آموزش JS-مقدمه جاوا اسکریپت

JavaScript (جاوا اسکریبت) یک زبان اسکریبت نویسی تحت وب است.

JavaScript (جاوا اسکریبت) در میلیون ها صفحه وب برای اضافه کردن توابع، اعتبار سنجی فرم ها، ارتباط برقرار کردن با سرور و ... استفاده شده است.

یادگیری **JavaScript** (جاوا اسکریبت) آسان است و از آموزش آن لذت خواهید برد.

آموزش با استفاده از مثال های "خودتان امتحان کنید"

در هر فصل، مثال های "خودتان امتحان کنید" آورده شده است.

این آموزش شامل صدها مثال "خودتان امتحان کنید" است.

می توانید با ویرایشگری که در اختیارتان است، کد **JavaScript** (جاوا اسکریبت) را ویرایش کنید و با کلیک روی دکمه "ویرایش کردن" نتیجه آنرا مشاهده کنید.

قبل از شروع آموزش چه چیزی را باید بدانید

پیشنهاد می شود اگر با بیاموزهای زیر آشنا نیستید، قبل از ادامه آنها را مطالعه نمایید:

- آموزش **HTML**
- آموزش **CSS**

JavaScript (جاوا اسکریبت) یک زبان اسکریبت نویسی است

- **JavaScript** (جاوا اسکریبت) محبوب ترین زبان اسکریبت نویسی تحت وب است و با عمده مرورگرها مانند **Firefox, Chrome, Opera ,Internet Explorer** و **Safari** کار می کند.
- **JavaScript** (جاوا اسکریبت) برای اضافه کردن تعامل بیشتر در صفحات **HTML**، بوجود آمده است.
- **JavaScript** (جاوا اسکریبت) معمولاً در بین کدهای **HTML** قرار می گیرد. (**embedded**)

- JavaScript (جاوا اسکریبت) یک زبان تفسیر شده است. (یعنی اینکه اسکریبت ها بدون مقدمات کامپایل، اجرا می شوند)
- هر کسی می تواند از JavaScript (جاوا اسکریبت) استفاده کند، بدون اینکه نیاز به خرید مجوز آن باشد.

آموزش JavaScript (جاوا اسکریبت): آموزش ایجاد خروجی HTML

```
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph</p>");
```

توجه: در کدهای JavaScript (جاوا اسکریبت) واقعی، از دستور document.write() استفاده نکنید. چون تمام صفحه دوباره بارگذاری می شود. با این وجود در اینجا دستور document.write() یک راه ساده برای ایجاد خروجی HTML در JavaScript (جاوا اسکریبت) را نشان می دهد.

آموزش JavaScript (جاوا اسکریبت): آموزش واکنش در برابر رویدادها

کدهای JavaScript، زمانی اجرا می شوند که صفحه بارگذاری شود (onload) اما این چیزی نیست که همیشه نیاز داریم.

بعضی مواقع می خواهیم زمانی که یک رویداد خاص اتفاق می افتد، کدمان اجرا شود، مانند زمانی که کاربر روی یک دکمه کلیک می کند.

```
<button type="button" onclick="alert('Welcome!')">Click Me!</button>
```

از تابع alert، معمولاً اگر بخواهید، کاربر را از اطلاعات ارسال شده به Server مطمئن کنید، استفاده کنید. (پیغام: عملیات با موفقیت انجام شد)

رویداد onclick، تنها یکی از رویدادهای HTML است و در آموزش های بعدی با آنها بیشتر آشنا خواهید شد.

آموزش JavaScript (جاوا اسکریبت): آموزش تغییر محتوای عناصر HTML

یکی از بهترین قابلیت های JavaScript (جاوا اسکریبت)، دستکاری محتوای عناصر HTML است. (JavaScript (جاوا اسکریبت) می تواند محتویات عناصر HTML را بخواند و آنها را تغییر دهد)

```
x=document.getElementById("demo") //Find the element
x.innerHTML="Hello JavaScript"; //Change the content
```

شما دستور ()document.getElementByID را در آینده بسیار خواهید دید. در قسمت HTML DOM تعریف شده است.

