CODEPATH*ORG

LEARN. BUILD. CHANGE THE WORLD.

JavaScript & the DOM

Week 1 Day 2 - June 7, 2022 - Doug Case

Week 1 Day 2 Goals

- Use JavaScript to add functionality to a website via interactive elements
- Explain JavaScript’s role in, and significance to, the internet
- Use JavaScript to manipulate a browser’s Document Object Model

JavaScript - What & Why

- Popular programming language first developed for Netscape Navigator in 1994
- Other browsers like IE, etc, followed
- Initially JS development painful due to browser-specific JS (and DOM)
- Much better now - standardized since ECMAScript 2015 (aka ES6)
- Despite the name not directly related to Java
- Most front-end (browser) code is JavaScript
- ReactJS and Angular have emerged as favorite JS frameworks
- ReactJS by far most popular JS framework since around 2018

Chrome debugger is effectively the standard

- Chrome: View -> Developer -> Developer Tools
- Console tab is good for looking at errors, and console.log messages

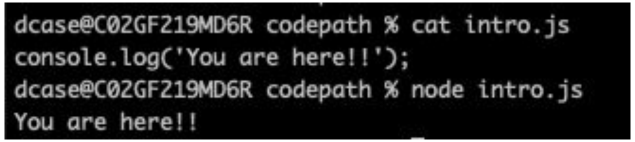
- Chrome: right mouse - View Page Source, or
- Chrome: View -> Developer -> View Source

- Amusing comment in source at
  https://developers.themoviedb.org/4/getting-started/authorization

CODEPATH*ORG

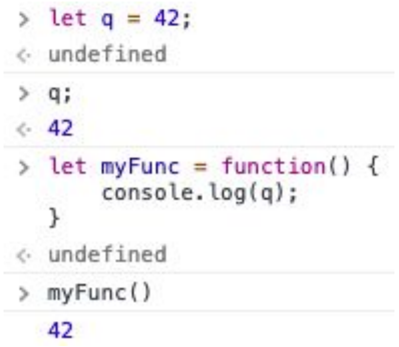
Ways to run JavaScript

- (1) Install node.js (via npm - or yarn or brew on Mac). Then “node <filename.js>”



```
dcase@C02GF219MD6R codepath % cat intro.js
console.log('You are here!!');
dcase@C02GF219MD6R codepath % node intro.js
You are here!!
```

- (2) Chrome: View -> Developer -> Developer Tools -> Console



```
> let q = 42;
< undefined
> q;
< 42
> let myFunc = function() {
      console.log(q);
  }
< undefined
> myFunc()
  42
```

- (3) Run JavaScript inside a browser

Coding standards - variable naming conventions

- Python uses snake case for variable name
  - Example: network_id or num_students_enrolled
- C/C++/Java/JavaScript use camel case for variable names
  - Example: networkId or numStudentsEnrolled
- Exception: all of above languages constants
  - Example: const MAX_NUM_STUDENTS = 40;

- Generally HTML indents by two spaces, other languages by 4
- Key is to be consistent with pre-existing code, company
- Story: networkId vs networkid vs networkID…
- Story: early 2000s no consensus indent 2 vs 4 spaces
- Moral of stories: be consistent w/ others on tools, etc

var vs let vs const

- const MAX_STUDENTS = 40;
- var is old, let is new in ES6 and generally considered better

try let vs var in for loop below

```
//for (let i = 0; i < 5; i++) {
for (var i = 0; i < 5; i++) {
    console.log(i);
}
console.log("Now i is: ", i);
```

JavaScript: “Variable hoisting” - pre-processing looks for var and
lifts it to start of document. Using let prevent this. And
var may “overlay” variables if same var name already used, but let won’t.

JS objects - example

```
let obj = {
  key: "valuuuuue",
  second: 2,
  third: “THREE”,
  four: {          // nested object
    five: 5,
    six: "SIXX"
  },
  eight:          // array
   [
    {
      color: "red"
    },{
      color: "blue"
    }
   ]
}
console.log(obj.third);       // THREE
console.log(obj.four.five); // 5
console.log(obj.eight);      // [ { color: 'red' }, { color: 'blue' } ]
console.log(obj.eight[1].color);    // blue
```

CODEPATH*ORG

JS functions - 3 examples

```
// this example has parameters, other examples do not
let myFunc = function(parameter1, parameter2) {
    console.log(“whatever”);
}

var myFunc2 = function() {
    console.log(“whatever”);
}

// this one returns a value
function myFunc3 = () {
    return 3;
}
```

JS functions - arrow functions, and callbacks

```
// arrow functions new in ES6, allow passing function to another function
let printMe = (arg) => {console.log(arg);}


// Why would you pass a function to a function? Used for a callback function.
// forEach is a predefined callback function
let arr = [11, 12, 13, 14, 15];
arr.forEach(printMe);
```

JS methods

```
// functions are a data type

let myObj = {
   key1: "valuuuuue",
   key2: 2,
   funk: () => { console.log(“Go Steph Curry!”);}
}
console.log(myObj.funk());

// toUpperCase() is a method for string objects
let str = “I am hungry”;
let upper = str.toUpperCase();

console.log(upper);


// JS is an Object-Oriented language - object-based
```

CODEPATH*ORG

JS to manipulate DOM (elements on web site):

```
let first = document.getElementById("image1");
let third = document.getElementById("image3");
let button = document.getElementById("clicker");

// addEventListener method has 2 parameters:
// one is event to listen for, second is callback function
first.addEventListener("mouseover", () => {
    first.style.display = "none";
    third.style.display = "block";
})

third.addEventListener("mouseover", () => {
    third.style.display = "none";
    first.style.display = "block";
})

button.addEventListener("click", () => {
    console.log("I felt a click");
})
```

JS Equality

JS transforms in equality check
(7 == “7”)   // evaluates to true

JS also has strict equality comparison operator which returns false for the values which are not similar type
(7 === “7”)   // evaluates to false

Happy Coding (after break)!

CODEPATH*ORG