Javascript

به زبان ساده

تقدیم به

همه جویندگان علم

راه‌های ارتباط با نویسنده

وب سایت: www.w3-farsi.com

لینک تلگرام: https://telegram.me/ebrahimi_younes

ID تلگرام: @ebrahimi_younes

پست الکترونیکی: younes.ebrahimi.1391@gmail.com

JavaScript چیست

جاواسکریپت (JavaScript) یک از زبان برنامه نویسی شیء گرا و پرطرفدار وب می‌باشد. این زبان را در ابتدا شخصی به نام Brendan Eich (برندان ایچ) در شرکت Netscape با نام Mocha طراحی نمود. این نام بعداً به LiveScript و نهایتاً به جاواسکریپت تغییر یافت. این تغییر نام تقریباً با افزوده شدن پشتیبانی از جاوا در مرورگر وب Netscape Navigator همزمانی دارد.

اولین نسخهٔ جاواسکریپت در نسخه B32.0 این مرورگر در دسامبر 1995 معرفی و عرضه شد. این نام گذاری منجر به سردرگمی‌های زیادی شده و این ابهام را ایجاد می‌کند که جاواسکریپت با جاوا مرتبط است در حالی که این‌طور نیست. عدهٔ زیادی این کار را یک ترفند تجاری برای به دست آوردن بخشی از بازار جاوا که در آن موقع زبان جدید مطرح برای برنامه‌نویسی تحت وب بود می‌دانند.

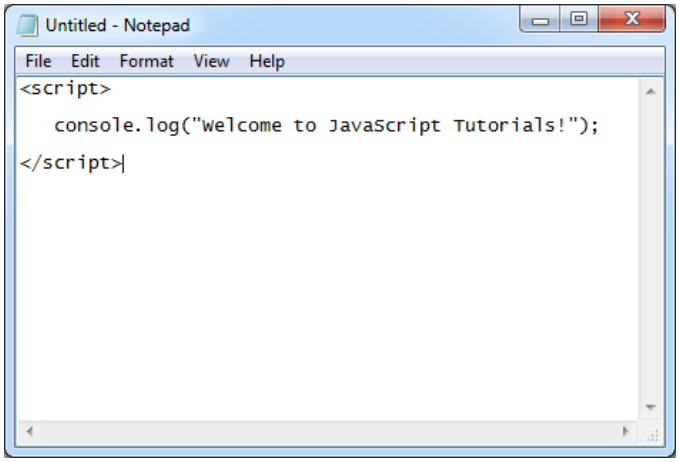
JavaScript به صورت «جاواسکریپت» خوانده می‌شود، ولی در فارسی به صورت »جاواسکریپت» ترجمه می‌شود و اگر به صورت «جاوا اسکریپت» ترجمه شود اشتباه است چون دو کلمه جدا از هم نیست و اگر به صورت دو کلمه جدا نوشته شود خطاهای نگارشی ایجاد می‌شود، به‌طور مثال ممکن است کلمه جاوا در انتهای خط و کلمه اسکریپت در ابتدای خط بعدی نوشته شود.

علیرغم اشتباه عمومی، زبان جاواسکریپت با زبان جاوا ارتباطی ندارد، اگر چه ساختار این زبان به سی پلاس پلاس (C++) و جاوا شباهت دارد که این امر برای یادگیری آسان در نظر گرفته شده است.

