JavaScript & DOM

Java vs. JavaScript: Early Development

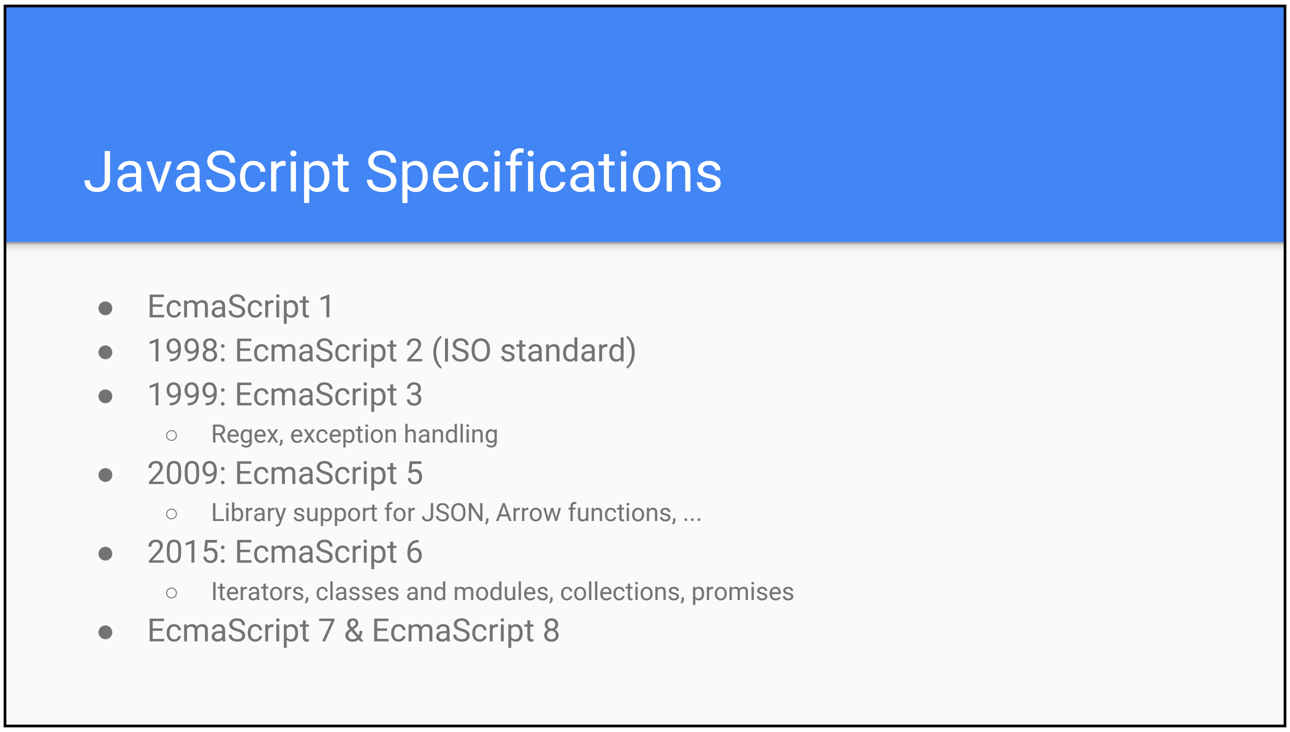
Java

- Created by Sun Microsystems
- 1991 initiated Java Project (Gosling cs)
- 1995 released Java 1.0
- Browsers support for Java Applets
- 1998 Java 2.0
- Interpreted by Java Virtual Machine
- Object-oriented
- (2010 Oracle acquired Sun Microsystems)

