Int. JavaScript (JS) Intro. jQuery

NOT YOUR PARENTAL FIGURE'S JAVASCRIPT (AND JQUERY)

MARK BROWN II & NATASSJA LINZAU (TA)

Featuring Beyonce

Who IS Mark Brown II?

Front-End Developer

Department of Commerce

Commerce Data Service

Lover of NodeJS

Practitioner of Crossfit

Student of Higher-Education Policy

Worshipper of Beyonce

Commerce Data Academy

A data education initiative of the Commerce Data Service (CDS).

• Launched by the CDS to offer data science, data engineering, and web development training to employees of the U.S. Department of Commerce.

• Course schedule and materials (e.g. slides, code, papers) produced for the Commerce Data Academy can be found on our website at http://dataacademy.commerce.gov .

• Questions? Feel free to write us at Data Academy (dataacademy@doc.gov)

Course Files

Today's code can be found at:

https://github.com/maabrown/IntermediateJSClass

GOALS

Review introductory JS concepts:

- Data Types
- Arrays vs Objects
- Looping (Array and Object)

Write Functions

HTML/CSS Interaction

Introductory jQuery

MAKE A FRIEND!!

Informative Conversation Statements

THERE ARE NO STUPID QUESTIONS!

ASK QUESTIONS OFTEN!

PLEASE CODE-ALONG WITH ME!

I ❤️ QUESTIONS!

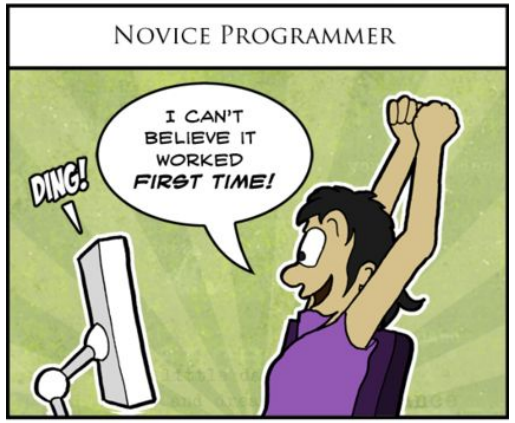
Our Agreement *(Repeat After Me)*

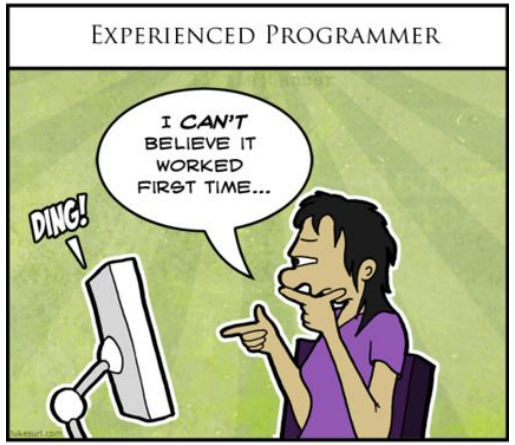
I am going to make mistakes.

I am going to get things wrong.

I am going to fail.

This is okay!

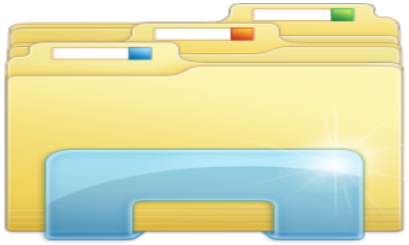




LETS GET TO IT!

Getting Started

- Download code from repo https://github.com/maabrown/IntermediateJSClass
- Find folder on your machine using ‘Finder’ (Mac) or ‘Windows Explorer’ (PC)
- Open ‘ReviewMaterials’ folder (directory) in Code editor – Sublime, Atom, or other of choosing
- MAC
  - Drag icon of ’index.html’ file to your web browser icon
- PC
  - Double-click index.html (or index)

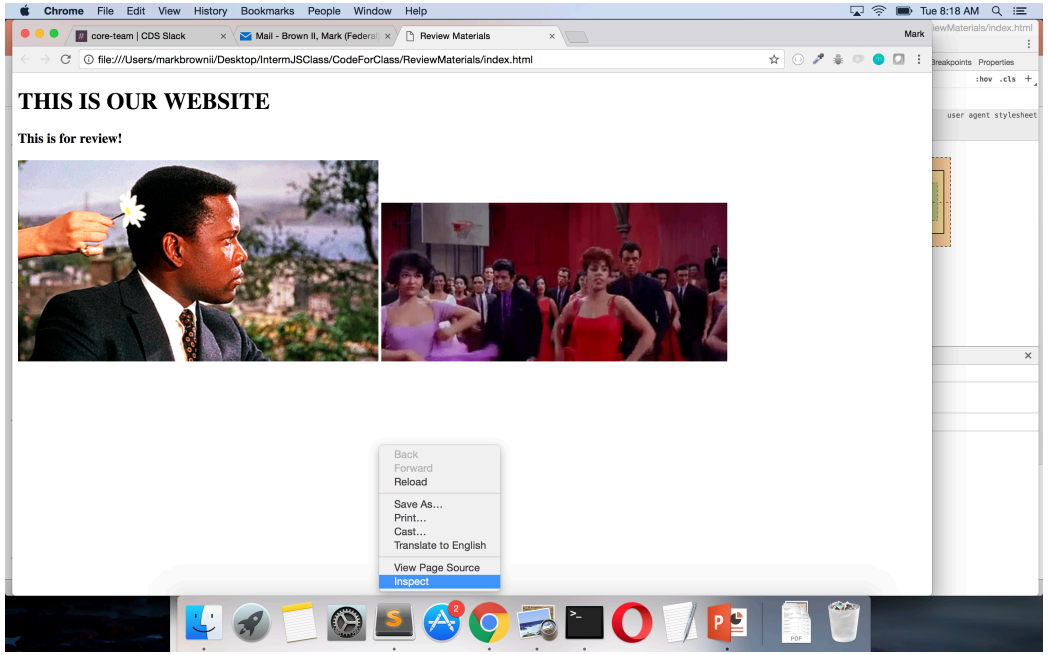


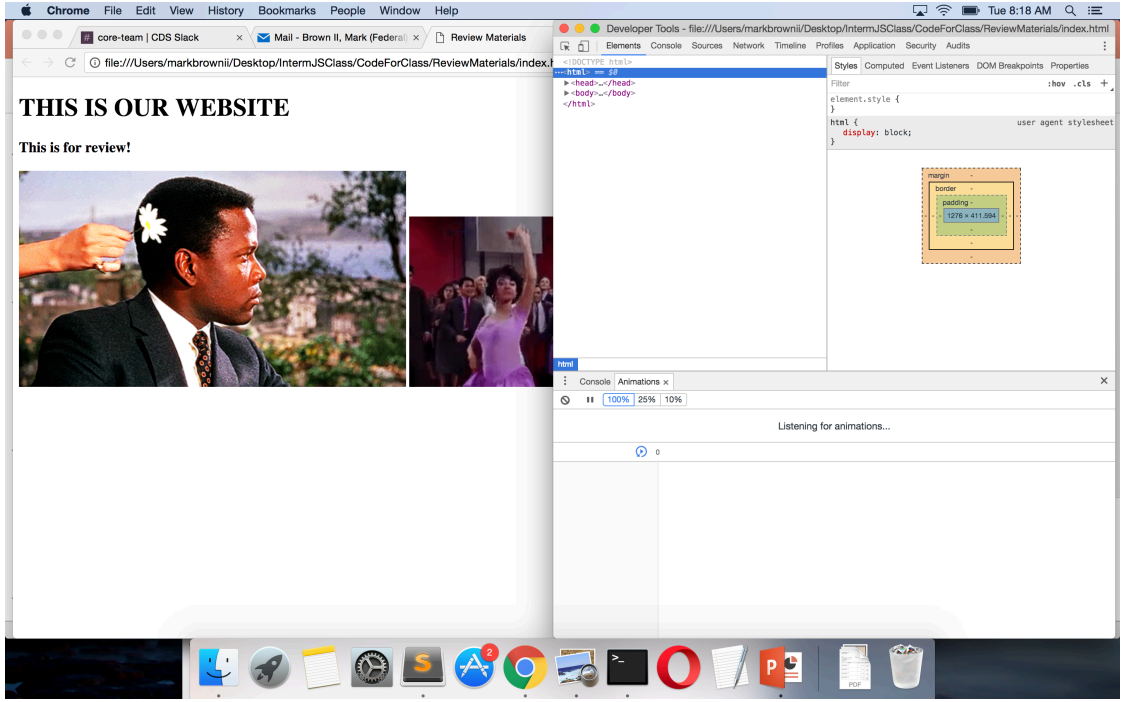
THIS IS OUR WEBSITE

This is for review!

Developer Tools (on Chrome)

- Right click page
- Select Inspect





Data Types

What is a data type?

- It is a classification identifying one of various types of data

7 Types of Data:

- String
- Number (Integer and Float)
- Boolean
- Array
- Object
- Null
- Undefined

STRINGS

Textual data

MUST Surrounded by double (sometimes single) quotes

Consists of characters

- Spaces *are* characters
- Zero-indexed (which includes spaces)

Examples:

- ”Hello”
- “5”
- “{ key: value }”
- “kittens”

NUMBERS - INTEGERS

Whole numbers – no decimals

Does not have a delimiter (a character that is used as a boundary in a data stream) – e.g. “,”

NaN is a number

Examples:

- 42
- -3
- 10000000000000000

NUMBERS - FLOATS

Numbers *with* decimals

No delimiter

Examples:

- 4.5
- -1.333333333
- 5968332244.33324
- 3.141592

BOOLEAN – NULL - UNDEFINED

Boolean

- Values of ‘true’ or ‘false’

Undefined

- Variable without a value

Null

- Essentially something that doesn’t exist

ARRAY

Ordered collection of data

- Zero-indexed

Can have mixed data types

Technically is an ‘object’ data type

Delimiters used:

- [] – to open and close an array
- , - to separate items within an array

To get the value in an array:

- arrayName[index]

- Examples:
  - [3, 2, “hello”, {}, 4.555555, true] ← One array
  - [“hello”], [37, 91] ← Two arrays
  - [[3,2,”hello”], 45, “laugh”] ← One array but an array within an array

ARRAYS

- Data can be added, deleted, and changed
- Some methods:
  - Reverse()
  - toString()
  - length (property – not a method)
  - push() – add a value to an array

- var names = [‘Beyonce’, ‘Bey’, ‘Yonce’];
- names[0] = ‘Beyonce’
- names[2] = ?

- var sailorNames = [‘Moon’, ‘Jupiter’, ‘Mars’, ’Venus’, ‘Mercury’];
- sailorNames[1] = ?
- sailorNames[2] = ?
- sailorNames[3][0] = ?
- sailorNames[1][6] = ?

OBJECT

Unordered collection of data

Can have mixed data types

Key-value pairs

- KEY: VALUE

Delimiters used:

- {} – used to open and close objects
- : - used to separate key from value
- , - used to separate different objects and key-value pairs in objects

Examples:

- {name: ”Beyonce”, num_of_awards: 195}
- {albums: [‘Bday’, ’Dangerously in Love’, ‘Lemonade’]}

OBJECTS

- Data can be added, deleted, and changed
- Order does not matter - indexed by the ‘key’
- Access values 2 ways:
  - *objectName[key] = value*
    - newObject[name] = ”Mark Brown II”
    - newObject[“position”] = ?
  - *objectName.key = value*
    - newObject.employer = ?

- var newObject = {

  name: ”Mark Brown II”,

  position: “ Developer”,

  employer: “Department of Commerce”,

  “love of life” : “Beyonce”

  }

- Use “ “ when key has number or other special characters
- Use same notation to add key/value to object
  - newObject.hair = “natural” OR
  - newObject[‘hair’] = ”natural”

OPERATORS

Know most from math

- -
  - Subtracts numbers
- /
  - Divides numbers
- *
  - Multiplies numbers
- +
  - Add numbers, concatenates Strings
- %
  - Returns the remainder from the division

Console.log - typeof

Console.log()

- Is a function
- Returns the value given to the function
- Put the computation you want in between the parentheses -- console.log(“hello”);

typeof

- Is an operator
- Tells you the data type of an input
- Type ‘typeof’ – give it a space – put the value you want to evaluate -- typeof {}

POP QUIZ!!