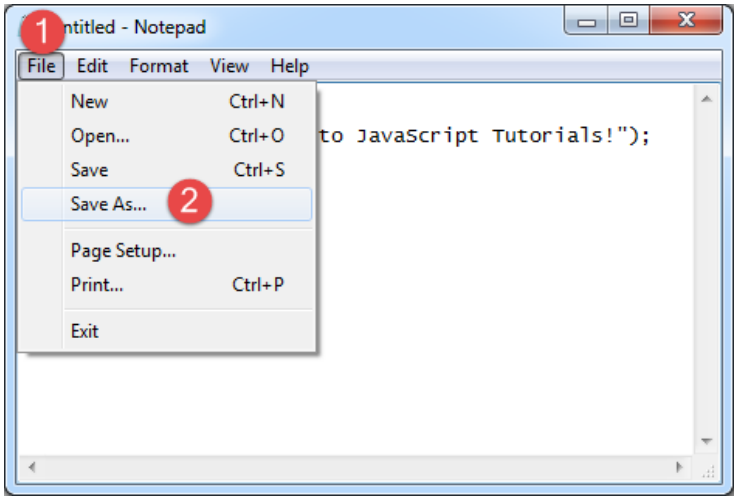
ساخت یک برنامه ساده JavaScript

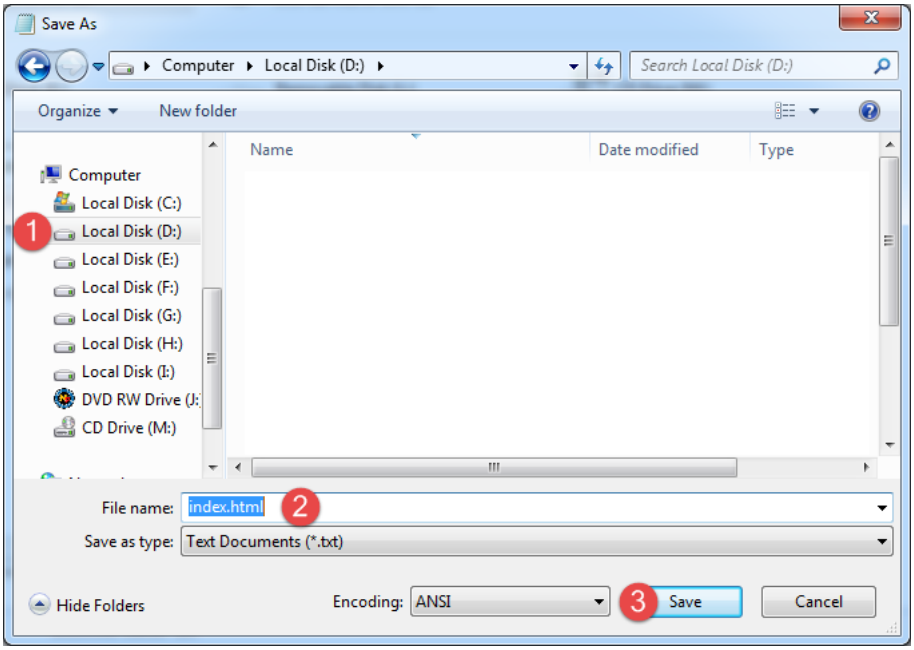
اجازه بدهید یک برنامه بسیار ساده به زبان جاواسکریپت بنویسیم. این برنامه یک پیغام را نمایش می‌دهد. در این درس می‌خواهم ساختار و دستور زبان یک برنامه ساده جاواسکریپت را توضیح دهم. برای اجرای کدهای جاواسکریپت، هیچ ابزار خاصی نیاز نیست. در واقع بر خلاف زبان‌های دیگر که نیاز به یک کامپایلر برای اجرای کدها دارند، جاواسکریپت روی مرورگرهای اینترنت اجرا می‌شود. برای نوشتن کدهای این زبان می توانید از یک ویرایشگر متن ساده، مانند NotePad پیش‌فرض ویندوز استفاده کنید؛ اما برای راحتی کار توصیه می‌کنیم از IDE (محیط‌ها و نرم‌افزارهای کدنویسی و توسعه) مناسب استفاده کنید. پیشنهاد ما به شما، Visual Studio Code مایکروسافت یا نرم افزار NotePad++ است. ما فرض را بر راحت ترین حالت ممکن می گذاریم. به منوی Start رفته و برنامه NotePad ویندوز را باز کرده و کدهای زیر را در داخل آن بنویسید:

```
<script>
    console.log("Welcome to JavaScript Tutorials!");
</script>
```

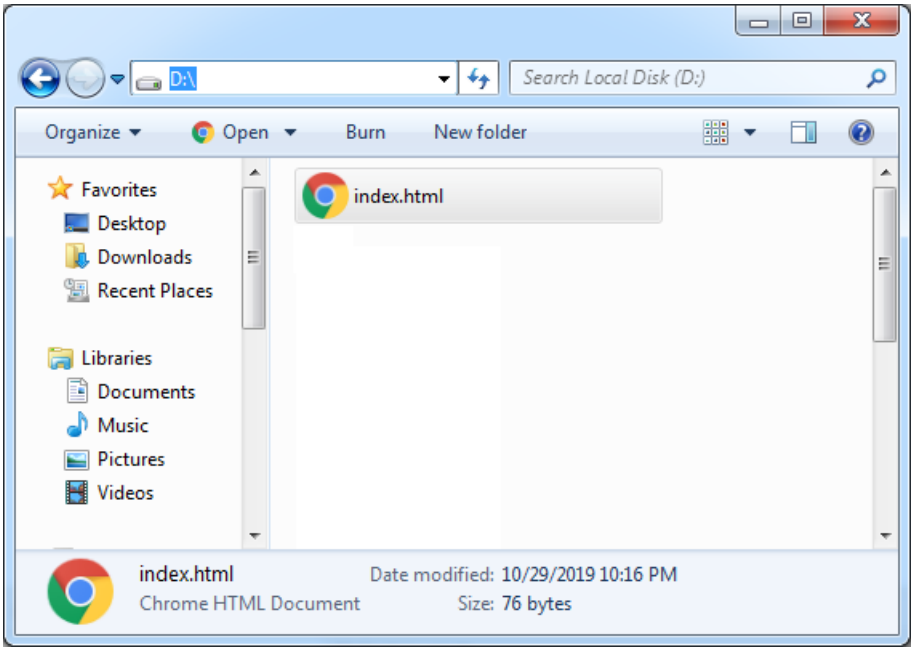


حال برنامه بالا را با نام `index.html` در درایو D به صورت زیر ذخیره کنید:

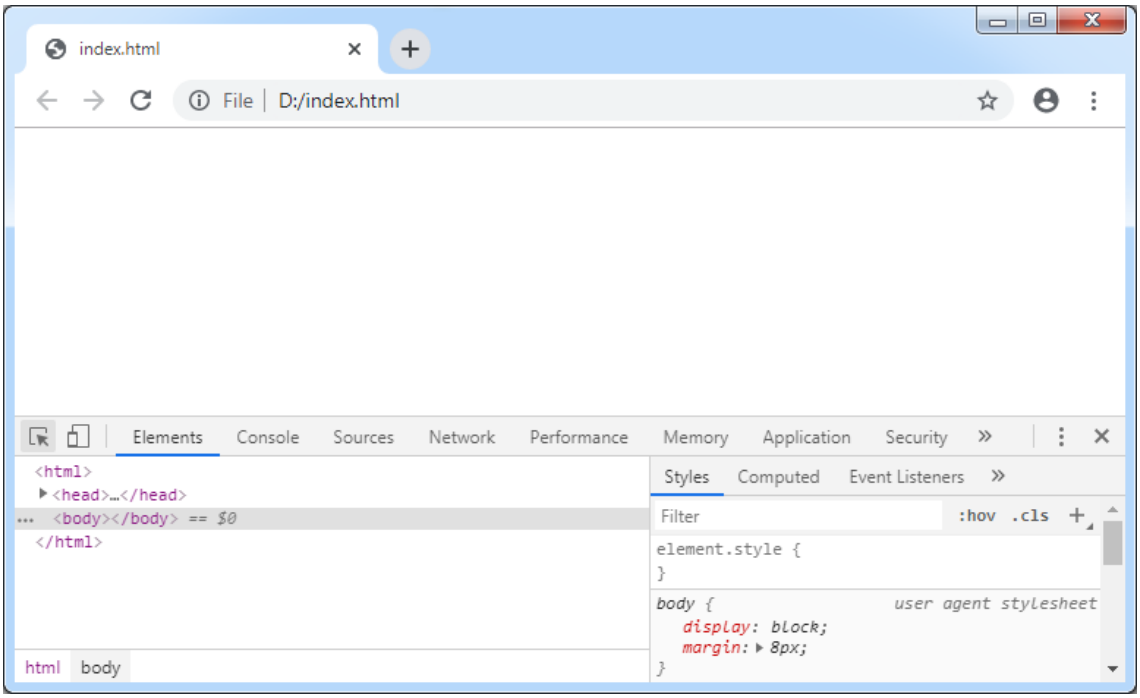




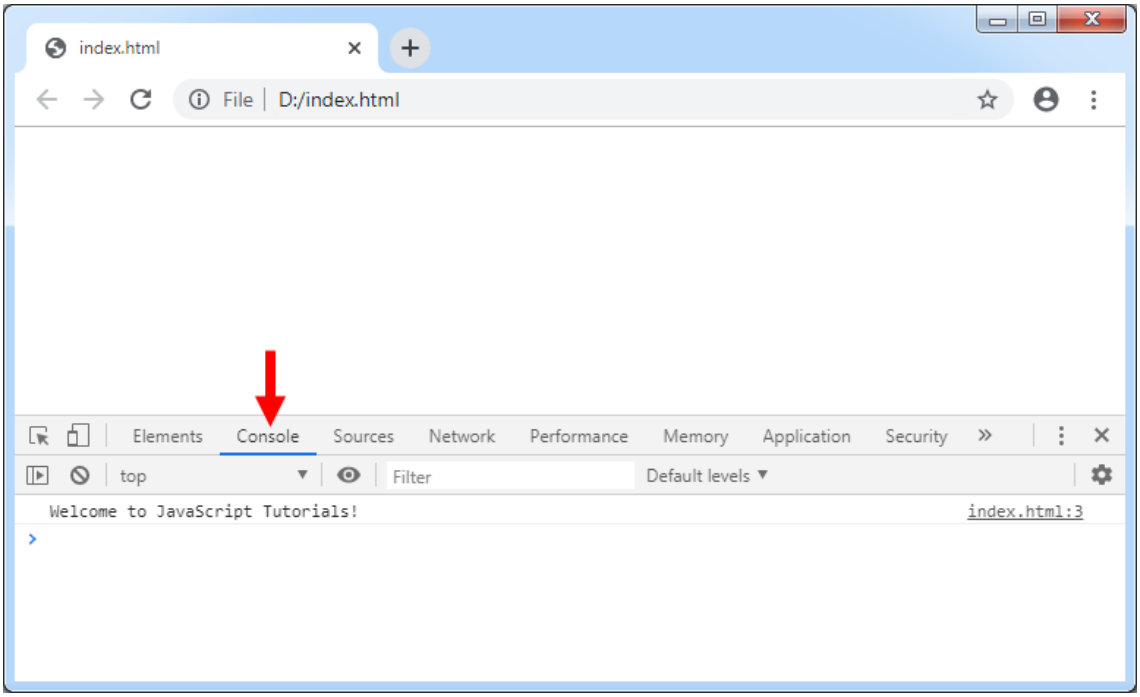
سپس به درایو D رفته و بر روی فایل `index.html` دو بار کلیک کنید تا به وسیله مرورگرتان اجرا شود:



اگر از یکی از مرورگرهای `Opera`، `FireFox` و یا `Chrome` استفاده می کنید، کافیست بعد از باز شدن مرورگر، بر روی دکمه های ترکیبی `Ctrl+Shift+C` بزنید تا پنجره ای به صورت زیر باز شود:



در این پنجره به سربرگ Console رفته تا نتیجه اجرای برنامه را ببینید :



مثال بالا ساده‌ترین برنامه‌ای است که شما می‌توانید در Javascript بنویسید. هدف در مثال بالا نمایش یک پیغام در صفحه نمایش است.