JavaScript

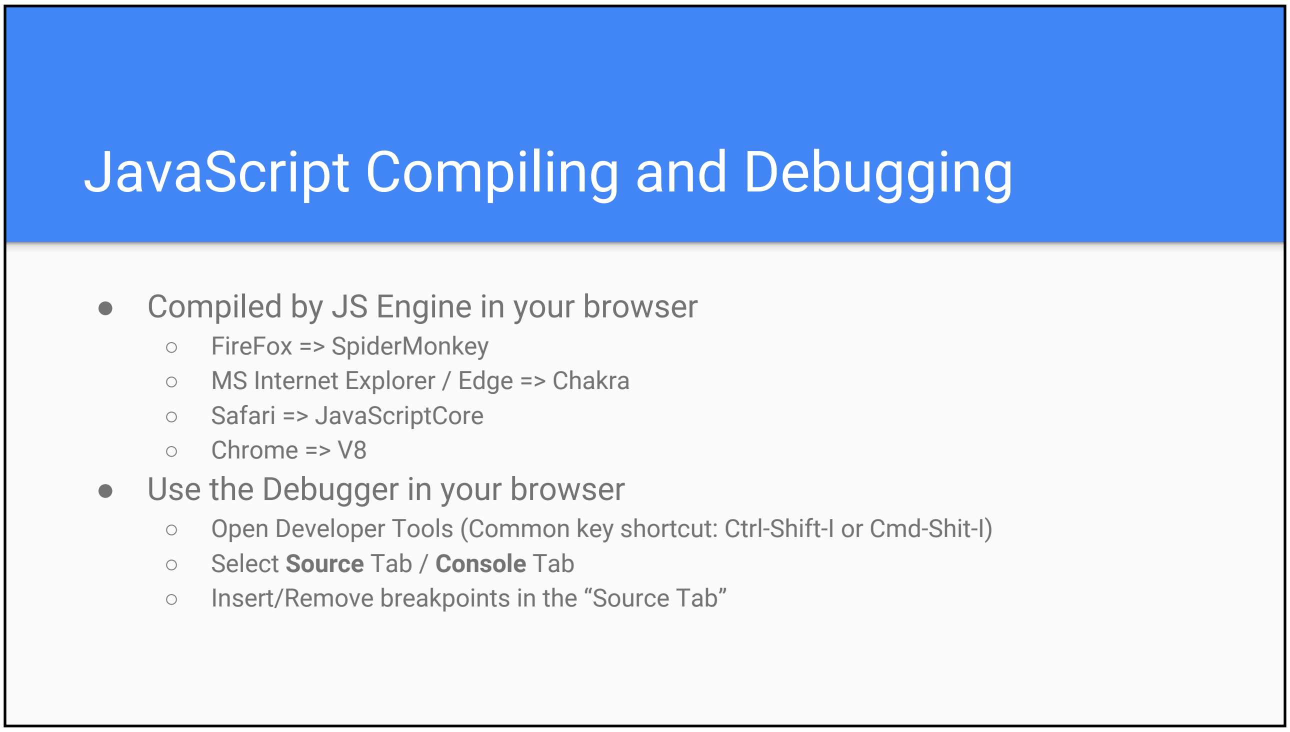
- Created by Netscape
- Goal: Run Scheme in Browser
- 1995 JavaScript was born
- Original name: Mocha ⇒ LiveScript
- Interpreted by JavaScript Engine
- Object-oriented

