WEEK 5 – JAVASCRIPT 101

GENERAL ASSEMBLY

FEWD

Joe Bliss

AGENDA

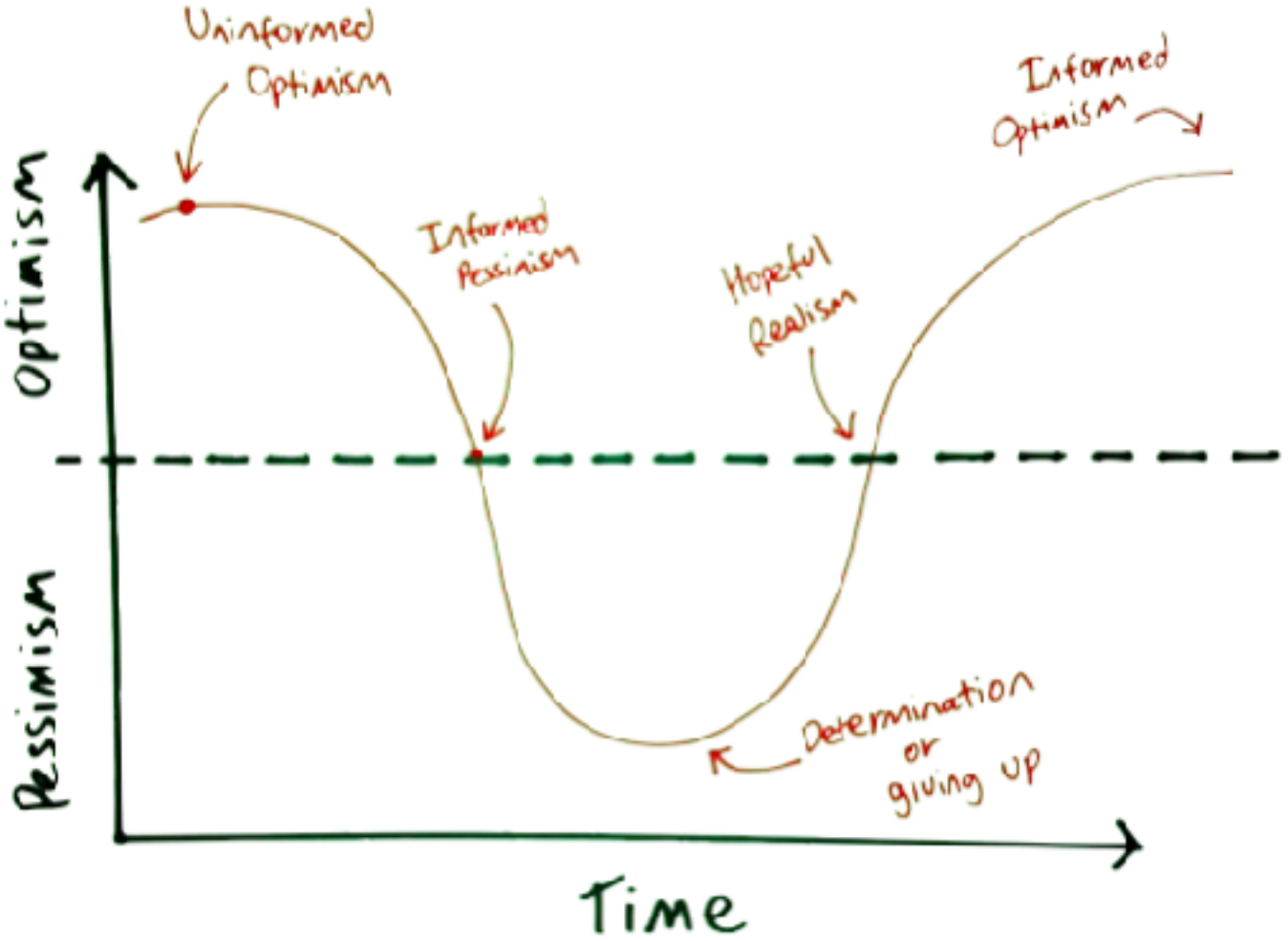
Homework Review / Questions

Thinking Programmatically

JavaScript 101:

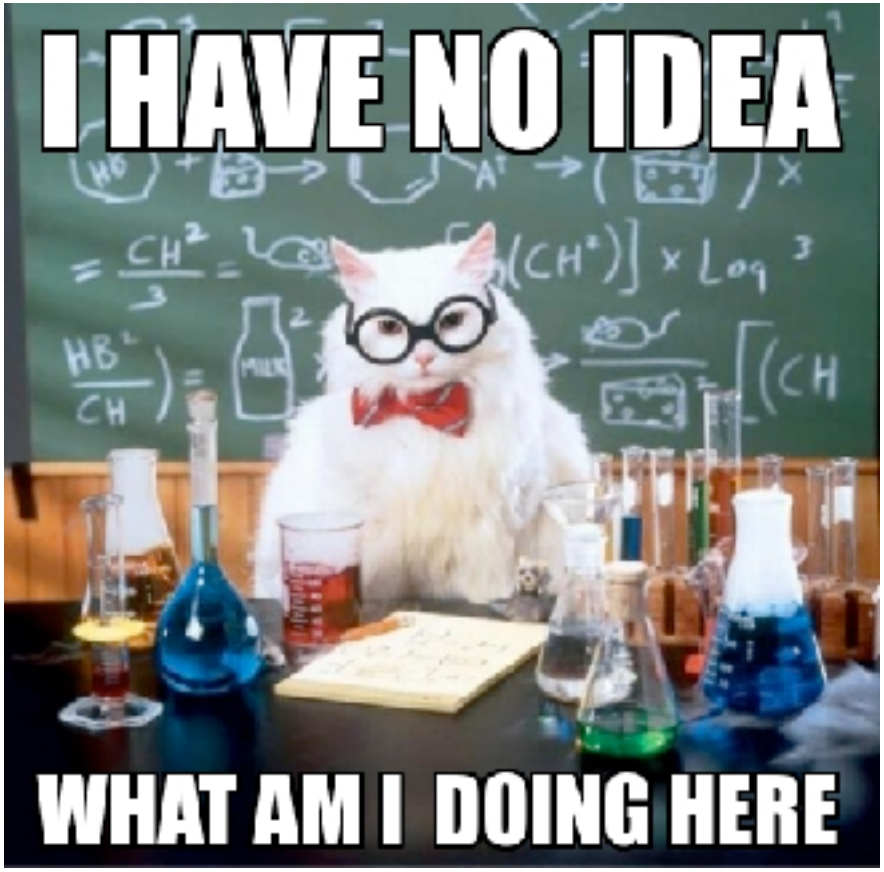
- Data Types and Data Structures
- The DOM
- Control Flow
- Events

WHERE WE ARE


Uninformed Optimism
Informed Optimism
Informed Pessimism
Hopeful Realism
Optimism
Pessimism
Determination or giving up
Time

JAVA ≠ JAVASCRIPT

SO WHY ARE WE LEARNING JAVASCRIPT?



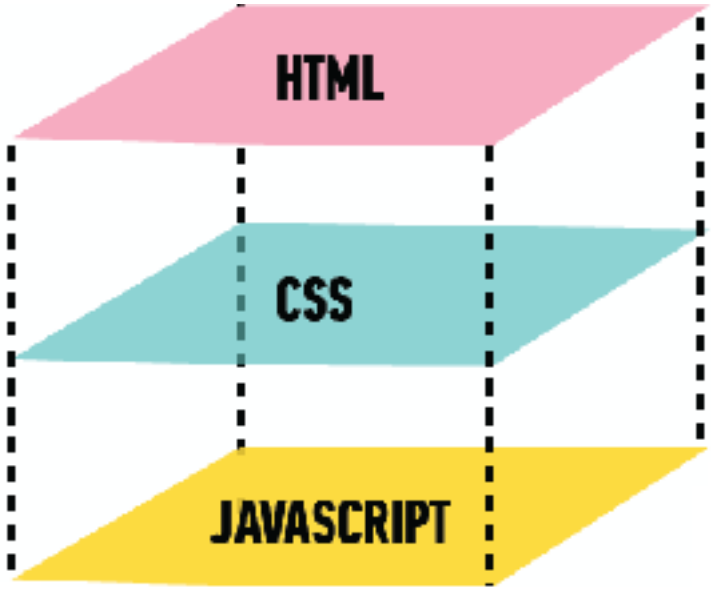
HOW JS DIFFERS FROM HTML AND CSS

HTML and CSS are used to define the initial state of our website.

