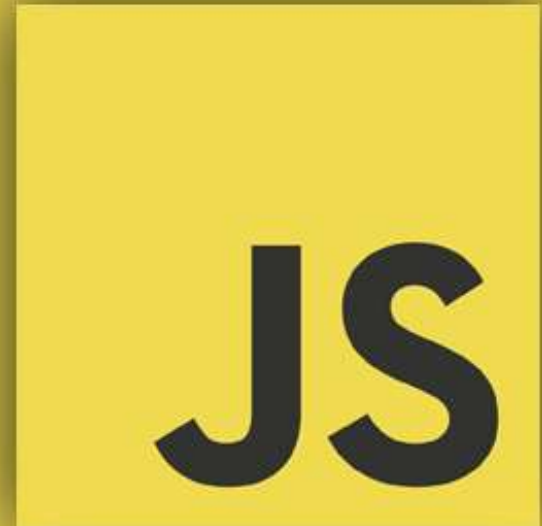


# *JavaScript 1.8.5*



By Vijay Shivakumar  
<https://dotjs.in>

**Before We Begin**

# Before We Begin

- What we should have

An IDE

Visual Studio Code (recommended)

- What you should know

Basics of HTML, CSS, XML, DOM etc...

# What we will learn

JavaScript Premier

Objects in JavaScript

Events

Build Web (Browser) based programs

Hands On Experience

# About Me

Vijay Shivakumar

Designer | Developer | Trainer

Training on web and Adobe products from past 10+ years



# About You ?

- Developer
- Designer
- Architect

# History Of JavaScript

- Developed by Netscape
- Client side support for Sun's JAVA
- Concept to Creation in 10 days
- Shipped with Netscape ver. 2.0 (1995)
- Code Name *MOCHA* officially called *LIVESCRIPT*
- Renamed to JavaScript as Microsoft had the patent on the name live.



**Brendan Eich**

# History Of JavaScript

- Microsoft's implementation is called *JScript*
- ECMA Script embraced JavaScript for standardization after 1996
- Officially called as *ECMA Script* but popularly known as *JavaScript*
- Netscape was acquired by AOL in 1999
- AOL and Sun alliance for iPlanet after dissolution iPlanet and JavaScript is retained by Sun.
- Now Oracle owns the name JavaScript and Mozilla owns the source code



# Who reads your program ?

Browser's JavaScript engines

V8 engine in Chromium browsers

(Chrome, Safari, IE Edge, Opera)

Chakra in IE9 & IE Edge with windows 10

SpiderMonkey in Firefox

Webkit in Old Safari

Futhark in Opera until version 10.10

They do

memory management

just in time compilation

(in olden days browsers used to interpret your  
JavaScript)

# What JavaScript is NOT ...!

- JAVASCRIPT is not JAVA
- Can't create or edit files  
(cookies are an exception)
- Can't be used to talk to databases
- Doesn't need to be compiled
- Can't keep track of user's interaction  
(stateless)

NOTE : How ever there is a version of JavaScript derived from google's V8 JavaScript engine called node.js and rhino.js from mozilla which can do all of the above...

# What is JavaScript ?

- A programming tool for HTML designers / developers
- Read, Modify and Create HTML elements
- React to events like click, swipe, drag, tap etc..
- Validate data
- Detect visitor's browser
- Create and read cookies

# Why JavaScript ?

Most used scripting language

Great for UI-coding

Flexible and powerful

Everything is an object (including functions)

AJAX makes it a must-know

# Bad things about JavaScript

- Global Scope
- + for adding and concatenation
- No need for a semicolon to terminate a line

# JavaScript Fundamentals

# Data Types in JavaScript

Floats

Decimals

Inters and

Unsigned integers



Number

true

false



Boolean

"vijay"