Common goals: run programs on a browser



JavaScript Specifications

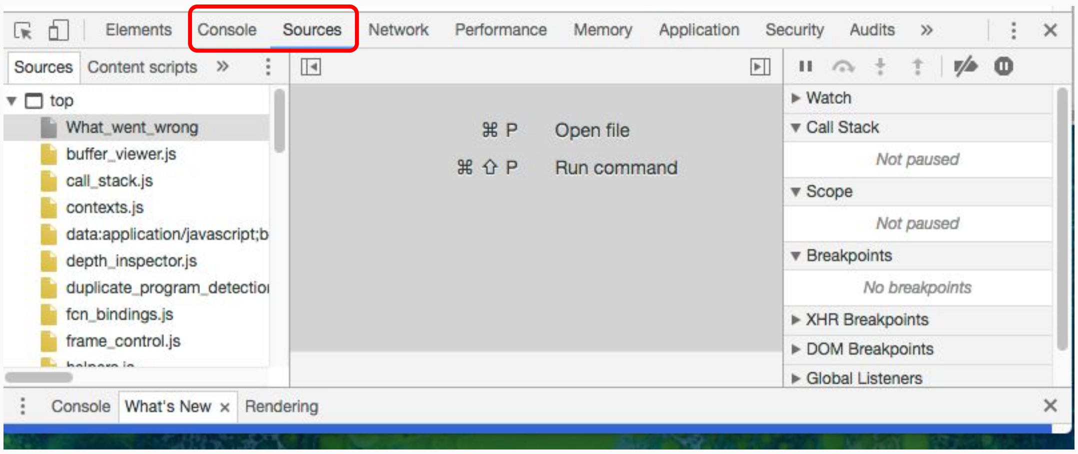
- EcmaScript 1
- 1998: EcmaScript 2 (ISO standard)
- 1999: EcmaScript 3
  - Regex, exception handling
- 2009: EcmaScript 5
  - Library support for JSON, Arrow functions, ...
- 2015: EcmaScript 6
  - Iterators, classes and modules, collections, promises
- EcmaScript 7 & EcmaScript 8



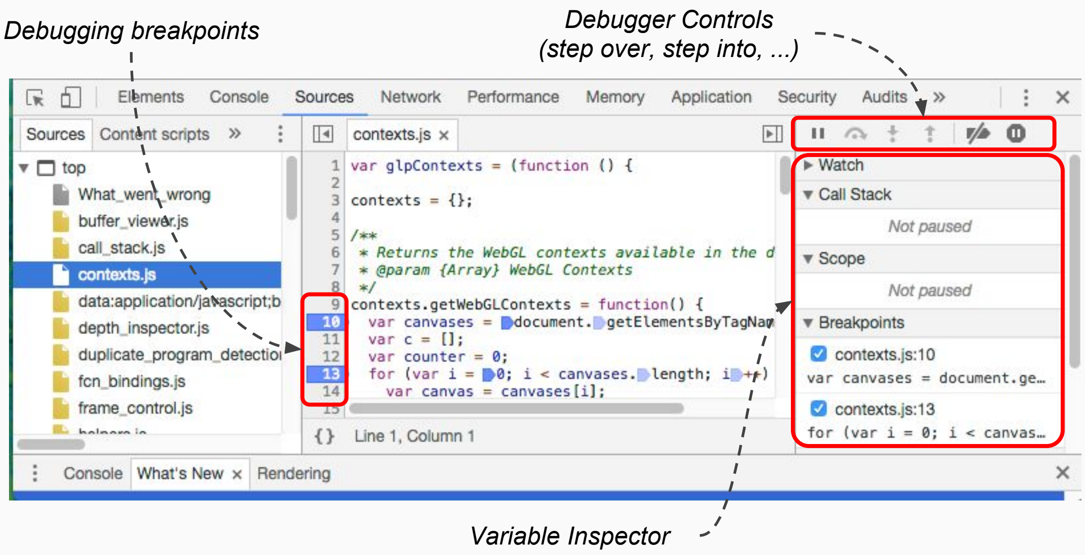
JavaScript Compiling and Debugging

- Compiled by JS Engine in your browser
  - FireFox => SpiderMonkey
  - MS Internet Explorer / Edge => Chakra
  - Safari => JavaScriptCore
  - Chrome => V8
- Use the Debugger in your browser
  - Open Developer Tools (Common key shortcut: Ctrl-Shift-I or Cmd-Shit-I)
  - Select **Source** Tab / **Console** Tab
  - Insert/Remove breakpoints in the “Source Tab”

Browser Developer Tool (Chrome)



