در HTML5 و مرورگرهای امروزی JavaScript زبان اسکریبت نویسی پیشفرض است.

جایگزینی برای دانلود

اگر نمی خواهید فایل jQuery را در بین فایل های پروژه تان قرار دهید، می توانید با استفاده از یک CDN آنرا به صفحه و بتان لینک کنید.

Microsoft و Google هر دو میزبان jQuery هستند:

```
<head>
    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js">
    </script>
</head>
```

✓ گرفتن آخرین نسخه jQuery از Google:

همانطور که در URL بالا مشاهده می کنید، نسخه jQuery در URL مشخص شده است (1.9.1). اگر آخرین نسخه jQuery مدنظر شماست می توانید هر کدام از آخرین اعداد را حذف کنید. برای مثال ۱.۹ آخرین نسخه قابل دسترس در سری ۱.۹ را برگرداند یا عدد ۱ آخرین نسخه در سری ۱ را برمی گرداند (از ۱.۹.۰ تا ۱.۹.۹).

```
<head>
    <script src="//ajax.aspnetcdn.com/ajax/jquery/jquery-1.9.1.min.js"></script>
</head>
```

```
<html>
<head>
<script src="jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
</script>
```

```

</head>

<body>

    <h2>This is a heading</h2>

    <p>This is a paragraph.</p>

    <p>This is another paragraph.</p>

    <button>Click me</button>

</body>

</html>

```

✓ یک مزیت بزرگ استفاده از CDN های Microsoft یا Google زمانی که بازدیدکنندگان سایت شما، قبلاً در سایت های دیگر، فایل jQuery را از CDN های دریافت کرده اند، در دفعات بعدی، فایل jQuery از Cache فراخوانی می شود که باعث افزایش سرعت می شود.

نحوه نوشتن jQuery

با jQuery یک عنصر یا تعدادی عنصر را انتخاب می کنید و عملی را روی آنها انجام می دهید بگونه ای بهینه شده که بتوانید با آسان ترین راه ممکن، عناصر HTML را انتخاب و عملی را روی آنها انجام دهید.

اصلی ترین دستور در jQuery

`$(selector).action()`

- علامت `$` مانند متدهای JavaScript در `document.getElementById()` عمل می کند (با این تفاوت `getElementById` فقط به عناصر با `id` مشخص دسترسی داشت ولی `$` می تواند به تمام عناصر دسترسی داشته باشد) و برای دسترسی به عناصر HTML درjQuery استفاده می شود.

- عنصر **(selector)** : نام کلاس، شناسه، نام تگ، نوع، نام خصوصیت و یا مقدار خصوصیت عنصر HTML است.

• عملی است که روی عنصر HTML انجام می شود.

مثال:

`$(this).hide() // عنصر جاری را پنهان می کند`

`$(".p").hide() // تمام پاراگراف ها را پنهان می کند`

`$(".test").hide() // تمام عناصری که شامل کلاس مشخص شده در این کد هستند را پنهان می کند`

`$("#test").hide() // عنصری که شامل شناسه مشخص شده در این کد هستند را پنهان می کند`

رویداد ready

همانطور که تا به حال در مثال ها مشاهده کرده اید، تمام متدهای jQuery داخل ready آورده شده

اند:

```
$(document).ready(function(){  
    // jQuery methods go here...  
});
```

این امر باعث می شود تا کدهای jQuery بعد از بارگذاری کامل صفحه، اجرا شوند. همچنین با این تکنیک، قادر خواهید بود کدهای JavaScript را در قسمت `<head>` صفحه قرار دهید.

در اینجا مثال هایی آورده شده است که اگر قبل از بارگذاری کامل صفحه اجرا شوند، عمل نخواهند کرد:

• تلاش برای پنهان کردن عنصری که هنوز ایجاد نشده است.

• تلاش برای گرفتن عرض و ارتفاع عکسی که هنوز بارگذاری نشده است.

نکته: تیم jQuery یک متدهای کوتاه تر برای رویداد ready ایجاد کرده است:

```
$(function(){  
    // jQuery methods go here...  
});
```

می توانید هر کدام از روش های بالا را که ترجیح می دهید استفاده کنید. اما توجه داشته باشید که روش اول قابل فهم تر و خواناتر است.

گزینشگرهای jQuery

گزینشگرهای jQuery یکی از مهترین بخش های کتابخانه jQuery است.

گزینشگرهای jQuery یا (jQuery Selectors)

گزینشگرهای jQuery به شما این اجازه را می دهند تا عناصر HTML را انتخاب و دستکاری کنید.

با گزینشگرهای jQuery می توانید عناصر HTML را براساس شناسه، کلاس، نوع، نام خصوصیت، مقدار خصوصیت و ... پیدا کنید. این تکنیک براساس گزینشگرهای CSS Selector (CSS Selector) و علاوه بر این، زعدادی گزینشگر سفارشی نیز دارد.

تمام گزینشگرها در `jQuery` با علامت "\$" آغاز و با پرانتز محصور می شوند(): :

گزینشگر نام تگ

گزینشگر نام تگ، به شما اجازه می دهد تا عناصر را براساس نام تگ، انتخاب کنید.

می توانید تمام تگ های `<p>` را در صفحه مانند زیر انتخاب کنید:

```
$("p")
```

•مثال : در مثال زیر، زمانی که کاربر روی دکمه کلیک می کند، تمام عناصر `<p>` پنهان می شوند:

```
$(document).ready(function(){  
    $("button").click(function(){  
        $("p").hide();  
    });  
});
```

گزینشگر شناسه (#id)

گزینشگر شناسه، از مقدار خصوصیت id تگ HTML برای پیدا کردن عنصر استفاده می کند.

شناسه در یک صفحه، باید یکتا باشد، بنابراین زمانی که بخواهید به یک عنصر منحصر به فرد در صفحه دسترسی داشته باشید از این روش استفاده نمایید.

برای پیدا کردن یک عنصر با شناسه مشخص، از کاراکتر "#" همراه با شناسه مورد نظر استفاده نمایید:

```
$("#test")
```

مثال :در این مثال، زمانی که کاربر روی دکمه کلیک کند، عنصر با شناسه "test" پنهان می شود:

```
$(document).ready(function(){  
    $("button").click(function(){  
        $("#test").hide();  
    });  
});
```

گزینشگر نام کلاس (.class)

گزینشگر نام کلاس، عناصر با کلاس مشخص را پیدا می کند.

برای پیدا کردن عناصر با نام کلاس مشخص، از کاراکتر ". " همراه با نام کلاس مورد نظر استفاده کنید:

```
$(".test")
```

مثال :در این مثال، زمانی که کاربر روی دکمه کلیک کند، عنصر با شناسه "test" پنهان می شود:

```
$(document).ready(function(){  
    $("button").click(function(){  
        $(".test").hide();  
    });  
});
```

مثال های بیشتر از گزینشگر های jQuery

نحوه نوشتمن	توضیحات
<code>\$("*")</code>	تمام عناصر را انتخاب می کند.
<code>\$(this)</code>	عنصر جاری را انتخاب می کند.
<code>\$("p.intro")</code>	تمام عناصر <p> با کلاس "intro" را انتخاب می کند.
<code>\$("p:first")</code>	اولین عنصر <p> را انتخاب می کند.
<code>\$("ul li:first")</code>	اولین آیتم از اولین عنصر را انتخاب می کند.
<code>\$("ul li:first-child")</code>	اولین آیتم از تمام عناصر را انتخاب می کند.
<code>\$("[href]")</code>	تمام عناصر با خصوصیت href را انتخاب می کند.
<code>\$("a[target='_blank']")</code>	تمام عناصر <a> که خصوصیت target آنها با مقدار "blank" تنظیم شده باشید را انتخاب می کند.
<code>\$("a[target!='_blank']")</code>	تمام عناصر <a> که خصوصیت target آنها با مقدار "blank" تنظیم نشده باشید را انتخاب می کند.
<code>\$(":button")</code>	تمام عناصر <button> و همچنین تمام عناصر <input> که خصوصیت type آنها "button" است را انتخاب می کند.
<code>\$("tr:even")</code>	تمام عناصر <tr> زوج را انتخاب می کند.
<code>\$("tr:odd")</code>	تمام عناصر <tr> فرد را انتخاب می کند.

توابع پر کاربرد در یک فایل مجزا

اگر وب سایت شما شامل تعداد زیادی صفحه است، و می خواهید توابع jQuery یکسانی را در آنها استفاده کنید، می توانید این توابع را در یک فایل مجزا با پسوند .js قرار دهید.

همانطور که در این آموزش دیده اید، توابع به طور مستقیم در قسمت <head> صفحه قرار داده شده اند. اما بعضی موقع مانند بالا، ترجیح داده می شود که آنها را در یک فایل JavaScript جداگانه با پسوند .js ذخیره کنیم و مانند زیر، توسط تگ <script> به صفحه لینک کرد:

رویدادها در jQuery

رویدادهای موس	رویدادهای صفحه کلید	رویدادهای فرم	رویدادهای صفحه/پنجره
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

jQuery برای پاسخ دادن به رویدادهایی که در یک صفحه HTML اتفاق می افتد، بهینه شده است.

رویداد (Event) چیست؟

تمام کارهایی که در یک صفحه وب اتفاق می افتد و می توانیم به آنها پاسخ دهیم، رویداد (Event) گفته می شود.

مثال:

- حرکت موس ببروی یک عنصر
- انتخاب یک radio button
- کلیک کردن ببروی یک عنصر

واژه "اتفاق افتادن" اغلب همراه با واژه "رویداد" استفاده می شود. مثال: رویداد keypress زمانی اتفاق می افتد که یکی از دکمه های صفحه کلید را فشار دهید.

در جدول زیر، تعدادی از رویدادهای رایج در مدل DOM آورده شده است:

نحوه استفاده از رویدادها در jQuery

بیشتر رویدادهای مدل DOM ، یک متاد معادل در jQuery دارند.

برای اختصاص دادن رویداد **onclick** به تمام تگ های **<p>**، می توانید مانند زیر عمل کنید:

```
$( "p" ).click();
```

قدم بعدی، مشخص کردن کاری است که باید به ازای رویداد **onclick** اتفاق بیافتد. باید یک تابع را به عنوان پارامتر به متده **click** ارسال کنید:

```
$( "p" ).click(function(){  
    // action goes here!!  
});
```

رویدادهای رایج در **jQuery**

\$(document).ready()

این متده به شما اجازه می دهد تا یک تابع را زمانی که صفحه کاملاً بارگذاری شد، اجرا کنید.

click() : این متده، یک تابع را برای پاسخ دادن به رویداد **onclick** یک عنصر، به آن اختصاص می دهد.

زمانی که کاربر روی یک عنصر **HTML** کلیک کند، رویداد **onclick** اتفاق می افتد.

در مثال زیر، زمانی که کاربر روی هر کدام از تگ های **<p>** در صفحه کلیک کند، پاراگراف مورد نظر را پنهان می کند:

```
$( "p" ).click(function(){  
    $(this).hide();  
});
```

```
<html>  
<head>  
<script src="jquery.min.js"></script>
```

```
<script>  
$(document).ready(function(){  
    $("p").click(function(){  
        $(this).hide();  
    });  
});  
</script>  
</head>  
<body>  
<p>If you click on me, I will disappear.</p>  
<p>Click me away!</p>  
<p>Click me too!</p>  
</body>  
</html>
```

dblclick()

این متده، یک تابع را برای پاسخ دادن به رویداد **ondblclick** یک عنصر، به آن اختصاص می دهد.

زمانی که کاربر روی یک عنصر HTML دوبار کلیک کند، رویداد **ondblclick** اتفاق می افتد.

```
$( "p" ).dblclick(function(){  
    $(this).hide();  
});
```

```
<!DOCTYPE html>

<html>
<head>
<script src="jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("p").dblclick(function(){
        $(this).hide();
    });
});
</script>
</head>
<body>
<p>If you click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>
</body>
</html>
```

خروجی

If you click on me, I will disappear.

Click me away!

Click me too!

mouseenter()

این متده، یک تابع را برای پاسخ دادن به رویداد `onmouseover` یک عنصر، به آن اختصاص می دهد.

زمانی که موس کاربر روی یک عنصر HTML قرار می‌گیرد، رویداد `onmouseover` اتفاق می‌افتد.

```
$("#p1").mouseenter(function(){
    alert("You entered p1!");
});
```

کد

```
<!DOCTYPE html>

<html>
    <head>
        <script src="jquery.min.js"></script>
        <script>
            $(document).ready(function(){
                $("#p1").mouseenter(function(){
                    alert("You entered p1!");
                });
            });
        </script>
    </head>
    <body>
        <p id="p1">Enter this paragraph.</p>
    </body>
</html>
```

خروجی

Enter this paragraph.

mouseleave()

این متده، یک تابع را برای پاسخ دادن به رویداد `onmouseout` یک عنصر، به آن اختصاص می‌دهد.

زمانی که موس کاربر از روی یک عنصر HTML خارج می شود، رویداد `onmouseout` اتفاق می افتد.

```
$("#p1").mouseleave(function(){
    alert("Bye! You now leave p1!");
});
```

کد

```
<html>
<head>
<script src="jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#p1").mouseenter(function(){
        alert("You entered p1!");
    });
});
</script>
</head>
<body>
<p id="p1">Enter this paragraph.</p>
</body>
</html>
```

mousedown()

این متده، یک تابع را برای پاسخ دادن به رویداد `onmousedown` یک عنصر، به آن اختصاص می دهد.

زمانی که موس کاربر روی یک عنصر HTML قرار دارد و چپ کلیک کند، رویداد `onmousedown` اتفاق می افتد.

```
$("#p1").mousedown(function(){
  alert("Mouse down over p1!");
});
```

```
<head>
<script src="jquery.min.js"></script>

<script>
$(document).ready(function(){
  $("#p1"). mousedown (function(){
    alert("You entered p1!");
  });
});
</script>
</head>
<body>
  <p id="p1">Enter this paragraph.</p>
</body>
```

mouseup()

این متده، یک تابع را برای پاسخ دادن به رویداد onmouseup یک عنصر، به آن اختصاص می دهد.

زمانی که موس کاربر روی یک عنصر HTML قرار دارد و چپ کلیک را رها کند، رویداد onmouseup اتفاق می افتد.

```
$("#p1").mouseup(function(){
  alert("Mouse up over p1!");
});
```

hover() : این متدهم زمان دو تابع را برای پاسخ دادن به رویداد **onmouseover** و **onmouseout** یک عنصر، به آن اختصاص می دهد.

زمانی که موس کاربر روی عنصر HTML قرار گیرد، تابع اول اجرا می شود و زمانی که موس از روی آن خارج می شود، تابع دوم اجرا خواهد شد.

```
$("#p1").hover(function(){
    alert("You entered p1!");
},
function(){
    alert("Bye! You now leave p1!");
});
```

```
<head>
<script src="jquery.min.js"></script>

<script>
$(document).ready(function(){
    $("#p1").hover(function(){
        alert("You entered p1!");
    },
    function(){
        alert("Bye! You now leave p1!");
    });
});
</script>
</head>

<body>
    <p id="p1">This is a paragraph.</p>
</body>
```

focus() : این متدهم یک تابع را برای پاسخ دادن به رویداد **onfocus** یک عنصر، به آن اختصاص می دهد.

زمانی که علامت چشمک زن مکان نما روی یکی از اجزای فرم، مانند **Text Box**: قرار گیرد، رویداد **onfocus** اتفاق می‌افتد.

```
$( "input" ).focus(function() {
  $(this).css("background-color", "#cccccc");
});
```

این متدها، یک تابع را برای پاسخ دادن به رویداد **onBlur** یک عنصر، به آن اختصاص می‌دهد.