String

Objects, Arrays, Functions,  
Regex, Error, Map etc.,



Object

stores any data type above or arrays & objects

# Data Types in JavaScript

Large Numbers



BigInt

unique and immutable



Symbol

A variable without a value



Undefined

When a variable is  
kept empty



Null

stores any data type above or arrays & objects



# Data Types in JavaScript

Number | 4.5 Any number not inside quote marks

Boolean | true or false A logical operator

String | "Vijay" A series of characters inside quote marks

Object | A virtual thing defined by its properties and methods (in javascript most of them are objects)

Undefined | Returns when a non existent value is called.  
undefined when you have not assigned any thing yet

Null | Usually assigned by developers when we initialize a variable but don't want to assign anything yet.  
null is assigned by developers as place holders

# Where to write JavaScript

## Inline JavaScript

```
<a href="javascript:callfun()">click me</a>
```

```
<a onclick="callfun()" href="#">click me</a>
```

## Infile (Embedded) JavaScript

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

```
    callfun()
```

```
</script>
```

## External JavaScript (best recommended)

```
yourscript.js
```

(do not use spaces or uppercase for file names)

# Programming in JavaScript

Variables

Operators

Strings

Arrays

Functions

Conditions

loops

# Variables

- A variable is a "container name" for information you want to store.
- A variable's value can change during the script.
- You can refer to a variable by name to access or to change its value.

- Rules for variable names:

Variable names are case sensitive

They must begin with a letter or the underscore character

IMPORTANT! JavaScript is case-sensitive! A variable named **uName** is not the same as a variable named **uname**

# Variables

- A variable when declared will have a value of undefined.
- Variable can take any data-type in JavaScript and even be changed later
- Variables must be declared and assigned in the beginning else they get hoisted to the top as undefined

# List of reserved words

break	delete	function	return
typeof case	do	if	
switch	var	catch	else
in	this	void	continue
false	throw	while	instanceof
debugger	finally	new	true
with	default	for	null
try			

class	const	enum	
export	extends	import	super

\* Reserved in ECMA 5

# List of reserved words

implements

let

private

public

yield

interface

package

protected

static

arguments

eval

# Variable LifeCycle

Declaration Phase

Initialization Phase

```
var user = "Vijay";
```

is

```
var user;
```

```
console.log(user);
```

```
    user = "Vijay";
```

declaration phase

undefined

initialization phase



# Variable Scope

Global scope: The default scope for all code running in script mode.

Module scope: The scope for code running in module mode.

Function scope: The scope created with a function.

# Operators | Basics

<code>x + y</code>	(Numeric)	Adds x and y together
<code>x + y</code>	(String)	Concatenates x and y together
<code>x - y</code>		Subtracts y from x
<code>x * y</code>		Multiplies x and y together
<code>x / y</code>		Divides x by y
<code>x % y</code>		Modulus
<code>x++</code> , <code>++x</code>		Adds 1
<code>x--</code> , <code>--x</code>		Subtracts 1

# Operators | Assignment

**x = y**

Sets **x** to the value of **y**

**x += y**

Same as **x = x + y**

**x -= y**

Same as **x = x - y**

**x \*= y**

Same as **x = x \* y**

**x /= y**

Same as **x = x / y**

**x %= y**

Same as **x = x % y**

# Operators | Comparison

<code>==</code>	Equals
<code>!=</code>	Does not equal
<code>===</code>	Strictly equals
<code>!==</code>	Strictly does not equal
<code>&gt;</code>	Is greater than
<code>&gt;=</code>	Is greater than or equal to
<code>&lt;</code>	Is less than
<code>&lt;=</code>	Is less than or equal to

# Operators | Comparison

<code>x == y</code>	Returns true if x and y are equal
<code>x === y</code>	Returns true if x and y are identical
<code>x != y</code>	Returns true if x and y are not equal
<code>x !== y</code>	Returns true if x and y are not identical
<code>x &gt; y</code>	Returns true if x is greater than y
<code>x &gt;= y</code>	Returns true if x is greater or equal to y
<code>x &lt; y</code>	Returns true if x is less than y
<code>x &lt;= y</code>	Returns true if x is less or equal to y
<code>x &amp;&amp; y</code>	Returns true if both x and y are true
<code>x    y</code>	Returns true if either x or y is true
<code>!x</code>	Returns true if x is false