Browser Debugger (Chrome)



```
var glpContexts = (function () {

contexts = {};

/**
 * Returns the WebGL contexts available in the d
 * @param {Array} WebGL Contexts
 */
contexts.getWebGLContexts = function() {
  var canvases = document.getElementsByTagNam
  var c = [];
  var counter = 0;
  for (var i = 0; i < canvases.length; i++)
    var canvas = canvases[i];
```

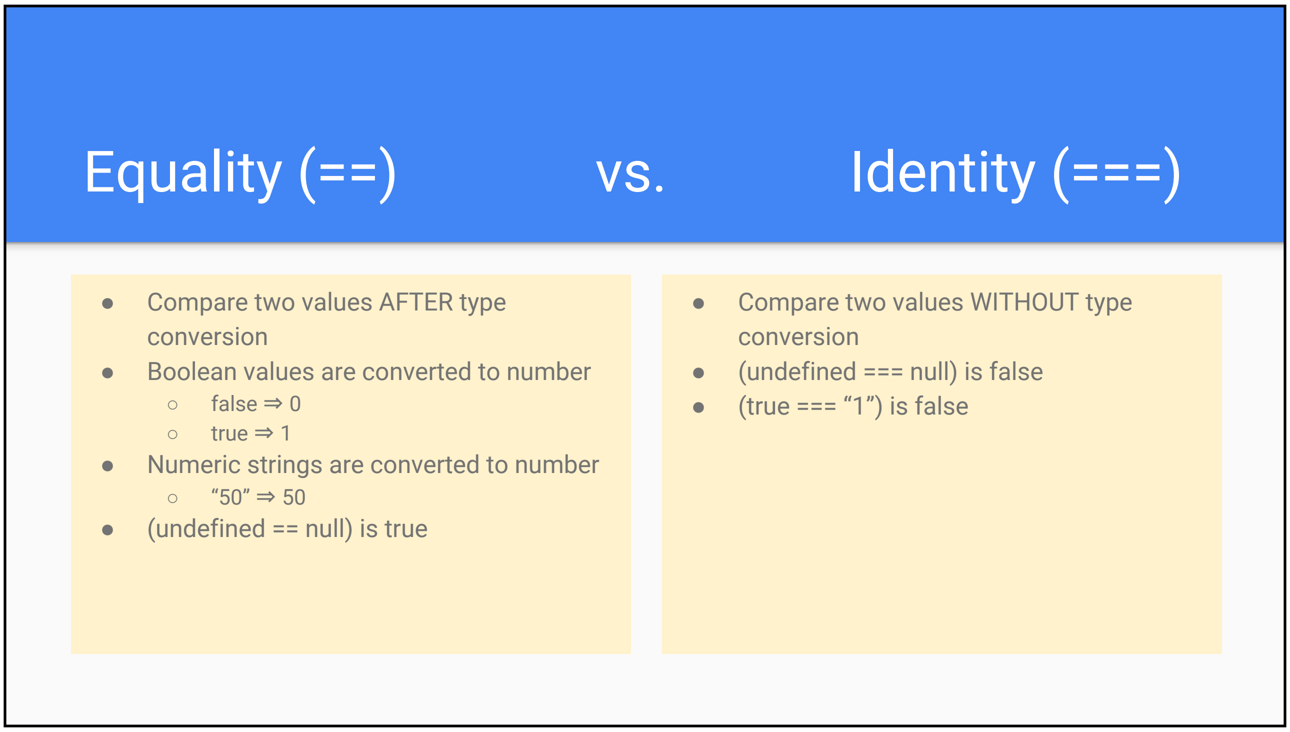
```
var canvases = document.ge…
for (var i = 0; i < canvas…
```

Disclaimer

These slides are not a full course on JavaScript.
They only highlight major differences between Java and JavaScript