زمانی که تمرکز از روی یکی از اجزای فرم خارج شود، رویداد **onBlur** اتفاق می‌افتد.

```
$( "input" ).blur(function() {
  $(this).css("background-color", "#ffffff");
});
```

```
<head>
<script src="jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("input").focus(function(){
    $(this).css("background-color", "#cccccc");
  });
  $("input").blur(function(){
    $(this).css("background-color", "#00ff00");
  });
});
</script>
</head>
<body>
  Name: <input type="text" name="fullname"><br>
  Email: <input type="text" name="email">
</body>
```

افکت ها در **jQuery**

پنهان و نمایان کردن عناصر در **jQuery**

متدهای **show()** و **hide()**

در **jQuery**, با متدهای **show()** و **hide()** می توانید عناصر HTML را پنهان و نمایان کنید:

مثال پنهان و نمایان کردن عناصر در **jQuery**

```
$("#hide").click(function(){
    $("p").hide();
});

$("#show").click(function(){
    $("p").show();
});
```

نحوه استفاده:

\$(selector).hide(speed,callback);

\$(selector).show(speed,callback);

پارامترها:

توضیح	پارامتر
اختیاری است، سرعت پنهان یا نمایان شدن را مشخص می کند و می تواند با مقادیر "fast" ، "slow" و یا میلی ثانیه مقدار دهی شود.	Speed
اختیاری است، تابعی است که بعد از اجرای کامل متدهای hide() و show() اجرا می شود.	Callback

در مثال زیر، پارامتر **speed** همراه با متدهای **hide()** و **show()** استفاده شده است:

مثال پنهان و نمایان کردن عناصر در **jQuery**

```
$( "button" ).click(function(){
    $("p").hide(1000,function(){alert('hi');});
});
```

این مثال نشان می دهد که چگونه با کلیک روی تگ **p** آنرا مخفی کنیم.

مثال **hide()**

```
<html>
<head>
    <script src="jquery.min.js"></script>
    <script>
        $(document).ready(function(){
            $("p").click(function(){
                $(this).hide();
            });
        });
    </script>
</head>
<body>
    <p>If you click on me, I will disappear.</p>
    <p>Click me away!</p>
    <p>Click me too!</p>
</body>
</html>
```

این مثال نشان می دهد که چگونه یک تگ **div** را بصورت آهسته مخفی کنیم.

مثال hide()

```
<head>

<script src="jquery.min.js"></script>

<script>

$(document).ready(function(){

    $(".ex .hide").click(function(){

        $(this).parents(".ex").hide("slow");

    });

});

</script>

<style type="text/css">

div.ex

{

    background-color:#e5eecc;

    padding:7px;

    border:solid 1px #c3c3c3;

}

</style>

</head>
```

```
<body>

    <h3>Island Trading</h3>

    <div class="ex">

        <button class="hide">Hide me</button>

        <p>Contact: Helen Bennett<br>

            Garden House Crowther Way<br>

            London</p>

    </div>

    <h3>Paris specialits</h3>

    <div class="ex">

        <button class="hide">Hide me</button>

        <p>Contact: Marie Bertrand<br>

            265, Boulevard Charonne<br>

            Paris</p>

    </div>

</body>
```

متدهای **toggle()**

با متدهای **show()** و **hide()** می توانید بین عناصر پنهان و نمایان حرکت کنید.

عناصر نمایان را پنهان و عناصر پنهان را نمایان می کند:

۳ مثال پنهان و نمایان کردن عناصر در jQuery

```
$( "button" ).click(function(){
  $( "p" ).toggle();
});
```

نحوه استفاده:

```
$(selector).toggle(speed,callback);
```

پارامتر ها

توضیح	پارامتر
اختیاری است، سرعت پنهان یا نمایان شدن را مشخص می کند و می تواند با مقادیر "fast" ، "slow" و "fast" یا میلی ثانیه مقدار دهی شود.	Speed
اختیاری است، تابعی است که بعد از اجرای کامل متدهای () hide() و () show() اجرا می شود.	Callback

محو کردن عناصر در jQuery

با jQuery می توان عناصر را کم کم ناپدید و یا نمایان کرد.

این مثال، چگونگی نمایان کردن عناصر را نشان می دهد.

fadeIn() مثال

```
<head>
  <script src="jquery.min.js"></script>
<script>
  $(document).ready(function(){
    $("button").click(function(){
      $("#div1").fadeIn();
      $("#div2").fadeIn("slow");
      $("#div3").fadeIn(3000); });
    });
  </script>
```

```

</head>

<body>

<p>Demonstrate fadeIn() with different parameters.</p>

<button>Click to fade in boxes</button>

<br><br>

<div id="div1" style="width:80px;height:80px;display:none;background-color:red;"></div><br>
<div id="div2" style="width:80px;height:80px;display:none;background-color:green;"></div><br>
<div id="div3" style="width:80px;height:80px;display:none;background-color:blue;"></div>

</body>

```

این مثال، چگونگی محو کردن عناصر را نشان می دهد.

مثال

```

<html>

<head>

<script src="jquery.min.js"></script>

<script>

$(document).ready(function(){

    $("button").click(function(){

        $("#div1").fadeOut();

        $("#div2").fadeOut("slow");

        $("#div3").fadeOut(3000);

    });

});

</script>

</head>

```

```

<body>

    <p>Demonstrate fadeOut() with different parameters.</p>

    <button>Click to fade out boxes</button>

    <br><br>

    <div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>

    <div id="div2" style="width:80px;height:80px;background-color:green;"></div>

    <div id="div3" style="width:80px;height:80px;background-color:blue;"></div>

</body>

</html>

```

این مثال، چگونگی محو یا نمایان کردن عناصر را نشان می دهد.

مثال

```

<head>

<script src="jquery.min.js"></script>

<script>

$(document).ready(function(){

    $("button").click(function(){

        $("#div1").fadeToggle();
        $("#div2").fadeToggle("slow");
        $("#div3").fadeToggle(3000);

    });

});

</script>

</head>

```

```

<body>

    <p>Demonstrate fadeToggle() with different speed parameters.</p>

    <button>Click to fade in/out boxes</button>

    <br/><br/>

    <div id="div1" style="width:80px;height:80px;background-color:red;"></div>

    <br/>

    <div id="div2" style="width:80px;height:80px;background-color:green;"></div>

    <br/>

    <div id="div3" style="width:80px;height:80px;background-color:blue;"></div>

</body>

```

متدهای fadeTo()

متدهای fadeTo()، یک عنصر را تا یک مقدار مشخص، محو می کند. (مقداری بین ۰ و ۱)

نحوه استفاده:

`$(selector).fadeTo(speed,opacity,callback);`

پارامترها:

پارامتر	توضیح
Speed	اختیاری است، سرعت پنهان یا نمایان شدن را مشخص می کند و می تواند با مقادیر "slow" ، "fast" و یا میلی ثانیه مقدار دهی شود.
Callback	اختیاری است، تابعی است که بعد از اجرای کامل متدهای hide() و show() اجرا می شود.

در مثال زیر، متدهای fadeTo() با پارامترهای مختلف نشان داده شده است:

مثال

```
$( "button" ).click(function(){
  $( "#div1" ).fadeTo("slow",0.15);
  $( "#div2" ).fadeTo("slow",0.4);
  $( "#div3" ).fadeTo("slow",0.7);
});
```

این مثال، چگونگی محو کردن عناصر تا یک مقدار مشخص را نشان می دهد.

مثال

```
<html>
<head>
<script src="jquery.min.js"></script>
<script>
$(document).ready(function(){

  $("button").click(function(){

    $( "#div1" ).fadeTo("slow",0.15);

    $( "#div2" ).fadeTo("slow",0.4);

    $( "#div3" ).fadeTo("slow",0.7);

  });

});

</script>
</head>
```

```
<body>

<p>Demonstrate fadeTo() with different parameters.</p>

<button>Click to fade boxes</button>

<br><br>

<div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>

<div id="div2" style="width:80px;height:80px;background-color:green;"></div><br>

<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>

</body>

</html>
```

پنهان و نمایان کردن عناصر بصورت اسلایدی در **jQuery**

با **jQuery** می توان عناصر را بصورت اسلایدی مخفی و یا نمایان کرد.

متدهایی که برای این منظور در **jQuery** استفاده می شود:

- slideDown() •
- slideUp() •
- slideToggle() •

slideDown() متدهایی که برای این منظور در **jQuery** استفاده می شود:

متدهایی که برای این منظور در **jQuery** استفاده می شود:

نحوه استفاده:

```
$(selector).slideDown(speed,callback);
```

پارامترها:

پارامتر	توضیح
Speed	اختیاری است، سرعت پنهان یا نمایان شدن را مشخص می کند و می تواند با مقادیر "slow" ، "fast" و یا میلی ثانیه مقدار دهی شود.
Callback	اختیاری است، تابعی است که بعد از اجرای کامل متدهای hide() و show() اجرا می شود.

مثال زیر، نحوه‌ی استفاده از متدهای slideDown() و slideUp() را نشان می دهد:

مثال
<pre>\$("#flip").click(function(){ \$("#panel").slideDown(); });</pre>

مثال: در این مثال، چگونگی نمایان کردن یک عنصر پنهان را بصورت اسلایدی نشان می دهد.

```
<head>
  <script src="jquery.min.js"></script>
  <script>
    $(document).ready(function(){
      $("#flip").click(function(){
        $("#panel").slideDown("slow");
      });
    });
  </script>
```

```

<style type="text/css">

    #panel,#flip

    {
        padding:5px;
        text-align:center;
        background-color:#e5eecc;
        border:solid 1px #c3c3c3;
    }

    #panel
    {
        padding:50px;
        display:none;
    }

</style>

</head>

<body>

    <div id="flip">Click to slide down panel</div>
    <div id="panel">Hello world!</div>

</body>

```

متد **slideUp()**

متد **slideUp()**، یک عنصر را به صورت اسلایدی پنهان می کند.

نحوه استفاده:

`$(selector).slideUp(speed,callback);`

پارامترها:

پارامتر	توضیح
Speed	اختیاری است، سرعت پنهان یا نمایان شدن را مشخص می کند و می تواند با مقادیر "slow" ، "fast" و یا میلی ثانیه مقدار دهی شود.
Callback	اختیاری است، تابعی است که بعد از اجرای کامل متدهای <code>hide()</code> و <code>show()</code> اجرا می شود.

مثال زیر، نحوه‌ی استفاده از متدهای `slideUp()` را نشان می‌دهد:

مثال
<pre>\$("#flip").click(function(){ \$("#panel").slideUp(); }); <head> <script src="jquery.min.js"></script> <script> \$(document).ready(function(){ \$("#flip").click(function(){ \$("#panel").slideUp("slow"); }); }); </script></pre>

```

<style type="text/css">

#panel,#flip

{
padding:5px;
text-align:center;
background-color:#e5eecc;
border:solid 1px #c3c3c3;

}

#panel

{
padding:50px;
}

</style>

</head>

<body>

<div id="flip">Click to slide up panel</div>

<div id="panel">Hello world!</div>

</body>

```

متدهای slideToggle()

متدهای slideUp() و slideDown() بین این دو حالت حرکت می کند.

عنصر پنهان را بصورت اسلایدی نمایان و یا عنصر نمایان را بصورت اسلایدی پنهان می کند.

نحوه استفاده:

`$(selector).slideToggle(speed,callback);`

پارامترها:

پارامتر	توضیح
Speed	اختیاری است، سرعت پنهان یا نمایان شدن را مشخص می کند و می تواند با مقادیر "slow" و "fast" یا میلی ثانیه مقدار دهی شود.
Callback	اختیاری است، تابعی است که بعد از اجرای کامل متدهای <code>hide()</code> و <code>show()</code> اجرا می شود.

مثال زیر، نحوه استفاده از متده استفاده از `slideToggle()` را نشان می دهد:

```
$("#flip").click(function(){
  $("#panel").slideToggle();
});

<head>
  <script src="jquery.min.js"></script>
  <script>
    $(document).ready(function(){
      $("#flip").click(function(){
        $("#panel").slideToggle("slow");
      });
    });
  </script>
```

```

<style type="text/css">

#panel,#flip

{
    padding:5px;
    text-align:center;
    background-color:#e5eecc;
    border:solid 1px #c3c3c3;
}

#panel
{
    padding:50px;
    display:none;
}

</style>

</head>

<body>

<div id="flip">Click to slide the panel down or up</div>

<div id="panel">Hello world!</div>

</body>

</html>

```

jQuery متحرک سازی در

متند (`animate()`)، برای متحرک سازی در jQuery استفاده می شود.

متد - **jQuery animate()** متحرک سازی در

متد **()** **animate()** ، برای متحرک سازی در **jQuery** استفاده می شود.

نحوه استفاده:

```
$(selector).animate({params},speed,callback);
```

پارامترها:

پارامتر	توضیح
Params	الزامی است، یک یا چند خصوصیت دلخواه CSS برای متحرک سازی است.
Speed	اختیاری است، سرعت پنهان یا نمایان شدن را مشخص می کند و می تواند با مقادیر "fast" ، "slow" و یا میلی ثانیه مقدار دهی شود.
Callback	اختیاری است، تابعی است که بعد از اجرای کامل متدهای hide() و show() اجرا می شود.

مثال: در مثال زیر، با استفاده از متد **()** **animate()** یک عنصر **<div>** را حرکت می دهیم تا خصوصیت **left** آن برابر با **250px** شود:

```
<head>
<script src="jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("div").animate({left:'250px'});
    });
});
</script>
</head>
```

```

<body>

    <button>Start Animation</button>

    <p>By default, all HTML elements have a static position, and cannot be moved.  

To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!</p>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;">

</div>

</body>

```

بصورت پیشفرض تمام عناصر HTML دارای یک مکان استاتیک هستند و نمی توانند حرکت داده شوند. برای دستکاری مکان یک عنصر، باید خصوصیت position آنرا با یکی از مقادیر absolute ،fixed و یا relative مقداردهی نمود.

تنظیم چند خصوصیت CSS در متدهای animate()

خصوصیت ها به طور همزمان برای متحرک سازی اعمال می شوند: مثال:

```

<head>

<script src="jquery.min.js"></script>

<script>

$(document).ready(function(){

    $("button").click(function(){

        $("div").animate({ left:'250px', opacity:'0.5', height:'150px', width:'150px' });

    });

});

</script>

</head>

```

```

<body>

<button>Start Animation</button>

<p>By default, all HTML elements have a static position, and cannot be moved. To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!</p>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;">

</div>

</body>

```

آیا تمام خصوصیت های CSS را می توان در متدهای `animate()` استفاده نمود؟

تقریباً بله! اما یک نکته مهم وجود دارد: برای استفاده از خصوصیت های CSS در متدهای `animate()` باید آنها را "دنبال هم" نوشت. مثلاً بجای `paddingRight` از `padding-right` و یا بجای `margin-left` از `marginLeft` استفاده نمود.