# **Array**

# Array

- `var arr = new Array(5) ;`
- `var arr = [] ;`
- `var arr = ["one",2,true,[],{}] ;`

# Array Properties

length

constructor

prototype



# Array Methods

<code>arr.concat(arr2)</code>	merge 2 arrays to create the 3 <sup>rd</sup>
<code>arr.join()</code>	convert array to string <code>join(" ")</code>
<code>arr.pop();</code>	removes the last value;
<code>arr.push(value);</code>	adds the value at the last;
<code>arr.unshift(value);</code>	adds the value in the first;
<code>arr.shift()</code>	removes the value in the first;

# Array Methods

```
arr.slice(startIndex [endIndex]);
```

will remove (return) from the start index to end index and create another array.

Will not modify the existing array

---

```
arr.splice(startIndex,deleteCount,"new val");
```

will remove from the start index to count and inserts the value in between.

Will modify the existing array

---

```
arr.reverse() will reverse the existing order
```

---

```
arr.sort() takes a function to custom sorting
```

---

```
arr.toString() inherited method from object
```

---

```
arr.toLocaleString() same as above
```

# Array Methods (in ES 5)

`forEach()`

`map()`

`filter()`

`every()`

`some()`

`reduce()`

`reduceRight()`

`indexOf()`

`lastIndexOf()`

# Conditions

# if . . . else conditions

```
if (condition) {  
  statement[s] if true  
}
```

=====

```
if (condition) {  
  statement[s] if true  
} else {  
  statement[s] if false  
}
```

=====

```
while(condition) {  
    // statement to execute  
}
```

# Loops

# loops

```
for ([initial expression]; [condition]; [update  
    expression]) {  
    statement[s] inside loop  
}
```

=====

```
for(item in items){  
    // loop for every property in an object  
}
```

=====

```
for(item of items){  
    // loop for an iterable  
}
```

**Math**



# Few math methods

<code>Math.abs(val)</code>	Absolute value of val
<code>Math.round(val)</code>	n+1 when val >= n.5; otherwise n
<code>Math.ceil(val)</code>	Next integer greater than or equal to val
<code>Math.floor(val)</code>	Next integer less than or equal to val
<code>Math.sqrt(val)</code>	Square root of val
<code>Math.max(val1, val2)</code>	The greater of val1 or val2
<code>Math.min(val1, val2)</code>	The lesser of val1 or val2
<code>Math.random()</code>	Random number between 0 and 1

**Number**

# Number methods

When Number is not a number

isNaN

=====

Converting to Number

Number()

parseInt()

parseFloat()

**String**

# String Methods

```
var s = "hello world" // Start with some text.  
s.charAt(0) // "h": the first character.  
s.charAt(s.length-1) // "d": the last character.  
s.substring(1,4) // "ell": start with and until.  
s.slice(1,4) // "ell": same thing  
s.slice(-3) // "rld": last 3 characters  
s.indexOf("l") // 2: position of first letter l.  
s.lastIndexOf("l") // 10: position of last letter l.  
s.indexOf("l", 3) // 3: position of first "l" at or after 3
```

# String Methods

```
s.split(", ") // ["hello", "world"] convert to array  
s.replace("h", "H") // "Hello, world": replaces all instances  
s.toUpperCase() // "HELLO, WORLD"  
s.toLowerCase() // "hello, world"
```