JavaScript vs. Java

- Share similar keywords (break, if, for, return, ...)
- Similar syntax (except function declarations)
- Both languages have Garbage Collector
- JavaScript keywords / operators not in Java
  - `undefined`, `function`, `typeof`
  - `===`, `!==` (testing equality without type conversion)

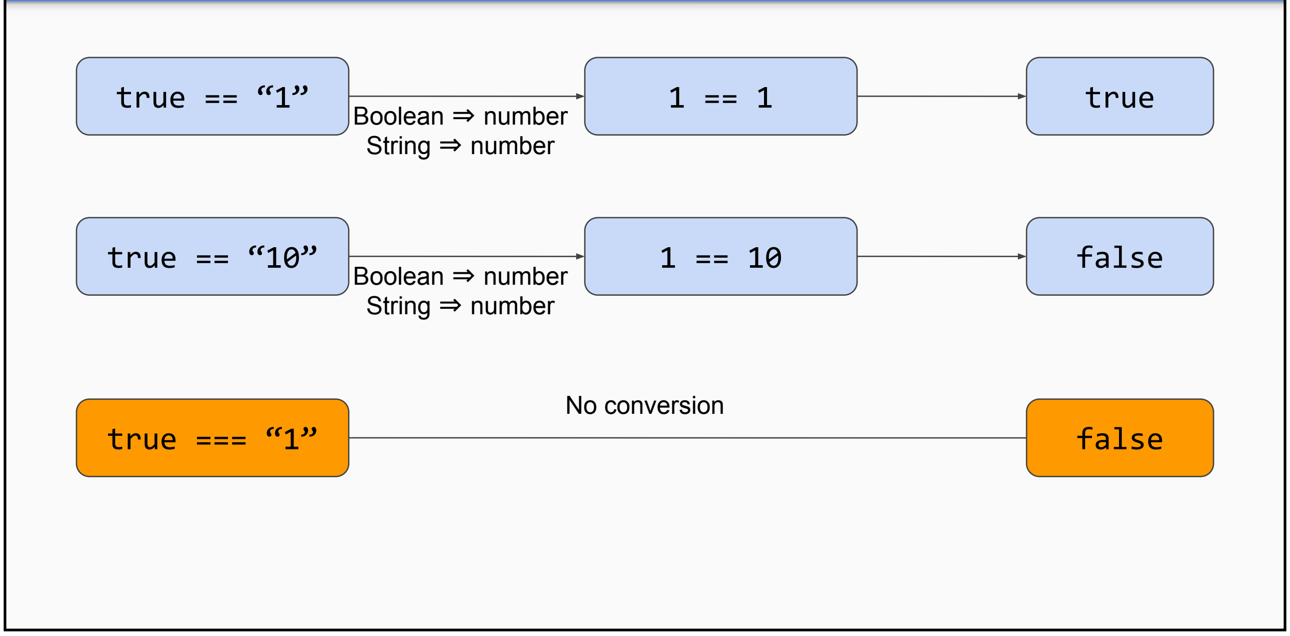


Equality (==) vs. Identity (===)

- Compare two values AFTER type conversion
- Boolean values are converted to number
  - false ⇒ 0
  - true ⇒ 1
- Numeric strings are converted to number
  - “50” ⇒ 50
- (undefined == null) is true

- Compare two values WITHOUT type conversion
- (undefined === null) is false
- (true === “1”) is false

Equality Conversion



Java vs. JavaScript

Java

- Integer division: 4/8 is 0
- `‘x’ // a single character`
- `“Coffee” // a string`

JavaScript

- No integer division, i.e. 4/8 is 0.5
- ‘Coffee’ or “Coffee” (both are strings)