توصیه می کنیم که برای اجرای کدهای Javascript از آخرین نسخه مرورگر Chrome استفاده کنید، چون از تمام قابلیت ها و امکانات نسخه نهایی جاوااسکریپت پشتیبانی می کند.

هر زبان برنامه نویسی دارای قواعدی برای کدنویسی است. Javascript دارای توابع از پیش تعریف شده‌ای است که هر کدام برای مقاصد خاصی به کار می‌روند. هر چند که در آینده در مورد توابع بیشتر توضیح می‌دهیم، ولی در همین حد به توضیح تابع بسنده می‌کنیم که توابع مجموعه‌ای از کدها هستند که دارای یک نام بوده و در جلوی نام آنها علامت () قرار می‌گیرد. یکی از این توابع، تابع log() است. از تابع log() برای چاپ یک رشته استفاده می‌شود. یک رشته گروهی از کاراکترها است، که به وسیله دابل کوتیشن (") محصور شده است. مانند "Welcome to javascript Tutorials!".

یک کاراکتر می‌تواند یک حرف، عدد، علامت یا ... باشد. در کل مثال بالا نحوه استفاده از تابع log() است. توضیحات بیشتر در درس‌های آینده آمده است. Javascript فضاهای خالی را نادیده می‌گیرد. مثلاً از کد زیر اشکال نمی‌گیرد:

```
console.log(
   "Welcome to JavaScript Tutorials!");
```

همیشه به یاد داشته باشید که Javascript به بزرگی و کوچکی حروف حساس است. یعنی به طور مثال MAN و man در Javascript با هم فرق دارند. رشته‌ها و توضیحات از این قاعده مستثنی هستند که در درس‌های آینده توضیح خواهیم داد. مثلاً کدهای زیر با خطا مواجه می‌شوند و اجرا نمی‌شوند:

```
Console.log("Welcome to JavaScript Tutorials!");
console.LoG("Welcome to JavaScript Tutorials!");
CONSOLE.log("Welcome to JavaScript Tutorials!");
```

تغییر در بزرگی و کوچکی حروف از اجرای کدها جلوگیری می‌کند. اما کد زیر کاملاً بدون خطا است:

```
console.log("Welcome to JavaScript Tutorials!");
```

حال که با خصوصیات و ساختار اولیه Javascript آشنا شدید در درسهای آینده مطالب بیشتری از این زبان برنامه نویسی قدرتمند خواهید آموخت.

توضیحات

وقتی که کدی تایپ می‌کنید شاید بخواهید که متنی جهت یادآوری وظیفه آن کد به آن اضافه کنید. در JavaScript (و بیشتر زبانهای برنامه نویسی) می‌توان این کار را با استفاده از توضیحات انجام داد. توضیحات متونی هستند که توسط مفسر نادیده گرفته می‌شوند و به عنوان بخشی از کد محسوب نمی‌شوند.

هدف اصلی از ایجاد توضیحات، بالا بردن خوانایی و تشخیص نقش کدهای نوشته شده توسط شما، برای دیگران است. فرض کنید که می‌خواهید در مورد یک کد خاص، توضیح بدهید، می‌توانید توضیحات را در بالای کد یا کنار آن بنویسید. از توضیحات برای مستند سازی برنامه هم استفاده می‌شود. در برنامه زیر نقش توضیحات نشان داده شده است :

```
 // This line will print the message hello world
console.log("Hello World!");
```

```
Hello World!
```

در کد بالا، خط 2 یک توضیح درباره خط 3 است که به کاربر اعلام می‌کند که وظیفه خط 3 چیست؟ با اجرای کد بالا فقط جمله Hello World چاپ شده و خط 2 در خروجی نمایش داده نمی‌شود چون مفسر توضیحات را نادیده می‌گیرد. توضیحات بر دو نوعند :

توضیحات تک خطی

```
// single line comment
```

توضیحات چند خطی

```
/* multi
   Line
   comment */
```

توضیحات تک خطی همانگونه که از نامش پیداست، برای توضیحاتی در حد یک خط به کار می‌روند. این توضیحات با علامت // شروع می‌شوند و هر نوشته‌ای که در سمت راست آنها قرار بگیرد جز توضیحات به حساب می‌آید. این نوع توضیحات معمولاً در بالا یا کنار کد قرار می‌گیرند. اگر توضیح درباره یک کد به بیش از یک خط نیاز باشد از توضیحات چند خطی استفاده می‌شود. توضیحات چند خطی با /* شروع و با */ پایان می‌یابند. هر نوشته‌ای که بین این دو علامت قرار بگیرد جز توضیحات محسوب می‌شود.