JS is used to define how this state changes.

HTML and CSS are static.

JS is dynamic.



JAVASCRIPT MAKES A PAGE DYNAMIC

Adding JavaScript to a page can have a subtle or drastic impact.

WHAT JAVASCRIPT CAN DO:

Add / Remove Elements
Change CSS “on-the-fly”
Animate content
Detect user interactions
Validate Forms
Load dynamic content
Etc. etc. etc. etc.

EVEN IF YOU KNOW HTML AND CSS, YOU HAVEN'T LEARNED TO PROGRAM YET.

SURPRISE!

WHAT IS A PROGRAM?

A program is a set of instructions that a person writes to tell a computer how to carry-out a task.

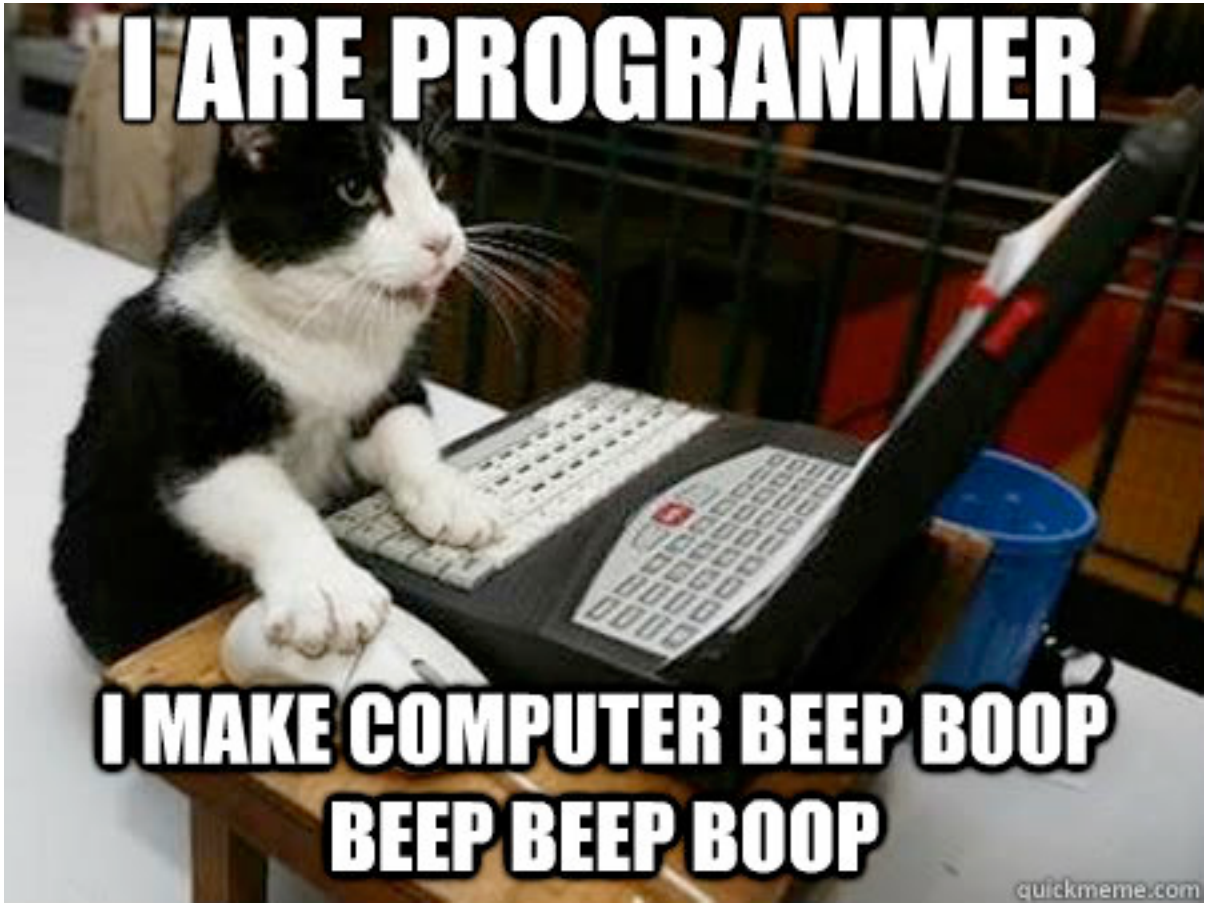
Programming is the task of writing those instructions in a language that the computer can understand.