JavaScript Data Type

- Number
- Boolean
- String
- Array (*associative arrays*) [2, 30, 12] [“Ann”, “Beth”, “Cindy”]
- Date
- Object
  `{ pages: 245, author: “Smith”, yearPublished: 2012 }`
- Function
  `function nameOfFun (x, y) { return x + y; }`

typeof

```
typeof 24.5    // is “number”    typeof {x: 24.5} // is “object”
typeof 245     // is “number”    typeof null      // is “object”
typeof “24.5”  // is “string”    typeof [1,4,11]  // is “object”
typeof true    // is “boolean”
var x;
typeof x  // is “undefined”
x = 24;
typeof x  // is “number”
```

var **quiz** = (typeof undefined == type of null);

Is **quiz** true or false?

for-in

```
var book = {
  pages: 245,
  author: “Smith”,
  yearPublished: 2012
};

for (var prop in book) {
  console.log(prop);
}
```

```
// Output
pages
author
yearPublished
```

Using JavaScript (with HTML)

- Manipulate Nodes (HTML elements / contents) in a DOM Tree
  - CRUD (Create, Read, Update, Delete) operations (Elements|Contents)
  - CRUD operations to element attributes
- Adding JavaScript program to an HTML page
  - Intenal: `<script> /* lines of code go here */ </script>`
  - External: `<script src=”url/to/your/script/here.js”></script>`
  - May add more than one <script> tags in a page

Local vs Global Variables

Local Variables

- Declared within a JS function
- Can only be accessed within the function
- Created when the function starts and deleted when the function returns

Global Variables

- Declared outside a function
- Visible to ALL scripts and functions on a web page

Minimize use of global variables (source of bugs)

Automatic Global Variable

Assigning a value to undeclared variables automatically makes them global

```
function sample() {
  var page = 200;     // page is a LOCAL var

  // totalCount is not previously declared
  totalCount = 400;  // it automatically becomes GLOBAL
}
```

JS Functions

```
// parameters are declared
// without type
function addThem (a, b) {
   return a + b;
}

// invocation
addThem (10, 20.4);
```

```
// Using Arrow Expressions
var sum = (a, b) => { return a + b; }

// invoke
sum (10, 20.4);
```

`(a, b) => { return a + b; }`

is actually an **anonymous** function

Variable Number of Arguments

```
// Java: three dots
public void sample(int... args)
{
  // args is an array of ints
  System.out.println (args.length);
}

sample(2, 5, 10, -11);  // output 4
sample(2, 5);           // output 2
sample();               // output 0
```

```
// JavaScript: special variable
// arguments
function sample()
{
  console.log(arguments.length);
}

sample(2, 5, 10, -11);  // output 4
sample(2, 5);           // output 2
sample();               // output 0
```

String Operations

String Interpolation (backquotes)

```
var x = “Eleven”;
var arr = [3, 7, 13];

var where = `${arr[1]}-${x}`;  // 7-Eleven
```

backquotes

Number operations

Array Operations

Spread operator: “unpack” an array

```
...arr
```

```
...[4, 11, 6, 20] ⇒ 4, 11, 6, 20
```

Spread operator (...):

1. Combine array
   - `arr1.push(...arr2);    // add arr2 items to the end of arr1`
   - `arr1.unshift(...arr2); // add arr2 items to the front of arr1`
2. Copying array

   -
     ```
     arr1 = [20, 11, 36];
     arr2 = arr1;           // arr1 and arr2 refer to the same data
     arr3 = [...arr1];      // arr3 is a copy of arr1
     ```
3. Convert NodeList to Array
   - `listItems = [...document.querySelectorAll(‘li’)];`

JS Objects = Associative Arrays

```
var book = { pages: 245, author: “Smith”, published: 2012 };

// add a new property (price) using “dot”
book.price = 174.97;
// add a new property (ebook_avail) using associative array
book[“ebook_avail”] = true;

console.log(book.ebook_avail);
console.log(book[“ebook_avail”]);
```

JavaScript Objects

```
var myBook = {
  author: “Smith”,
  pages: 245,
  published: 2015,
  toc: [
    “Introduction”,
    “Ready to Go Offline?”,
    “Conclusion”
  ],
  ebook_avail: true
}
```