کاراکترهای کنترلی

کاراکترهای کنترلی، کاراکترهای ترکیبی هستند که با یک بک اسلش (\) شروع می‌شوند و به دنبال آنها یک حرف یا عدد می‌آید و یک رشته را با فرمت خاص نمایش می‌دهند. برای مثال برای ایجاد یک خط جدید و قرار دادن رشته در آن می‌توان از کاراکتر کنترلی \n استفاده کرد :

```
console.log("Hello\nWorld!");
```

```
Hello
World
```

مشاهده کردید که مفسر بعد از مواجهه با کاراکتر کنترلی \n نشانگر ماوس را به خط بعد برده و بقیه رشته را در خط بعد نمایش می‌دهد.

جدول زیر لیست کاراکترهای کنترلی و کارکرد آنها را نشان می‌دهد :

عملکرد	کاراکتر کنترلی	عملکرد	کاراکتر کنترلی
Form Feed	\f	چاپ کوتیشن	\’
خط جدید	\n	چاپ دابل کوتیشن	\”
سر سطر رفتن	\r	چاپ بک اسلش	\\
حرکت به صورت افقی	\t	چاپ فضای خالی	\0
حرکت به صورت عمودی	\v	صدای بیپ	\a
چاپ کاراکتر یونیکد	\u	حرکت به عقب	\b

ما برای استفاده از کاراکترهای کنترلی، از بک اسلش (\) استفاده می‌کنیم. از آنجاییکه \ معنای خاصی به رشته‌ها می‌دهد برای چاپ بک اسلش (\) باید از (\\) استفاده کنیم :

```
console.log("We can print a \\ by using the \\\\ escape sequence.");
```

```
We can print a \ by using the \\ escape sequence.
```

یکی از موارد استفاده از \\، نشان دادن مسیر یک فایل در ویندوز است :

```
console.log("C:\\Program Files\\Some Directory\\SomeFile.txt");
```

```
C:\Program Files\Some Directory\SomeFile.txt
```

از آنجاییکه از دابل کوتیشن (“) برای نشان دادن رشته‌ها استفاده می‌کنیم برای چاپ آن از \” استفاده می‌کنیم :

```
console.log("I said, \"Motivate yourself!\".");
```

```
I said, "Motivate yourself!".
```

همچنین برای چاپ کوتیشن (‘) از \’ استفاده می‌کنیم :

```
console.log("The programmer\'s heaven.");
```

```
The programmer's heaven.
```

برای ایجاد فاصله بین حروف یا کلمات از \t استفاده می‌شود :

```
console.log("Left\tRight");
```

```
Left  Right
```

برای چاپ کاراکترهای یونیکد می‌توان از \u استفاده کرد. برای استفاده از \u، مقدار در مبنای 16 کاراکتر را درست بعد از علامت \u قرار می‌دهیم. برای مثال اگر بخواهیم علامت کپی رایت (©) را چاپ کنیم، باید بعد از علامت \u مقدار A900 را قرار دهیم مانند :

```
console.log("\u00A9");
```

```
©
```

برای مشاهده لیست مقادیر مبنای 16 برای کاراکترهای یونیکد به لینک زیر مراجعه نمایید :

```
http://www.ascii.cl/htmlcodes.htm
```

اگر مفسر به یک کاراکتر کنترلی غیر مجاز برخورد کند، برنامه پیغام خطا می‌دهد. بیشترین خطا زمانی اتفاق می افتد که برنامه نویس برای چاپ اسلش (\) از \\ استفاده می‌کند.

متغیر

متغیر مکانی از حافظه است که شما می‌توانید مقادیری را در آن ذخیره کنید. می‌توان آن را به عنوان یک ظرف تصور کرد که داده‌های خود را در آن قرار داده‌اید. محتویات این ظرف می‌تواند پاک شود یا تغییر کند. هر متغیر دارای یک نام نیز هست. که از طریق آن می‌توان متغیر را از دیگر متغیرها تشخیص داد و به مقدار آن دسترسی پیدا کرد. همچنین دارای یک مقدار می‌باشد که می‌تواند توسط کاربر انتخاب شده

باشد یا نتیجه یک محاسبه باشد. مقدار متغیر می‌تواند تهی نیز باشد. متغیر دارای نوع نیز هست بدین معنی که نوع آن با نوع داده‌ای که در آن ذخیره می‌شود یکی است.

متغیر دارای عمر نیز هست که از روی آن می‌توان تشخیص داد که متغیر باید چقدر در طول برنامه مورد استفاده قرار گیرد. و در نهایت متغیر دارای محدوده استفاده نیز هست که به شما می‌گوید که متغیر در چه جای برنامه برای شما قابل دسترسی است. ما از متغیرها به عنوان یک انبار موقتی برای ذخیره داده استفاده می‌کنیم. هنگامی که یک برنامه ایجاد می‌کنیم احتیاج به یک مکان برای ذخیره داده، مقادیر یا داده‌هایی که توسط کاربر وارد می‌شوند، داریم. این مکان، همان متغیر است.