1. What are the differences between Integers and Floats?

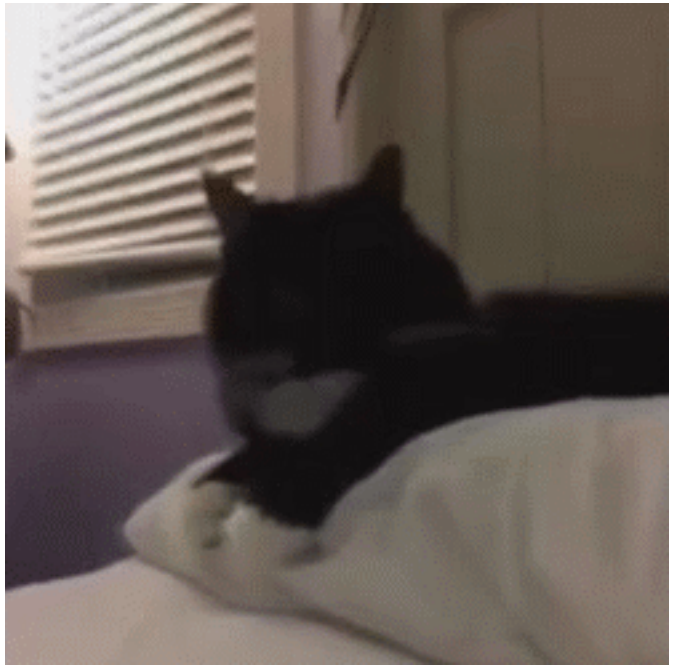
2. Is “5” the same as 5?

3. ‘True’ and ‘False’ are what type of data?

4. What are the differences between an array and an object?

5. True/False: I can have mixed data types in my array.

6. How can I check a datatype using my computer?



VARIABLES

Used to store information to computer's memory for future use

Declaring

- Always use keyword 'var'
- var myName

Assigning

- Can assign variables any value – including arrays, objects, functions
- var myName = "Mark";
- Always use a ";"

Calling

- How you access the variable's value
- Use variable name – not the value

Returns 'undefined' if variable isn't assigned a value

PRACTICE!

```
var practiceArray = [
        "JavaScript",
        "Ruby on Rails",
        "C++",
        "AngularJS"
 ]
```

1. What is practiceArray[2]?
2. How would I get the value of 'AngularJS?
3. How would I get the length of practiceArray?

```
var practiceObject = {
        develop: "web",
        language: "JavaScript",
        dataType: "object",
        bureaus: [
            "BEA", "NOAA", "NIST", "MBDA"
        ]

 }
```

1. What are the keys in this object?
2. What is the value of practiceObject["dataType"]?
3. How would I get the value 'MBDA'?

REPEAT AFTER ME!

ALL ARRAYS *ARE* OBJECTS

ALL OBJECTS ARE *NOT* ARRAYS

QUESTIONS????

LOOPING (aka ‘For Loops’) - ARRAY

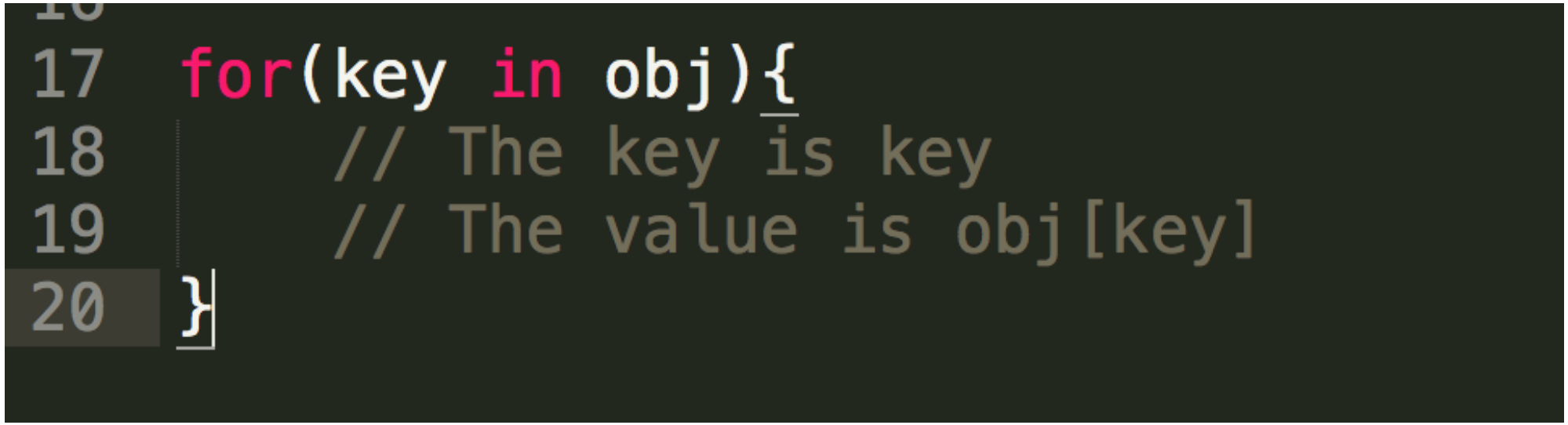
```
3 for (var i = 0; i < array.length; i++) {
4     //DO SOMETHING WITH ARRAY
5 }
```

Incrementors:

- ++ increments by 1
- -- decreases by 1

”i” is just a counter

LOOPING (aka ‘For-In’) - OBJECT



PRACTICE

Go to ReviewCode directory/folder

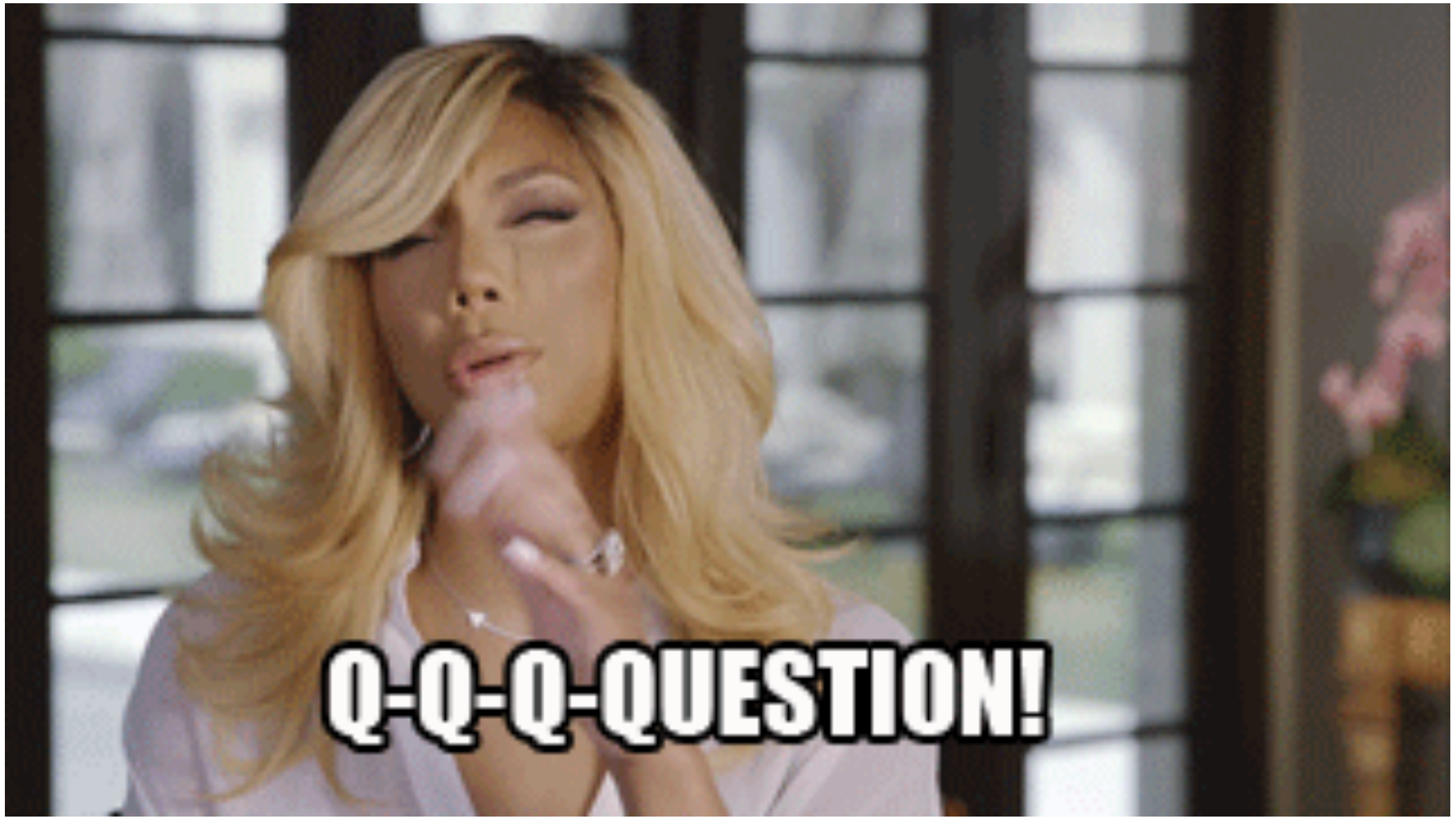
Open script.js

1. Write a ‘for’ loop that logs every item in the destinyAlbums array.

2. Write a ‘for’ loop that logs every key in the whitneyAlbums object.

3. Write a ‘for’ loop that logs every value in the whitneyAlbums object.

SHOW/TELL ME YOUR CODE!!!


Q-Q-Q-QUESTION!

BREAK! – 5 MIN

FUNCTIONS

Reusable statement or group of statements

Can be called anywhere within the program

DRY – Don't repeat yourself

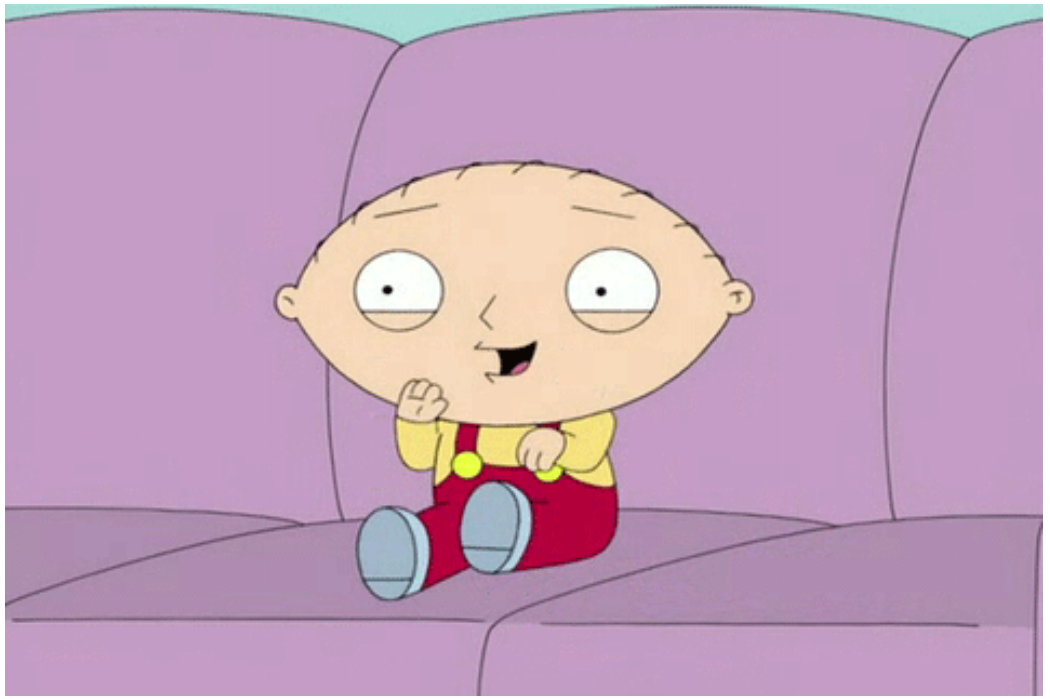
Use keyword 'function'

