JAVASCRIPT WORKSHOP

TYLER SCHWEHR

WHAT IS JAVASCRIPT?

- JavaScript is a very important language for running code on a client's device, through the web browser
- As the code is run on the client's computer and not the server, we refer to it as client-side code
- Despite being the language of insanity, it is the only universal client-side language
- Some ~~dickheads~~ *people* even like it so much, that they use it as a server-side language
- JavaScript is kind of object-oriented because it can do a lot of the same stuff as an object-oriented language, but it achieves this in a very weird way
- This is only a very basic introduction; you can find more information on W3Schools
- I'm also focusing primarily on the differences between JavaScript on Python, so I don't spend too much time covering programming basics that you already know

JAVASCRIPT SYNTAX AND CODE BLOCKS

- A feature that might seem different to Python users are code blocks, which is an encapsulated block of code
- Python actually has code blocks but hides them from users by using whitespace to mark them
- In JS code blocks are marked with curly braces, they are required to be used in many situations
- Line comments are denoted with //
- Section comments use /* to start the section and */ to end the section

- camelCase is the norm for JS
- Every line must be ended with a semi-colon
- What you call a list in Python is called an array is JS, there are more differences but you don't really need to know them here

JAVASCRIPT HOW TO ADD SCRIPTS

- Much like CSS, JavaScript can be included as an external script, and internal script, or an inline script
- I strongly advise against using inline scripts, as these become very difficult to manage
- When using either external or internal scripts, I include mine in the <footer> of the document as a shortcut of dealing with load order, there's probably a better way of handling this
- External scripts can help improve loading times due to caching
- The syntax for an internal script is to simply encapsulate the code with the <script> tags, using the attribute type as text/javascript
- For an external script we still use the <script> tags wit the type attribute, but rather than put the code within the tags, we use the src attribute to reference the script
- You can add as many scripts to a single page as you would like

JAVASCRIPT ACCESSING THE WINDOW

- There are three main objects that we can access in script, the document, the browser window, and the browser console
- The browser object model (BOM) can change between browsers but is typically the same across most
- The BOM actually contains everything, and is referenced by default, but it is best practice to call it explicitly when referencing it directly
- We can reference the window by simply using the "window" keyword
- The first method to be aware for interacting with the window is the alert(<string>) method, it creates a popup for the user, displaying the string parameter
- The other handy aspect of accessing the window is to access the screen properties
- Accessing the screen properties allows to view a range of variables on the users' device

JAVASCRIPT ACCESSING THE HTML ELEMENTS

- The HTML is served as a document within the BOM
- Since the window is the default object that we are interacting with, we can exclude the reference to it
- To access the document we simply use the reference "document"
- We can write directly to the document using "document.write()", this will overwrite the entire document
- There are several methods to reference some HTML on the document, but the main one you should be aware of is "getElementById(<string>)"
- This method will return the HTML object with the matching ID, if none are found, it will return NULL
- Note that it returned an object, not a string or anything else that is directly modifiable
- To modify the internal HTML we access the property "innerHTML"
- Putting it all together "document.getElementById('myBanner').innerHTML = '<h1>some new content</h1>'

JAVASCRIPT DEBUGGING IN THE CONSOLE

- The console is an object within the BOM, and can be accessed with the "console" keyword
- The main purpose of using the console is to log information for developers, but not obvious to users
- There are three main console log methods that we can call, each of them takes a string as a parameter
- Each of them escalating the emphasis on the log, furthermore browsers allow us to filter based on emphasis
- The "console.log()" method is the most basic, and simply prints the string to the console, it's useful for seeing what's going on at runtime
- The "console.warn()" method is very similar but marks the log as a warning
- And finally we have "console.error()" which marks the log as an error
- These logs are really handy in debugging, and should be used often

JAVASCRIPT VARIABLES

- Variables are declared with either the keyword "let" for mutable variables, and "const" for immutable variables
- Unlike Python, JS separates out variable declaration and initialisation
- Variable declaration reserves some space in main memory, while initialisation is the first time you store information in that space
- Here is an example of an un-initialised variable "let myVariable;"
- An uninitialized variable will return the value "null", you can check for this with the null keyword
- Variables are only ever declared once, but can have a value assigned to it many times
- JS is a dynamically typed language
- You don't declare what type a variable is, and it can be changed during run time
- Variables have three levels of scope, global, block, and function

JAVASCRIPT OPERATORS

- Most JS operators are the same as Python, so we'll focus on what's different
- JS has the increment and decrement operators using i++ or i--
- JS includes the not equal comparison operator, using !=
- JS has a comparison operator to check both value and type, using ===
- Python technically has a ternary operator, but it's shit, JS uses the ? for a ternary operator
- Python replaces logical operators with keyword, JS uses logical operators,
- && denotes AND
- || denotes OR
- ! Denotes NOT