برای این از کلمه متغیر استفاده می‌شود چون ما می‌توانیم بسته به نوع شرایط هر جا که لازم باشد، مقدار آن را تغییر دهیم. متغیرها موقتی هستند و فقط موقعی مورد استفاده قرار می‌گیرند که برنامه در حال اجراست و وقتی شما برنامه را می‌بندید محتویات متغیرها نیز پاک می‌شود. قبلاً ذکر شد که به وسیله نام متغیر می‌توان به آن دسترسی پیدا کرد. برای نامگذاری متغیرها باید قوانین زیر را رعایت کرد :

- نام متغیر باید با یکی از حروف الفبا (a-z or A-Z) یا علامت _ شروع شود.
- نمی‌تواند شامل کاراکترهای غیرمجاز مانند . ,$,^ ,? ,# باشد.
- نمی‌توان از کلمات رزرو شده در جاوااسکریپت برای نام متغیر استفاده کرد.
- نام متغیر نباید دارای فضای خالی (spaces) باشد.
- اسامی متغیرها نسبت به بزرگی و کوچکی حروف حساس هستند. در جاوااسکریپت دو حرف مانند a و A دو کاراکتر مختلف به حساب می‌آیند.

دو متغیر با نامهای myNumber و MyNumber دو متغیر مختلف محسوب می‌شوند چون یکی از آنها با حرف کوچک m و دیگری با حرف بزرگ M شروع می‌شود. متغیر دارای نوع هست که نوع داده‌ای را که در خود ذخیره می‌کند را نشان می‌دهد. در درس بعد در مورد انواع داده ها در جاوااسکریپت توضیح می دهیم. لیست کلمات کلیدی جاوااسکریپت، که نباید از آنها در نامگذاری متغیرها استفاده کرد در زیر آمده است:

abstract	arguments	*await	boolean
break	byte	case	catch
char	*class	const	continue
debugger	default	delete	do

eval	*enum	else	double
final	false	*extends	*export
function	for	float	finally
*import	implements	if	goto
interface	int	instanceof	in
new	native	long	*let
protected	private	package	null
static	short	return	public
this	synchronized	switch	*super
true	transient	throws	throw
void	var	typeof	try
yield	with	while	volatile

انواع داده

انواع داده هایی که در JavaScript وجود دارند عبارتند از :

داده	توضیح
عددی (Numeric)	شامل اعداد مثبت و منفی (اعشاری و صحیح) می باشد.
رشته ای (String)	به مجموعه ای از کاراکترها که بین دو علامت کوتیشن یا دابل کوتیشن قرار گرفته باشند، اطلاق می شود.
آرایه (Array)	مجموعه ای از آیتم ها هستند که بین دو علامت [] قرار گرفته و با علامت کاما (,) از هم جدا شده اند.

شیء (Object)	مجموعه ای از آیتم ها هستند که به صورت کلید و مقدار بوده، بین دو علامت {} قرار گرفته، و با علامت کاما (,) از هم جدا شده اند.
boolean	شامل دو مقدار true یا false می باشد.
Null	یک مقدار خاص است که به معنای هیچ چیز، خالی و یا مقدار ناشناخته است.
undefined	اگر متغیری تعریف شود و مقداردهی اولیه نشود آنگاه مقدار آن برابر با undefined خواهد بود

در مورد انواع داده های بالا و نحوه استفاده از آنها در متغیرها، در درس بعد توضیح می دهیم.

استفاده از متغیرها

بر خلاف زبان هایی مثل جاوا و سی شارپ، که هنگام تعریف متغیر باید نوع متغیر را هم مشخص می کردیم، در JavaScript کافیست که ابتدا کلمه کلیدی var و سپس نام متغیر را نوشته و به وسیله علامت مساوی یک مقدار به آن اختصاص دهیم :

```
var variableName = Value
```

در مثال زیر نحوه تعریف و مقداردهی متغیرها نمایش داده شده است :

```
1  var numericVar = 10
2  var boolVar    = true
3  var StringVar  = "Hello World!"
4  var arrayVar   = [1, 5, 8]
5  var objectVar  = { 'Name': 'jack', 'family': 'Scalia', 'Age': 7 }
6
7  console.log("numericVar = ", numericVar)
8  console.log("boolVar    = ", boolVar)
9  console.log("StringVar  = ", StringVar)
10 console.log("arrayVar   = ", arrayVar)
11 console.log("objectVar  = ", objectVar)
```

```
numericVar = 10
boolVar    = true
StringVar  = Hello World!
arrayVar   = Array(3)
    0: 1
    1: 5
    2: 8
    length: 3
objectVar = Object
    Name: "jack"
    family: "Scalia"
    Age: 7
```