مقدار دهی نسبی در متدهای `animate()`

می توان خصوصیت ها را نسبت به مقادیر فعلی شان، مقدار دهی نمود. برای انجام این کار هنگام مقدار دهی از `"+="` یا `"-="` استفاده کنید: مثال:

```

<head>

<script src="jquery.min.js"></script>

<script>

$(document).ready(function(){

    $("button").click(function(){

        $("div").animate({ left:'250px', height:'+=150px', width:'+=150px' });

    });

});

</script>

</head>

```

```
<body>

    <button>Start Animation</button>

    <p>By default, all HTML elements have a static position, and cannot be moved. To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!</p>

    <div style="background:#98bf21;height:100px;width:100px;position:absolute;">

    </div>

</body>
```

تنظیم مقادیر از پیش تعریف شده در متده **animate()**

شما حتی می توانید خصوصیت ها را با مقادیر "show", "hide" و یا "toggle" مقداردهی کنید: مثال:

```
<head>

<script src="jquery.min.js"></script>

<script>

$(document).ready(function(){

    $("button").click(function(){

        $("div").animate({

            height:'hide'

        });

    });

});

</script>

</head>
```

```

<body>

    <button>Start Animation</button>

    <p>By default, all HTML elements have a static position, and cannot be moved. To
manipulate the position, remember to first set the CSS</p>

    <div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>

</body>

</html>

```

قابلیت صف کردن چندین متدهای animate()

اگر از متدهای `animate()` چندین بار پشت سر هم استفاده کنید، jQuery یک صف داخلی از آنها ایجاد می کند. سپس آنها را یکی بعد از دیگری فراخوانی می کند.

بنابراین اگر می خواهید متحرک سازی های مختلفی را یکی بعد از دیگری اجرا کنید، بهتر است از قابلیت صف در jQuery استفاده کنید: مثال:

```

<head>

<script src="jquery.min.js"></script>

<script>

$(document).ready(function(){

    $("button").click(function(){

        var div=$("#div");

        div.animate({height:'300px',opacity:'0.4'},"slow");

        div.animate({width:'300px',opacity:'0.8'},"slow");

        div.animate({height:'100px',opacity:'0.4'},"slow");

        div.animate({width:'100px',opacity:'0.8'},"slow");

    });

});

</script>

```

```

</head>

<body>

    <button>Start Animation</button>

    <p>By default, all HTML elements have a static position, and cannot be moved. To manipulate the position, remember to first set the CSS position property </p>

    <div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>

</body>

</html>

```

در مثال زیر، ابتدا عنصر `<div>` به سمت راست حرکت می کند و سپس اندازه متن داخل آن افزایش می یابد:

```

<head>

    <script src="jquery.min.js"></script>

    <script>

        $(document).ready(function(){

            $("button").click(function(){

                var div=$("#div");

                div.animate({left:'100px'},"slow");

                div.animate({fontSize:'3em'},"slow");

            });

        });

    </script>
</head>

<body>

    <button>Start Animation</button>

    <p>By default, all HTML elements have a static position, and cannot be moved. To manipulate the position</p>

    <div style="background:#98bf21;height:100px;width:200px;position:absolute;">HELLO</div>

</body>

```

متوقف کردن متحرک سازی متدها **stop()**

متدها **stop()**، برای متوقف کردن متحرک سازی استفاده می شود.

متدها **stop()**، برای متوقف کردن متحرک سازی استفاده می شود.

نحوه استفاده `$(selector).stop(stopAll,goToEnd);`

پارامترها:

پارامتر	توضیح
<code>stopAll</code>	اختیاری است، مشخص می کند که آیا تمام اینیمیشن هایی که در صف قرار دارند متوقف شوند یا نه . مقدار پیشفرض "false" است و معنی آن این است که فقط اینیمیشنی که در حال اجراست متوقف خواهد شد و اجازه می دهد که بقیه اینیمیشن هایی که در صف قرار دارند اجرا شوند.
<code>goToEnd</code>	اختیاری است، مشخص می کند که آیا اینیمیشن جاری فوراً به انتهای اجرا برود یا در همان جا متوقف شود. مقدار پیشفرض "false" است.

بنابراین بصورت پیشفرض، متدها **stop()**، اینیمیشن جاری را تا همان جایی که اجرا شده متوقف می کند و سپس بقیه اینیمیشن هایی که در صف هستند را در ادامه، اجرا می کند.

در مثال زیر، متدها **stop()** بدون پارامتر نشان داده شده است:

```
<head>
<script src="jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("#flip").click(function(){ $("#panel").slideDown(5000); });
    $("#stop").click(function(){ $("#panel").stop(); });
});
</script>
```

```
<style type="text/css">  
  
#panel,#flip  
{  
padding:5px;  
text-align:center;  
background-color:#e5eecc;  
border:solid 1px #c3c3c3;  
}  
  
#panel  
{  
padding:50px;  
display:none;  
}  
  
</style>  
  
</head>  
  
<body>  
    <button id="stop">Stop sliding</button>  
    <div id="flip">Click to slide down panel</div>  
    <div id="panel">Hello world!</div>  
</body>
```

با پارامتر: مثال:

```
<head>

<script src="jquery.min.js"></script>

<script>

$(document).ready(function(){

    $("#start").click(function(){

        $("div").animate({left:'100px'},5000);

        $("div").animate({fontSize:'3em'},5000);

    });

    $("#stop").click(function(){

        $("div").stop();

    });

    $("#stop2").click(function(){

        $("div").stop(true);

    });

    $("#stop3").click(function(){

        $("div").stop(true,true);

    });

});

</script>

</head>
```

```

<body>

    <button id="start">Start</button>

    <button id="stop">Stop</button>

    <button id="stop2">Stop all</button>

    <button id="stop3">Stop but finish</button>

    <p>The "Start" button starts the animation.</p>

    <p>The "Stop" button stops the current active animation, but allows the </p>

    <p>The "Stop all" button stops the current active animation and clears the </p>

    <p>The "Stop but finish" rushes through the current active animation, then it stops.</p>

    <div style="background:#98bf21;height:100px;width:200px;position:absolute;">HELLO</div>

</body>

```

تابع **jQuery** در **callback**

تابع **callback** ، بعد از اجرای کامل متحرک سازی اجرا می شود.

دستورات **JavaScript** ، خط به خط اجرا می شوند. اما هنگام استفاده از دستورات متحرک سازی، خط بعدی می تواند بدون اجرای کامل دستور قبلی اجرا شود. این امر می تواند موجب بروز خطا شود.

برای جلوگیری از بروز خطا، می توان از توابع **callback** استفاده نمود.

تابع **callback** ، بعد از اجرای کامل متحرک سازی اجرا می شود.

یک نمونه از نحوه استفاده تابع **callback**

```
$(selector).hide(speed,callback);
```

مثال : در مثال زیر، پارامتر **callback** در متده **hide()** با یک تابع تنظیم شده است، این تابع بعد از اجرای کامل افکت پنهان سازی، اجرا خواهد شد: مثال:

```
<head>

<script src="jquery.min.js"></script>

<script>

$(document).ready(function(){

    $("button").click(function(){

        $("p").hide("slow",function(){

            alert("The paragraph is now hidden");

        });

    });

});

</script>

</head>

<body>

    <button>Hide</button>

    <p>This is a paragraph with little content.</p>

</body>
```

در مثال زیر، پارامتر **callback** استفاده نشده است، و قبل از اینکه افکت پنهان سازی کامل شود، پنجره **alert** نمایش داده خواهد شد:

مثال

```
<!DOCTYPE html>

<html>

<head>

<script src="jquery.min.js"></script>

<script>

$(document).ready(function(){

    $("button").click(function(){

        $("p").hide(1000);

        alert("The paragraph is now hidden");

    });

});

</script>

</head>

<body>

    <button>Hide</button>

    <p>This is a paragraph with little content.</p>

</body>

</html>
```

زنجیره ای کردن متدها در **jQuery**

با **jQuery**، می توانید متدها را به یکدیگر زنجیر کنید.

زنجیره ای کردن به ما اجازه می دهد تا تنها با یک دستور، چندین متده را روی یک عنصر مشخص، اجرا کنیم.

زنجیره ای کردن (chaining) متدها در **jQuery**

تا حالا دستورات **jQuery** را یکی بعد از دیگری نوشته ایم. اما تکنیکی به نام "زنجیره ای کردن" یا "chaining" وجود دارد که به ما اجازه می دهد تا تنها با یک دستور، چندین متده را روی یک عنصر مشخص، اجرا کنیم.

توجه: به این ترتیب، برای اعمال کارهای مختلف، مرورگرها مجبور نیستند که عنصرهای یکسان را بیش از یک مرتبه شناسایی کنند.

برای زنجیره ای کردن کارها، به صورت ساده عمل دلخواه تان را به عمل قبلی متصل کنید.

در مثال زیر، متدهای `css()` و `slideDown()` و `slideUp()` به یکدیگر زنجیر شده اند. عنصر با شناسه "#p1" ابتدا به رنگ قرمز، سپس به صورت اسلایدی پنهان می شود و در نهایت به صورت اسلایدی نمایان خواهد شد:

```
<head>
<script src="jquery.min.js"></script>
<script>
$(document).ready(function()
{
    $("button").click(function(){
        $("#p1").css("color","red").slideUp(2000).slideDown(2000);
    });
});
</script>
</head>
```

```
<body>  
    <p id="p1">jQuery is fun!!</p>  
    <button>Click me</button>  
</body>
```

شما می توانید هر تعداد متدهای دیگری که نیاز دارید را به انتهای کد بالا اضافه نمایید.

توجه: زمانی که در حال زنجیره ای کردن متدها هستید، می توانید هر متدها را در یک خط مانند زیر قرار دهید:

```
<head>  
    <script src="jquery.min.js"></script>  
  
    <script>  
        $(document).ready(function()  
        {  
            $("button").click(function(){  
                $("#p1").css("color","red").slideUp(2000).slideDown(2000);  
            });  
        });  
    </script>  
</head>  
<body>  
    <p id="p1">jQuery is fun!!</p>  
    <button>Click me</button>  
</body>
```

jQuery، خطوط سفید اضافی را حذف می کند و کد بالا را مانند یک خط کد اجرا خواهد کرد.

مطالب بیشتر:

انواع رویدادها:

رویداد	معنی و کاربرد
Onchange	این رویداد در حالتی اجرا می شود که مقادیر یکی از فیلدهای فرم در صفحات وب تغییر کند.
onclick	این رویداد در حالتی اجرا می شود که کاربر بر روی یک عنصر کلیک کند، در واقع پس از فشردن و رها کردن ماوس یا هنگامی که از صفحه کلید برای اجرای عناصر استفاده می شود.
Ondblclick	این رویداد در حالتی اتفاق می افتد که کاربر بر روی یک عنصر طی بازه زمانی استاندارد، دابل کلیک کند.
Onhashchange	این رویداد در حالتی اتفاق می افتد که مقادیر hash در نوار آدرس مرورگر تغییر کند، منظور از در اینجا، علامت # در آدرس های وب است.
Onfocus	این رویداد در حالتی اتفاق می افتد که کاربر یک عنصر را به حالت focus در می آورد، به فرض انتخاب یک فیلد در فرم های وب.
Onblur	این رویداد در حالتی اجرا می شود که یک عنصر از حالت focus و انتخاب خارج شود، مانند فرم های وب.
Onfocusin	این رویداد مشابه onfocus است.
Onfocusout	این رویداد مشابه onblur است.
Onkeydown	این رویداد در حالتی کاربرد دارد که کاربر یک دکمه از صفحه کلید را فشرده و نگه دارد.
Onkeypress	این رویداد زمانی اتفاق می افتد که کاربر یک حرف از صفحه کلید را که برای عنصر خاصی تعریف شده است، فشار دهد.

این رویداد زمانی اتفاق می افتد که کاربر یک دکمه را فشرده و رها کند.	Onkeyup
این رویداد زمانی اتفاق می افتد که کاربر کلیک کرده و ماوس را همچنان نگه دارد.	Onmousedown
این رویداد زمانی اتفاق می افتد که کاربر ماوس را حرکت دهد.	Onmousemove
این رویداد زمانی اتفاق می افتد که ماوس از محدوده یک عنصر خارج شود.	Onmouseout
این رویداد زمانی اتفاق می افتد که ماوس وارد محدوده یک عنصر شود.	Onmouseover
این رویداد زمانی اتفاق می افتد که ماوس از حالت کلیک خارج شود.	Onmouseup
این رویداد در حالتی اتفاق می افتد که ماوس وارد محدوده یک عنصر شود، مشابه رویداد onmouseover.	Onmouseenter
این رویداد در حالتی اتفاق می افتد که ماوس از محدوده یک عنصر خارج شود، مشابه رویداد onmouseout.	Onmouseleave
این رویداد در حالتی اتفاق می افتد که کاربر دکمه چرخان روی ماوس را حرکت دهد، حتی اگر اسکرول صفحه تغییر نکند.	Onmousewheel
این رویداد مشابه رویداد onmousewheel است.	Onwheel
این رویداد در حالتی کاربرد دارد که کاربر یک متن را انتخاب می کند.	Onselect
این رویداد در حالتی کاربرد دارد که صفحه دچار خطای جوا اسکریپتی بوده یا دسترسی به یک فایل ضروری ممکن نباشد.	Onerror
این رویداد در حالتی اتفاق می افتد که یک صفحه html یا یک فایل به طور کامل بارگذاری می شود.	Onload
این رویداد زمانی اتفاق می افتد که اندازه پنجره مرورگر تغییر کند.	Onresize
این رویداد در حالتی کاربرد دارد که کاربر صفحه فعلی را ترک می کند.	Onunload

این رویداد در حالتی اتفاق می افتد که کاربر کلیک راست را انتخاب کند.	Oncontextmenu
این رویداد در حالتی کاربرد دارد که کاربر یک متن را کپی می کند.	Oncopy
این رویداد در حالتی کاربرد دارد که کاربر یک متن را <code>cut</code> می کند.	Oncut
این رویداد در حالتی کاربرد دارد که کاربر یک متن را <code>paste</code> می کند.	Onpaste
این رویداد در حالتی کاربرد دارد که کاربر یک فرم را <code>reset</code> می کند.	Onreset
این رویداد در حالتی اتفاق می افتد که صفحه یا پنجره ای اسکرول شود.	Onscroll
این رویداد در حالتی کاربرد دارد که کاربر یک فرم را ارسال می کند.	Onsubmit
این رویداد در حالتی کاربرد دارد که یک کاراکتر با یک رویداد در هنگام وارد کردن اطلاعات در یک فرم، نسبت داده شده باشد.	Ontextinput
این رویداد هنگامی اتفاق می افتد که کاربر، بارگذاری یک <code></code> یا را بی نتیجه می گذارد.	Onabort

خواص شی `string`

در جدول زیر ، مجموعه خواص شی `string` ارائه شده است .

نام خاصیت	شرح
<code>constructor</code>	نام تابع سازنده شی مورد نظر را بر می گرداند.
<code>prototype</code>	به برنامه نویس امکان اضافه کردن خواص و متدهای جدید را به شی می دهد.

خاصیت constructor

این خاصیت، نام تابع سازنده شی مورد نظر را برمی‌گرداند. تابع سازنده، تابعی است که در هنگام تعریف اولیه، شی را به وجود آورده است. شکل کلی استفاده از این خاصیت به شرح زیر است:

Syntax	object.constructor * object = نام شی مورد نظر
--------	--

مثال: در مثال زیر با استفاده از تابع `string()` یک شی `string` جدید به نام `txtname` ایجاد کرده‌ایم. به وسیله دستور `document.write`، نوع تابع سازنده شی را نشان داده‌ایم:
نکته: توجه شود شکل صحیح نوشتاری تابع `String` با `S` بزرگ است. مثال:

< script type="text/javascript" > var txtname = new String () ; document.write (txtname.constructor) ; </script >	کد
function String() { [native code] }	خروجی

خاصیت prototype

به وسیله این خاصیت می‌توان خواص و متدهای جدید مورد نظر خود را به شی اضافه کرد. پس از تعریف خاصیت یا متدهای خواص ذاتی شی استفاده کرد. شکل کلی استفاده از این خاصیت به شرح زیر است:

Syntax	object.prototype.name = value * object = نام خاصیت یا متدهای خواص ذاتی شی name = نام شی مورد نظر value = مقدار اولیه خاصیت جدید یا نام متدهای خواص ذاتی شی
--------	--

مثال: در مثال زیر ابتدا یک کلاس به نام `Student` که دارای ۳ خاصیت نام (Name)، نام خوانندگی (Family) و شماره دانشجویی (ID) است را ایجاد کرده و یک شی جدید به نام `St2` از روی آن ساخته‌ایم.