DOM یا (Document Object Model) یک استاندارد رسمی در W3C است و برای دسترسی به عناصر HTML است. (در آینده بیشتر توضیح داده خواهد شد)

آموزش JavaScript (جاوا اسکریبت): آموزش تغییر تصاویر HTML

این مثال، بصورت داینامیک، خصوصیت (src) عنصر <img> را تغییر می دهد:

```
element=document.getElementById('myimage')
if (element.src.match("bulbon"))
{
element.src="/pic_bulboff.gif";
}
else
{
```

```
element.src="/pic_bulbon.gif";
}
}
```

JavaScript (جاوا اسکریبت) بسیاری از خصوصیت های عناصر HTML را می تواند تغییر دهد.

آموزش JavaScript (جاوا اسکریبت): آموزش تغییر ظاهر (Style) عناصر HTML

تغییر Style یک عنصر نسبت به تغییر خصوصیت کمی متفاوت است.

```
x=document.getElementById("demo") //Find the element
x.style.color="#ff0000"; //Change the style
```

آموزش JavaScript (جاوا اسکریبت): آموزش اعتبار سنجی فرم (Validate)

JavaScript (جاوا اسکریبت) می تواند برای معتبر ساختن اطلاعات فرم، قبل از اینکه به سرور ارسال شود استفاده شود. چنین فرم هایی جلوی پردازش های اضافی توسط سرور را می گیرند.

```
if isNaN(x) {alert("Not Numeric")};
```

آیا JavaScript (جاوا اسکریبت) و Java یکی اند؟

نه!

JavaScript (جاوا اسکریبت) و java از نظر مفهوم و طراحی، دو زبان کاملا متفاوت هستند.

java (که توسط شرکت sun توسعه یافته است) قوی و بسیار پیچیده تر از JavaScript (جاوا اسکریبت) است و در دسته بندی زبان هایی مانند C و C++ قرار می گیرد.

JavaScript = ECMAScript

JavaScript (جاوا اسکریبت) یک پیاده سازی از زبان استاندارد ECMAScript است.
ECMA-262 استاندارد رسمی JavaScript (جاوا اسکریبت) است.

این زبان توسط "Brendan Eich" در Nets cape (با Navigator2.0) ابداع شده است و تقریباً در تمام مرورگرها از سال 1996 ظاهر شده است.

توسعه ی استاندارد ECMA_262 از سال 1996 و اولین ویرایش آن در 1997 توسط مجمع عمومی ECMA پذیرفته شد.

استاندارد ECMA به عنوان یک استاندارد ISO جهانی (ISO/IEC 16262) در سال 1998 تصویب شده است.

توسعه استاندارد هنوز هم در حال پیشرفت است.

محل قرارگیری کدهای JavaScript

کدهای JavaScript در صفحه HTML باید بین تگهای <script> و </script> قرار داده شود.

کدهای JavaScript را می توان در بدنه صفحه HTML (بین تگهای باز و بسته <body> و </body>) و یا در قسمت <head> صفحه قرار داد.

تگ <script>

از تگ <script>، برای وارد کردن کدهای JavaScript در صفحه HTML استفاده می شود.

تگهای باز و بسته <script> و </script> می گوید، کدهای JavaScript کجا شروع و کجا پایان یافته است.

خطوط بین <script> و </script> شامل کدهای JavaScript است:

```
<script>
alert("My First JavaScript");
</script>
```

مرورگر، کد JavaScript بین <script> و </script> را ترجمه و اجرا می کند.

توجه: بدون تگ های باز و بسته <script> و </script> ، مرورگر با دستورات بین آنها مانند یک متن معمولی برخورد می کند و آنها را عیناً در صفحه HTML نمایش می دهد.

توجه: ممکن است برای تگ <script> خصوصیت type را تنظیم کنیم ("text/javascript"). نیاز به انجام این کار نیست، چون در تمام مرورگرها و در HTML5 زبان اسکریپت پیشفرض، JavaScript است.