JAVASCRIPT FUNCTION DECLARATION

- Standard functions work about the same as Python, but the syntax is a little different
- We begin with the "function" keyword, followed by the function name, and the parameters are included in parentheses.
- The code which gets executed in stored in a code block following the function signature

```
function myFunction(a, b, c){
Do some stuff
return
}
```

- JS also allows the use of generic functions, that is a function that has no name
- The declaration is the same but without the function name
- Generic functions are typically either called right away, or are assigned as a variable
- Here is a classic example:

```
document.getElementById("myButton").onclick =
function() {
Do some stuff
}
```

JAVASCRIPT OBJECTS

- So as I mentioned earlier JS is kind of object-oriented, because it has objects
- For JS an object is an unordered collection of properties
- These properties can be variables or methods (functions)
- We declare an object much like we do a variable, but we assign a collection of properties within a code block (example on the right)
- You might be thinking that it looks a lot like JSON, because it is JSON, JavaScript Object Notation

```
myObj = {
  name:"John",
  age:30,
  myCars: {
    car1:"Ford",
    car2:"BMW",
    car3:"Fiat"
  }
}
```

- If else blocks work as you expect but with some syntax changes
- The comparison statement is encased in parentheses
- And the code to be executed is encased in curly braces
- The "else" statement, or the "else if" statement is called after the curly braces

- Earlier I mentioned getting a HTML object by id, which is a unique identifier, meaning only one object has that id
- We can also get objects by less unique identifiers, such as class, or tags
- The function is "getElementsByClassName()" or replace class with Tag
- This returns a collection of objects instead of a single object

JAVASCRIPT IF ELSE BLOCKS & GETTING MULTIPLE ELEMENTS

JAVASCRIPT FOR LOOPS

- Most languages keep C style loops and iteration loops separate by using a different keyword for each…
- JS doesn't do that and the keyword for both is "for", Fuck you JS
- Furthermore there are two types of iteration loops, they both use "for", Fuck you JS
- If you want to iterate through a list, the syntax looks like this "for (object of objects) {}"
- If you want to iterate through the properties of an object we use "for (property in object) {}"
- The C style loop is pretty typical, it has three parameters; a variable declaration, an exit condition, and an increment/decrement statement
- Now is a good time to tell you about the length property, every array has this property to see the length of it
- Here is an example:

```
for (let i = 0; i < myArray.length; i++) {
Some logic
}
```

JAVASCRIPT INTERACTING WITH FORMS

- JS can be used to interact with forms and the data that the user puts into it, this is done by interacting with the Document Object Model (DOM)
- However, JS is run on the client's computer, meaning that they can directly modify the JS, so do not use this for any security purposes
- One of the main things to use JS for is client-side form validation
- JS can also be used to make on page calculators

- Form elements can be selected by element id
- There is a better method however, using the "forms" keyword, we can access any form on the page by using the form name followed by the field name
- Form elements have a property name value which stores the user input
- Here is an example:

"document.forms['myForm']['fname'].value;"

JAVASCRIPT EVENTS

- The DOM handles many different events that happen when a user interacts with a webpage
- You can use JS to react to these events, an example would be writing logic to react to a mouse click on a given element
- This is how we create things like buttons
- To attach logic to an event, you must select the element and assign a function to the event
- Event properties are always lowercase

- There are many different events that you can use, but we're only going to cover three today
- The first is the "onclick" event, this event triggers when the given element is clicked, here is an example of the syntax:

```
document.getElementById("myBtn").onclick = function(){
Some stuff happens
}
```

- The next two are "onmouseover" and "onmouseout", these are useful for dealing with logic regarding mouse hover

JAVASCRIPT INTERACTING WITH CSS

- Interacting with the CSS and modifying it is quite straight-forward, but also a pain to write, so have fun with that
- To access the CSS of any given element, we select the element, and select the style property, from here we can read the property or overwrite it
- I'm pretty sure that every style is stored as a string, I don't think I've ever encountered an integer
- The style property contains every possible style type, but the syntax is slightly different than CSS
- JS doesn't like hyphens, so hypens are removed and it is written in camelCase
- Here is an example of changing the font size:

```
document.getElementById('myText').style.fontSize =
'24px'
```

- Using the getElement method isn't overly demanding but I recommend storing the element in a variable before modifying the styles, for sanity reasons

JAVASCRIPT ANIMATING ELEMENTS

- There are a few key things to keep in mind when animating elements using JS
- Firstly the element that we want to animate must be set to "position: absolute", which is a position absolute to the nearest relative parent
- That means the element must be contained in a parent element with "position: relative"
- Animations should always have an entry condition and an exit condition
- The BOM has a function which allows us to periodically call our own function at a given time interval, the interval is in milliseconds
- We use "window.setInterval(<function>,<integer>)" to begin the trigger, a reference to this interval should be stored in a variable
- We can then use "window.clearInterval(<function>)" to stop the trigger
- By attaching a function which moves an element, we can create an animation

THANK YOU

TYLER SCHWEHR