سپس به کلاس یک خاصیت جدید به نام Major اضافه کرده ایم ، که قرار است رشته تحصیلی را برای دانشجو نگهداری کند . در پایان ، خاصیت جدید را برای شی St2 مقدار دهی کرده و آن را در خروجی نشان داده ایم

<pre><script type="text/javascript"> function Student (n , f , i) { this.Name = n ; this.Family = f ; this.ID = i ; } var St2 = new Student ("Mehrdad" , "Fattahi" , 122092) ; Student.prototype.Major = null ; Major St2.Major = "Soft Ware" ; St2 document.write (St2.Major) ; </script></pre>	کد
Soft Ware	خروجی

متدهای string

شرح	نام خاصیت
برای نمایش متن یک متغیر رشته ای با اندازه بزرگتر استفاده می شود .	big ()
برای نمایش متن یک متغیر رشته ای به صورت چشمک زن استفاده می شود .	blink ()
برای نمایش متن یک متغیر رشته ای به صورت توپر (درشت) استفاده می شود .	bold ()
این متدها برای نمایش متن متغیر رشته ای با قلمی font (شبیه نوشه های تلگراف استفاده می شود .	fixed ()
از این متدها برای نمایش متن یک متغیر رشته ای به یک رنگ دلخواه استفاده می شود .	fontcolor ()
از این متدها برای نمایش یک متغیر رشته ای در یک اندازه خاص استفاده می شود .	fontsize ()

از این متدهای نمایش متن یک متغیر رشته ای به صورت کج استفاده می شود .	italics ()
از این متدهای تبدیل متن یک متغیر رشته ای به یک پیوند (HyperLink) استفاده می شود .	link ()
از این متدهای نمایش متن یک متغیر متنی با اندازه ای کوچکتر از حد معمول استفاده می شود .	small ()
از این متدهای نمایش متن یک متغیر متنی با یک خط کشیده شده بر روی آن استفاده می شود .	strike ()
از این متدهای نمایش یک متغیر به صورت اندیس استفاده می شود .	sub ()
از این متدهای برش تعداد معینی از کاراکترهای یک متغیر متنی استفاده می شود .	substr ()
از این متدهای نمایش یک متغیر به صورت زیرنویس استفاده می شود .	sup ()

متدهایی که از این متدهای نمایش متن یک متغیر رشته ای با اندازه ای بزرگتر استفاده می شود .

Syntax	stringobject.big ()
--------	----------------------

مثال : در مثال زیر یک متغیر رشته ای به نام matn را با یکباره صورت ساده و بار دیگر با استفاده از متدهای نمایش داده ایم . به تفاوت ۲ حالت دقت کنید : مثال :

حالت اول بدون متدهای big	حالت دوم با متدهای big
<pre><script type ="text/javascript"> var matn = "Developer Studio" ; document.write (matn); </script></pre>	<pre><script type ="text/javascript"> var matn = "Developer Studio" ; document.write (matn.big ()); </script></pre>

متدهایی که از این متدهای نمایش متن یک متغیر رشته ای با اندازه ای بزرگتر استفاده می شود .

این متدهای نمایش یک متغیر متنی ، به صورت چشمک زن بر روی صفحه استفاده می شود .

نکته : متساقنه این متده در مرورگر (Internet Explorer) پر کاربردترین مرورگر ! ، پشتیبانی نمی شود ، ولی در سایر مرورگرهای از جمله Opera و NetScape ، Firefox و پشتیانی می شود . بنابرین متن خروجی مثال این قسمت فقط در مرورگرهای اشاره شده چشمک خواهد زد .

Syntax	stringobject.blink()
--------	-----------------------

مثال : در مثال زیر یک متغیر رشته ای به نام matn را با متده blink به کار بردہ ایم . مثال :

```
<script type ="text/javascript">
var matn = "Developer Studio" ;
document.write ( matn.blink ( ) );
</script>
```

کد

متده () **bold** : از این متده برای نمایش یک متغیر رشته ، به صورت توپر (درشت) استفاده می شود . عملکرد این متده همانند تگ در HTML است .

Syntax	stringobject.bold()
--------	----------------------

متده () **fixed()** : این متده ، برای نمایش متن متغیر رشته ای با قلمی (font) شبیه نوشه های تلگراف استفاده می شود . عملکرد این متده همانند تگ <tt> </tt> یا تگ <code> </code> در HTML است .

Syntax	stringobject.fixed ()
--------	------------------------

مثال : در مثال زیر یک متغیر رشته ای به نام matn را در حالت اول به صورت ساده و در حالت دوم با متده fixed کار بردہ ایم . به تفاوت در روش دقت کنید : مثال :

<script type ="text/javascript"> var matn = "Developer Studio" ; document.write (matn) ; </script>	<script type ="text/javascript"> var matn = "Developer Studio" ; document.write (matn.fixed ()); </script>	کد
--	---	----

متده است **fontcolor()** : از این متده برای نمایش متن یک متغیر رشته ای به یک رنگ دلخواه استفاده می شود .
رنگ مورد نظر توسط خاصیت **color** تعیین می شود .

تعیین رنگ می تواند به یکی از ۳ روش زیر صورت بگیرد :

۱. نام رنگ مثل **red** یا **blue**

۲. تعیین رنگ به وسیله تابع **rgb** به صورت زیر:

مقدار رنگ قرمز ، مقدار رنگ سبز ، مقدار رنگ آبی **rgb**

این تابع مقدار ۳ رنگ را با هم ترکیب کرده و یک رنگ را ایجاد می کند . که مقدار رنگ توسط عددی بین ۰ تا ۲۵۵ تعیین می شود ، که هر چه عدد بزرگتر باشد میزان آن رنگ در کل رنگ بیشتر خواهد بود .

Example : **rgb (10,65,232)**

۳. تعیین رنگ به صورت عددی ترکیبی در مبنای ۱۶ هگزادسیمال:

میزان رنگ آبی میزان رنگ سبز میزان رنگ قرمز #

Example : **#08FF00**

Syntax	stringobject.fontcolor(color) * color = رنگ مورد نظر
--------	--

مثال : در مثال زیر سه متغیر رشته ای را با متده است **fontcolor** ، هر کدام را یکی از ۳ روش فوق به صورت رنگی نمایش داده ایم: مثال:

```
<script type ="text/javascript">
var Srt1 = "String Object 1" ;
var Srt2 = "String Object 2" ;
var Srt3 = "String Object 3" ;
document.write ( Srt1.fontcolor ( "red" ) + "<br /> " ) ;
document.write ( Srt2.fontcolor ( "rgb(220,10,190)" ) + "<br /> " ) ;
document.write ( Srt1.fontcolor ( "#FA390B" ) ) ;
</script>
```

متدها () : از این متدها برای نمایش یک متغیر رشتہ ای در یک اندازه خاص استفاده می‌شود. اندازه مورد نظر توسط خاصیت `size` در تعیین شده و می‌تواند عددی بین ۱ تا ۷ باشد. با بزرگتر شدن عدد، سایز نوشه نیز افزایش می‌یابد.

Syntax	<code>stringobject.fontsize (size)</code> * size = اندازه مورد نظر
--------	---

مثال : در مثال زیر یک متغیر رشتہ ای را با متدهای `fontsize` در ۲ اندازه متفاوت در خروجی نمایش داده ایم:

```
<script type ="text/javascript">
var S1 = "Text Size 2" ;
var S2 = "Text Size 5" ;
document.write ( S1.fontsize( 2 ) + "<br />" );
document.write ( S2.fontsize ( 5 ) );
</script>
```

متدهای `italics()` : از این متدهای برای نمایش متن یک متغیر رشتہ ای، به صورت کج (`italics`) استفاده می‌شود. عملکرد این متدهای همانند تگ `<i>` یا `` در HTML است.

Syntax	<code>stringobject.italics ()</code>
--------	--------------------------------------

مثال : در مثال زیر یک متغیر رشتہ ای به نام `matn` را یکبار به صورت معمولی و بار دیگر با متدهای `italics` نمایش داده ایم. به تفاوت دقت کنید:

حالت اول بدون متدهای <code>italics</code>	حالت دوم با متدهای <code>italics</code>	
<pre><script type ="text/javascript"> var matn = "Developer Studio" ; document.write (matn); </script></pre>	<pre><script type ="text/javascript"> var matn = "Developer Studio" ; document.write (matn.italics ()); </script></pre>	کد
Developer Studio	<i>Developer Studio</i>	خروجی

متدهایی که ممکن است برای نمایش متن یک متغیر رشتہ ای ، به صورت یک پیوند (Hyperlink) روی صفحه استفاده می شود . عملکرد این متدهای همانند تگ <a> در HTML است . آدرس صفحه یا فایل مقصد پیوند به وسیله خاصیت url تعیین می شود.

Syntax	stringobject.link (url) * url = آدرس صفحه یا فایل مقصد پیوند
--------	---

مثال : در مثال زیر یک متغیر رشتہ ای به نام matn یه صورت یک پیوند در خروجی نمایش داده ایم . مقصد پیوند نیز صفحه اصلی سایت تعیین شده است:

```
<script type ="text/javascript">
var matn = "Developer Studio" ;
document.write ( matn.link ( "http://www.dadekav21.com/contacts2.php" ) );
</script>
```

متدهای sub() : از این متدهای نمایش یک متغیر رشتہ ای ، به صورت زیر نویس استفاده می شود . عملکرد این متدهای همانند تگ <sub> در HTML است.

Syntax	stringobject.sub ()
--------	----------------------

مثال : در مثال زیر یک متغیر رشتہ ای به نام matn و یک متغیر به نام ex ایجاد کرده ایم . در خروجی ابتدا متغیر matn را چاپ کرده و متغیر ex را به صورت اندیس برای آن قرار داده ایم: مثال:

<script type ="text/javascript"> var matn = "www.dadekav21.com" ; var ex = "Website" ; document.write (matn + ex.sub ()); </script>	کد
www.dadekav21.com Website	خروجی

متدهای sup() : از این متدهای نمایش یک متغیر رشتہ ای ، به صورت زیر نویس استفاده می شود . عملکرد این متدهای همانند تگ <sup> در HTML است.

Syntax	stringobject.sup ()
--------	----------------------

مثال : در مثال زیر یک متغیر رشته ای به نام matn و یک متغیر به نام ex ایجاد کرده ایم . در خروجی ابتدا متغیر matn را چاپ کرده و متغیر ex را به صورت اندیس برای آن قرار داده ایم:

```
<script type ="text/javascript">
var matn = "www.dadekav21.com" ;
var ex = "Website" ;
document.write ( matn + ex.sup ( ) ) ;
</script>
```

www.dadekav21.com	website
--	---------

 کد | خروجی |

متده است **substr()**: از این متغیر برای برش و جدا کردن تعداد حروف معینی از یک متغیر متنی ، از یک نقطه مشخص در طول آن استفاده می شود .

در این متده ، نقطه شروع عمل برش را بر حسب شماره یک کاراکتر توسط خاصیت start و تعداد کاراکترهایی که می خواهیم از نقطه شروع برش داده شوند را توسط خاصیت length تعیین می کنیم.

Syntax	stringobject.substr (start , length)
--------	--

* start = شماره کاراکتر آغاز نقطه برش در طول متغیر

* length = تعداد کاراکترهایی که می خواهیم از نقطه شروع برش داده شوند

نکته ۱ : تعیین تعداد کاراکترهای مورد نظر برای عملیات برش اختیاری بوده و می تواند تعیین نشود . در صورت عدم تعیین آن ، انتهای متغیر به عنوان نقطه پایان برش در نظر گرفته می شود .

نکته ۲ : شماره گذاری حروف یک عبارت رشته ای در جاوا اسکریپت ، از سمت چپ بوده و از شماره گذاری از عدد صفر شروع می شود . بنابراین در کلمه ای مثل "Java Script" حرف شماره ۲ ، حرف ۷ و شماره ۷ حرف ۰ خواهد بود .

نکته ۳ : فاصله خالی بین حروف نیز یک کاراکتر حساب شده و دارای شماره خواهد بود .

مثال : در مثال زیر در ۲ حالت به برش متغیر متن پرداخته ایم . در حالت اول عملیات برش را از کاراکتر ۴ و به تعداد ۶ حرف انجام داده ایم . در حالت دوم عملیات برش را از کاراکتر ۴ انجام داده ایم ، ولی تعداد کاراکتر

معینی را در نظر نگرفته ایم . در این حالت انتهای متغیر به عنوان نقطه پایان عملیات برش در نظر گرفته شده است :

<pre><script type ="text/javascript"> var matn = "Developer Studio" ; document.write (matn.substr (4 , 7) + "
"); document.write (matn.substr (4)); </script></pre>	کد
loper S loper Studio	خروجی

متد **(small)**: از این متد برای نمایش یک متغیر رشته ای ، با اندازه ای کوچکتر از حد معمول استفاده می شود . عملکرد این متد همانند تگ **<small>** در HTML است .

Syntax	stringobject.small ()
--------	------------------------

مثال : در مثال زیر یک متغیر رشته ای به نام **matn** را یکبار به صورت معمولی و بار دیگر با متد **small** نمایش داده ایم . به تفاوت دقت کنید :

حالات اول بدون متدمall	حالات دوم با متدمall
<pre><script type ="text/javascript"> var matn = "Developer Studio" ; document.write (matn); </script></pre>	<pre><script type ="text/javascript"> var matn = "Developer Studio" ; document.write (matn.small ()); </script></pre>
Developer Studio	Developer Studio

متد **(strike)**: از این متد برای نمایش یک متغیر رشته ای ، با یک خط کشیده شده بر روی آن استفاده می شود . عملکرد این متد همانند خاصیت **text-decoration: line-through** در CSS است .

Syntax	stringobject. strike ()
--------	--------------------------

مثال : در مثال زیر یک متغیر رشته ای به نام **matn** را با متد **strike** نمایش داده ایم . دقت کنید :

```
<script type="text/javascript">
var matn = "Dadekav21";
document.write ( matn.strike ( ) );
</script>
```

کد

Dadekav21

خروجی

متدهای شی: Math

متد **acos**: برگرداندن \arccos یک عدد:

`Math.acos(0.5);`

نتیجه کد بالا:

1.0471975511965979

تعریف و کاربرد

متد (acos) , آرک کسینوس یک عدد را برمی گرداند به طوری که مقدار برگشتی بین 0 و π رادیان است.

نکته: در صورتی که پارامتر x , بین -1 و 1 نباشد، این متد NaN را برمی گرداند.

نکته: $1 - \text{مقدار } \pi$ را برمی گرداند.