Must be defined first → Called (usually) – when writing code

Technically – functions are *objects*

- Related to concept of 'first-class objects'

Reminder: JS works LINE BY LINE, TOP TO BOTTOM



FUNCTIONS

```
function learningJS(param1, param2) {
    console.log(param1 + " " + param2);
}
```

4 Parts:

- 1. Keyword ‘function’
- 2. Name of the function (Optional)
- 3. Parameters (Optional)
- 4. Code inside the function (Optional)

FUNCTIONS

DECLARATION

Names the function within the definition

```
32  function newFunction() {
33      // DO SOMETHING HERE
34  };
35
```

EXPRESSION

Declares variable to name the function

```
37  var newFunction = function() {
38      // DO SOMETHING HERE
39  };
40
```

FUNCTIONS – Calling

Functions are called by () directly after their defined name

```
function newFunction() {
    // CODE
};

newFunction();
```

```
function subtractNumbers() {
    console.log(3-1);
};

subtractNumbers();  // => 2
```

FUNCTIONS – Parameters

Write parameters in initial function definition

Uses those values in its computation

Can have an infinite number of parameters

Pass in values for the parameters to function when calling it

```
56
57  function giveMeMyName(firstName, lastName) {
58      console.log(firstName + " " + lastName);
59  };
60
61  giveMeMyName("Mark", "Brown II"); // => "Mark Brown II"
62
```

FUNCTIONS – Calling and Parameters

```
65
66  function addNumbers(number1, number2) {
67      console.log(number1 + number2);
68  };
69
70  addNumbers(2,4); // => 6
71
72
```

FUNCTIONS – Calling and Parameters

```
70
71  var oneNumber = 3;
72  var secondNumber = 7;
73
74  function addNumbers(number1, number2) {
75      console.log(number1 + number2);
76  };
77
78  addNumbers(oneNumber, secondNumber);
79  // What do you think is going to happen?
80
```

FUNCTIONS – Calling and Parameters

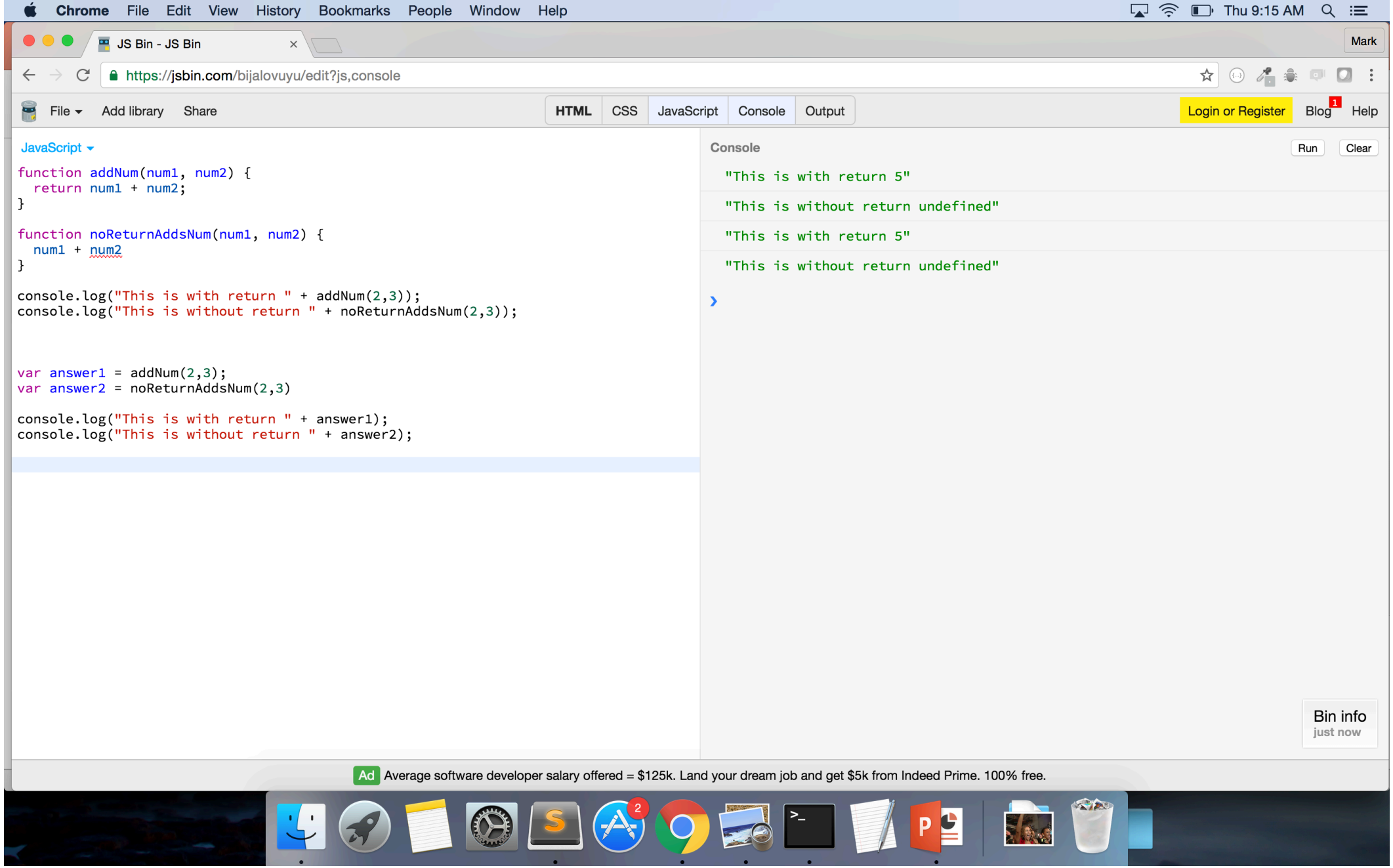
```
84
85  var oneNumber = 3;
86  var secondNumber = 7;
87
88  function addNumbers() {
89      console.log(oneNumber + secondNumber);
90  };
91
```

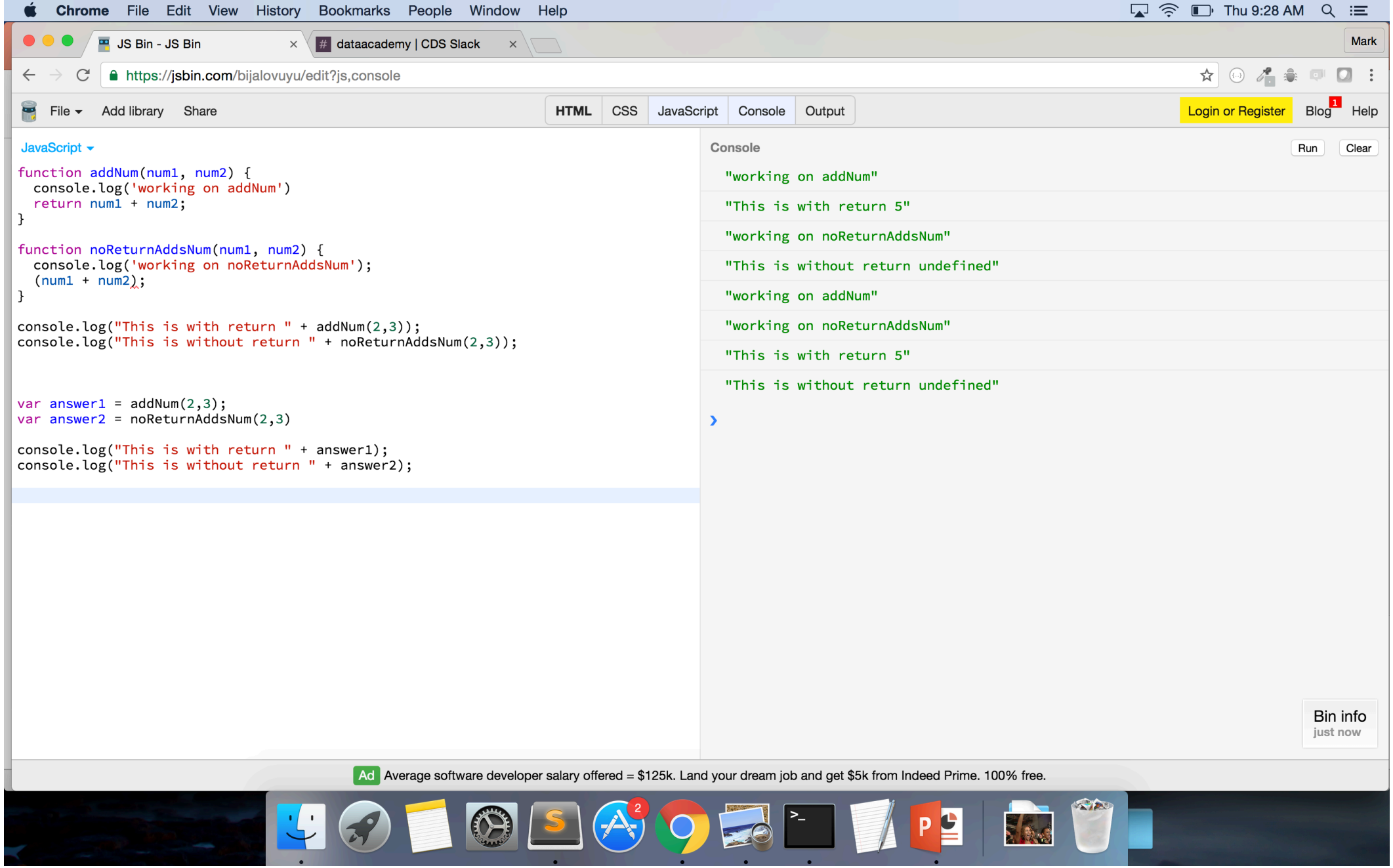
FUNCTIONS – keyword ‘return’

Returns the value of your function’s computation to the function’s moment of execution

‘return’ stops the function’s execution

```
91
92  var oneNumber = 3;
93  var secondNumber = 7;
94
95  function addNumbers() {
96      return oneNumber + secondNumber;
97  };
98
```


Chrome File Edit View History Bookmarks People Window Help
Thu 9:15 AM
JS Bin - JS Bin
Mark
https://jsbin.com/bijalovuyu/edit?js,console
File
Add library
Share
HTML
CSS
JavaScript
Console
Output
Login or Register
Blog
Help
JavaScript
function addNum(num1, num2) {
return num1 + num2;
}
function noReturnAddsNum(num1, num2) {
num1 + num2
}
console.log("This is with return " + addNum(2,3));
console.log("This is without return " + noReturnAddsNum(2,3));
var answer1 = addNum(2,3);
var answer2 = noReturnAddsNum(2,3)
console.log("This is with return " + answer1);
console.log("This is without return " + answer2);
Console
Run
Clear
"This is with return 5"
"This is without return undefined"
"This is with return 5"
"This is without return undefined"
Bin info
just now
Ad
Average software developer salary offered = $125k. Land your dream job and get $5k from Indeed Prime. 100% free.