# Escape Characters

`\b Backspace (\u0008)`

`\t Horizontal tab (\u0009)`

`\n Newline (\u000A)`

`\v Vertical tab (\u000B)`

`\r Carriage return (\u000D)`

`\" Double quote (\u0022)`

`\' Apostrophe or single quote (\u0027)`

**Date**



# Date

```
var dt = new Date(); // Returns current date
var dt = new Date(yyyy,mm,dd); // set date
dt.getFullYear(); // returns current year
dt.getMonth(); // zero-based months
dt.getDate(); // one-based days
dt.getDay(); // 0 is Sunday.
dt.getHours(); // 24hrs time
dt.getUTCHours(); // hours in UTC time depends on timezone
dt.toString(); // converts date info to string
```

# Date

```
dt.toLocaleDateString() //"01/01/2015"  
dt.toLocaleTimeString() //"09:10:30 AM"
```

# Functions

# Functions in JavaScript

Functions : a code block with a name

Methods : when inside an object

Class : that contain private, public members

Constructor : used to create instances

Module : self containing code block

# Function Anatomy

```
function myFun(arg1, arg2) {  
    alert(arg1 + arg2);  
};
```

```
function : expression  
myFun : name (optional)  
arg1, arg2 : parameters  
{ } : body of the function
```

# Functions are first class citizens

Can be passed as an argument to a function

Can be returned from a function

Can be assigned to a variable

Can be stored in an array

=====

Inherit from `Function.prototype`

Always have a return

If the function doesn't return anything it returns  
undefined

# How to write a function ?

Function can be statement or an expression

Function statement:

```
function myFun() {  
}
```

Function expression :

```
var myFun = function() {  
}
```

**Object**



# Top Level Objects

document

window

location

navigator

screen

history

# window.methods

`.open()`

```
var win = window.open("url.html", "winName",  
    "status,height=200,width=300");
```

`.close()`

```
close(), window.close(), self.close(), windowName.close();
```

`.alert()`

`.prompt()`

`.confirm()`

`.setInterval()` ,

`.clearInterval()`

`.setTimeout()` ,

`.clearTimeout()`

# DOM with JavaScript



# DOM Manipulation

- DOM selection
- DOM creation
- DOM attributes
- DOM removing

# Properties Of DOM

`document.body`

`document.title`

`document.forms[0]`

`document.forms["formName"]`

`document.formName`

`document.images[]`

`document.scripts`

`document.links`

`document.cookie`

`document.domain`

# Methods Of DOM

`document.write()` // open write layout stream

`document.close()` // close layout stream

`document.createElement()`

`document.createTextNode()`

`document.getElementById()`

`document.getElementsByTagName()`

`document.getElementsByName()`

`document.renameNode(selectnode, namespace, targetnode)`

# DOM Properties, Methods & events

<code>cookie</code>	<code>images[]</code>	<code>focus()</code>
<code>height</code>	<code>links[]</code>	<code>detachEvent()</code>
<code>width</code>	<code>scripts[]</code>	<code>write()</code>
<code>lastChild</code>	<code>styleSheets[]</code>	<code>writeln()</code>
<code>firstChild</code>	-----	<code>hasFocus()</code>
<code>location</code>	<code>onkeydown()</code>	<code>open()</code>
<code>nextSibling</code>	<code>onkeypress()</code>	<code>getElementById()</code>
<code>nodeName</code>	<code>onkeyup()</code>	<code>getElementsByName()</code>
<code>nodeType</code>	<code>onmouseover()</code>	<code>getElementsByTagName()</code>
<code>parentNode</code>	<code>onmousedown()</code>	
<code>parentWindow</code>	<code>onmousemove()</code>	
<code>previousSibling</code>	<code>onmouseup()</code>	
<code>readyState</code>	<code>onmouseout()</code>	
<code>title</code>	<code>onpropertychange()</code>	
<code>forms[]</code>	<code>onreadystatechange()</code>	
<code>frames[]</code>		