BECOMING A PROGRAMMER

… is about learning how to “think” like a computer. It is not about learning a particular language.

So, we have to know how the computer “thinks” to change how we think.

THINK LIKE A COMPUTER



CLOSE YOUR LAPTOPS

Seriously.

WHAT DID WE LEARN?

You have to be speaking the same language.

You have to know what's pre-defined in the language.

Steps execute sequentially.

Steps must be small, granular.

The computer will do ONLY and EXACTLY what you tell it to do.

THINKING PROGRAMMATICALLY

Play the following game based on FROZEN:
http://studio.code.org/s/frozen/stage/1/puzzle/1

THE CODE NEVER BOTHERED ME ANYWAY

JAVASCRIPT 101

- Data Types and Data Structures
- The DOM
- Control Flow
- Events

DATA TYPES AND DATA STRUCTURES

CODEALONG – OUR FIRST JAVASCRIPT

Open up http://codepen.io/pen/

Type into the JS panel:

```
alert("Hello, World!");
```

or

```
document.write("Hello, World!");
```

TYPES OF DATA – NUMBERS

Integers

1, 2, 3, 4, 5

Floats (numbers with decimal points)

3.14159, 2.718281828459045

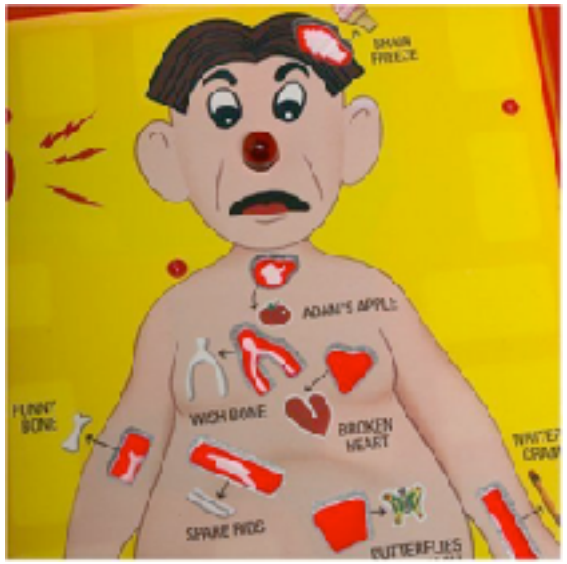
Can be Signed or Unsigned (- or +)

6, -8.2

We can perform arithmetic on number data types

NUMERICAL OPERATIONS

Operator	Description	Example
+	Addition	1+1
-	Subtraction	3-2
*	Multiplication	5 * 3
/	Division	10 /2



TYPES OF DATA – STRINGS

Strings

- A sequence of characters enclosed in quotes, i.e. “I am a String”, “Hello!”, “Joe Bliss”
- Stores textual information
- Can be “double” or ‘single’ quoted

MORE ON STRINGS

Double vs single quoted strings:

- 'They "purchased" it'
- "It's a beautiful day"

Escaping

- "They \"purchased\" it"
- 'It\'s a beautiful day'

EXERCISE – OUR FIRST JAVASCRIPT

Add to our script the following alerts / Document Write messages:

- The product of 5 and 23 (using *)
- The difference of 4 and 2 (using -)
- The quotient of 42 and 6 (using /)
- The sum of 7 and 8 (using +)
- Your first and last name (“Joe Bliss”)
- A warning message of your choosing (“These are not the droids you are looking for!”)