```
var hisBook = {
  author: “Smith”,
  pages: 245,
  published: 2015,
  chapters: [
    { sub: “Introduction”, page: 23 },
    { sub: “Ready to Go Offline?”, page: 50 },
    { sub: “Conclusion”, page: 117 }
  ],
  ebook_avail: true
}
```

```
console.log(myBook.toc[0]);                 // “Introduction”

console.log(hisBook.chapters[0].page);   // 23
```

Objects & Arrays may include functions

```
var myBook = {
  author: “Smith”,
  pages: 245,
  published: 2015,

  addPage: function (numpg)
  {
    this.pages += numpg;
  },
  ebook_avail: true
}
```

```
// Add a new function def
myBook.delPage = function (numpg) {
  this.pages -= numpg;
};

// Or using arrow oper
myBook.delPage = (numpg) => {
  this.pages -= numpg;
};
```

JS (Predefined) Objects

- `document`: the current HTML document that hosts the script
  - Provides functions for manipulating the DOM tree
- `window`: the current window where the HTML doc is rendered
- `console`: the browser console window (mostly for debugging)

Common Methods

- `window.alert()` shows a windowed message
- `window.prompt()` shows an input dialog
- `console.clear()` clears the JS console output
- `console.log()` prints logging information
- `console.error()` prints error messages
- `console.warn()` prints warning messages
- References

Document and Element Methods

Methods for manipulating the DOM tree

- `document.getElementById()`
- `document.getElementsByClassName()`
- `document.getElementsByTagName()`
- `document.querySelector()`
- `document.querySelectorAll()`
- HTML DOM APIs, Document APIs and Element APIs

Manipulating HTML: contents, styles, and class

```
<!-- HTML doc -->
<body>
  <div id=”container”></div>
</body>
```

```
// JavaScript
var elem = document.getElementById(“container”);
elem.innerHTML = “<b>Hello</b>”;
elem.style.background = “red”;          // NOT RECOMMENDED!!!
elem.className += “filled”;
```

```
// JavaScript
var elems = document.getElementsByTagName(“div”);
elems[0].innerHTML = “<b>Hello</b>”;
elems[0].style.background = “red”;      // NOT RECOMMENDED!!!
elems[0].className += “filled”;
```

```
// JavaScript
var elems = document.querySelectorAll(“body > div”);
elems[0].innerHTML = “<b>Hello</b>”;
elems[0].style.background = “red”;      // NOT RECOMMENDED!!!
elems[0].className += “filled”;
```

innerHTML

Document APIs
Element APIs

.innerHTML
vs.
Document.createElement() +
Element.appendChild()

JavaScript Events

- Window events: onload, onresize, onunload
- Document events: ondblclick, onkeydown, onkeyup, onmousedown, onmouseup
- Text element events: onblur, onfocus
- Button events: onclick, ondblclick, onmousedown, onmouseup
- Link events: onclick, ondblclick, onmouseout, onmouseover
- Image events: onabort, ondblclick, onkeydown, onkeyup, onmousedown, onmouseup,
- Complete Reference: Event APIs

Setting Up Event Handler

- Which Event?
- Which object events are delivered to?
  - Resize => window
  - Key presses => document
  - Load => document
  - Click => button, image, ….
  - Focus => input elements
- Details of the event properties

Example: Setting Up a Keyboard Event Handler

```
document.onkeydown = function(event) {   // traditional “function” syntax (pre ES2015)
  // your code here
};
```

```
document.onkeydown = event => {
  switch (event.key) {
    case “ArrowLeft”:
      /* code here */

      break;
    case “ArrowLeft”:
      /* code here */

      break;
  }
};
```

```
// “keydown”, and NOT “onkeydown”
document.addEventListener(“keydown”,
  event => {
      /* event handling logic here */
  }
);
```