Chrome File Edit View History Bookmarks People Window Help
Thu 9:28 AM
JS Bin - JS Bin
dataacademy | CDS Slack
Mark
https://jsbin.com/bijalovuyu/edit?js,console
File
Add library
Share
HTML
CSS
JavaScript
Console
Output
Login or Register
Blog
Help
JavaScript
function addNum(num1, num2) {
console.log('working on addNum')
return num1 + num2;
}
function noReturnAddsNum(num1, num2) {
console.log('working on noReturnAddsNum');
(num1 + num2);
}
console.log("This is with return " + addNum(2,3));
console.log("This is without return " + noReturnAddsNum(2,3));
var answer1 = addNum(2,3);
var answer2 = noReturnAddsNum(2,3)
console.log("This is with return " + answer1);
console.log("This is without return " + answer2);
Console
Run
Clear
"working on addNum"
"This is with return 5"
"working on noReturnAddsNum"
"This is without return undefined"
"working on addNum"
"working on noReturnAddsNum"
"This is with return 5"
"This is without return undefined"
Bin info
just now
Ad
Average software developer salary offered = $125k. Land your dream job and get $5k from Indeed Prime. 100% free.

FUNCTIONS – keyword ‘return’

```
 99
100  var oneNumber = 3;
101  var secondNumber = 7;
102
103  function addNumbers() {
104      return oneNumber + secondNumber;
105      console.log(oneNumber + 1) // THIS WILL NOT HAPPEN
106  }
107
```

JAVASCRIPT IS NATIVELY SYNCHRONOUS!!

FUNCTIONS - Synchronous

```
22
23  thisOldFunction();
24  thisNewerFunction();
25  thisEvenNewerFunction();
```

FUNCTIONS – keyword ‘return’

```
100  var oneNumber = 3;
101  var secondNumber = 7;
102
103  function addNumbers() {
104      var newAddition = oneNumber + secondNumber;
105      return newAddition;
106  };
107
```

FUNCTIONS

```
 99
100  var oneNumber = 3;
101  var secondNumber = 7;
102
103  function addNumbers() {
104      var newAddition = oneNumber + secondNumber;
105      return newAddition;
106  };
107
108
109  function subtractOne(value) {
110      return value - 1;
111  }
112
113  subtractOne(addNumbers());
114
```

FUNCTIONS – keyword ‘return

```
3  function addNum(num1,num2) {
4      console.log(num1 + num2);
5  };
6
7  function sqNum(value) {
8    console.log(value * value);
9  }
10
11 sqNum(addNum(5,3));
12
```

```
15 function addNum(num1,num2) {
16     return (num1 + num2);
17 };
18
19 function sqNum(value) {
20   console.log(value * value);
21 }
22
23 sqNum(addNum(5,3))
24
```

FUNCTIONS - Readability

Break down complex tasks

- Readability
- Lessens margin of human error
- Allows for code flexibility
- Allows for code reuse

```
108
109  var oneNumber = 3;
110  var secondNumber = 7;
111
112  function addNumbers() {
113      var newAddition = oneNumber + secondNumber;
114      return newAddition;
115  };
116
117
118  function subtractOne(value) {
119      return value - 1;
120  }
121
122  subtractOne(addNumbers());
```

```
167
168  function oneBigFunction() {
169      return ( (oneNumber + secondNumber) - 1 );
170  }
171
```

CODING OBSERVATION

Coding is FLEXIBLE!

Different code by *different* people can have *same* result!!

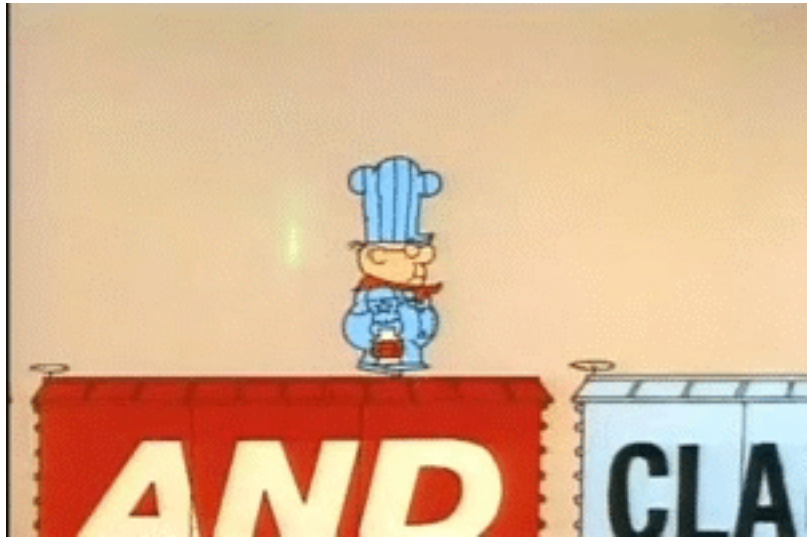
Not all code has to, or will, look alike!

FUNCTION PRACTICE! (with me)

1. Write a function that console.log's your name. (Remember to pass in a string!)

2. Write a function that will square any number passed into it.

3. Write a function that will take a value and add it to an array that is outside the function.



CODE-ALONG!

FUNCTION PRACTICE!

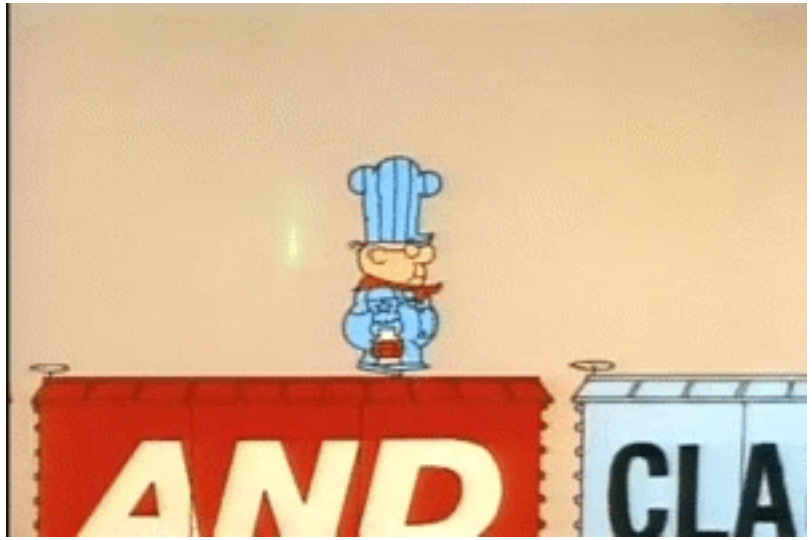
1. Write a function that will return the name of an animal you pass in.

2. Write a function that will divide the number you pass in by 5.

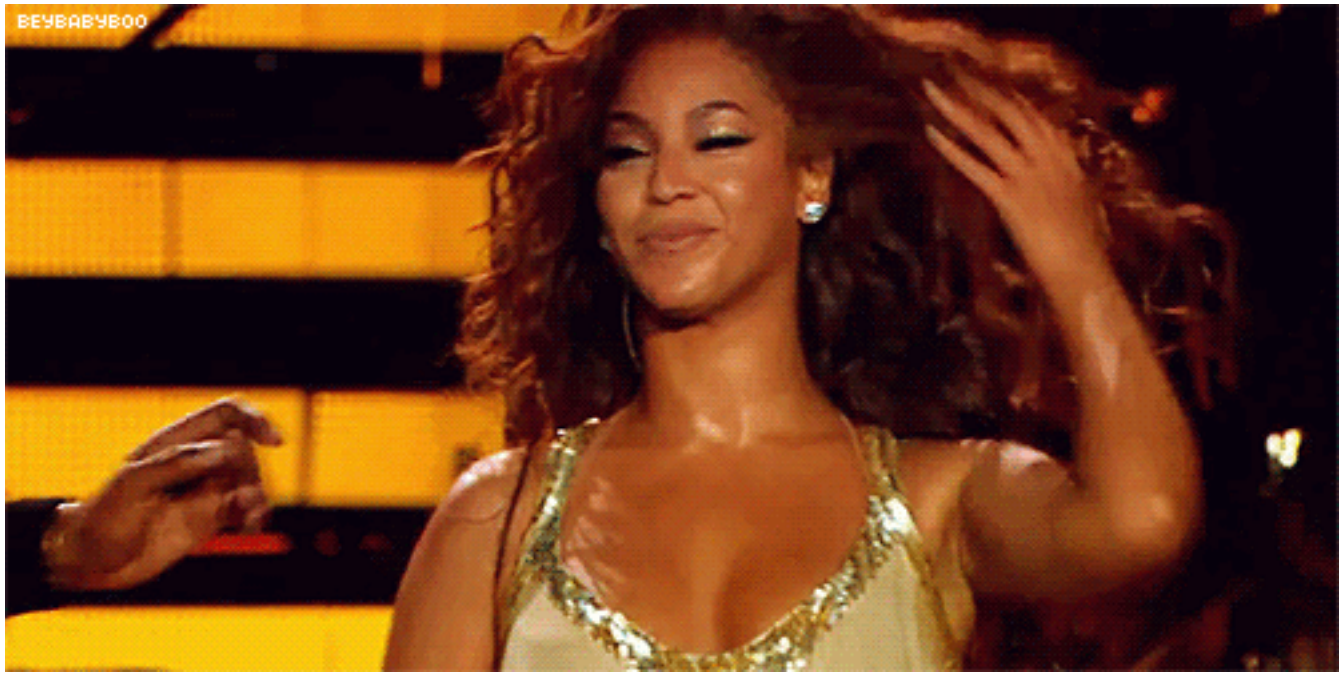
3. Write a function that will console.log the values of any object. Use object captainPlanet or whitneyAlbums to check your fxn.

4. Write a function that will add 1 to every element of an array that is passed in and put those new values in a new array. Use addMeUp to check your fxn.

5. Write a function that will tell me the dataType of every element in an array that is passed in. Use mixedUpTypes to check your fxn.



Remember – YOU CAN DO THIS!



FUNCTION PRACTICE!

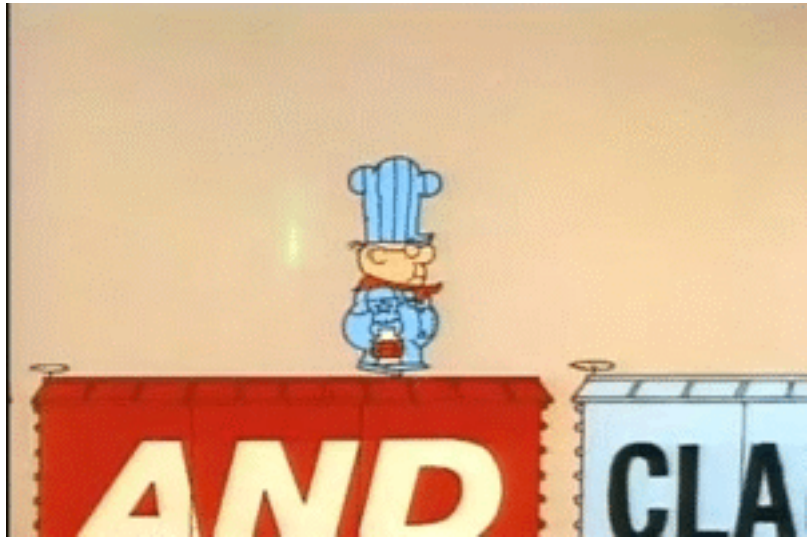
1. Write a function that will return the name of an animal you pass in.

2. Write a function that will divide the number you pass in by 5.

3. Write a function that will console.log the values of any object. Use object captainPlanet to check your fxn.

4. Write a function that will add 1 to every element of an array that is passed in. Use addMeUp to check your fxn.

5. Write a function that will tell me the dataType of every element in an array that is passed in. Use mixedUpTypes to check your fxn.



OMG POP QUIZ!