کدهای JavaScript در بدنه صفحه HTML

در مثال زیر، هنگام بارگذاری صفحه یا (onload)، خروجی HTML در قسمت <body> چاپ می شود:

```
<!DOCTYPE html>
<html>
<body>
.
.
<script>
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph</p>");
</script>
.
.
</body>
</html>
```

همچنین در مثال زیر، تاریخ جاری سیستم، هنگام بارگذاری صفحه، داخل عنصر با شناسه "demo" نوشته می شود:

```
<html>
<body>
<h1>My First Web Page</h1>
<p id="demo"></p>
<script type="text/javascript">
document.getElementById("demo").innerHTML=Date();
</script>
</body>
</html>
```

توجه: برای اطمینان بیشتر، کد JavaScript در پایین صفحه HTML قرار داده شده است (نباید قبل از ایجاد عنصر <p> کدمان اجرا شود)

توابع و رویدادها در JavaScript

کدهای JavaScript در صفحه HTML، زمانی که صفحه بارگذاری شود (onload) اجرا خواهند شد. اما این چیزی نیست که ما همیشه نیاز داریم.

بعضی مواقع می خواهیم زمانی که یک رویداد خاص اتفاق می افتد، کدمان اجرا شود، مانند زمانی که کاربر روی یک دکمه کلیک می کند. در این مورد می توانیم کدمان را داخل یک تابع قرار دهیم.

رویدادها در حالت عادی با توابع ترکیب می شوند، مانند صدا زدن یک تابع هنگامی که رویداد رخ می دهد.

در آموزش های بعدی درباره توابع و رویدادهای **JavaScript** بیشتر خواهید آموخت.

اسکریپت ها در دو قسمت <body> و <head>

می توان به صورت نامحدود، هر تعداد خط اسکریپت را در صفحه **HTML** قرار داد، و همچنین می توانید اسکریپت ها را همزمان در دو قسمت <**body**> و <**head**> بیاورید.

در حالت کلی، تمام توابع را در قسمت <**head**> صفحه و یا در پایین صفحه قرار می گیرد. این روش باعث می شود که تداخلی با محتویات صفحه پیش نیاید.

کدهای **JavaScript** در قسمت <head>

در مثال زیر، زمانی که روی دکمه کلیک شود یا به عبارتی رویداد **onclick** عنصر مورد نظر رخ دهد، تابع **displayDate**() صدا زده می شود:

```
<html>
<head>
<script type="text/javascript">
function displayDate()
{
document.getElementById("demo").innerHTML=Date();
}
</script>
</head>
<body>
<h1>My First Web Page</h1>
<p id="demo"></p>
<button type="button" onclick="displayDate()">Display
Date</button>
</body>
</html>
```

همچنین در مثال ساده زیر، هنگامی که روی دکمه "Try it" کلیک شود، محتوای عنصر با شناسه "demo" تغییر می کند:

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
document.getElementById("demo").innerHTML="My First
JavaScript Function";
}
</script>
</head>
<body>
<h1>My Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
</body>
</html>
```

یک تابع JavaScript در قسمت <body>

در این مثال، تابع "myFunction" در انتهای <body> قرار داده شده است و وقتی کاربر روی دکمه "Try it" کلیک کند، myFunction صدا زده می شود:

```
<!DOCTYPE html>
<html>
<body>
<h1>My Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
document.getElementById("demo").innerHTML="My First
JavaScript Function";
}
</script>
</body>
</html>
```

کدهای JavaScript را همچنین می توان در یک فایل خارجی قرار داد.

فایل های خارجی JavaScript اغلب شامل کدهایی است که در چندین صفحه HTML مختلف استفاده می شود.

این فایل ها را باید با فرمت ".js" ذخیره نمود.

توجه: اسکریپت خارجی نمی تواند شامل تگ های باز و بسته <script> و </script> باشد.

برای استفاده از یک فایل اسکریپت خارجی در صفحه HTML، باید خصوصیت "src" تگ <script> را با آدرس فایل، تنظیم کنید:

```
<!DOCTYPE html>
<html>
<body>
<script src="/myScript.js"></script>
</body>
</html>
```

JavaScript بطور معمول برای دستکاری عناصر HTML استفاده می شود.

تغییر محتوای عناصر HTML

در JavaScript برای دسترسی به یک عنصر HTML از طریق ID می توان از دستور document.getElementById() استفاده کرد.

این متد بوسیله شناسه یا (ID) یک عنصر، به آن دسترسی پیدا می کند.

در مثال زیر، محتوای عنصر با شناسه "demo" به مقدار جدید، تغییر می کند:

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p id="demo">My First Paragraph</p>
<script>
document.getElementById("demo").innerHTML="My First
JavaScript";
```

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

همچنین در مثال زیر، تاریخ جاری سیستم، داخل عنصری با شناسه "demo" نوشته می شود (عنصر <p>):

```
<html>
<body>
<h1>My First Web Page</h1>
<p id="demo"></p>
<script type="text/javascript">
document.getElementById("demo").innerHTML=Date();
</script>
</body>
</html>
```

دستور document.write()

در مثال زیر، یک عنصر <p> با محتوی "My First JavaScript" در ادامه فایل HTML نوشته می شود:

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<script>
document.write("<p>My First JavaScript</p>");
</script>
</body>
</html>
```

توجه: در کدهای JavaScript واقعی، از دستور document.write() استفاده نکنید. چون تمام صفحه دوباره نوشته می شود. با این وجود در اینجا دستور document.write() یک راه ساده برای ایجاد خروجی در JavaScript را نشان می دهد. (به مثال زیر توجه نمایید)

```
<!DOCTYPE html>
```

```
<html>
<body>
<h1>My First Web Page</h1>
<p>My First Paragraph.</p>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction()
{
document.write("Oops! The document disappeared!");
}
</script>
</body>
</html>
```

JavaScript در Windows 8

Microsoft در Windows 8 برای ایجاد برنامه های تحت ویندوز، JavaScript را پشتیبانی می کند.

در آینده بطور قطع، JavaScript هم در اینترنت و هم در ویندوز استفاده خواهد شد.

JavaScript، ترتیبی از دستورات است که توسط مرورگر اجرا می شود.

حساس به حروف کوچک و بزرگ

برخلاف HTML کدهای JavaScript به حروف کوچک و بزرگ حساس است (case sensitive)، بنابراین هنگامی که کد می نویسید به حروف بزرگ به دقت نگاه کنید.

دستورات JavaScript

یک دستور JavaScript درواقع یک فرمان به مرورگر است. هدف این دستورات این است که به مرورگر بگویید که چه کاری باید انجام دهد.

در مثال زیر، دستور JavaScript به مرورگر می گوید که رشته "Hello world" را بر روی صفحه وب بنویسد:

```
document.write("Hello world");
```

به طور معمول در پایان هر دستور قابل اجرا، یک سمیکالن (;) اضافه می کنیم.

براساس استانداردهای JavaScript قرار دادن سمیکالن، اختیاری است و مرورگر فرض می کند که آخر خط، انتهای دستور است. به همین خاطر در اغلب مثال ها سمیکالن را در آخر خط نمی بینید.

نکته: استفاده از سمیکالن، این امکان را به شما می دهد که چندین دستور را در یک خط بنویسید.

کدهای JavaScript

کدهای JavaScript (یا فقط JavaScript) یک ترتیبی از دستورات جاوا اسکریپت است.

هر دستور جاوا اسکریپت به ترتیبی که نوشته شده است، توسط مرورگر اجرا می شود.

این مثال، دو عنصر با شناسه "demo" و "myDIV" را دستکاری می کند:

```
document.getElementById("demo").innerHTML="Hello Dolly";
document.getElementById("myDIV").innerHTML="How are you?";
```

بلوکی از دستورات

دستورات JavaScript می توانند با هم در یک بلوک باشند.

بلوک ها با "{" شروع وبا "}" خاتمه می یابند.

هدف از این بلوکها این است که یک ترتیبی از دستوراتی که با هم اجرا می شوند ایجاد شود.

یک مثال خوب برای گروه بندی دستورات در یک بلوک، توابع هستند.

این مثال، تابعی را برای دستکاری دو عنصر، صدا می زند:

```
function myFunction()
{
document.getElementById("demo").innerHTML="Hello Dolly";
document.getElementById("myDIV").innerHTML="How are you?";
}
```

در فصل های بعدی با توابع بیشتر آشنا خواهید شد.

فضاهای خالی

JavaScript، فاصله های اضافی را درنظر نمی گیرد. برای خواناتر شدن کد می توانید فضاهای خالی را اضافه نمایید. دو خط کد زیر یکسان هستند:

```
var person="Hege";
var person = "Hege";
```

شکستن یک خط کد

با استفاده از "\" می توانید یک خط کد را بشکنید. مثال زیر بدرستی نمایش داده خواهد شد:

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

یک خط کد را مانند زیر نمی توانید بشکنید:

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

توضیحات در JavaScript

توضیحات را می توان برای شرح چگونگی انجام یک بلوک از دستورات به کد JavaScript اضافه کرد، در این صورت کد خواناتری خواهیم داشت.

توضیحات یک خطی با "//" شروع می شود.

در مثال زیر، از توضیحات یک خطی، برای شرح کدها استفاده شده است:

```
<script type="text/javascript">
// چاپ یک عنوان
document.write("<h1>This is a heading</h1>");
// چاپ دو پاراگراف
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

توضیحات چند خطی در JavaScript

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

در مثال زیر، از توضیحات چند خطی استفاده شده است:

```
<script type="text/javascript">
/*
کدهای زیر یک عنوان و
دو پاراگراف را چاپ خواهد کرد
*/
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

استفاده از توضیحات برای جلوگیری از اجرا

در مثال زیر، یک خط کد، به توضیحات تبدیل شده و از اجرای آن جلوگیری می کنیم (این راه می تواند برای کشف خطا مناسب باشد):

```
<script type="text/javascript">
//document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

در مثال زیر، بلوکی از کدها به توضیحات تبدیل شده است و مانند مثال قبلی از اجرای آنها جلوگیری خواهد شد (این راه می تواند برای کشف خطا مناسب باشد):

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

قرار دادن توضیحات در انتهای خطوط

در مثال زیر، توضیحات در پایان یک خط ، قرار داده شده است:

```
<script type="text/javascript">
document.write("Hello"); // Write "Hello"
document.write(" Dolly!"); // Write " Dolly!"
</script>
```

آموزش JS-متغییرها

متغییرها، ظرفی برای ذخیره اطلاعات اند:

```
var x=5;
var y=6;
var z=x+y;
```

آیا جبر را از زمان مدرسه به یاد دارید؟

x=5, y=6, z=x+y

آیا به یاد می آورید که یک حرف (مانند x) می تواند برای نگهداری یک مقدار (مثل 5) به کار رود و شما می توانید طبق اطلاعات بالا مقدار z را برابر 11 ارزیابی کنید.

این حروف، متغییر نامیده می شوند و آنها را می توان برای نگهداری مقادیر (x=5) و یا عبارات (z=x+y) استفاده کرد.

متغییرهای JavaScript

مانند جبر، متغییرهای JavaScript نیر برای نگهداری مقادیر و یا عبارات به کار می روند.

متغییرها می توانند اسمی کوتاه داشته باشند مانند X و یا کمی توصیفی باشند مانند Carname

قواعد نامگذاری متغییرهای JavaScript:

نام متغییرها به حروف کوچک و بزرگ حساس (case sensitive) هستند. (y و Y دو متغیر متفاوت اند)

نام متغییر می تواند با حروف و یا آندرلاین و یا $ شروع شود.

توجه: چون جاوا اسکریپت Case-sensitive است، نام متغییرهای آن نیز Case-sensitive است.

انواع داده در JavaScript

در متغییرهای JavaScript می توان، انواع دیگر داده، مانند: متن را نیز ذخیره نمود. ("علی احمدی"=person)

در JavaScript یک متن مانند "علی احمدی" یک رشته نامیده می شود.