در خطوط 1-5، متغیر ها تعریف شده اند. اما نوع این متغیرها چیست؟ JavaScript نوع متغیرها را بسته به مقداری که به آنها اختصاص داده می شود در نظر می گیرد. مثلا نوع متغیر StringVar در خط 3 از نوع رشته است، چون یک مقدار رشته ای به آن اختصاص داده شده است. به خطوط 4 و 5 کد بالا توجه کنید. در خط 4 یک متغیر تعریف شده است و نوع داده ای که به آن اختصاص داده شده است از نوع array است. همانطور که در درس قبل اشاره شد، برای تعریف array علامت [] به کار می رود و آیتم های داخل آن به وسیله کاما از هم جدا می شوند :

```
var arrayVar = [1, 5, 8]
```

در خط 5 هم یک متغیر تعریف شده است و یک مقدار از نوع Object به آن اختصاص داده شده است. در تعریف Object به جای علامت [] از {} استفاده می شود. آیتم ها به صورت کلید/مقدار تعریف می شوند. بین کلید و مقدار علامت : و بین هر دو کلید/مقدار، هم علامت , قرار می گیرد :

```
var objectVar = { Key1: Value1, Key2: Value2, Key3: Value3 }
```

مثلا در مثال بالا یک Object تعریف کرده ایم که سه آیتم یا کلید/مقدار دارد که بین آنها علامت کاما (,) قرار داده ایم. ولی بین یک کلید و مقدار مربوط به آن علامت : قرار گرفته است. برای اختصاص یک مقدار به چند متغیر می توان به صورت زیر عمل کرد :

```
var identifier1 = identifier2 = ...indentifierN = Value
```

به مثال زیر توجه کنید :

```
var num1 = num2 = num3 = num4 = num5 = 10
var message1 = message2 = message3 = "Hello World!"

console.log(num1)
console.log(num4)
console.log(message1)
console.log(message3)
```

```
10
10
Hello World!
Hello World!
```

دقت کنید که برای متغیرهای تعریف شده در حالت بالا یک خانه حافظه تخصیص داده می شود، یعنی مقدار 10 در حافظه ذخیره شده و متغیرهای num1 و num2 و num3 و num4 و num5 به آن خانه از حافظه اشاره می کنند. همچنین می توان چند متغیر را تعریف کرد و برای هر یک از آن ها مقدار جداگانه ای مشخص نمود:

```
var identifier1, identifier2, ...indentifierN = Value1, Value2, ...ValueN
```

به مثال زیر توجه کنید:

```
var [num1, num2, message1] = [10, 12.5, "Hello World!"]

console.log(num1)
console.log(num2)
console.log(message1)
```

```
10
12.5
Hello World!
```

در کد بالا مقدار num1 برابر 10، num2 برابر 12.5 و message1 برابر Hello World! می باشد. در JavaScript، متغیر ها هم باید تعریف و هم مقداردهی شوند. یعنی اگر متغیری را تعریف کرده و به آن مقداری را اختصاص ندهید و برنامه را اجرا کنید با خطا مواجه می شوید :

```
var number

console.log(number)
```

```
Uncaught ReferenceError: number is not defined
```

همانطور که در درس قبل هم اشاره کردیم، یک رشته در اصل یک مجموعه از کاراکترهاست که در داخل علامت "" یا ' ' قرار دارند. هر کدام از این کاراکترها دارای یک اندیس است که به وسیله آن اندیس قابل دسترسی هستند. اندیس کاراکترها در رشته از 0 شروع می شود.

به رشته زیر توجه کنید :

```
var message = "Hello World!"
```

در رشته بالا اندیس کاراکتر O برابر 4 است. برای درک بهتر به شکل زیر توجه کنید :

```
H e l l o   W o r l d  !
0 1 2 3 4 5 6 7 8 9 10 11
```

حال برای چاپ یک کاراکتر (مثلا W) از این رشته کافیست که به صورت زیر عمل کنیم :

```
var message = "Hello World! "

console.log(message[6])
```

```
W
```

همانطور که در کد بالا مشاهده می کنید کافیست که نام متغیر را نوشته، در جلوی آن یک جفت کروشه و در داخل کروشه ها اندیس آن کاراکتری را که می خواهیم چاپ شود را بنویسیم. چاپ مقدار با استفاده از اندیس در مورد آرایه هم صدق می کند :

```
var arrayVar = [1, 5, 8]

console.log(arrayVar[2])
```

```
8
```

و اما در مورد Object، شما باید نام کلید را بنویسید تا مقدار آن برای شما نمایش داده شود:

```
var objectVar = { 'Name': 'jack', 'Family': 'Scalia', 'Age': 7 }

console.log(objectVar['Family'])
```

```
Scalia
```

نکته ای که بهتر است در همین جا به آن اشاره کنیم این است که کلید/مقدارها در Object می توانند از هر نوعی باشند و شما برای چاپ مقدار مربوط به یک کلید باید نام کلید را دقیق بنویسید. به مثال زیر توجه کنید :

```
var objectVar = { 1: 'Jack', '2': 'Scalia', 3: 7 }

console.log(objectVar['2'])
```

```
Scalia
```

در مثال بالا ما مقدار کلید '2' را چاپ کرده ایم. حال اگر به جای '2' عدد 2 را بنویسیم، نتیجه همان است :

```
var objectVar = { 1: 'Jack', '2': 'Scalia', 3: 7 }

console.log(objectVar[2])
```

```
Scalia
```