1. What's the difference between a float and integer?

2. True/False: You can only get the value from an object using dot.notation. *e.g. object.key = value*

3. What are 2 major differences between arrays and objects?

4. True/False: Functions must always take a parameter.

5. True/False: Less code is always better code.

6. What is the difference between console.log() and the keyword 'return'?

WHAT'S WRONG?

```
56
57  var exampleArray = [3, 2, 1];
58
59  for (var i = 0; i < array.charAt(); i++) {
60      var newElement = array[i] + 1;
61      console.log(newElement);
62  }
63
```

WHAT'S WRONG?

```
64
65  var exampleArray = ["LaToya", "Farrah", "LaTavia"]
66
67  for (var i = exampleArray.length; i > 0; i++) {
68      var newElement = exampleArray[i] + " isn't in Destiny's Child";
69      console.log(newElement);
70  }
```

IF…ELSE IF….ELSE Statements

Conditional Statements: they determine if a statement is ‘true’ or ‘false’

Used to perform different actions based on different conditions

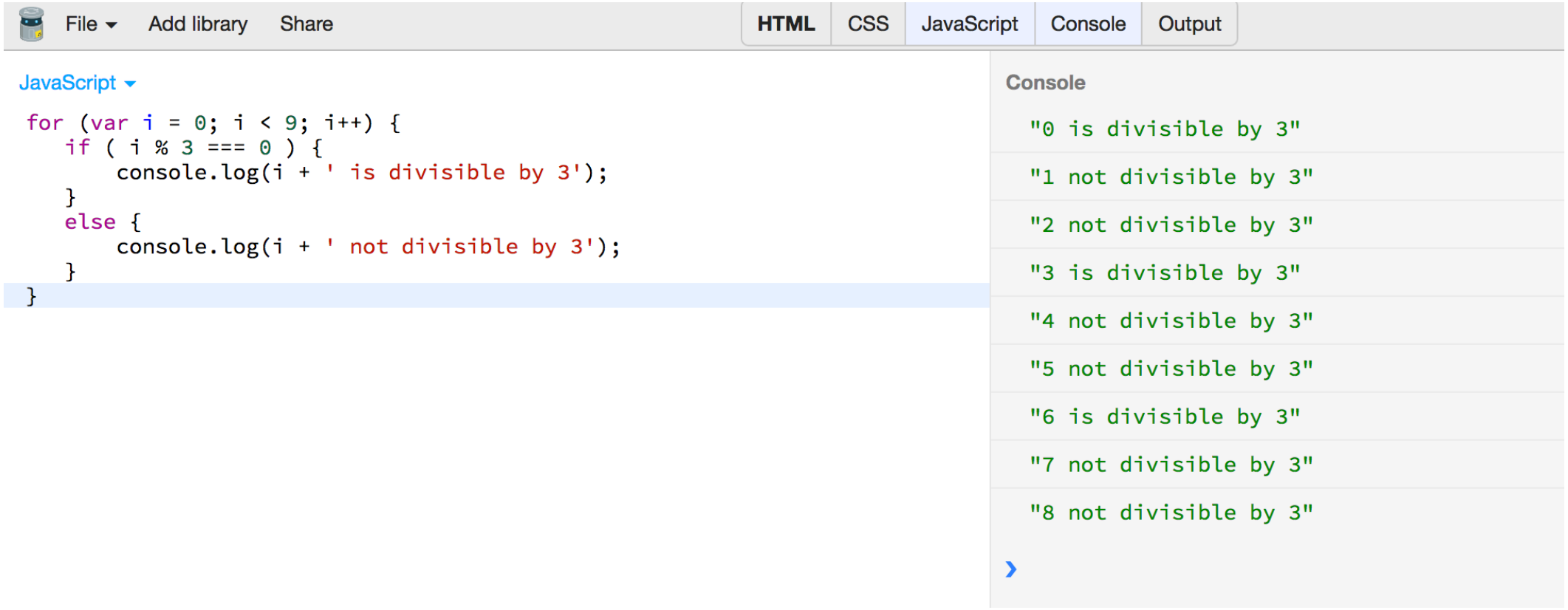
```
if (True) {
    // EXECUTE THIS CODE
}
else {
    // EXECUTE THIS CODE
}
```

IF…ELSE IF…ELSE Statements

SYNTAX:

```
if (TRUE) {
    // EXECUTE THIS CODE
}
else if (TRUE) {
    // EXECUTE THIS CODE
}
else {
    // EXECUTE THIS CODE
}
```

- MUST begin with an if statement
- Can have infinite ‘else if’ statements
- MUST end with ‘else’
  - Browser will throw an error
- Can make your conditionals
  - Truthy
    - (2 < 3)
  - Falsy
    - !(2 < 3)
    - Mostly see this for validation
    - !(’does this ’div’ have ’x’ class)
      - if it doesn’t, it will return true and you can add the class
      - if it does, it will return false and move to next statement


File
Add library
Share
HTML
CSS
JavaScript
Console
Output
JavaScript
for (var i = 0; i < 9; i++) {
if (i % 3 === 0) {
console.log(i + ' is divisible by 3');
}
else {
console.log(i + ' not divisible by 3');
}
}
Console
"0 is divisible by 3"
"1 not divisible by 3"
"2 not divisible by 3"
"3 is divisible by 3"
"4 not divisible by 3"
"5 not divisible by 3"
"6 is divisible by 3"
"7 not divisible by 3"
"8 not divisible by 3"


File
Add library
Share
HTML
CSS
JavaScript
Console
Output
JavaScript
for (var i = 0; i < 9; i++) {
if (i % 2 === 0) {
console.log(i + ' is divisible by 2');
}
else if (i % 3 === 0) {
console.log(i + ' is divisible by 3');
}
else {
console.log(i + ' is not divisbile by 2 or 3')
}
}
Console
"0 is divisible by 2"
"1 is not divisbile by 2 or 3"
"2 is divisible by 2"
"3 is divisible by 3"
"4 is divisible by 2"
"5 is not divisbile by 2 or 3"
"6 is divisible by 2"
"7 is not divisbile by 2 or 3"
"8 is divisible by 2"
›

JavaScript

```
function checkDivisibility(value) {
    for (var i = 1; i <= value; i++) {
        if ( i % 3 === 0 ) {
            console.log(i + ' is divisible by 3');
        }
        else {
            console.log(i + ' not divisible by 3');
        }
     }
}

checkDivisibility(12);
```

Console

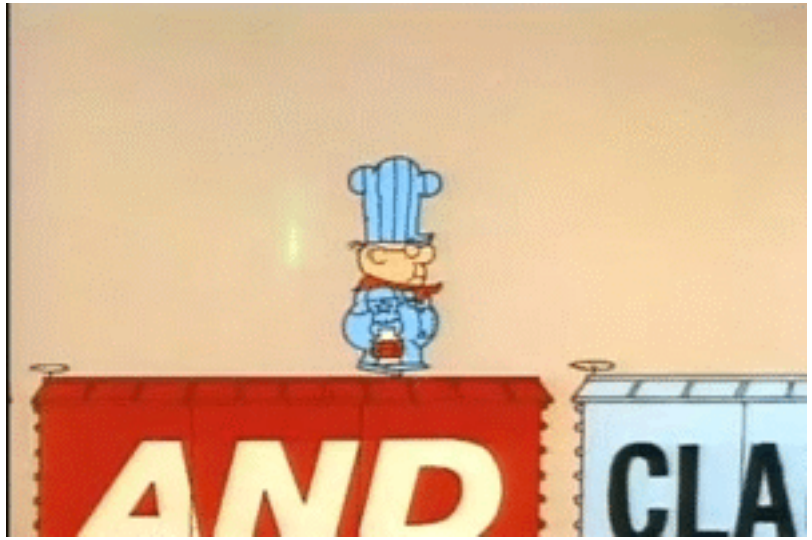
```
"1 not divisible by 3"
"2 not divisible by 3"
"3 is divisible by 3"
"4 not divisible by 3"
"5 not divisible by 3"
"6 is divisible by 3"
"7 not divisible by 3"
"8 not divisible by 3"
"9 is divisible by 3"
"10 not divisible by 3"
"11 not divisible by 3"
"12 is divisible by 3"
```

›

IF ELSE PRACTICE!

1. Write an if..else function that will take in a number:
   - If it is less than 3, print ‘less than 3’
   - If it is greater than 3, print ‘greater than 3’

2. Write an if…else function that will take in an array:
   - If array item is a number, print ‘value is a number’
   - If array item is a string, print ‘value is a string’
   - If array item is an object, print ‘value is an object’
   - If none of them, print ‘value is crazy’



QUESTIONS????

HTML REVIEW

HTML is Hypertext Markup Language

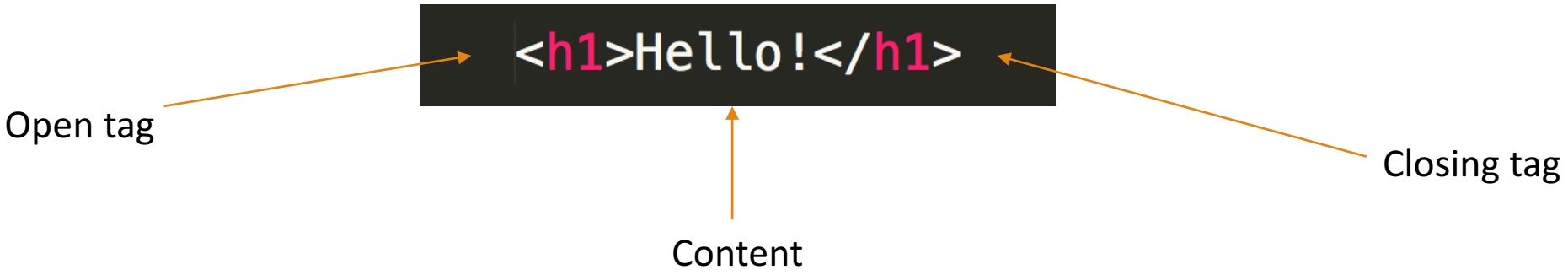
Consists of group of ‘elements’ (everything between start and end tag)

Can have ’nested’ HTML elements (divs within divs, lists, etc.)

HTML REVIEW

HTML Syntax:

- Open tag
- Content
- Close tag (the open tag with a “/”)



HTML REVIEW

Header Tags
- <h1></h1>, <h2></h2>, etc.
- As numbers increase, size of font decreases

Paragraph Tag
- This is what a paragraph element looks like.

Image Tag
-

Link Tag
-

HTML REVIEW

List Tag (Ordered and Unordered)

```
<ol>
    <li>Item 1</li>
    <li>Item 2</li>
</ol>
```

```
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
</ul>
```

HTML REVIEW

Div Tag (Division)

- Block element
- Used for code 'blocks'
- Breaks up the HTML page

```
<div>
    <p>This is a nested element</p>
</div>
```

Span Tag

- Inline element
- Surrounds small portions of text, images

```
<span>This is a span tag!</span>
```

HTML REVIEW

Attributes – Gives more information to the browser – connected to CSS

- ID, Class, Alt (Alternative Text, Accessibility), In-line CSS, Href

```
<div id="exampleID">
    <p>Hello!</p>
</div>
```

```
<span class="exampleClass">How are you?</span>
```

```
<p text-align="center">I am fine.</p>
```

CSS REVIEW

CSS = Cascading Stylesheets

- Cascading: styles can be overwritten and overruled
- Stylesheets: document(s) that tell the browser how to render the page

Style rules are followed top to bottom

- In the sheet
- In your HTML document

Consists of selector, property, and value

CSS REVIEW

CSS Syntax

- Selector: div element or HTML attribute
  - IDs begin with “#”, Classes begin with “.”