در JavaScript انواع مختلف متغییر وجود دارد، اما در حال حاضر، فقط به متغییرهای عددی و رشته ای می پردازیم.

زمانی که یک مقدار رشته ای را به یک متغییر انتساب می دهید، باید آنرا در کوتیشن یا دابل کوتیشن (' یا ") قرار دهید.

زمانی که یک عدد را به یک متغییر انتساب می دهید، نیازی به کوتیشن نیست. اگر آنرا در کوتیشن قرار دهید با آن متغیر، مانند یک رشته برخورد خواهد شد.

```
var pi=3.14;
var person="John Doe";
var answer='Yes I am!';
```

مثال

مقدار یک متغییر در طول اجرای اسکریپت می تواند تغییر کند. شما می توانید به یک متغییر از طریق نام آن برای تغییر یا نمایش مقدارش دستیابی پیدا کنید.

مثال زیر به شما چگونگی دسترسی و مقداردهی یک متغییر را نشان خواهد داد.

```
<html>
<body>
<script type="text/javascript">
var firstname;
firstname="Hege";
document.write(firstname);
document.write("<br />");
firstname="Tove";
document.write(firstname);
</script>
<p>The script above declares a variable,
assigns a value to it, displays the value, changes the value,
and displays the value again.</p>
</body>
</html>
```

خروجی کد بالا :

```
Hege
Tove
The script above declares a variable, assigns a value to it, displays the value,
changes the value, and displays the value again.
```

اعلان (ایجاد) متغییرهای JavaScript

در JavaScript ایجاد متغییر بیشتر اوقات به اعلان متغییر معروف است.

شما می توانید متغیبرهای JavaScript را با کلمه کلیدی var اعلان کنید:

```
var x;
```

```
var carname;
```

بعد از تعریف به صورت بالا متغییرها خالی هستند(هنوز هیچ اطلاعاتی داخل آنها نیست)، اگر چه شما می توانید آنها را موقع تعریف مقداردهی نمایید.

```
var x=5;
var carname="Volvo";
```

بعد از اجرای دستورات بالا، متغییر **x** مقدار **5** و **carname** مقدار **"volvo"** را در خود نگه می دارد.

توجه: اگر یک متغییر **JavaScript** را بعد از مقدار دهی، دوباره اعلان کنید، مقدار اولیه اش را از دست نخواهد داد.

تعریف چندین متغییر، در یک دستور

می توانید چندین متغییر را در یک دستور اعلان نمایید. فقط دستور را با **var** شروع کنید و متغییرها را با کاما از هم جدا نمایید:

```
var lastname="Doe", age=30, job="carpenter";
```

اعلان ها، می توانند در چندین خط باشند:

```
var lastname="Doe",
age=30,
job="carpenter";
```

Value = undefined

در برنامه های کامپیوتری، اغلب متغییرها بدون مقدار اعلان می شوند این متغییرها مقدار **undefined** را خواهند داشت.

بعد از اجرای دستور زیر، متغییر **carname** مقدار **undefined** را خواهد داشت:

```
var carname;
```

متغییر های محلی (LOCAL) در JavaScript

متغییرهایی که داخل یک تابع اعلان شده اند، تنها داخل همان تابع، قابل دسترسی اند. (متغییرهای با قلمرو محلی)

شما می توانید در توابع مختلف، متغییرهای محلی با نام های یکسان داشته باشید. (متغییرهای محلی تنها بوسیله تابعی که آنها را اعلان کرده است شناخته می شوند)

متغییرهای محلی بمحض اینکه عملیات تابع تکمیل شد، حذف می شوند.

در فصل های بعدی، درباره توابع بیشتر خواهید آموخت.

متغییر های عمومی (Global) در JavaScript

متغییرهایی که خارج از توابع اعلان شده اند، عمومی می شوند، و در تمام اسکریبت و توابع داخل یک صفحه به آنها دسترسی خواهد بود.

زمانی که یک صفحه وب را ببندید، متغییرهای عمومی حذف خواهند شد.

مقداردهی متغییرهای اعلان نشده

اگر به متغییری که هنوز اعلان نشده مقداری را اختصاص دهید، آن متغییر به طور خودکار اعلان خواهد شد.