ثابت ها

ثابت‌ها، انواعی هستند که مقدار آنها در طول برنامه تغییر نمی‌کند. ثابت‌ها حتماً باید مقدار دهی اولیه شوند و اگر مقدار دهی آنها فراموش شود در برنامه خطا به وجود می‌آید. بعد از این که به ثابت‌ها مقدار اولیه اختصاص داده شد هرگز در زمان اجرای برنامه نمی‌توان آن را تغییر داد. برای تعریف ثابت‌ها باید از کلمه کلیدی const استفاده کرد. معمولاً نام ثابت‌ها را طبق قرارداد با حروف بزرگ می‌نویسند تا تشخیص آنها در برنامه راحت باشد. نحوه تعریف ثابت در زیر آمده است :

```
const data_type identifier = initial_value;
```

مثال :

```
const NUMBER = 1;

NUMBER = 10; //Uncaught TypeError: Assignment to constant variable.
```

در این مثال می‌بینید که مقدار دادن به یک ثابت، که قبلاً مقدار دهی شده برنامه را با خطا مواجه می‌کند. ممکن است این سؤال برایتان پیش آمده باشد که دلیل استفاده از ثابت‌ها چیست؟ اگر مطمئن هستید که مقادیری در برنامه وجود دارند که هرگز در طول برنامه تغییر نمی‌کنند بهتر است که آنها را به صورت ثابت تعریف کنید. این کار هر چند کوچک کیفیت برنامه شما را بالا می‌برد.

تبدیل انواع داده

در زبان جاوااسکریپت امکان تبدیل یک نوع به نوع دیگر وجود دارد. این زبان دارای مجموعه‌ای از توابع از پیش تعریف شده است، که می‌توانند مقادیر را از یک نوع بـه نوع دیگر تبدیل کنند. جاوااسکریپت دو متد برای تبدیل انواع غیر عددی به عددی فراهم کرده است:

- praseInt()
- parseFloat()

توجه کنید که حروف F و I باید به صورت حرف بزرگ نوشته شوند.

این متدها فقط بر روی رشته‌های حاوی عدد کار می‌کنند و بر روی بقیه انواع مقدار NaN را بر می‌گردانند. متد praseInt() از اولین کاراکتر رشته شروع می‌کند اگر عدد بود آن را بر می‌گرداند در غیر این صورت مقدار NaN را بر می‌گردانند. این روند تا آخرین کاراکتر ادامه پیدا می‌کند تا اینکه به کاراکتری غیر عددی برسد. به مثال زیر توجه کنید:

```
console.log(parseInt("25Number"));
```

```
25
```

متد parseFloat() نیز همانند parseInt() عمل کرده و از اولین کاراکتر شروع به جستجو می‌کند. البته در این متد اولین کاراکتر نقطه حساب نمی‌شود و آن را به همان صورت می‌گرداند. به مثال زیر توجه کنید:

```
console.log(parseFloat("25.5Number"));

console.log(parseFloat("25.45.2Number"));
```

```
25.5
25.45
```

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

- Boolean()
- Number()

- String()

متد Boolean() زمانی مقدار true را بر می‌گرداند که پارامتر دریافتی‌اش، رشته‌ای شامل حداقل یک کارکتر، یک عدد غیر از صفر و یا یک شیء باشد. مقدار false را نیز زمانی بر می‌گرداند که پارامتر دریافتی‌اش رشته‌ای تهی، عدد صفر یا یکی مقادیر null و undefined باشد:

```
var b1 = Boolean("");
var b2 = Boolean("String");
var b3 = Boolean(100);
var b4 = Boolean(null);
var b5 = Boolean(0);
var b6 = Boolean(new Object());

console.log(b1);
console.log(b2);
console.log(b3);
console.log(b4);
console.log(b5);
console.log(b6);
```

```
false
true
true
false
false
true
```

متد Number() کاری شبیه به متدهای parseInt() و parseFloat() انجام می‌دهد ولی تفاوت‌هایی هم با این دو متد دارد. اگر به یاد داشته باشید متدهای parseInt() و parseFloat() آرگومان دریافتی را فقط تا اولین کاراکتر بی ارزش بر می‌گرداندند. مثلاً رشته "25.5.2" را به ترتیب به 25 و 25.5 تبدیل خواهند کرد. اما متد Number() مقدار NaN را می‌گرداند. زیرا این رشته از نظر متد Number() امکان تبدیل به یک عدد را ندارد. اگر رشته‌ای امکان تبدیل به یک عدد را داشته باشد متد Number()، خود برای استفاده از یکی از توابع parseInt() و parseFloat() تصمیم می‌گیرد. مثال زیر حاصل اجرای تابع Number() برای انواع داده‌ها را نشان می‌دهد:

```
console.log(Number(false));
console.log(Number(true));
console.log(Number(undefined));
console.log(Number(null));
console.log(Number("5.5"));
console.log(Number("56"));
console.log(Number("5.6.7"));
console.log(Number(new Object()));
console.log(Number(100));
```

```
0
1
NaN
0
5.5
```