CSS REVIEW

```
div {
    background-color: red;
}
```

```
#exampleID {
    font-size: 10px;
}
```

```
div#exampleID {
    color: white;
    border-style: solid;
}
```

CSS Selectors

Multiple selectors

```
div#firstParagraph.bio {
    color: yellow;
}
```

Use of Commas

```
p.bio, p#work {
    color: blue;
}
```

CSS REVIEW

Descendants

```
p.bio p#work {
    color: green;
}
```

Direct Child

```
p.bio > p#work {
    color: red;
}
```

How will it render?

index1.html

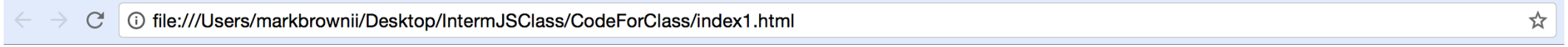
index1.html descendants.css

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Descendents and Child Selectors</title>
5      <link rel="stylesheet" type="text/css" href="descendants.css">
6  </head>
7
8  <body>
9
10     <div id="bio">
11         <h1>Solange Knowles</h1>
12
13         <div>
14             <h1>Singer and DJ</h1>
15         </div>
16     </div>
17
18 </body>
19 </html>
```

descendants.css

index1.html descendants.css

```
1
2  div#bio h1 {
3      color: red;
4  }
5
6  div#bio > h1 {
7      color: green;
8  }
9
10
```



Solange Knowles

Singer and DJ

How will it render?

index1.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Descendents and Child Selectors</title>
    <link rel="stylesheet" type="text/css" href="descendants.css">
</head>

<body>

    <div id="bio">
        <h1>Solange Knowles</h1>

        <div>
            <h1>Singer and DJ</h1>
        </div>
    </div>

</body>
</html>
```

descendants.css

```css


div#bio > h1 {
    color: green;
}

div#bio h1 {
    color: red;
}

```

file:///Users/markbrownii/Desktop/IntermJSClass/CodeForClass/index1.html

Solange Knowles

Singer and DJ

CSS – Pseudo-Classes

Pseudo-classes:

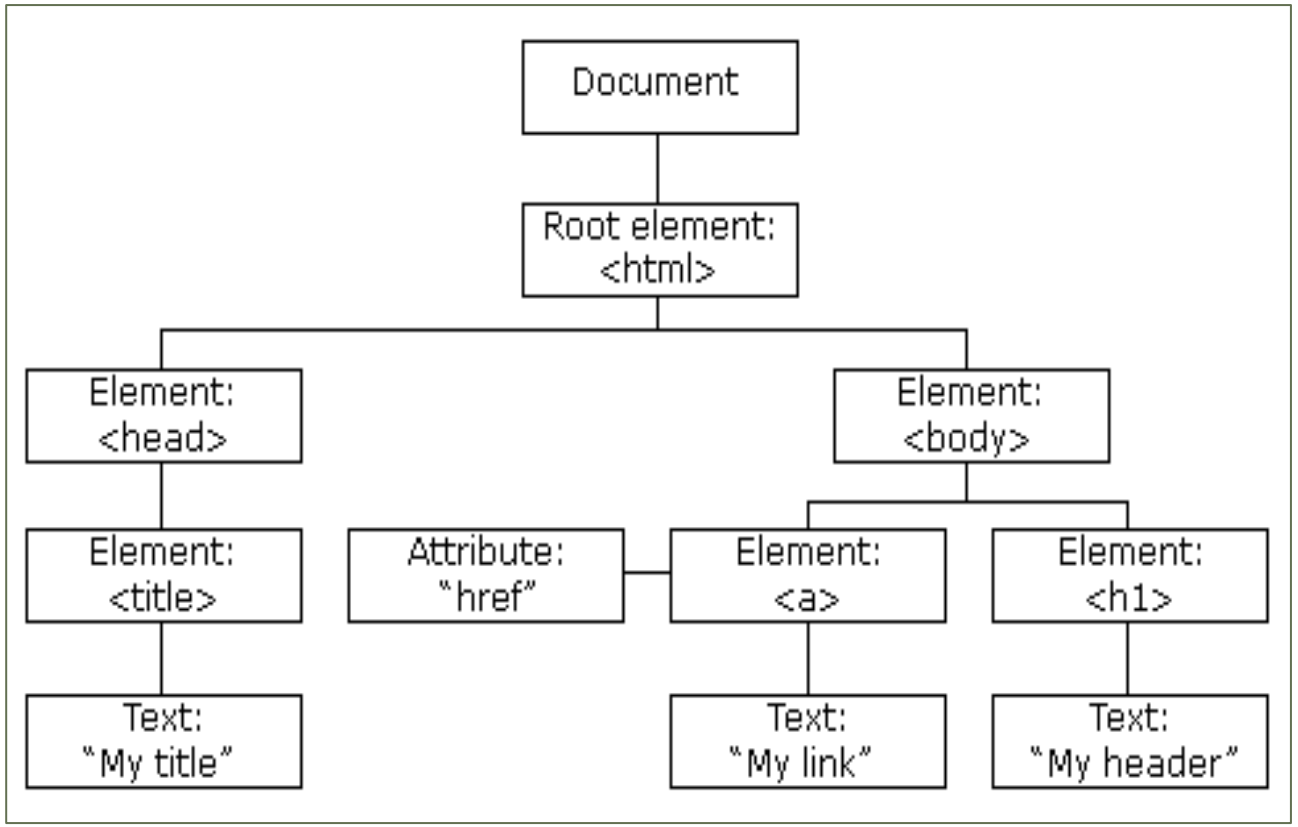
- Keyword added to selectors that specify a special state of the element to be selected (Mozilla Foundation)
- Let you apply a style to an element:
  - In relationship to the document tree (traditional selectors)
  - History of the navigator (:visited)
  - Status of the content (:checked)
  - Position of the mouse (:hover)

QUESTIONS????

DOM Manipulation

DOM – Document Object Model

- Platform and and interface that allows programs and scripts to dynamically access and update the context structure and style of a document (W3C standard)
- Created by browser when page is loaded
- Created using HTML and CSS files
- Used to create DOM tree (creates elements, text nodes, attribute nodes, etc.)
- Can be used by JS to create dynamic HTML:
  - Change elements on a page
  - Change attributes on a page
  - Change CSS styles on a page
  - React to events on a page
  - And others
- Is represented by the 'document' object in JavaScript




```
<!DOCTYPE html>
<html>

    <head>
        <title>My title</title>
    </head>

    <body>
        <a href="">My Link</a>
        <h1>My header</h1>
    </body>
</html>
```

DOM Manipulation

Can manipulate 3 ways:

1. Inline JavaScript

(not a best practice)

```
<h1 onclick="doSomething()"></h1>
```

2. Script tag in HTML document

(not a best practice)

```
<div>
    <script type="text/javascript">
        function doSomething();
    </script>
</div>
```

3. JavaScript files

```
<script type="text/javascript" src="./script.js"></script>
```

Please open DOM_Manip directory/folder

in Text Editor and Browser

Beyonce Knowles

Biography

Beyonce Knowles is the mother of twins and Blue Ivy.

She is also a former member of Destiny's Child.

Members of Destiny's Child

- **Beyonce Knowles**
- Kelly Rowland
- Michelle Williams
- Farah

DOM Manipulation

1. Access the element (using methods)

document.getElementByID()

document.getElementByClassName()

document.querySelector() → Returns first match

document.querySelectorAll() → Returns all matches

DOM Manipulation - Lists

Some methods will return arrays:

- getElementsByClassName()
- querySelectorAll()

To access individual elements:

- Item(index-of-list-item)
  - memberElements.item(1)
- Array syntax
  - memberElements[1]

DOM Manipulation - Loops

Since methods return arrays – we can do loops!

DOM Manipulation – Text Content

To get text from the DOM:

1. innerHTML → returns text and the HTML content

2. textContent → returns text, does not 'obey' CSS

3. innerText→ returns text, does 'obey' CSS
   - If CSS has text 'hidden' it will not return it

PRACTICE WITH DOM MANIP

Open DOM_Manip_Practice#1

Use JavaScript to do the following tasks:

1. Print Solange's name to the console
2. Print Solange's 3rd album to the console – just text
3. Print Solange's 2nd album to the console – with markup
4. Print *all* of Solange's albums to the console

DOM Manipulation – Change Text

Using the same methods you used to grab text – you can change it!

Remember these are attributes on an object, how do you think you can change it??

```
//Changing Text
document.getElementById('title').textContent = 'Tina Knowles';
```

DOM Manipulation – CSS Classes

To manipulate CSS classes uses the ‘classList’ property

Syntax:

- [selector].classList.[method]

Methods:

- add(’className’, ‘className’)
- remove(‘className’)
- contains(‘className’)
  - Returns true if present
  - Returns false if not

```
//Changing Class
var farah = document.getElementsByClassName('secondIter')[0];
farah.classList.add('green', 'red');
```

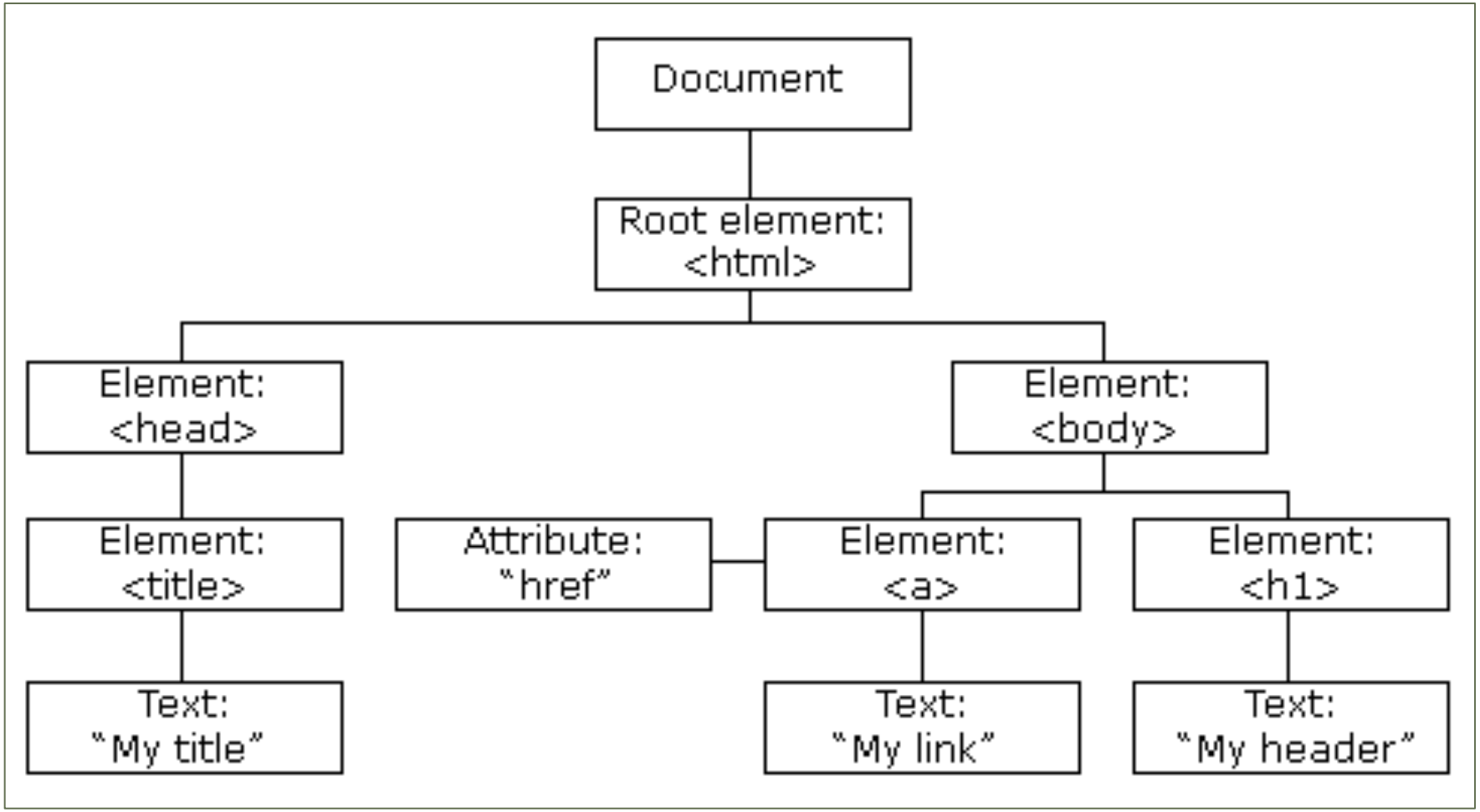
```
farah.classList.remove('green');
```