مثال:

```
x=5;
carname="Volvo";
```

اگر متغییرهای x و carname پیش از این وجود نداشته باشند، به طور خودکار، به عنوان متغییرهای عمومی اعلان خواهند شد(البته اگر دستورات بالا خارج از توابع باشد)

محاسبات در JavaScript

همانند جبر ، شما می توانید عملگرهای محاسباتی را با متغیرهای JavaScript به کار ببرید.

```
y=5;
```

```
x=y+2;
```

آموزش JS–انواع داده

```
String, Number, Boolean, Array, Object, Null, Undefined
```

متغییرهای رشته ای (String) در JavaScript

در متغییرهای رشته ای، یک سری از کاراکترها مانند "John Doe" ذخیره می شود.

زمانی که یک مقدار رشته ای را به یک متغییر انتساب می دهید، باید آنرا در کوتیشن یا دابل کوتیشن (' یا ") قرار دهید.

```
var carname="Volvo XC60";
var carname='Volvo XC60';
```

می توانید از علامت کوتیشن (') داخل رشته استفاده کنید:

```
var answer="It's alright";
var answer="He is called 'Johnny'";
var answer='He is called "Johnny"';
```

با متغییرهای رشته ای در قسمت پیشرفته بیشتر آشنا خواهید شد.

متغییرهای عددی (Number) در JavaScript

متغییرهای عددی می توانند اعشاری یا صحیح باشند:

```
var x1=34.00;     //Written with decimals
var x2=34;       //Written without decimals
```

اعدادی که بسیار بزرگ یا بسیار کوچک هستند را می توان بصورت "نماد علمی" نوشت:

```
var y=123e5;     // 12300000
var z=123e-5;    // 0.00123
```

متغییر Boolean در JavaScript

متغییرهای Boolean تنها دو مقدار می توانند داشته باشند: True یا False

```
var x=true;
var y=false;
```

متغییر Boolean اغلب برای تست یک شرط استفاده می شود.

آرایه ها (Array) در JavaScript

در JavaScript، آرایه را به یکی از روش های زیر می توان اعلان کرد:

```
var cars=new Array();
cars[0]="Saab";
cars[1]="Volvo";
cars[2]="BMW";
```

یا

```
var cars=new Array("Saab","Volvo","BMW");
```

یا

```
var cars=["Saab","Volvo","BMW"];
```

توجه: ایندکس آرایه از صفر شروع می شود، یعنی اولین آیتم [0] است، دومین آیتم [1] و ...

اشیاء (Object) در JavaScript

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

```
var person={firstname:"John", lastname:"Doe", id:5566};
```

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

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

```
var person={
firstname : "John",
lastname  : "Doe",
id       :  5566
};
```

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

```
name=person.lastname;

name=person["lastname"];
```

با اشیا در قسمت پیشرفته، بیشتر آشنا خواهید شد.

Undefined و Null

در برنامه های کامپیوتری، اغلب متغییرها بدون مقدار اعلان می شوند این متغییرها مقدار undefined را خواهند داشت.

متغییرها را می توان با null مقداردهی کرد:

```
cars=null;
person=null;
```

یک متغییر یکسان را می تواند برای انواع مختلف داده استفاده نمود:

```
var x;               // Now x is undefined
var x = 5;           // Now x is a Number
var x = "John";      // Now x is a String
```

اعلان نوع متغییر

زمانی که متغییر 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 متغییرها در JavaScript شیء اند، زمانی که متغییری را اعلان می کنید، در واقع یک شیء ایجاد کرده اید.

عملگرها در JavaScript

عملگر انتساب (=) برای اختصاص یک مقدار به یک متغیر به کار می رود.

عملگر محاسباتی (+) برای جمع کردن مقادیر با یکدیگر به کار می رود.

```
y=5;
z=2;
x=y+z;
```

بعد از اجرای دستورات بالا مقدار x برابر 7 است.

عملگرهای محاسباتی JavaScript

عملگرهای ریاضی برای اجرای محاسبات بین متغیرها و یا مقادیر به کار می رود.

