IWDD100

Introduction to JavaScript

What is JavaScript?

- A client-side scripting language
- Meant to run entirely on the user’s browser
- Defined by the ECMAscript standard, published by the ECMA foundation
- JavaScript != Java. Don’t mess this up or web developers will get very mad at you.

Main uses of JavaScript

- Originally used to handle simple tasks like on-screen calculations

- Can be used to manipulate objects on the page that are in the DOM (Document Object Model), so you can:
  - Show and Hide elements
  - Animate elements
  - Replace elements with other elements
  - Make requests to the server without reloading the page

Request flow

- A user makes a request to the server, server-side scripting languages (Ruby, PHP, Python) are processed
- HTML, JavaScript, and CSS are returned to the user’s browser
- The browser typically runs each line of HTML line-by-line and by extension each JavaScript included in the page in the order indicated
- If using a JavaScript library, always include it before any code that references it!

Writing and running JavaScript

- Use your text editor of choice to write .js files
- Save them and include them in your HTML to run them

```
<script src=”myscript.js”></script>
```

or write JavaScript directly in your
HTML:

```
<script>
  alert(‘hi!’);
</script>
```

Comments

- There are two different ways to write comments in JavaScript:

```
// This is a single line comment

/* I'm
a
nifty
multi-line
comment! */
```

Alerts and the Console

- To see the output of your script, you could use an alert
- To call an alert, use:
  `alert(“Content of your alert”);`
- To bring up the JavaScript console in Chrome, use Command + Option + J
- Arbitrary JavaScript can be run directly in the console line by line
- To log to the JavaScript console, use:
  `console.log(“This is logged to the console”);`

Your first JavaScript

- Create a JavaScript that sends an alert to the user that says “Hello World”

Data Types in JavaScript

- Why are we putting any text being logged or alerted inside of quotes?

- By encapsulating our text in quotes, we are telling JavaScript the data inside is a **string**, a basic JavaScript data type

Basic Data Types

- String - `“Hello World”`
- Number - `5, 5.5, 1000` (all numbers in JS are floats)
- Boolean - `true`, `false`
- Undefined (no value, var with no value)

Identifying Data Types

- Curious about the type of a piece of data?
- Use the typeof JS function:
  - `yourData = “This is my data.”;`
  - `typeof(yourData);`

    ```
    >string
    ```

Variables in JavaScript

- Remember high school Algebra?

  $x + y = 10$
  If x is 4, what is y equal to?

- In JavaScript, variables are declared with the `var` keyword
  ```
  var x = 10;
  ```

Basic Math

- JavaScript can do all basic math:

```
10 + 10;
> 20

var x = 100;
x * 40;
> 4000
```

Further Data Types: Arrays

- Arrays are used to hold a collection of data, of any data type

```
[“Snoopy”, “Charlie Brown”, “Patty”, “Violet”];
```

- They can hold multiple data types

```
[11, 15, 25, 48, 79, “elephant”];
```

- They can be stored in variables

```
var class_names = [“Julie”, “Sophie”, “Rob”, “John”];
```

Accessing Array Items

- Using an index

```
var myArr = [“Snoopy”, “Charlie Brown”, “Patty”,
“Violet”];
console.log(myArr[0]);
>”Snoopy”
```

- When unsure of an index number

```
 var snoopyPosition = myArr.indexOf(“Snoopy”);
console.log(myArr[snoopyPosition]);
>”Snoopy”
```

A R R A Y C E P T I O N

- An array can store other arrays

```
var toyotas = [“Camry”, “Prius”];
var porsches = [“Camero”, “Boxer”];
var cars = [toyotas, porsches];
console.log(cars[0][0]);>”Camry”
```

- This is called a multi-dimensional array

A note on semicolons

- Semicolons are traditionally used to end statements in JavaScript
- Code will still execute without them
- They should be used to indicate the end of a statement

Exercises

- Create a script with two variables assigned to two numbers. Add them together and output the result to the console

- Try to add two strings together and output the result to an alert

- Create a multi-dimensional array related to a subject that interests you. Output two of the items in sub-arrays to the console

Logic

- The control flow of your program

- Think of logic as a river that branches off in a few different ways

- It allows you to make the computer do the thinking for you!

Testing

- Any test returns a boolean `true` or `false`
- To test if two strings are equal:
  ```
  “stringone” === “string two”;
  >false
  ```
- Using three equals signs instead of two also checks the object type
- If you don’t check type, these are both true:
  - `(10-5) == 5;`
  - `(10-5) == “5”;`
- Can cause bugs down the road!

Testing

- To test if two strings are NOT equal:
  ```
  “stringone” !== “string two”;
  > true
  ```

- To test if one number is greater than another:
  ```
  5 > 10;
  > false
  ```

- `<, >, <=, >=` are also valid comparison operators

If...Then...Else...Then

- Now that we have learned testing, we can implement gates into our program

```
if(5>10){
  console.log(“You’ll never see this 
because 5 is not greater than 10”);
} else{
   console.log(“You will see this because 
5 is not greater than 10”);
}
```

Exercise

- Write a program that checks if a variable is less than 10. If it is, alert the user that their variable is less than 10. If it is not, let the user know what the variable was and that it was greater than 10.

- Try running the script and then changing the variable’s value to see how this affects the program output

- If you have extra time, write a similar program to check if a string stored in a variable is the same as another string

If...Then...**Else If...Then...**Else...Then

- Else if is another condition to evaluate in the case where `if` is not true and you have another condition to look at before `else`:

```
if(5>10){
  console.log(“You’ll never see this because 5 is not 
greater than 10”);
}else if(5===5){
  console.log(“Yes, 5 really is equal to 5.”)
}else{
  console.log(“You will see this because 5 is not 
greater than 10”);
}
```

Functions

- A function is a way to encapsulate code for later use

- It can take **arguments**, which are used as variables inside the function

```
function addTwo(some_number){
  return some_number + 2;
}
console.log(addTwo(4));
>6
```

Functions

- Once a function is declared, it can be called later on (invoked) by calling its name and supplying it with any arguments

```
function alertName(somePersonsName){
  return alert(somePersonsName);
}
alertName(“Zach”);
```

Final Exercises

- Declare a function that takes a name as an argument and tells the user what name they’ve entered, try running it after it has been declared

- Declare a function that takes no arguments but prints something to the console, try running it after it has been declared

- Declare a function that, depending upon which virtual “door” was entered, tells the user they’ve received a different “prize” in an alert. Try running it after it has been declared a few times with each door option.