WORKING LOCALLY WITH JAVASCRIPT

A text file with the “.js” extension. Like CSS, we include it one of two ways:

External (most common)

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

Internal

```
<script type=”text/javascript”>
    //Do stuff here
</script>
```

Group multiple scripts into a folder, such as “js” or “scripts”.

RETAINING INFORMATION

What does the following do?

```
alert(2+3);
```

What about this?

```
2+3;
```

VARIABLES

We can tell our program to remember values for us to use later on. The entity we use to store the value is called a variable.

We use the keyword “var” to reserve a variable in JS.

VAR OUT, MAN

Declaration - Creating a variable, reserves a space in memory and gives it a name.

- var age;

Assignment - gives that variable a value.

- age = 2;

Intialization (Declaration and Assignment):

- var age = 2;

WHAT'S IN A NAME?

Variable Naming Conventions:

- Start with a lowercase letter. If they contain multiple words, subsequent words will start with an uppercase letter.
- var number = 5;
- var numberOfClasses = 10;

RESERVED WORDS – I.E. DON'T USE!

abstract
boolean
break
byte
case
catch
char
class
const
continue
debugger
default
delete
do
double
else
enum
export
extends
false
final
finally
float
for
function
goto
if
implements
import
in
instanceof
int
interface
long
native
new
null
package
private
protected
public
return
short
static
super
switch
synchronized
this
throw
throws
transient
true
try
typeof
var
void
volatile
while
with

IN A GALAXY VAR, VAR AWAY ...

Re-assignment:

- var name = “Joe”;
- name = “Steve”;

alert(name);

- Will display “Steve”

ARRAYS

A complex Data Type, referred-to as a Data Structure

Arrays are used when we want to keep track of multiple values in a single variable.

HIP! HIP! ARRAY!

Rather than create 5 variables with different values:

```
var fruit1 = “Pineapple”;
var fruit2 = “Lemon”;
var fruit3 = “Apple”;
var fruit4 = “Orange”;
var fruit5 = “Peach”;
```

I can create one Array and give it 5 different values.

HIP! HIP! ARRAY!

var fruits = ["Pineapple", "Lemon", "Apple", "Orange", “Peach”];

Each fruit in the above Array can now be referenced by its “index”, that is, it’s numeric place in the Array, starting at 0.

fruits[0] is equal to “Pineapple”

fruits[3] is equal to “Orange"

HIP! HIP! ARRAY!

You can think of the structure / behavior of an Array as being like a pill sorter.

OBJECTS

Objects, like Arrays, can contain many values.

```
var car = {make:"Honda", model:"FIT", color:"green"};
```

alert(car.make); will alert: Honda
alert(car.color); will alert: green

Great for storing complex data and modeling real-life objects.

OBJECTS

Objects have properties, i.e. name:value pairs.

In the previous example, make, model, color are all properties of the car.

Objects can also have methods. Methods are actions that can be performed on objects. start() or drive() might be methods defined on a car.

JAVASCRIPT 101

- Data Types and Data Structures
- The DOM
- Control Flow
- Events

THE DOM

Dom, dom, dom, dom, doooommmmmmm….

NO, NOT THAT ONE ...

DOM – THE DOCUMENT OBJECT MODEL

To the browser, your webpage looks nothing like the way the user sees it. Instead, the browser interacts with what is called the “Document Object Model” of your website.

The browser takes the values you set in the HTML and CSS and *builds* the DOM from them. Each tag in HTML becomes a JavaScript Object which all make-up the DOM.

DOM – THE DOCUMENT OBJECT MODEL

The browser is showing you the DOM, not the HTML/CSS. When the page first loads, the DOM matches the defaults stored in the HTML / CSS, but these are manipulated as your JavaScript *changes* the DOM.

Javascript changes the DOM, not the underlying HTML/CSS.

WHAT'S IN A DOM OBJECT?

Every Javascript Object in the DOM has a bunch of Properties and Methods associated with it:

http://www.w3schools.com/jsref/dom_obj_all.asp

And Events (which are like *special* methods):
http://www.w3schools.com/jsref/dom_obj_event.asp

Manipulating these values is how we effect change on our pages.