متد **asin**: برگرداندن آرک سینوس یک عدد:

`Math.asin(0.5);`

نتیجه کد بالا:

0.5235987755982989

تعریف و کاربرد

متد (asin) , آرک سینوس یک عدد را برمی گرداند، به طوری که مقدار برگشتی بین $-\frac{\pi}{2}$ و $\frac{\pi}{2}$ است.

در صورتی که پارامتر X بین -1 و 1 نباشد، مرورگر مقدار NaN را برخواهد گرداند.

نکته: عدد 1 مقدار $\text{PI}/2$ را برخواهد گرداند و عدد -1 مقدار $-\text{PI}/2$ را برخواهد گرداند.

متدهای atan: برگرداندن آرک تانژانت یک عدد مشخص:

`Math.atan(2);`

نتیجه کد بالا:

1.1071487177940904

تعريف و کاربرد

متدهای `atan()`، آرک تانژانت یک عدد را برمی گرداند، به طوری که مقدار برگشتی بین $-\text{PI}/2$ و $\text{PI}/2$ رادیان است.

متدهای atan2: فرض کنید که یک نقطه به مختصات (x, y) دارد و می خواهید زاویه بین این نقطه و قسمت مثبت محور X را به دست آورید:

`Math.atan2(8, 4);`

خروجی کد بالا:

1.1071487177940904

تعريف و کاربرد

متدهای `atan2()`، آرک تانژانت خارج قسمت آرگومان هایش را برمی گرداند، به طوری که مقدار برگشتی بین $-\text{PI}$ و PI رادیان است.

عدد برگشت داده شده، زاویه ی پادساعتگرد (به رادیان) بین محور X و نقطه (y, x) را برمی گرداند.

نکته: در متدهای `atan2()`، مختصات y به عنوان اولین آرگومان و مختصات x به عنوان دومین آرگومان قرار می گیرد.

متدهای sin: برگرداندن سینوس یک عدد:

```
Math.sin(3);
```

0.1411200080598672

نتیجه کد بالا :

متدهای sqrt: برگرداندن جذر یک عدد:

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

نتیجه کد بالا: ۳

متدهای tan: برگرداندن تانژانت یک عدد(زاویه):

```
Math.tan(90);
```

نتیجه کد بالا: -1.995200412208242

متدهای cos(): برگرداندن کسینوس یک عدد:

```
Math.cos(3);
```

خروجی کد بالا: -0.9899924966004454

تعريف و کاربرد

متدهای cos، کسینوس یک عدد را برمی گرداند.

نکته: این متدهای کسینوس یک عدد را برمی گردانند. این عدد همان کسینوس زاویه‌ی داده شده است

متدهای exp(): برگرداندن E^x هنگامی که x برابر با ۱ است:

```
Math.exp(1);
```

نتیجه کد بالا: 2.718281828459045

تعريف و کاربرد

متده است $\exp(x)$ مقدار E^x را بر می گرداند. در واقع همان عدد اول است (حدود ۲.۷۱۸۳) و x عددی است که به این متده داده ایم.

استفاده از متده $\exp()$ بر روی اعداد مختلف:

```
var a = Math.exp(-1);  
var b = Math.exp(5);  
var c = Math.exp(10);
```

خروجی a و b و c :

0.36787944117144233
148.4131591025766
22026.465794806718

متده $\log()$ برگرداندن لگاریتم طبیعی عدد ۲:

```
Math.log(2);
```

0.6931471805599453

نتیجه کد بالا:

تعريف و کاربرد

متده $\log()$ لگاریتم طبیعی یک عدد در مبنای E را بر می گرداند.

نکته: در صورتی که پارامتر x منفی باشد، مقدار NaN برگشت داده می شود.

نکته: در صورتی که پارامتر x برابر با 0 باشد، Infinity - برگشت داده خواهد شد.

استفاده از متده $\log()$ بر روی اعداد مختلف:

```
var a = Math.log(2.7183);  
var b = Math.log(2);  
var c = Math.log(1);  
var d = Math.log(0);  
var e = Math.log(-1);
```

1.0000066849139877

0.6931471805599453

0

-Infinity

NaN

متدهای شی: Date

متد	توضیحات
getTime()	از این متد در Javascript، برای برگرداندن میلی ثانیه ها از نیمه شب ۱ زانویه ۱۹۷۰ تا اکنون استفاده می شود.
getTimezoneOffset()	از این متد در Javascript، برای برگرداندن اختلاف بین زمان جهانی (UTC) و زمان محلی در دقیقه استفاده می شود.
getUTCDate()	از این متد در Javascript، برای برگرداندن روز نسبت به ماه طبق زمان جهانی استفاده می شود(از ۱ تا ۳۱).
getUTCDay()	از این متد در Javascript، برای برگرداندن روز نسبت به هفته طبق زمان جهانی استفاده می شود(از ۰ تا ۶).
getUTCFullYear()	از این متد در Javascript، برای برگرداندن سال بعنوان یک عدد چهار رقمی طبق زمان جهانی استفاده می شود.
getUTCHours()	از این متد در Javascript، برای برگرداندن ساعت طبق زمان جهانی استفاده می شود(از ۰ تا ۲۳).
getUTCMilliseconds()	از این متد در Javascript، برای برگرداندن میلی ثانیه ها طبق زمان جهانی استفاده می شود(از ۰ تا ۹۹۹).
getUTCMinutes()	از این متد در Javascript، برای برگرداندن دقیقه ها طبق زمان جهانی، استفاده می شود(از ۰ تا ۵۹).

getUTCMonth()	از این متدها در Javascript، برای برگرداندن ماه، طبق زمان جهانی استفاده می شود(از ۰ تا ۱۱).
getUTCSeconds()	از این متدها در Javascript، برای برگرداندن ثانیه ها، طبق زمان جهانی، استفاده می شود(از ۰ تا ۵۹).
getYear()	غیرقابل استفاده ، به جای آن از متدهای <code>getFullYear()</code> استفاده کنید.
parse()	از این متدها در Javascript، برای تجزیه یک تاریخ و برگرداندن تعداد میلی ثانیه ها از ۱ ژانویه ۱۹۷۰ تا آن تاریخ استفاده می شود.
 setDate()	از این متدها در Javascript، برای ساخت کردن روز نسبت به ماه در یک شیء تاریخ استفاده می شود.
setFullYear()	از این متدها در Javascript، برای ساخت کردن سال(۴ رقمی) در یک شیء تاریخ استفاده می شود.
setHours()	از این متدها در Javascript، برای ساخت کردن ساعت در یک شیء تاریخ استفاده می شود.
setMilliseconds()	از این متدها در Javascript، برای ساخت کردن میلی ثانیه ها در یک شیء تاریخ استفاده می شود.
setMinutes()	از این متدها در Javascript، برای ساخت کردن دقیقه ها در یک شیء تاریخ استفاده می شود.
setMonth()	از این متدها در Javascript، برای ساخت کردن ماه در یک شیء تاریخ استفاده می شود.
setSeconds()	از این متدها در Javascript، برای ساخت کردن ثانیه ها در یک شیء تاریخ استفاده می شود.
 setTime()	از این متدها در Javascript، برای ایجاد یک تاریخ از تعداد مشخصی میلی ثانیه، قبل یا بعد از تاریخ ۱ ژانویه ۱۹۷۰، استفاده می شود.
setUTCDate()	از این متدها در Javascript، طبق زمان جهانی، برای ساخت کردن روز نسبت به ماه در یک شیء تاریخ استفاده می شود.
setUTCFullYear()	از این متدها در Javascript، طبق زمان جهانی، برای ساخت کردن سال(چهار رقمی) در یک شیء تاریخ استفاده می شود.

setUTCHours()	از این متدها در Javascript، برای ساخت کردن ساعت در یک شیء تاریخ طبق زمان جهانی استفاده می شود.
setUTCMilliseconds()	از این متدها در Javascript، برای ساخت کردن میلی ثانیه ها در یک شیء تاریخ، طبق زمان جهانی استفاده می شود.
setUTCMilliseconds()	از این متدها در Javascript، برای ساخت کردن دقیقه ها در یک شیء تاریخ طبق زمان جهانی استفاده می شود.
setUTCMonth()	از این متدها در Javascript، برای ساخت کردن ماه در یک شیء تاریخ طبق زمان جهانی، استفاده می شود.
setUTCSeconds()	از این متدها در Javascript، برای ساخت کردن ثانیه ها در یک شیء تاریخ طبق زمان جهانی استفاده می شود.
setYear()	غیر قابل استفاده به جای آن از متدها <code>setFullYear</code> استفاده کنید.
toDateString()	از این متدها در Javascript، برای جدا کردن قسمت مربوط به تاریخ از یک تاریخ، و تبدیل آن بصورت یک رشته‌ی قابل خواندن، استفاده می شود.
toGMTString()	غیر قابل استفاده به جای آن از متدها <code>toUTCString</code> استفاده کنید.
toISOString()	از این متدها در Javascript، برای برگرداندن تاریخ بعنوان یک رشته، طبق استاندارد ISO استفاده می شود.
toJSON()	از این متدها در Javascript، برای برگرداندن تاریخ بعنوان یک رشته و تغییر فرمت آن طبق فرمت تاریخ در JSON استفاده می شود.
toLocaleString()	از این متدها در Javascript، برای برگرداندن تاریخ بعنوان یک رشته که بصورت معمولی نوشتن تاریخ است، استفاده می شود.
toUTCString()	از این متدها در Javascript، برای تبدیل یک شیء تاریخ به رشته، طبق ساعت جهانی، استفاده می شود.
UTC()	از این متدها در Javascript، برای برگرداندن تعداد میلی ثانیه های یک تاریخ از ۱ ژانویه ۱۹۷۰ تا آن تاریخ، طبق ساعت جهانی استفاده می شود.
valueOf()	از این متدها در Javascript، برای برگرداندن مقدار اولیه‌ی یک شیء تاریخ استفاده می شود.

متدهای Date

متدهای Date میتوانند بازگرداندن میلی ثانیه ها را از نیمه شب تاریخ January 1, 1970 تا یک تاریخ مشخص برمی گردانند.

```
<script type ="text/javascript">
var d = new Date();
var n = d.getTime();
</script>
```

1419180699360

مثال (متدهای Date)

محاسبه تعداد سال ها از تاریخ ۱/۱/۱۹۷۰ تا اکنون:

```
var minutes = 1000 * 60;
var hours = minutes * 60;
var days = hours * 24;
var years = days * 365;
var d = new Date();
var t = d.getTime();
var y = Math.round(t / years);
```

45

خروجی ۴۵ در کد بالا:

متدهای Date

اختلاف زمانی بین ساعت جهانی (UTC) و زمان محلی را برمی گرداند:

متدهای Date اختلاف زمانی بین زمان جهانی (UTC) و زمان محلی را در دقیقه برمی گردانند.

عنوان مثال اگر منطقه زمانی **GMT+2** باشد، خروجی عدد ۱۲۰- خواهد بود.

نکته: مقدار خروجی یک عدد ثابت نیست.

نکته: زمان جهانی (UTC) یک زمان است که توسط استاندارد زمان جهانی تعیین شده است.

نکته: زمان UTC همانند زمان GMT است.

```
<script type ="text/javascript">
var d = new Date();
var n = d.getTimezoneOffset();
</script>
```

1419180699360

متدها

متدها() طبق ساعت جهانی، از شیء date روز را نسبت به ماه برمی‌گرداند.

با اینکه شیء date زمان و تاریخ محلی را مشخص می‌کند، اما متدهای UTC می‌توانند از آن تاریخ و زمان جهانی را محاسبه کنند.

نکته: زمان جهانی(UTC) یک زمان است که توسط استاندارد زمان جهانی تعیین شده است.

نکته: زمان UTC همانند زمان GMT است.

نحوه استفاده: Date.getUTCDate()

(مثال) متدها

طبق ساعت جهانی روز را نسبت به ماه برمی‌گرداند:

```
var d = new Date();
var n = d.getUTCDate();
```

21

متدها

متدها() طبق ساعت جهانی، از شیء date روز را نسبت به ماه برمی‌گرداند.

با اینکه شیء date زمان و تاریخ محلی را مشخص می‌کند، اما متدهای UTC می‌توانند از آن تاریخ و زمان جهانی را محاسبه کنند.

نکته: زمان جهانی(UTC) یک زمان است که توسط استاندارد زمان جهانی تعیین شده است.

نکته: زمان UTC همانند زمان GMT است.

نحوه استفاده: Date.getUTCDate()

(مثال) متدها

طبق ساعت جهانی روز را نسبت به ماه برمی‌گرداند:

```
var d = new Date();
var n = d.getUTCDate();
```

21

(getUTCDate() مثال)

طبق زمان جهانی (UTC) از یک زمان مشخص یا زمان محلی، روز را نسبت به ماه برمی گرداند.

```
var d = new Date("July 21, 1983 01:15:00");
var n = d.getUTCDate();
```

20

متدهای getUTCDay()

متدهای getUTCDay() طبق زمان جهانی، از یک تاریخ مشخص، روز را نسبت به هفته (از ۰ تا ۶) برمی گرداند.

با اینکه شیء date زمان و تاریخ محلی را مشخص می کند، اما متدهای UTC می توانند از آن تاریخ و زمان جهانی را محاسبه کنند.

نکته: یکشنبه (Sunday) هم ارز با ۰ است و دوشنبه (Monday) هم ارز با ۱ است و به همین ترتیب الی آخر.

نکته: زمان جهانی (UTC) یک زمان است که توسط استاندارد زمان جهانی تعیین شده است.

نکته: زمان UTC همانند زمان GMT است.

نحوه استفاده : Date.getUTCDay()

(getUTCDay() مثال)

طبق زمان جهانی، روز را نسبت به هفته برمی گرداند:

```
var d = new Date();
var n = d.getUTCDay();
```

0

(getUTCDay() مثال)

نام روز را نسبت به هفته برمی گرداند (نه فقط یک عدد):

```

var d = new Date();
var weekday = new Array();
weekday[0] = "Sunday";
weekday[1] = "Monday";
weekday[2] = "Tuesday";
weekday[3] = "Wednesday";
weekday[4] = "Thursday";
weekday[5] = "Friday";
weekday[6] = "Saturday";
var n = weekday[d.getUTCDay()];

```

Sunday

متد getUTCFullYear

متد `getUTCFullYear()` طبق زمان جهانی، از یک تاریخ مشخص یک عدد چهار رقمی را بعنوان سال، برمی گرداند.

با اینکه شیء `date` زمان و تاریخ محلی را مشخص می کند، اما متدهای UTC می توانند از آن تاریخ و زمان جهانی را محاسبه کنند.

نکته: زمان جهانی(UTC) یک زمان است که توسط استاندارد زمان جهانی تعیین شده است.

نکته: زمان UTC همانند زمان GMT است.

نحوه استفاده: `Date.getUTCFullYear()`

مثال (متد getUTCFullYear)

سال را نسبت به زمان جهانی برمی گرداند:

```

var d = new Date();
var n = d.getUTCFullYear();

```

2014

مثال (متد getUTCFullYear)

از یک تاریخ مشخص سال را طبق زمان جهانی برمی گرداند:

```

var d = new Date("July 21, 1983 01:15:00");
var n = d.getUTCFullYear();

```

1983

متد getUTCHours

متد `getUTCHours()` طبق زمان جهانی، از یک تاریخ و زمان مشخص، ساعت را برمی گرداند (از ۰ تا ۲۳).

با اینکه شیء date زمان و تاریخ محلی را مشخص می کند، اما متدهای UTC می توانند از آن تاریخ و زمان جهانی را محاسبه کنند.

نکته: زمان جهانی(UTC) یک زمان است که توسط استاندارد زمان جهانی تعیین شده است.