DOM Manipulation – DOM elements

Create a DOM element:

- 1. Create the element
  - var newEl = document.createElement('li');
- 2. Create a text node (if needed)
  - var newText = document.createTextNode('New Member');
- 3. Append the text node to the element
  - newEl.appendChild(newText)
- 4. Find (cache) the position of desired element
  - var position = document.getElementByTagName('ul')[0]
- 5. Append new element to the position
  - position.appendChild(newEl);

```
//Appending Element
var newEl = document.createElement('li');
var newText = document.createTextNode('New Member');
newEl.classList.add('green');
newEl.appendChild(newText);
var position = document.getElementsByTagName('ul')[0];
position.appendChild(newEl);
```


Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
"href"
Element:
<a>
Element:
<h1>
Text:
"My title"
Text:
"My link"
Text:
"My header"

DOM Manipulation – DOM elements

Remove a DOM element:

- 1. Find the element you want to remove
  - var removeEl = document.getElementsByClassName('newMember')[0];
- 2. Find the parent of the element
  - var parentEl = removeEl.parentNode;
- 3. Use removeChild() method to remove the child
  - parentEl.removeChild(removeEl);
  - Shorter Syntax: removeEl.parentNode.removeChild(removeEl);
    - Doesn't require you create a variable for the parent node

```
//Removing Element
newEl.parentNode.removeChild(newEl);
```

PRACTICE WITH DOM MANIP

Open DOM_Manip_Practice#2:

1. Change header of the page from ‘African-American Writers’ to ‘African-American Authors’

2. Add a class to the <ul> element that adds a border to the list

3. Add a list item to the unordered list with the text: Alice Walker

4. Add a class to your Alice Walker list item that will change the text color to any color other than black

POP QUIZ ON MATERIAL

1. How do you write a loop for arrays?
2. How do you write a loop for objects?
3. What is the difference between a function declaration and a function expression?
4. What is a conditional statement?
5. What does the ‘return’ keyword do?
6. Using JavaScript, what is the object we use to manipulate the DOM?
7. Do all nodes have attributes? Do all nodes have text nodes?

BREAK!

Please open Event_Practice#1 directory/folder

in Text Editor and Browser

EVENTS

Users tend to interact with websites using ‘events’

Events include actions such as: (List found here)

- Clicking
- Key-up, Key-down
- Doubleclicking
- Submitting a form
- Selecting a checkbox

The browser listens for events

We are able to make changes to our website based on these events

Events are *fired/raised* → Events *trigger* scripts

EVENT HANDLING

Steps that are used to get HTML to trigger scripts

- 1. Select the element
- 2. Specify which event
- 3. Call the code

3 Ways to Bind Events to HTML elements

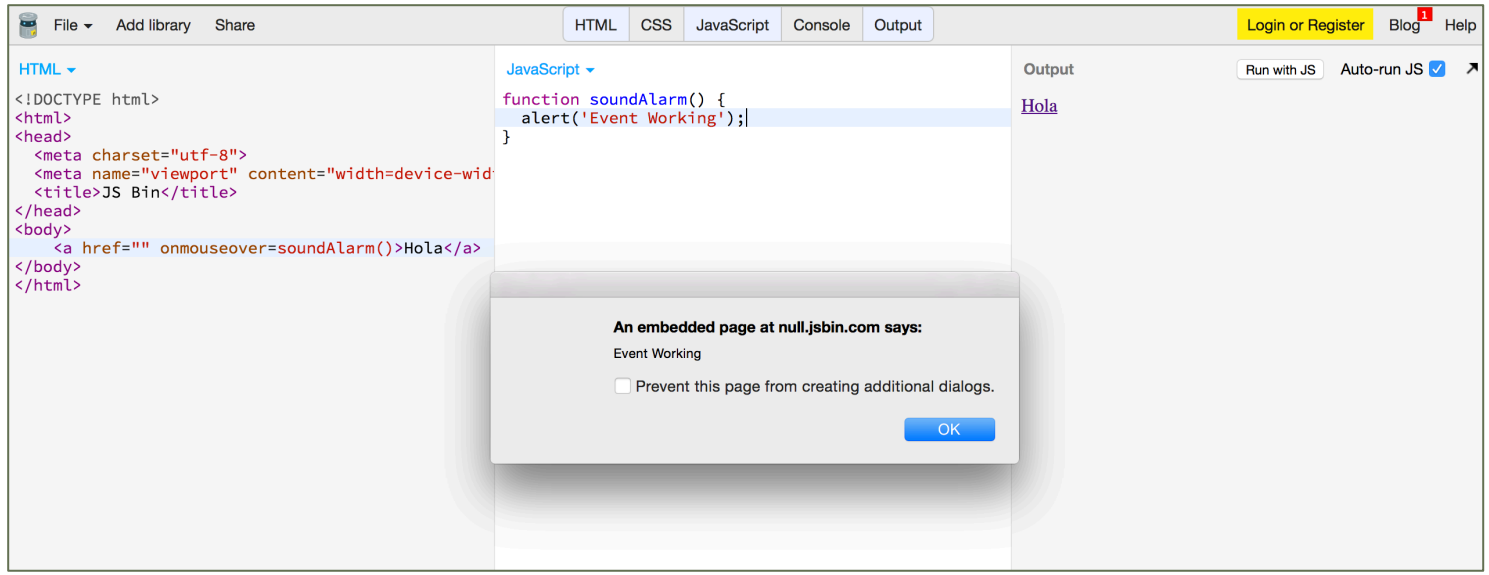
- HTML Event Handlers (bad practice)
- DOM Event Handlers
- Event Listeners

HTML EVENT HANDLERS

List of HTML Event Attributes (here)

Steps that are used to get HTML to trigger scripts

- 1. Select the element
- 2. Specify which event
- 3. Call the code



EVENT HANDLING - DOM Event

Steps that are used to get HTML to trigger scripts

- 1. Select the element
- 2. Specify which event
- 3. Call the code

Syntax:

- ELEMENT.onEVENT = functionName; (found in JS file)

```
// DOM EVENT HANDLER
var eventHolder = document.getElementsByTagName('a')[0]
var responseEl = document.getElementsByClassName('response')[0];

function onTheTwenty() {
    responseEl.style.visibility = 'visible';
    console.log('working');
}

eventHolder.ondblclick = onTheTwenty;
```

EVENT HANDLING – Event Listeners

Steps that are used to get HTML to trigger scripts

- 1. Select the element
- 2. Specify which event
- 3. Call the code

Syntax:

- element.addEventListener(‘event’, ’functionName’) (found in JS file)

```
var secondQuestion = document.getElementsByTagName('h2')[0];
var secondResponseEl = document.getElementsByClassName('response')[0];

function showAnswer() {
    secondResponseEl.style.visibility = "visible";
}

function hideAnswer() {
    secondResponseEl.style.visibility = "hidden";
}

secondQuestion.addEventListener('mousemove', showAnswer);
secondQuestion.addEventListener('mouseleave', hideAnswer);
```

EVENT OBJECT

Every time an event is called – an ‘event’ object is made (shorthand: ‘e’)

Must pass in the ‘e’ object to function to get access to it

Important properties of object

- e.target (target of the event)
- e.type (type of event)

Important methods

- e.preventDefault() – cancels default behavior
- e.stopPropagation() – stops events from bubbling or capturing

```
// EVENT OBJECT
var thirdQuestion = document.getElementsByTagName('h2')[0];
var thirdChoiceOne = document.getElementsByTagName('p')[0];
var thirdChoiceTwo = document.getElementsByTagName('p')[1];

thirdChoiceOne.addEventListener('click', function(e) {
    console.log(e);
    alert(e.target.outerHTML);
    correctAnswer(e);

})

thirdChoiceTwo.addEventListener('click', function(e) {
    alert(e.target.outerHTML);
    correctAnswer(e);
})
```

EVENT DELEGATION

What happens if we have a long list of items?

Use parent element to listen to all of its children → event delegation

Looking back at our ’event’ object

- event.target gives you a variety of properties that you can use

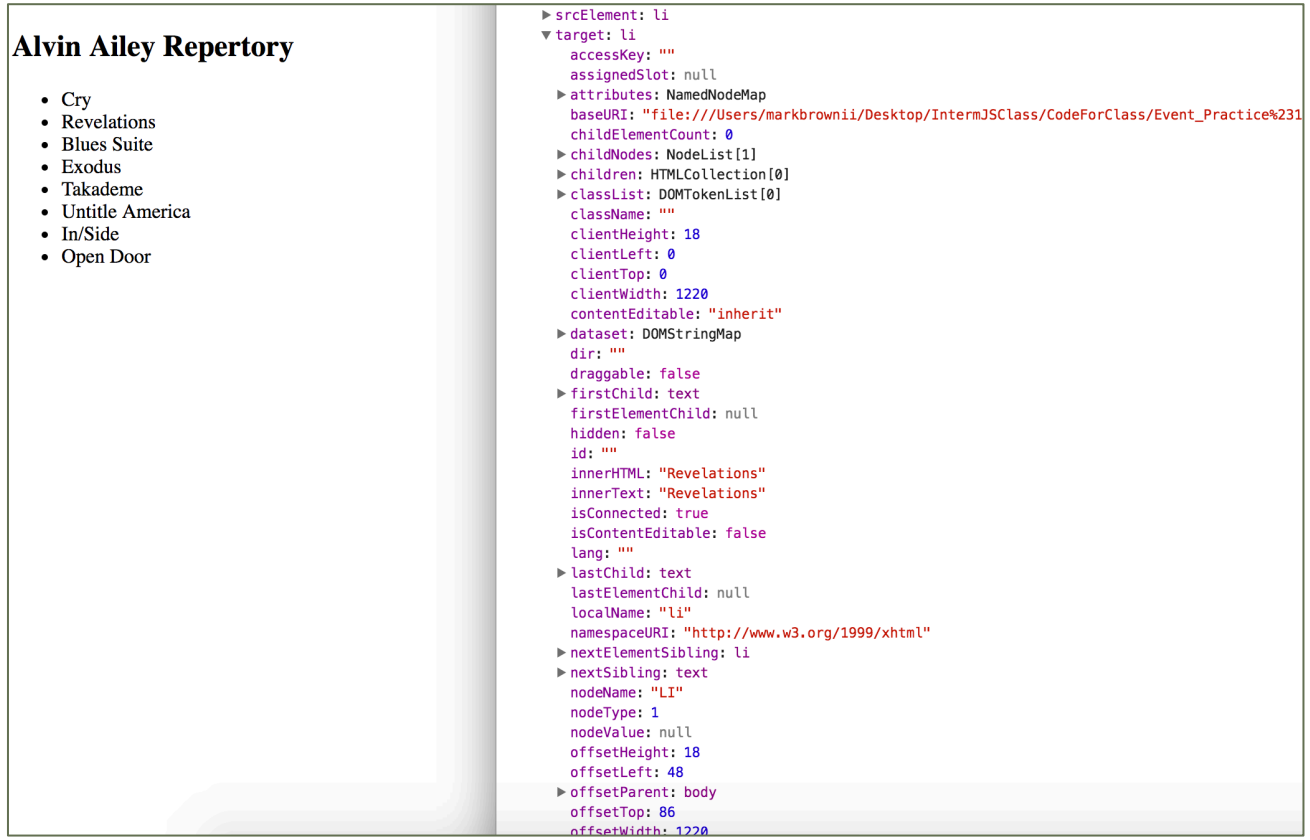
Syntax:

- parentElement.addEventListener(‘event’, function(e) {

```
// RUN CODE HERE
// Access to ‘clicked’ li is found at e.target
}
```

```
// EVENT DELEGATION
var aileyList = document.getElementsByTagName('ul')[0];

aileyList.addEventListener('click', function(e) {
    console.log(e);
})
```



QUESTIONS?

YOUR TURN!!