جدول زیر، عملگرهای محاسباتی JavaScript را توضیح می دهد (با فرض y=5):

عملگر (Operator)	توضیحات	مثال	نتیجه	
+	جمع	x=y+2	x=7	y=5
–	تفریق	x=y-2	x=3	y=5
*	ضرب	x=y*2	x=10	y=5
/	تقسیم	x=y/2	x=2.5	y=5
%	باقیمانده تقسیم	x=y%2	x=1	y=5
++	افزایش یک واحد	x=++y	x=6	y=6

y=6	x=5	++x=y		
y=4 y=4	x=4 x=5	x=--y --x=y	کاهش یک واحد	--

عملگرهای انتسابی JavaScript

عملگرهای انتسابی، مقادیر را به متغیرها اختصاص می دهند.

جدول زیر، عملگرهای انتسابی JavaScript را توضیح می دهد (با فرض y=5 و x=10):

عملگر	مثال	یکسان با مثال	نتیجه
=	x=y		X=5
+=	x+=y	X= X+Y	X=15
-=	x-=y	X= X-Y	X=5
*=	x*=y	X= X*Y	X=50
/=	x/=y	X= X/Y	X=2
%=	x%=y	X= X%Y	X=0

عملگر (+) در رشته ها

عملگر (+) علاوه بر استفاده در محاسبات ریاضی برای جمع دو متغیر رشته ای و یا مقادیر متنی نیز بکار می رود.

مثال:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

بعد از اجرای کد بالا، متغییر txt3 برابر "what a verynice day" می شود.

برای اضافه کردن فضای خالی بین دو متن، در انتها یا ابتدای یکی از آنها فضای خالی قرار می دهیم:

```
txt1="What a very ";
txt2="nice day";
```

```
txt3=txt1+txt2;
```

یا بین دو متغییر یک فضای خالی قرار می دهیم:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

بعد از اجرای کد بالا، متغییر txt3 برابر "what a very nice day" می شود.

جمع رشته ها و اعداد

یک قانون وجود دارد! اگر شما یک عدد و یک رشته را با هم جمع کنید حاصل یک رشته خواهد بود.

به مثال های زیر توجه کنید:

```
x=5+5;
document.write(x);
x="5"+"5";
document.write(x);
x=5+"5";
document.write(x);
x="5"+5;
document.write(x);
```

عملگرهای مقایسه ای JavaScript

عملگرهای مقایسه ای در دستورات منطقی برای تعیین تساوی یا اختلاف بین متغیرها و یا مقادیر استفاده می شوند.

جدول زیر، عملگرهای مقایسه ای JavaScript را توضیح می دهد (با فرض x=5):

عملگر	توضیحات	مثال	نتیجه
==	تساوی	X==8 X==5	false true

===	تساوی از نظر نوع و مقدار	x===5 x==="5"	True false
!=	مخالف	x!=8	True
>	بزرگتر از	x>8	false
<	کوچکتر از	X<8	True
=<	بزرگتر و مساوی	x>=8	false
=>	کوچکتر و مساوی	X<=8	True

چگونه از آن استفاده کنیم؟

عملگرهای مقایسه ای در دستورات شرطی برای مقایسه مقادیر استفاده می شوند.

```
if (age<18) document.write("Too young");
```

در فصل های آینده درباره دستورات شرطی بیشتر خواهید آموخت.

عملگرهای منطقی JavaScript

عملگرهای منطقی برای تعیین منطق بین متغیرها و یا مقادیر استفاده می شود.

جدول زیر، عملگرهای منطقی JavaScript را توضیح می دهد (با فرض x=6 و y=3):

عملگر	توضیحات	مثال	نتیجه
&&	and	(x < 10 && y > 1)	true
||	or	(x==5 || y==5)	false
!	not	!(x==y)	true

عملگر شرطی (?)

عملگر (?) یک عملگر شرطی است که مقداری را به یک متغیر، مبنی بر یک یا چند شرط انتساب می دهد.

نحوه استفاده

```
variablename=(condition)?value1:value2
```