نکته: زمان UTC همانند زمان GMT است.

نحوه استفاده : Date.getUTCHours()

مثال (متدهای getUTCHours)

ساعت را نسبت به زمان جهانی برمی گرداند:

```
var d = new Date();
var n = d.getUTCHours();
```

مثال (متدهای getUTCHours)

برگرداندن ساعت(UTC) از یک تاریخ و زمان مشخص:

var d = new Date("July 21, 1983 01:15:00"); var n = d.getUTCHours();	20
---	----

متدهای getUTCMilliseconds

متدهای getUTCMilliseconds() طبق زمان جهانی، از یک تاریخ و زمان مشخص، میلی ثانیه ها را بر می گرداند (از ۰ تا ۹۹۹).

با اینکه شیء date زمان و تاریخ محلی را مشخص می کند، اما متدهای UTC می توانند از آن تاریخ و زمان جهانی را محاسبه کنند.

نکته: زمان جهانی(UTC) یک زمان است که توسط استاندارد زمان جهانی تعیین شده است.

نکته: زمان UTC همانند زمان GMT است.

نحوه استفاده : Date.getUTCMilliseconds()

مثال (متدهای getUTCMilliseconds)

میلی ثانیه ها را نسبت به زمان جهانی(UTC)، برمی گرداند:

```
var d = new Date();
var n = d.getUTCMilliseconds();
```

966

(**getUTCMilliseconds** متد) مثال

برگرداندن میلی ثانیه های زمان جهانی (UTC)، از یک تاریخ و زمان مشخص:

```
var d = new Date("July 21, 1983 01:15:00:195");
var n = d.getUTCMilliseconds();
```

NaN

متد **getUTCMinutes**

متد (**getUTCMinutes**) طبق زمان جهانی، از یک تاریخ و زمان مشخص، دقیقه ها را برمی گرداند (از ۰ تا ۵۹).

با اینکه شیء **date** زمان و تاریخ محلی را مشخص می کند، اما متد های UTC می توانند از آن تاریخ و زمان جهانی را محاسبه کنند.

نکته: زمان جهانی (UTC) یک زمان است که توسط استاندارد زمان جهانی تعیین شده است.

نکته: زمان UTC همانند زمان GMT است.

نحوه استفاده: **Date.getUTCMinutes()**:

(**getUTCMinutes** متد) مثال

طبق زمان جهانی، دقیقه ها را برمی گرداند:

```
var d = new Date();
var n = d.getUTCMinutes();
```

51

(**getUTCMinutes** متد) مثال

برگرداندن دقیقه های زمان جهانی (UTC)، از یک تاریخ و زمان مشخص:

```
var d = new Date("July 21, 1983 01:15:00");
var n = d.getUTCMinutes();
```

45

متد **getUTCMonth**

متد (**getUTCMonth**) طبق زمان جهانی، از یک تاریخ مشخص، ماه را برمی گرداند (از ۰ تا ۱۱).

نکته : (ژانویه) هم ارز با `January` است و `February` (فوریه) هم ارز با ۱ است و همینطور الی آخر.

با اینکه شیء `date` زمان و تاریخ محلی را مشخص می کند، اما متد های UTC می توانند از آن تاریخ و زمان جهانی را محاسبه کنند.

نکته: زمان جهانی(UTC) یک زمان است که توسط استاندارد زمان جهانی تعیین شده است.

نکته: زمان UTC همانند زمان GMT است.

نحوه استفاده : `Date.getUTCMonth()`

(مثال) متد `getUTCMonth`

طبق زمان جهانی ماه را برمی گرداند:

```
var d = new Date();
var n = d.getUTCMonth();
```

11

(مثال) متد `getUTCMonth`

طبق زمان جهانی، نام ماه را برمی گرداند(نه فقط یک عدد):

```
var d = new Date()
var month = new Array(12);
month[0] = "January";
month[1] = "February";
month[2] = "March";
month[3] = "April";
month[4] = "May";
month[5] = "June";
month[6] = "July";
month[7] = "August";
month[8] = "September";
month[9] = "October";
month[10] = "November";
month[11] = "December";
var n = month[d.getUTCMonth()];
```

December

متدهای `getUTCSeconds`

متدهای `getUTCSeconds` طبق زمان جهانی، از یک تاریخ و زمان مشخص، ثانیه ها را برمی گرداند(از ۰ تا ۵۹).

با اینکه شیء date زمان و تاریخ محلی را مشخص می کند، اما متدهای UTC می توانند از آن تاریخ و زمان جهانی را محاسبه کنند.

نکته: زمان جهانی (UTC) یک زمان است که توسط استاندارد زمان جهانی تعیین شده است.

نکته: زمان UTC همانند زمان GMT است.

نحوه استفاده: Date.getUTCSeconds()

مثال (متدهای getUTCSeconds)

طبق زمان جهانی، ثانیه ها را برمی گرداند:

```
var d = new Date();
var n = d.getUTCSeconds();
```

مثال (متدهای getUTCMilliseconds)

استفاده از متدهای getUTCSeconds() و getUTCMilliseconds() و getUTCMinutes() و getUTCHours() برای به نمایش درآوردن زمان جهانی (به همراه میلی ثانیه ها):

```
function addZero(x,n) {
  if (x.toString().length < n) {
    x = "0" + x;
  }
  return x;
}

function myFunction() {
  var d = new Date();
  var x = document.getElementById("demo");
  var h = addZero(d.getUTCHours(), 2);
  var m = addZero(d.getUTCMinutes(), 2);
  var s = addZero(d.getUTCSeconds(), 2);
  var ms = addZero(d.getUTCMilliseconds(), 3);
  x.innerHTML = h + ":" + m + ":" + s + ":" + ms;
}
```

متدهای parse

متدهای parse() یک تاریخ را تجزیه می کند و تعداد میلی ثانیه ها را از ۱ ژانویه ۱۹۷۰ تا آن تاریخ برمی گرداند.

Date.parse(datestring)

(parse) مثال (متد

برگرداندن تعداد میلی ثانیه ها از تاریخ ۱ ژانویه ۱۹۷۰ تا تاریخ ۲۱ مارس ۲۰۱۲:

var d = Date.parse("March 21, 2012");	1332271800000
---------------------------------------	---------------

(parse) مثال (متد

محاسبه تعداد سال ها از تاریخ ۱ ژانویه ۱۹۷۰ تا ۲۱ مارس ۲۰۱۲:

var d = Date.parse("March 21, 2012"); var minutes = 1000 * 60; var hours = minutes * 60; var days = hours * 24; var years = days * 365; var y = Math.round(d / years);	42
---	----

setDate متده

متده setDate() در یک شیء date، روز را نسبت به ماه تعیین می کند.

Date.setDate(day)

(setDate) مثال (متد

قرار دادن و تعیین روز از ماه:

var d = new Date(); d.setDate(15);	Mon Dec 15 2014 20:21:51 GMT+0330 (Iran Standard Time)
---------------------------------------	--

(setDate) مثال (متد

روز(از ماه) را برابر با آخرین روز ماه قبل قرار می دهد:

var d = new Date(); d.setDate(0);	Sun Nov 30 2014 20:21:51 GMT+0330 (Iran Standard Time)
--------------------------------------	--

(setDate) مثال (متد

روز(از ماه) را برابر با یک تاریخ مشخص قرار می دهد:

var d = new Date("July 21, 1983 01:15:00"); d.setDate(15);	Fri Jul 15 1983 01:15:00 GMT+0430 (Iran Daylight Time)
---	--

متدهای setFullYear()

متدهای setFullYear() سال(۴ رقمی) را در یک شیء تاریخ تغییر می دهد.

همچنین این متدهای تواند در تغییر ماه و روز هم استفاده شود.

Date.setFullYear(year,month,day)

(مثال) متدهای setFullYear()

تغییر سال به ۲۰۲۰:

var d = new Date(); d.setFullYear(2020);	Mon Dec 21 2020 20:21:52 GMT+0330 (Iran Standard Time)
---	--

(مثال) متدهای setFullYear()

تغییر دادن تاریخ به ۳ نوامبر ۲۰۲۰:

var d = new Date(); d.setFullYear(2020, 10, 3);	Tue Nov 03 2020 20:21:52 GMT+0330 (Iran Standard Time)
--	--

(مثال) متدهای setFullYear()

تغییر تاریخ به ۶ ماه قبل:

var d = new Date(); d.setFullYear(d.getFullYear(), d.getMonth() - 6);	Sat Jun 21 2014 20:21:52 GMT+0430 (Iran Daylight Time)
---	--

متدهای setHours()

متدهای setHours() ساعت را در یک شیء تاریخ تعیین می کند.

همچنین این متدهای تواند در سیستم کردن دقیقه ها و ثانیه ها و میلی ثانیه ها ،استفاده شود.

Date.setHours(hour,min,sec,millisec)

(setHours) مثال (متد

ست کردن ساعت به ۱۵:

var d = new Date(); d.setHours(15);	Sun Dec 21 2014 15:21:52 GMT+0330 (Iran Standard Time)
--	--

مقادیر پارامترها

پارامتر	توضیحات
hour	(ضروری) یک عدد صحیح که ساعت را مشخص می کند. مقادیر قابل قبول از ۰ تا ۲۳ هست اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none">-1: آخرین ساعت از روز قبل را نشان می دهد.24: اولین ساعت از روز بعد را نشان می دهد
min	(اختیاری) یک عدد صحیح که دقیقه ها را مشخص می کند مقادیر قابل قبول از ۰ تا ۵۹ هست اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none">-1: آخرین دقیقه از ساعت قبل را مشخص می کند.60: اولین دقیقه از ساعت بعد را مشخص می کند.
sec	(اختیاری) یک عدد صحیح که ثانیه ها را مشخص می کند مقادیر قابل قبول از ۰ تا ۵۹ هستند اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none">-1: آخرین ثانیه از دقیقه قبل را مشخص می کند60: اولین ثانیه از دقیقه بعد را مشخص می کند
millisec	(اختیاری) یک عدد صحیح که میلی ثانیه ها را مشخص می کند. مقادیر قابل قبول ۰ تا ۹۹۹ هستند اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none">-1: آخرین میلی ثانیه از دقیقه قبل را مشخص می کند1000: اولین میلی ثانیه از دقیقه بعد را مشخص می کند

(setHours) مثال (متد

var d = new Date(); d.setHours(15, 35, 1);	Sun Dec 21 2014 15:35:01 GMT+0330 (Iran Standard Time)
---	--

(مثال) متد setHours

تغییر زمان به ۴۸ ساعت قبل:

var d = new Date(); d.setHours(d.getHours() - 48);	Fri Dec 19 2014 20:21:52 GMT+0330 (Iran Standard Time)
---	--

متد setMilliseconds

متد () setMilliseconds() تعداد میلی ثانیه ها را در یک شیء تاریخ مشخص می کند.

Date.setMilliseconds(millisecond)

(مثال) متد setMilliseconds

ست کردن میلی ثانیه ها به ۱۹۲:

var d = new Date(); d.setMilliseconds(192); var n = d.getMilliseconds();	192
--	-----

مقادیر پارامترها

پارامتر	توضیحات
millisecond	(ضروری). یک عدد صحیح که میلی ثانیه ها را مشخص می کند مقادیر قابل قبول اعداد ۰ تا ۹۹۹ هستند اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none"> - آخرین میلی ثانیه از ساعت قبل را مشخص می کند اولین میلی ثانیه را از ثانیه بعد مشخص می کند دومین میلی ثانیه از ثانیه بعد را مشخص می کند

جزئیات تکنیکی

مقدار برگشتی	یک عدد که میلی ثانیه ها را از ۱ ژانویه ۱۹۷۰ تا تاریخ مشخص شده را برمی گرداند
نسخه JavaScript	1.3

متد setMinutes

متد (setMinutes() دقيقه ها را در شیء تاریخ مشخص می کند.

همچنین این متد برای ست کردن ثانیه ها و میلی ثانیه ها نیز بکار می رود.

Date.setMinutes(min,sec,millisecond)

(setMinutes (متد مثال

ست کردن دقیقه ها به ۱۷ :

var d = new Date(); d.setMinutes(17);	Sun Dec 21 2014 20:17:54 GMT+0330 (Iran Standard Time)
--	--

مقادیر پارامترها

پارامتر	توضیحات
min	(ضروری). یک عدد صحیح که دقیقه ها را مشخص می کند. مقادیر قابل قبول اعداد از ۰ تا ۵۹ هستند اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none">• -1: آخرین دقیقه از ساعت قبل را مشخص می کند.• 60: اولین دقیقه از ساعت بعد را مشخص می کند.
sec	(اختیاری). یک عدد صحیح که ثانیه ها را مشخص می کند. مقادیر قابل قبول اعداد از ۰ تا ۵۹ هستند اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none">• -1: آخرین ثانیه از دقیقه قبل را مشخص می کند.• 60: اولین ثانیه از دقیقه بعد را مشخص می کند.
millisec	(اختیاری). یک عدد صحیح که میلی ثانیه ها را مشخص می کند. مقادیر قابل قبول اعداد از ۰ تا ۹۹۹ هستند اما مقادیر زیر نیز قابل قبول هستند:

- | | |
|--|---|
| | <ul style="list-style-type: none"> • 1: آخرین میلی ثانیه از ثانیه قبل را مشخص می کند. • 1000: اولین میلی ثانیه از ثانیه بعد را مشخص می کند. |
|--|---|

(setMinutes) مثال (متدها)

بردن زمان به ۹۰ دقیقه قبل در یک تاریخ:

var d = new Date(); d.setMinutes(d.getMinutes() - 90);	Sun Dec 21 2014 18:51:54 GMT+0330 (Iran Standard Time)
---	--

setMonth متدها

متدها setMonth() ماه را در یک شیء تاریخ مشخص می کند.

نکته: زانویه هم ارز با ۰ است و فوریه هم ارز با ۱ و همینطور الی آخر.

همچنین می توان از این متدهای سیستمی "روز" نیز استفاده کرد.

Date.setMonth(month,day)

(setMonth) مثال (متدها)

ست کردن ماه به ۴(ماه مه):

var d = new Date(); d.setMonth(4);	Wed May 21 2014 20:21:54 GMT+0430 (Iran Daylight Time)
---------------------------------------	--

مقادیر پارامترها

پارامتر	توضیحات
Month	<p>(ضروری). یک عدد صحیح که ماه را مشخص می کند.</p> <p>مقادیر قابل قبول اعداد از ۰ تا ۱۱ هستند، اما مقادیر زیر قابل قبول هستند:</p> <ul style="list-style-type: none"> • 1: آخرین ماه از سال قبل را مشخص می کند. • 12: اولین ماه از سال بعد را مشخص می کند. • 13: دومین ماه از سال بعد را مشخص می کند.
Day	(اختیاری). یک عدد صحیح که "روز" را مشخص می کند.

	<p>مقادیر قابل قبول اعداد از ۱ تا ۳۱ هستند اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none"> • ۰: آخرین روز از ماه قبل را مشخص می کند. • ۱: روز یکی مانده به آخر از ماه قبلی را مشخص می کند. <p>اگر ماه ۳۱ روزه باشد:</p> <ul style="list-style-type: none"> • ۳۲: روز اول از ماه بعد را مشخص می کند. <p>اگر ماه ۳۰ روزه باشد:</p> <ul style="list-style-type: none"> • ۳۲: روز دوم ماه بعد را مشخص می کند.
--	--

(setMonth) متد

ست کردن ماه به ۴(ماه مه) و ست کردن روز به، ۲۰:

var d = new Date(); d.setMonth(4, 20);	Tue May 20 2014 20:21:54 GMT+0430 (Iran Daylight Time)
---	--

(setMonth) متد

ست کردن تاریخ به آخرین روز از آخرین ماه سال قبل:

var d = new Date(); d.setMonth(d.getMonth(), 0);	Sun Nov 30 2014 20:21:54 GMT+0330 (Iran Standard Time)
---	--

setSeconds متد

متد (setSeconds) ثانیه ها را در یک شیء تاریخ تعیین می کند.