Open Event_Practice#2

1. Use HTML Handler to change title to ‘First Ladies of Jazz’

2. Use an Event Handler (DOM or Event Listener) to show the list of Jazz Singers

3. Use Event Delegation so when a person’s name is clicked – a fact shows up about them!

QUIZ

1. What is an integer? What is a float?
2. What is the difference between an array and an object?
3. What does () do after a function name?
4. What do we call the data we put in-between () when we write and call a function?
5. What is a benefit of using Event Listeners?
6. What is ‘e’ in Event Listeners and DOM Event Handlers?

ANYTHING YOU CAN DO WITH jQUERY YOU CAN DO WITH VANILLA JAVASCRIPT!!

Please open jQuery_Exercises directory/folder

in Text Editor and Browser

jQuery

JavaScript library designed to help simplify client-side scripting of HTML

Free and open-source

Most widely-used JS library

Makes it easier to:

- Navigate a document
- Select DOM elements
- Handle events
- Create animations
- Develop Ajax applications

Must include script tag above your JS file to use jQuery

jQuery - Syntax

Basic Syntax:

- $(‘selector’).method()

Must use “$”

Must use ‘ ‘ with the selector

DOM element selectors are just like CSS selectors

jQuery comes with helpful methods such as: (List)

- .addClass()
- .hide()
- click()
- .removeClass()

jQuery – DOM Manipulation

To get all “li” on a page

- $(‘li’)

To get a ‘p’ with a class of ‘orange’

- $(‘p.organge’)

To get elements with ’href’ attribute

- $(‘attr=href’)

To get “div” with a “span” with a ’onclick’ attribute

- $(’div:has(span[onclick])) ’

jQuery – DOM Manipulation

Add Class – addClass()

Remove Class – removeClass()

Add Element – after() or before()

Remove Element – remove()

Change CSS – css()

jQuery – Event Listeners

All event listeners require a callback function (A function within a function)

Click Event – click()

Hover - hover()

Mouseover - mouseover()

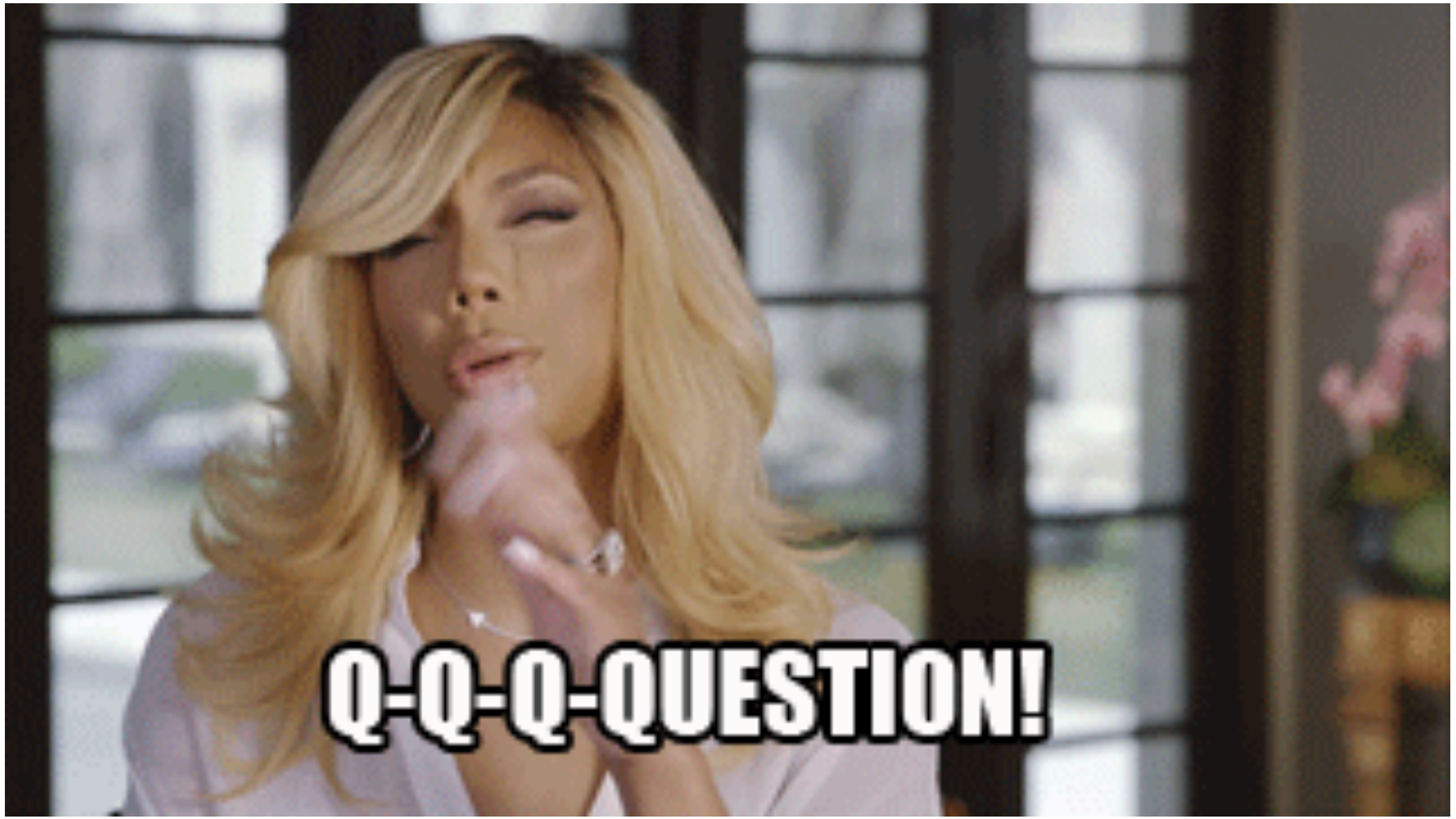
jQuery – Event Listeners

All event listeners require a callback function (A function within a function)

’on’ Method – allows you to set multiple events

- Pass in an object
- Object key/val pairs are ‘events’ (keys): ‘functions’ (value)

- $(document).on({
  click: function(),
  mouseover: function()
})


Q-Q-Q-QUESTION!

NOW IT IS YOUR TURN TO HAVE FUNSIES!!!

RESOURCES

jQuery Events: https://api.jquery.com/category/events/

jQuery Mouseevents: https://api.jquery.com/category/events/mouse-events/

jQuery Fading: https://api.jquery.com/category/effects/fading/

jQuery CSS: https://api.jquery.com/category/css/

jQuery Animate: http://api.jquery.com/animate/

jQuery Attributes: http://api.jquery.com/category/attributes/

jQuery – AJAX Requests

$.get()

Uses a callback function!

Documentation: http://api.jquery.com/jQuery.get/

FORMULA IS:

```
$.get( 'URL')
    .done( function(data) {
        // DO SOMETHING WITH DATA IF SUCCESSFUL
    })
    .fail( function(err) {
        // DO SOMETHING IF THERE IS AN ERROR
    })
```

jQuery - AJAX

4 Types

- GET
  - Get information
- PUT
  - Update or insert new information
- POST
  - Add new information
- DELETE
  - Delete information

Each API is set up differently with how to make ‘calls’

- Look at documentation
- It is okay to make mistakes hitting an API

```
// AJAX API CALLS
$.get( "https://ptabdata.uspto.gov/ptab-api/documents?id=16")
    .done(function(data) {
        $('#dataDiv').append(data.results[0].documentNumber);
        console.log(data);
    })
    .fail(function(err) {
        console.log(err);
    });
```

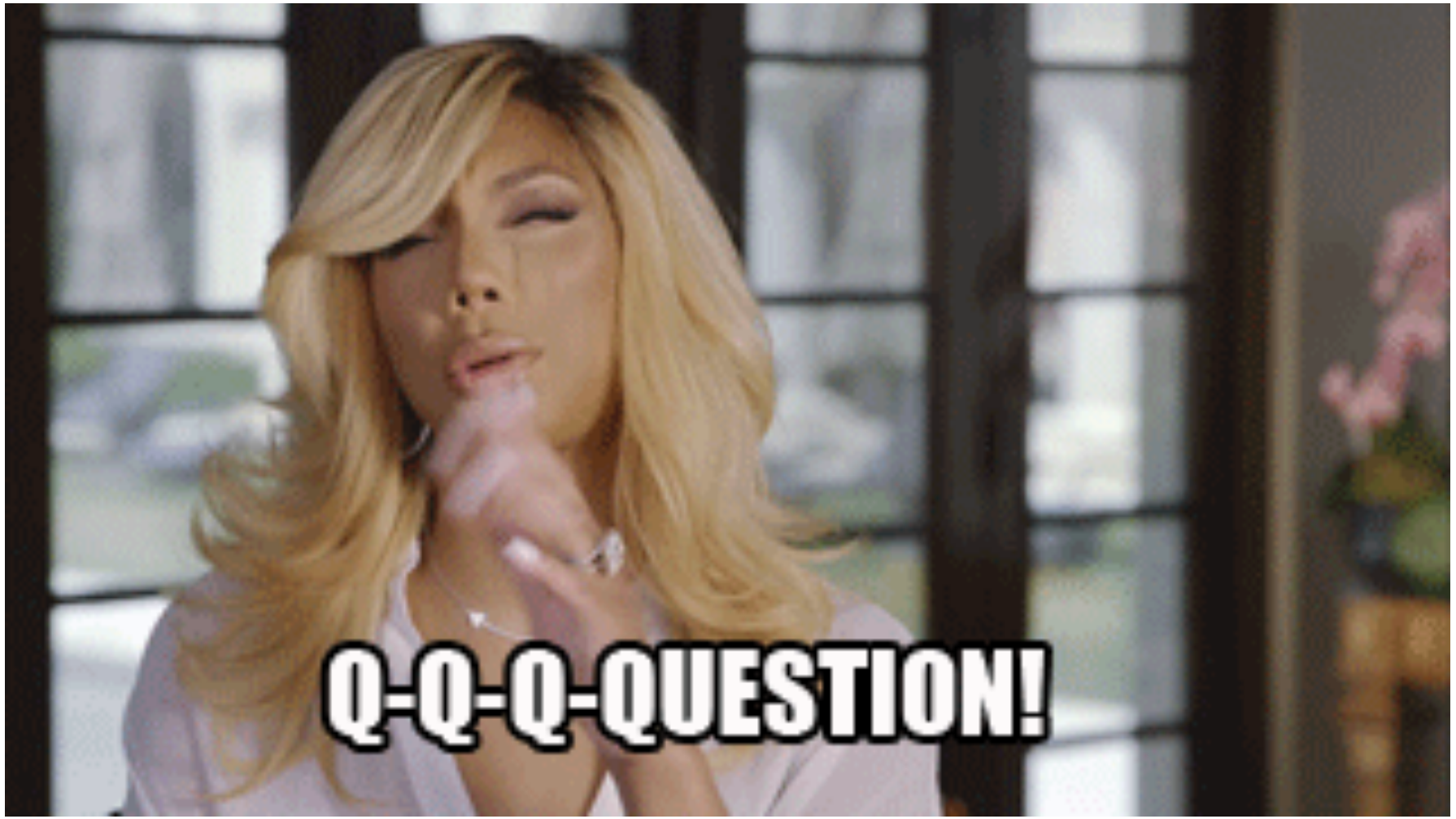
JSON

JavaScript Object Notation

Data objects consisting of attribute-value pairs

Passed back from AJAX calls – calls to API (Application Program Interface)

Based on JS – can be read by many other languages


Q-Q-Q-QUESTION!

Resources for Continued Learning

Code School (www.codeschool.io)

Eloquent JavaScript (http://eloquentjavascript.net/)

Pluralsight (https://www.pluralsight.com)

Udacity (https://www.udacity.com/)

jQuery Documentation (http://jquery.com/)

Javascript & jQuery – Book by Jon Duckett (http://javascriptbook.com/)

THANK YOU FOR COMING!

Hope you had as much fun as I did!!