HOW TO “GET” OBJECTS IN THE DOM

getElementById() - Your best friend

document.getElementById("someID");

- Gives you the tag in the HTML with id=“someID" as a JS Object that you can then manipulate.

There are tons of properties associated with this object that we will explore, for instance: innerHTML, style, onclick

JAVASCRIPT 101

- Data Types and Data Structures
- The DOM
- Control Flow
- Events

CONTROL FLOW

CONTROL FLOW

… is the order in which individual statements, instructions or function calls […] are executed or evaluated.

https://en.wikipedia.org/wiki/Control_flow

COMPARISONS

We often times have things that we want to compare.

Think about a Weather App.

We will also want to check if the current temperature is greater than, less than, or equal to some value.

VAR X = 3;

Logical Operators			
Operator	Description	Comparing	Returns
==	equal to	x == 8	FALSE
!=	is not equal	x!=8	TRUE
>	greater than	x>8	FALSE
<	less than	x<8	TRUE
>=	greater than or equal to	x>=8	FALSE
<=	less than or equal to	x<=8	TRUE

IF A PICTURE PAINTS A THOUSAND WORDS, THEN …

```
if (this condition is true) {
    //Execute this code
}
//Otherwise continue, skipping the code above
```

IF / ELSE

```
if (condition is true) {
    alert(“The condition is true”);
}
else {
    alert(“The condition is false”);
}
```

IF / ELSE IF / ELSE

```
if (condition is true) {
    alert(“The condition is true”);
}
else if (some other condition is true){
    alert(“The first condition was false, but this one is true”)’;
}
else {
    alert(“Neither was true”);
}
```