# Events of DOM

<code>onblur</code>	When the element loses focus
<code>onchange</code>	When the element changes
<code>onclick</code>	When an object is clicked
<code>ondblclick</code>	When an object is double-clicked
<code>onfocus</code>	When the element gets focus
<code>onkeydown</code>	When key is pressed
<code>onkeypress</code>	When key is pressed and released
<code>onkeyup</code>	When key is released
<code>onmousedown</code>	When mouse button is pressed
<code>onmousemove</code>	When mouse pointer moves
<code>onmouseout</code>	When mouse pointer moves out of an element
<code>onmouseover</code>	when mouse pointer moves over an element
<code>onmouseup</code>	when mouse button is released
<code>onreset</code>	when the form is reset
<code>onselect</code>	when the element is selected
<code>Onsubmit</code>	when the form is submitted

# **OOP in JavaScript**

# What is Object Oriented Programming ?

A paradigm that uses objects to create your program.

Any thing that is usually self contained and re-usable...

An object has the resources to work on its own to achieve the objective or can inherit properties and methods from other objects.

# Why OOP ?

Makes code easy to re-use

|No Re- write

Makes code easy to update

|Less Bugs

Code easily accessible through APIs

|Minimize Mistakes

( Hides what is not required by other objects, Provides access to only what is required )

# Objects

Objects contain properties and methods

Objects are made up of key value pairs

Key : value

If more than one property they are separated by comma " ,  
"

Keys can not be reserved key words

eg do, while, class, for etc

If so you can use quotes to overcome them eg., "class"

Values can be of any data type.

If values are functions we call them methods.

# Objects Creation

Objects can be created using

```
var obj = new Object();
```

```
var obj = {};
```

```
var obj = Object.create(null);
```

# OOP Concepts

## Creation

- creating Instances a piece of code via classes, functions or duplication

## Inheritance

- Extending the behavior of other classes

## Encapsulation

- Protect the internal functionalities from being accessed or modified

## Polymorphism

- Modify properties and methods of the parent class to achieve a customized performance





# Scope in JavaScript

# Scope... what is it ?

scope refers to the current context of your code  
(it simply is: "where to look for")

Scopes can be *globally* or *locally* defined

Scope also depends on who is looking for what and where

# Closure in JavaScript

# Closure in javascript

```
var user = function() {  
  var data = "Hello World";  
  return function() {  
    console.log(data);  
  }  
}
```

# Creating Objects

# Object.defineProperty

`Object.defineProperty(obj, prop, descriptor)`

`obj` : The object on which to define the property.

`prop` : The name of the property to be defined or modified.

`descriptor` : The descriptor for the property being defined or modified.

# Descriptor Object

`configurable`: true if the descriptor itself can be changed, defaults to false.

`enumerable`: true if this property shows up only while enumeration defaults to false.

`value` : The value for the property.

`writable` : true if the value can be changed. default is false

`get` : A function which serves as a getter for the property defaults to undefined.

`set` : A function which serves as a setter for the property, defaults to undefined.

# Exception Handling



# Exception Handling

What are Exceptions ?

A way to deal with errors that interrupt your program from working normally.

When do they happen?

On the runtime when an error has occurred which will cause the browser to create an exception

Or when programmatically you create an error with the throw method.

How can they be handled ?

You can use try catch and finally statements.

# Try | Catch | Finally

```
try{
```

This is the section of code that is expected to execute normally. But if any error occurs then its passed to the nearest catch block.

```
}
```

```
catch(err){
```

This is the section deals with the error that's thrown by try block.

```
}
```

```
finally{
```

The default section that executes in either case (if error or if no error)

```
}
```

# Throw Exception | Catch Error

```
throw "can throw a string error";  
throw 123456;  
throw new Error("this is my error message");
```

```
catch (error)
```

The error properties vary from IE and W3C browsers

But the name and the message is the same

name : will be the type of error usually Error

message : will be the message thrown

**[vijay.shivu@gmail.com](mailto:vijay.shivu@gmail.com)**