از این متد همچنین می توان در ست کردن میلی ثانیه ها استفاده کرد.

Date.setSeconds(sec,millisecc)

(setSeconds) متد

ست کردن ثانیه ها به، ۳۵:

```
var d = new Date();
d.setSeconds(35);1
```

Sun Dec 21 2014 20:21:35 GMT+0330 (Iran Standard Time)

مقادیر پارامترها

پارامتر	توضیحات
Sec	(ضروری). یک عدد صحیح که ثانیه ها را مشخص می کند . مقادیر قابل قبول اعداد ۰ تا ۵۹ هستند اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none">• ۱- آخرین ثانیه از دقیقه قبل را مشخص می کند.• ۶۰: اولین ثانیه از دقیقه بعد را مشخص می کند.
Millisec	(اختیاری). یک عدد صحیح که میلی ثانیه ها را مشخص می کند . مقادیر قابل قبول اعداد از ۰ تا ۹۹۹ هستند اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none">• ۱- آخرین میلی ثانیه از ثانیه قبل را مشخص می کند.• ۱۰۰۰: اولین میلی ثانیه از ثانیه بعد را مشخص می کند.

مثال (متدهای setSeconds و setTime)

ست کردن هر دوی ثانیه ها و میلی ثانیه ها:

```
var d = new Date();
d.setSeconds(35,825);
var n = d.getSeconds() + ":" +
d.getMilliseconds();
```

35:825

متدهای setSeconds و setTime

متدهای setSeconds و setTime، با اضافه کردن و یا کم کردن تعدادی میلی ثانیه از نیمه شب ۱ ژانویه ۱۹۷۰، یک تاریخ و زمان را معین می کنند.

Date.setTime(millisec)

مثال (متدهای setSeconds و setTime)

اضافه کردن ۱۳۳۲۴۰۳۸۸۲۵۸۸ میلی ثانیه به تاریخ ۱ ژانویه ۱۹۷۰ و برگرداندن تاریخ و زمان جدید:

```
var d = new Date();
d.setTime(1332403882588);
```

Thu Mar 22 2012 12:41:22 GMT+0430 (Iran Daylight Time)

مقادیر پارامترها

پارامتر	توضیحات
Millisec	(ضروری). تعداد مشخصی میلی ثانیه که به تاریخ نیمه شب ۱ ژانویه ۱۹۷۰ اضافه یا کم خواهد شد.

مثال (متدهای مرتبط)

کم کردن ۱۳۳۲۴۰۳۸۸۲۵۸۸ میلی ثانیه از تاریخ ۱ ژانویه ۱۹۷۰، و نشان دادن تاریخ و زمان جدید:

```
var d = new Date();
d.setTime(-1332403882588);
```

Wed Oct 12 1927 19:18:37 GMT+0330 (Iran Standard Time)

متدهای مرتبط

متدهای مرتبط با زمان جهانی (UTC) هستند که روز را نسبت به ماه، در شیء تاریخ سنجیده می‌کنند.

نکته: زمان جهانی (UTC)، زمانی است که توسط استاندارد زمان جهانی تعیین می‌شود.

نکته: زمان UTC همانند زمان GMT است.

Date.setUTCDate(day)

مثال (متدهای مرتبط)

ست کردن روز نسبت به ماه، طبق زمان جهانی:

```
var d = new Date();
d.setUTCDate(15);
```

Mon Dec 15 2014 20:21:56 GMT+0330 (Iran Standard Time)

پارامتر	توضیحات
day	<p>(ضروری). یک عدد صحیح که روز را (نسبت به ماه) تعیین می کند.</p> <p>مقادیر قابل قبول اعداد از ۱ تا ۳۱ هستند، اما مقادیر زیر نیز قابل قبول هستند.</p> <ul style="list-style-type: none"> • ۰: آخرین ساعت از ماه قبل را مشخص می کند. • -۱: ساعت یکی مانده به آخر از ماه قبل را مشخص می کند. <p>اگر ماه ۳۱ روز داشته باشد:</p> <ul style="list-style-type: none"> • ۳۲: اولین روز از ماه بعد را مشخص می کند. <p>اگر ماه ۳۰ روز داشته باشد:</p> <ul style="list-style-type: none"> • ۳۲: دومین روز از ماه بعد را مشخص می کند.

مثال (متد **setUTCDate**)

ست کردن روز (نسبت به ماه)، به طوری که آخرین روز از ماه قبل را مشخص کند:

var d = new Date(); d.setUTCDate(0);	Sun Nov 30 2014 20:21:56 GMT+0330 (Iran Standard Time)
---	--

مثال (متد **setUTCDate**)

ست کردن روز (نسبت به ماه)، در یک تاریخ مشخص شده:

var d = new Date("July 21, 1983 01:15:00"); d.setUTCDate(15);	Sat Jul 16 1983 01:15:00 GMT+0430 (Iran Daylight Time)
--	--

متد **setUTCFullYear**

مثال (متد **setUTCFullYear**) ست کردن سال به ۱۹۹۲

<pre>var d = new Date(); d.setUTCFullYear(1992);</pre>	Mon Dec 21 1992 20:21:57 GMT+0330 (Iran Standard Time)
--	--

متده است `setUTCFullYear()`، طبق زمان جهانی، سال (چهار رقمی) را برای یک شیء تاریخ معین می کند.

نکته: زمان جهانی (UTC)، زمانی است که توسط استاندارد زمان جهانی تعیین می شود.

نکته: زمان UTC همانند زمان GMT است.

`Date.setUTCFullYear(year,month,day)`

مقادیر پارامترها

پارامتر	توضیحات
<code>year</code>	(ضروری). یک عدد چهار رقمی که سال را مشخص می کند، مقادیر منفی نیز قابل قبول هستند.
<code>month</code>	(اختیاری). یک عدد صحیح که ماه را مشخص می کند. مقادیر قابل قبول اعداد از ۰ تا ۱۱ هستند، اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none">-۱: آخرین ماه از سال قبل را مشخص می کند.۱۲: اولین ماه از سال بعد را مشخص می کند.۱۳: دومین ماه از سال بعد را مشخص می کند.
<code>day</code>	(اختیاری). یک عدد صحیح که روز را (نسبت به ماه) مشخص می کند. مقادیر قابل قبول اعداد از ۱ تا ۳۱ هستند، اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none">۰: آخرین ساعت از ماه قبل را مشخص می کند.-۱: ساعت یکی مانده به آخر، از ماه قبل را مشخص می کند. اگر ماه ۳۱ روزه باشد: <ul style="list-style-type: none">۳۲: اولین روز از ماه بعد را مشخص می کند. اگر ماه ۳۰ روزه باشد: <ul style="list-style-type: none">۳۲: دومین روز از ماه بعد را مشخص می کند.

مثال ۲ ست کردن تاریخ به، ۳ نوامبر ۲۰۲۰:

<code>var d = new Date(); d.setUTCFullYear(2020, 10, 3);</code>	Tue Nov 03 2020 20:21:57 GMT+0330 (Iran Standard Time)
---	--

مثال (متد `setUTCFullYear`) ست کردن تاریخ به ۶ ماه قبل، طبق زمان جهانی (UTC):

<code>var d = new Date(); d.setUTCFullYear(d.getUTCFullYear, d.getUTCMonth() - 6);</code>	Sat Jun 21 2014 21:21:57 GMT+0430 (Iran Daylight Time)
---	--

متد `setUTCHours`

متدها (`setUTCHours`) ، طبق زمان جهانی، ساعت را برای یک شیء تاریخ معین می کند.

از این متدها همچنین می توان در ست کردن دقیقه ها و ثانیه ها و میلی ثانیه ها نیز، استفاده کرد.

نکته: زمان جهانی (UTC)، زمانی است که توسط استاندارد زمان جهانی تعیین می شود.

نکته: زمان UTC همانند زمان GMT است.

`Date.setUTCHours(hour,min,sec,millisec)`

مثال (متد `setUTCHours`) ست کردن ساعت به ۱۵، طبق زمان جهانی:

<code>var d = new Date(); d.setUTCHours(15);</code>	Sun Dec 21 2014 19:21:58 GMT+0330 (Iran Standard Time)
---	--

مقادیر پارامترها

پارامتر	توضیحات
Hour	(ضروری). یک عدد صحیح که ساعت را مشخص می کند. مقادیر قابل قبول اعداد از ۰ تا ۲۳ هستند، اما مقادیر زیر نیز قابل قبول هستند: ۱- آخرین ساعت، از روز گذشته را مشخص می کند. ۲۴- اولین ساعت از روز بعد را مشخص می کند.
Min	(اختیاری). یک عدد صحیح که دقیقه ها را مشخص می کند.

	<p>مقادیر قابل قبول اعداد از ۰ تا ۵۹ هستند اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none"> • ۱: آخرین دقیقه از ساعت قبل را مشخص می کند. • ۶۰: اولین دقیقه از ساعت بعد را مشخص می کند.
Sec	<p>(اختیاری). یک عدد صحیح که ثانیه ها را مشخص می کند.</p> <p>مقادیر قابل قبول اعداد از ۰ تا ۵۹ هستند، اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none"> • ۱: آخرین ثانیه از دقیقه قبل را مشخص می کند. • ۶۰: اولین ثانیه از دقیقه بعد را مشخص می کند.
millisec	<p>(اختیاری). یک عدد صحیح که میلی ثانیه ها را مشخص می کند.</p> <p>مقادیر قابل قبول اعداد از ۰ تا ۹۹۹ هستند اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none"> • ۱: آخرین میلی ثانیه از ثانیه قبل را مشخص می کند. • ۱۰۰۰: اولین میلی ثانیه از ثانیه بعد را مشخص می کند.

مثال (متده است کردن زمان به ۱۵:۳۵:۰۱ طبق زمان جهانی: UTC)

```
var d = new Date();
d.setUTCHours(15, 35, 1);
```

در کد بالا خروجی Sun Dec 21 2014 19:05:01 GMT+0330 (Iran Standard Time)

مثال (متده است کردن زمان به ۴۸ ساعت قبل، طبق زمان جهانی: UTC)

```
var d = new Date();
d.setUTCHours(d.getUTCHours() - 48);
```

خروجی d در کد بالا طبق زمان محلی برابر است با:

Fri Dec 19 2014 20:21:58 GMT+0330 (Iran Standard Time)

setUTCMilliseconds متده

متده setUTCMilliseconds() میلی ثانیه هارا (از ۰ تا ۹۹۹)، طبق زمان جهانی، تعیین می کند.

نکته: زمان جهانی (UTC)، زمانی است که توسط استاندارد زمان جهانی تعیین می شود.

نکته: زمان UTC همانند زمان GMT است.

Date.setUTCMilliseconds(milliseconds)

مثال (متد setUTCMilliseconds) ست کردن میلی ثانیه ها به ۱۹۲، طبق زمان جهانی:

```
var d = new Date();
d.setUTCMilliseconds(192);
var n = d.getUTCMilliseconds();
```

خروجی n در کد بالا: 192

مقادیر پارامترها

پارامتر	توضیحات
millisecond	(ضروری). یک عدد صحیح که میلی ثانیه ها را مشخص می کند. مقادیر قابل قبول اعداد از ۰ تا ۹۹۹ هستند اما مقادیر زیر نیز قابل قبول هستند: ۱- آخرین میلی ثانیه از ثانیه قبل را مشخص می کند. ۱۰۰۰: اولین میلی ثانیه از ثانیه بعد را مشخص می کند. ۱۰۰۱: دومین میلی ثانیه از ثانیه بعد را مشخص می کند.

متد setUTCMinutes

متد () setUTCMinutes() طبق زمان جهانی، دقیقه ها را در یک شیء تاریخ تعیین می کند.

نکته: زمان جهانی (UTC)، زمانی است که توسط استاندارد زمان جهانی تعیین می شود.

نکته: زمان UTC همانند زمان GMT است.

Date.setUTCMinutes(min,sec,millisecond)

مثال (متد setUTCMinutes) ست کردن دقیقه ها به ۱۷، طبق زمان جهانی:

```
var d = new Date();
d.setUTCMinutes(17);
```

مقادیر پارامترها

پارامتر	توضیحات
min	<p>(ضروری). یک عدد صحیح که دقیقه ها را مشخص می کند .</p> <p>مقادیر قابل قبول اعداد از ۰ تا ۵۹ هستند اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none"> • -1: آخرین دقیقه از ساعت قبل را مشخص می کند. • 60: اولین دقیقه از ساعت بعد را مشخص می کند.
sec	<p>(اختیاری). یک عدد صحیح که ثانیه ها را مشخص می کند .</p> <p>مقادیر قابل قبول اعداد از ۰ تا ۵۹ هستند اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none"> • -1: آخرین ثانیه از دقیقه قبل را مشخص می کند. • 60: اولین ثانیه از دقیقه بعد را مشخص می کند.
millisec	<p>(اختیاری). یک عدد صحیح که میلی ثانیه ها را مشخص می کند .</p> <p>مقادیر قابل قبول اعداد از ۰ تا ۹۹۹ هستند اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none"> • -1: آخرین میلی ثانیه از ثانیه قبل را مشخص می کند. • 1000: اولین میلی ثانیه از ثانیه بعد را مشخص می کند.

مثال (متده setUTCMMinutes)

ست کردن زمان به ۹۰ دقیقه قبل، طبق زمان جهانی:

```
var d = new Date();
d.setUTCMMinutes(d.getUTCMMinutes() - 90);
```

خروجی d در کد بالا طبق زمان محلی برابر است با:

Sun Dec 21 2014 18:51:59 GMT+0330 (Iran Standard Time)

setUTCMonth متده

تعریف و کاربرد

متده است (setUTCMonth)، طبق زمان جهانی، ماه را از ۰ تا ۱۱ (در یک شیء تاریخ، تعیین می کند).

نکته: ژانویه هم ارز با ۰ است و فوریه هم ارز با ۱ است و همینطور الی آخر.

از این متده همچنین می توان برای ست کردن روز (نسبت به ماه) استفاده کرد.

نکته: زمان جهانی (UTC)، زمانی است که توسط استاندارد زمان جهانی تعیین می شود.

نکته: زمان UTC همانند زمان GMT است.

نحوه استفاده

Date.setUTCMonth(month,day)

(setUTCMonth) مثال (متده است)

ست کردن ماه به ۴ (ماه مه):

```
var d = new Date();
d.setUTCMonth(4);
```

خروجی d در کد بالا:

Wed May 21 2014 21:22:00 GMT+0430 (Iran Daylight Time)

مقادیر پارامترها

پارامتر	توضیحات
month	(ضروری). یک عدد صحیح که ماه را مشخص می کند. مقادیر قابل قبول اعداد از ۰ تا ۱۱ هستند اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none">-1: آخرین ماه از سال قبل را مشخص می کند.12: اولین ماه از سال بعد را مشخص می کند.13: دومین ماه از سال بعد را مشخص می کند.
day	(اختیاری). یک عدد صحیح که روز را (نسبت به ماه) مشخص می کند. مقادیر قابل قبول اعداد از ۱ تا ۳۱ هستند اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none">0: آخرین ساعت از ماه قبل را مشخص می کند.