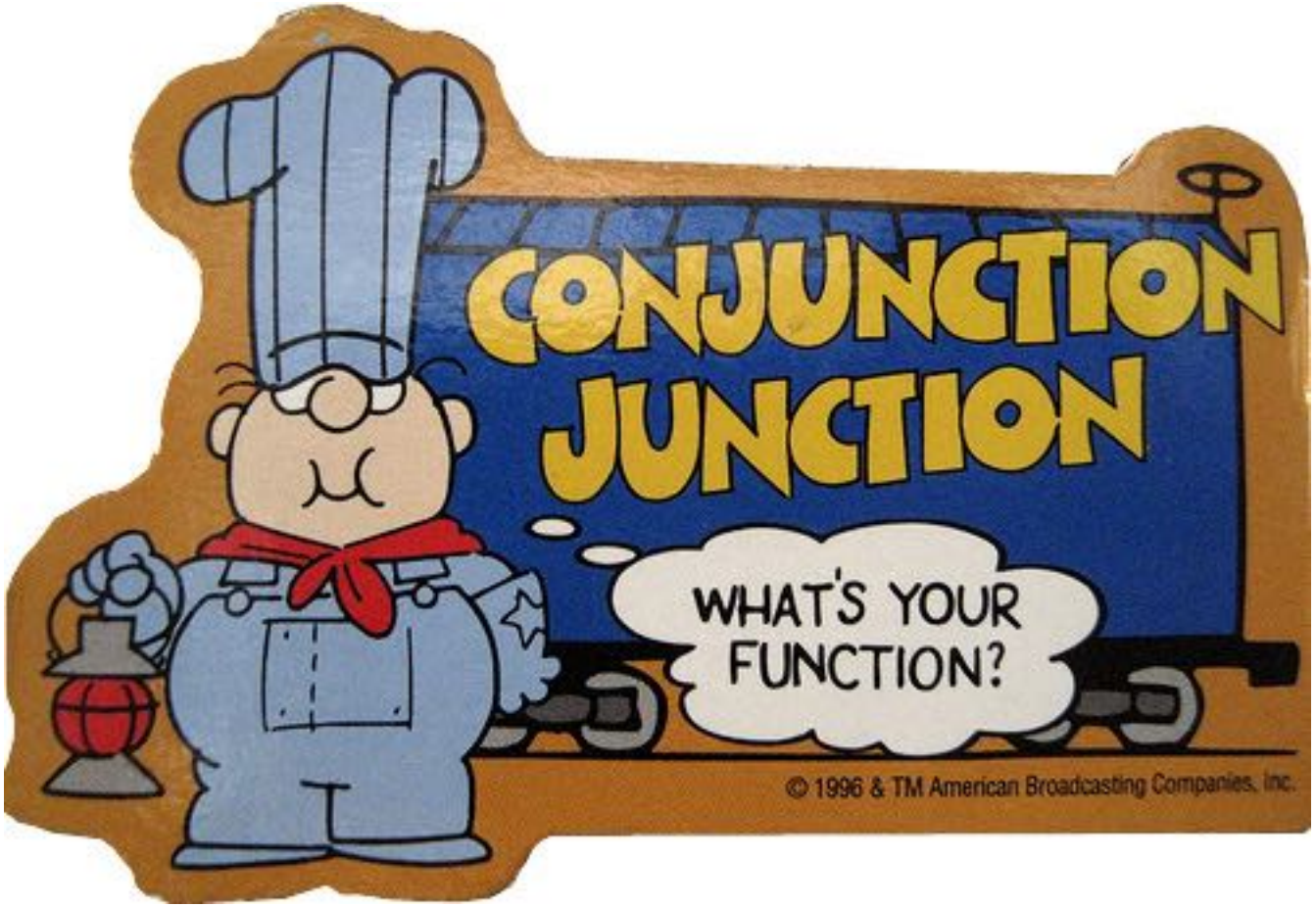
CODEALONG – CLICK COUNT

Change the DOM in some way when the count hits a certain threshold.

EXERCISE – CLICK COUNT

Add some more functionality to the click counter by adding an else if and an else to the existing code.

FUNCTIONS


CONJUNCTION JUNCTION
WHAT'S YOUR FUNCTION?
© 1996 & TM American Broadcasting Companies, Inc.

FUNCTIONS

Functions are a reusable collection of statements (lines of code) that you group together so that you can:

- Execute them at a given time
- Reuse them
- Respond to user input

FUNCTIONS

Declare a function:

```
function sayMyName( ) {
    document.write(“Joe Bliss”);
}
```

Call a function: //Will write “Joe Bliss” on the document twice

```
sayMyName( );
sayMyName( );
```

FUNCTION AS VARIABLE

Can also be written as:

```
var sayMyName = function() {
    document.write("Joe Bliss");
}

sayMyName( );
```

FUNCTIONS – ANONYMOUS

Some commands expect a function as an argument. The anonymous function lets us use a function without naming and separately defining it.

```
unction(){};
```

FUNCTIONS – ANONYMOUS

```
function() {
    //do stuff
}
```

Just like when we use numbers or strings without attaching them to a variable, when we use anonymous functions, they become more "ephemeral". They only exist when that bit of code is being run.

They are good for JS events (like onclick, onchange) because they mean you don't have to create a separate named function.

JAVASCRIPT 101

- The DOM
- Data Types and Data Structures
- Control Flow
- Events

EVENTS

EVENTS

Events are when something “happens” on the page.

There are browser events and user events. Some examples of events:

A web page has finished loading
An input field was changed
A button was clicked

You can detect when an event has occurred and perform an action.

EVENT HANDLING

When we attach a behavior to a specific event, it is referred-to as “binding”. We bind a function, called a “handler” to the event.

```
var someButton = document.getElementById(“submitbutton”);
someButton.onclick = doStuff;

function doStuff() {
  //These steps will be performed
  //every time the button is clicked
}
```

ANONYMOUS FUNCTION

We could also have written that function anonymously, like the following:

```
var someButton = document.getElementById(“submitbutton”);
someButton.onclick = function() {
  //These steps will be performed
  //every time the button is clicked
};
```

MOUSE EVENTS

onclick

- The event occurs when the user clicks on an element

onmouseenter

- The event occurs when the pointer is moved onto an element

onmouseleave

- The event occurs when the pointer is moved out of an element

EXERCISE – TRAFFIC LIGHT

Take the following Codepen and see if you can understand what it is doing. Then, fix it!

https://codepen.io/ga-joe/pen/yXyXLw

CODEALONGS