	<ul style="list-style-type: none"> • ۱- ساعت یکی مانده به آخر از ماه قبل را مشخص می کند.
	<p>اگر ماه ۳۱ روز داشته باشد:</p> <ul style="list-style-type: none"> • ۲- اولین روز از ماه بعد را مشخص می کند.
	<p>اگر ماه ۳۰ روز داشته باشد:</p> <ul style="list-style-type: none"> • ۳- دومین روز از ماه بعد را مشخص می کند.

جزئیات تکنیکی

مقدار برگشتی	یک عدد، که تعداد میلی ثانیه ها را از تاریخ نیمه شب ۱ ژانویه ۱۹۷۰ تا شیء تاریخ برمی گرداند
نسخه JavaScript	1.3

(مثال (متده است: setUTCMonth

ست کردن ماه به ۴(ماه مه) و همچنین ست کردن روز به ۲۰

```
var d = new Date();
d.setUTCMonth(4, 20);
```

خروجی d در کد بالا طبق زمان محلی برابر است با:

Tue May 20 2014 21:22:00 GMT+0430 (Iran Daylight Time)

(مثال (متده است: setUTCMonth

ست کردن تاریخ به آخرین روز از آخرین ماه:

```
var d = new Date();
d.setUTCMonth(d.getUTCMonth(), 0);
```

خروجی d در کد بالا طبق زمان محلی برابر است با:

Sun Nov 30 2014 20:22:00 GMT+0330 (Iran Standard Time)

متده setUTCSeconds

تعريف و کاربرد

متده است `setUTCSeconds()`، طبق زمان جهانی، ثانیه های یک شیء تاریخ را تعیین می کند.

از این متده همچنین می توان برای ست کردن میلی ثانیه ها نیز استفاده کرد.

نکته: زمان جهانی (UTC)، زمانی است که توسط استاندارد زمان جهانی تعیین می شود.

نکته: زمان UTC همانند زمان GMT است.

نحوه استفاده

`Date.setUTCSeconds(sec,millisec)`

(`setUTCSeconds` متد)

ست کردن ثانیه ها به ۳۵، طبق زمان جهانی:

```
var d = new Date();
d.setUTCSeconds(35);
```

خروجی `d` در کد بالا:

Sun Dec 21 2014 20:22:35 GMT+0330 (Iran Standard Time)

مقادیر پارامترها

پارامتر	توضیحات
Sec	<p>(ضروری). یک عدد صحیح که ثانیه ها را مشخص می کند .</p> <p>مقادیر قابل قبول اعداد از ۰ تا ۵۹ هستند، اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none">۱- آخرین ثانیه از دقیقه قبل را مشخص می کند.۶۰: اولین ثانیه از دقیقه بعد را مشخص می کند.
Millisec	<p>(اختیاری). یک عدد صحیح که میلی ثانیه ها را مشخص می کند .</p> <p>مقادیر قابل قبول اعداد از ۰ تا ۹۹۹ هستند اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none">۱- آخرین میلی ثانیه از ثانیه قبل را مشخص می کند.۱000: اولین میلی ثانیه از ثانیه بعد را مشخص می کند.

جزئیات تکنیکی

مقدار برگشته	یک عدد، که تعداد میلی ثانیه ها را از تاریخ نیمه شب ۱ ژانویه ۱۹۷۰ تا شیء تاریخ برمی گرداند
نسخه	1.3

مثال (setUTCSeconds)

ست کردن ثانیه ها و میلی ثانیه ها طبق زمان جهانی:

```
var d = new Date();
d.setUTCSeconds(35,825);
var n = d.getUTCSeconds() + ":" + d.getUTCMilliseconds();
```

خروجی n در کد بالا : 35:825

متده **toISOString**

مثال (متده **toISOString**) برگرداندن شیء تاریخ مثل یک رشته، طبق استاندارد ISO

```
var d = new Date();
var n = d.toISOString();
```

خروجی n در کد بالا : 2014-12-21T16:52:03.988Z

متده **toISOString()** طبق استاندارد ISO، یک شیء تاریخ را تبدیل به رشته می کند.

نام استاندارد، ISO-8601 است و فرمت آن بصورت YYYY-MM-DDTHH:mm:ss.sssZ است.

Date.toISOString()

متده **toJSON**

متده **toJSON()** یک شیء تاریخ را به رشته تبدیل می کند و فرمت آن را همچون فرمت تاریخ در JSON قرار می دهد.

فرمت تاریخ در JSON، همانند فرمت تاریخ در استاندارد ISO-8601 است : YYYY-MM-DDTHH:mm:ss.sssZ

Date.toJSON()

مثال (متد **toJSON**)

برگرداندن یک شیء تاریخ، بعنوان یک رشته با فرمت تاریخ در JSON:

```
var d = new Date();
var n = d.toJSON();
```

خروجی n در کد بالا

2014-12-21T16:52:04.298Z

متد **toLocaleString**

متد () **toLocaleString**، تمام یک شیء تاریخ را به یک رشته معمولی تبدیل می کند:

نحوه استفاده

Date.toLocaleString()

مثال (متد **toLocaleString**)

تبدیل تمام شیء تاریخ به یک رشته معمولی:

```
var d = new Date();
var n = d.toLocaleString();
```

خروجی n در کد بالا :

متد **toUTCString**

متد () **toUTCString** یک شیء تاریخ را طبق زمان جهانی تبدیل به رشته می کند.

نکته: زمان جهانی (UTC)، زمانی است که توسط استاندارد زمان جهانی تعیین می شود.

نکته: زمان UTC همانند زمان GMT است.

Date.toUTCString()

مثال (متد **toUTCString**) تبدیل شیء تاریخ به یک رشته، طبق زمان جهانی:

```
var d = new Date();
var n = d.toUTCString();
```

متده UTC

متده (UTC)، تعداد میلی ثانیه ها را طبق زمان جهانی، از ۱ ژانویه ۱۹۷۰ تا یک تاریخ مشخص، برمی گرداند.

نکته: زمان جهانی (UTC)، زمانی است که توسط استاندارد زمان جهانی تعیین می شود.

نکته: زمان UTC همانند زمان GMT است.

نحوه استفاده

Date.UTC(year,month,day,hours,minutes,seconds,millisecond)

(مثال) متده UTC

برگرداندن تعداد میلی ثانیه ها از نیمه شب ۱ ژانویه ۱۹۷۰ تا یک تاریخ مشخص:

```
var d = Date.UTC(2012,02,30);
```

خرجی d در کد بالا: 1333065600000

مقادیر پارامترها

پارامتر	توضیحات
year	(ضروری). یک عدد چهار رقمی که سال را مشخص می کند، همچنین اعداد منفی نیز قابل قبول هستند.
month	(ضروری). یک عدد صحیح که ماه را مشخص می کند. مقادیر قابل قبول از ۰ تا ۱۱ هستند، اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none"> - آخرین ماه از سال قبل را مشخص می کند. ۱۲: اولین ماه از سال بعد را مشخص می کند. ۱۳: دومین ماه از سال بعد را مشخص می کند.
day	(ضروری). یک عدد صحیح که روز را (نسبت به ماه) مشخص می کند. مقادیر قابل قبول اعداد از ۱ تا ۳۱ هستند، اما مقادیر زیر نیز قابل قبول هستند: <ul style="list-style-type: none"> ۰: آخرین ساعت از ماه قبل را مشخص می کند.

	<ul style="list-style-type: none"> -1: ساعت یکی مانده به آخر از ماه قبل را مشخص می کند. <p>اگر ماه ۳۱ روز داشته باشد:</p> <ul style="list-style-type: none"> 32: اولین روز از ماه بعد را مشخص می کند. <p>اگر ماه ۳۰ روز داشته باشد:</p> <ul style="list-style-type: none"> 32: دومین روز از ماه بعد را مشخص می کند.
hour	<p>(ضروری). یک عدد صحیح که ساعت را مشخص می کند، همچنین مقدار پیش فرض آن ۰ است.</p> <p>مقادیر قابل قبول اعداد از ۰ تا ۲۳ هستند اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none"> -1: آخرین ساعت از روز قبل را مشخص می کند. 24: اولین ساعت از روز بعد را مشخص می کند.
min	<p>اختیاری. یک عدد صحیح که دقیقه ها را مشخص می کند و مقدار پیش فرض آن ۰ است.</p> <p>مقادیر قابل قبول اعداد از ۰ تا ۵۹ هستند اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none"> -1: آخرین دقیقه از ساعت قبل را مشخص می کند. 60: اولین دقیقه از ساعت بعد را مشخص می کند.
sec	<p>اختیاری. یک عدد صحیح که ثانیه ها را مشخص می کند و مقدار پیش فرض آن ۰ است.</p> <p>مقادیر قابل قبول اعداد از ۰ تا ۵۹ هستند اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none"> -1: آخرین ثانیه از دقیقه قبل را مشخص می کند. 60: اولین ثانیه از دقیقه بعد را مشخص می کند.
millisec	<p>اختیاری. یک عدد صحیح که میلی ثانیه ها را مشخص می کند و مقدار پیش فرض آن ۰ است.</p> <p>مقادیر قابل قبول اعداد از ۰ تا ۹۹۹ هستند اما مقادیر زیر نیز قابل قبول هستند:</p> <ul style="list-style-type: none"> -1: آخرین میلی ثانیه از ثانیه قبل را مشخص می کند. 1000: اولین میلی ثانیه از ثانیه بعد را مشخص می کند.

مثال (متد UTC) ساختن یک شیء تاریخ با استفاده از زمان UTC به جای استفاده از زمان محلی:

```
var d = new Date(Date.UTC(2012,02,30));
```

خروجی D در کد بالا: Fri Mar 30 2012 04:30:00 GMT+0430 (Iran Daylight Time)

متدهای **valueOf**

متدهای **valueOf**، مقدار اولیه‌ی یک شیء رشته را برمی‌گرداند.

نکته: این متدهای طور اتوماتیک، به هنگام نیاز در جاوا اسکریپت فراخوانده می‌شود.

```
string.valueOf()
```

مثال (متدهای **valueOf**) برگرداندن مقدار اولیه‌ی یک شیء رشته:

```
var str = "Hello World!";
var res = str.valueOf();
```

خروجی **res** در کد بالا: Hello World!

BOM

متدهای دیگر شیء **window**

- **window.open()**: یک پنجره جدید باز می‌کند.
- **window.close()**: پنجره جاری را می‌بندد.
- **window.moveTo()**: پنجره جاری را حرکت می‌دهد.
- **window.resizeTo()**: پنجره جاری را تغییر اندازه می‌دهد.

window.open()

```
<body>

    <p>Click the button to open a new browser window.</p>

    <button onclick="myFunction()">Try it</button>

<script>

    function myFunction() {

        window.open("https://www.w3schools.com");

    }

</script>

</body>
```

window.close()

```
<body>

    <button onclick="openWin()">Open "myWindow"</button>

    <button onclick="closeWin()">Close "myWindow"</button>

<script>

    var myWindow;

    function openWin() {

        myWindow = window.open("", "myWindow", "width=200,height=100");

        myWindow.document.write("<p>This is 'myWindow'</p>");

    }

    function closeWin() {

        myWindow.close();

    }

</script>

</body>
```

window.moveTo()

```
<body>

<p>Open "myWindow" and move the new window to the top left corner of the screen:</p>

<button onclick="openWin()">Open "myWindow"</button>

<button onclick="moveWin()">Move "myWindow"</button>

<script>

    var myWindow;

    function openWin() {

        myWindow=window.open("", "myWindow", "width=200, height=100");

        myWindow.document.write("<p>This is 'myWindow'</p>");

    }

    function moveWin() {

        myWindow.moveTo(500, 100);

        myWindow.focus();

    }

</script>

</body>
```

: window.resizeTo()

```
<body>

<p>Open a new window, and resize the width and height to 500px:</p>

<button onclick="openWin()">Create window</button>

<button onclick="resizeWin()">Resize window</button>

<script>

    var myWindow;

    function openWin() {

        myWindow = window.open("", "", "width=100, height=100");

    }

    function resizeWin() {

        myWindow.resizeTo(250, 250);

        myWindow.focus();

    }

</script>

</body>
```

شیء JavaScript در history

شیء `window.history`، شامل تاریخچه مرورگر است.

شیء window.history

شیء `window.history` را می توان بدون پیشوند `window` نیز نوشت.

با خاطر حفظ حریم کاربران، محدودیت هایی برای چگونگی دسترسی JavaScript به این شیء وجود دارد.

بعضی از متدهای شیء `history`:

• `history.back()` : همانند کلید back در مرورگر عمل می کند.

• همانند کلید forward : در مرورگر عمل می کند.

history.back() متده

متده history.back() ، با توجه به تاریخچه صفحات بازدید شده، مرورگر را به صفحه قبلی هدایت می کند.

این متده، مانند کلید back در مرورگر عمل می کند.

ایجاد کلید back در یک صفحه

```
<html>
<head>
<script>
function goBack()
{
    window.history.back()
}
</script>
</head>
<body>
<input type="button" value="Back" onclick="goBack()">

</body>
</html>
```

history.forward() متده

متده history.forward() ، با توجه به تاریخچه صفحات بازدید شده، مرورگر را به صفحه بعدی هدایت می کند.

این متده، مانند کلید Forward در مرورگر عمل می کند.

ایجاد کلید forward در یک صفحه

```
<head>
<script>
function goForward()
{
    window.history.forward()
}
</script>
</head>
<body>
    <input type="button" value="Forward" onclick="goForward()">
</body>
```

شیء JavaScript در navigator

شیء window.navigator شامل اطلاعاتی درباره مرورگر بازدیدکننده است.

window.navigator

شیء window.navigator را می توان بدون پیشوند window نیز نوشت. مثال:

```
<div id="example"></div>
<script>
    txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";
    txt+= "<p>Browser Name: " + navigator.appName + "</p>";
    txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";
    txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";
    txt+= "<p>Platform: " + navigator.platform + "</p>";
    txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";
    txt+= "<p>User-agent language: " + navigator.systemLanguage + "</p>";
    document.getElementById("example").innerHTML=txt;
</script>
```

اخطار!!!

اطلاعاتی که از شیء `navigator` دریافت می شود، اغلب گمراه کننده است و نباید برای یافتن نسخه مرورگر استفاده شود.

دلیل:

- داده های `navigator` می توانند توسط مالک مرورگر تغییر داده شود.
- بعضی مرورگرها برای دور زدن تست سایت، در تشخیص هویت خودشان خلل ایجاد می کنند.
- مرورگرهایی که قبل از یک سیستم عامل جدید آمده باشند، نمی توانند اطلاعات آن سیستم عامل را گزارش دهند.

تشخیص نوع مرورگر

به خاطر اینکه اطلاعات شیء `navigator` در مورد نوع مرورگر گمراه کننده است، می توانید از یک سری اشیاء مختلف برای پیدا کردن نوع مرورگر استفاده نمایید.

چونکه مرورگرهای مختلف، اشیاء مختلفی را پشتیبانی می کنند، می توانید از این نوع اشیاء برای پیدا کردن نوع مرورگر استفاده نمایید. برای مثال، چونکه فقط مرورگر Opera خصوصیت "window.opera" را پشتیبانی می کند، می توانید از آن برای مشخص کردن مرورگر Opera استفاده نمایید:

Example: if (window.opera) {...some action...}