**JavaScript notes**

GENERAL

**Insert JS code into HTML** using the <script></script>

**Function definition** is done with **function x() { }**

**Insert numbers** into a line of printer text using ${ } where the text is defined in ` ` operands

**DOM** document object model – the structure of the page

**Arrow notation** is a way to shorthand function definition, where **function(input)** can be replaced with **input =>** and **function()** with **() =>**

Here’s an example of how to use functions as inputs to other functions, e.g. forEach():

let text = ""

const fruits = ["apple", "orange", "cherry"];

function myFunction(item, index) {

text += index + ": " + item + "<br>";

}

// method 1 of adding the strings in **fruits** to **text:**

fruits.forEach(myFunction);

// method 2 of adding the strings in **fruits** to **text:**

fruits.forEach((item,index) => {

text += index + ": " + item + "<br>"; })

// method 3 of adding the strings in **fruits** to **text:**

fruits.forEach(function(item,index){

text += index + ": " + item + "<br>"; })

**API (application programming interfaces)** a mechanism which allows you to communicate with an online service in a structured manner.

* Often this data will be in .json format (javascript object notation)

ENTITIES

**Statements –** are lines that end with semicolons.

**Methods –** functions within objects

**Constructor function** – a function which initializes a new object. E.g. :

* function Point(x,y) {this.x = x, this.y = y}
* to create instance:
  + var pt = new Point(1,1)

FUNCTIONS

*numeric*

**isFinite(num)** – checks if a number is finite

**.toFixed (n)** – converts numbers to n decimals

**window.onload = function() {}** will performs the function entries when the window loads up with all its resources (images, scripts, etc.)

**window.localStorage** – will check if local storage is supported

COMMANDS

**this** will refer to the element that received the event in an event handler.

**console.log()** will print the input to the terminal

**alert()** will create an alert on the webpage

**const** variable definition means I will never let the variable equal something else than what it was set to at initialization.

**let** variable definition is just the standard definition initialization.

**document.querySelector()** will in a tag that can reference a name, id, or class.

*Attributes*

.**value** = will get the value of an element specified in a form.

**.innerHTML** = gets the address of the value of a given HTTP entity, the value of which can then be manipulated either by getting or setting it.

.**style =** will change a given element of the object’s style

*Event handlers via accessing object*

**.onclick** = **function** will run a function with value name ‘function’ when the button is clicked

**.onsubmit = function** will run a function when the form is submitted

**.onchange** = **function** will run the function when the entity is changed

**.onmouseover**

**.onkeyup / .onkeyup**

*Event handlers via accessing HTML*

**Onclick = “function1(); function2(); …”** can be specified inside the brackets of any element

**document.querySelectorAll()** will return the array of .js elements with a given type, id, or class.

.**forEach()** will add an event handler for each of the entries in that array.

**setInterval(functionname, timeinterval)** will apply functionname every timeinerval milliseconds

**document.addEventListener( eventname, functionname )** will listen to the eventname event and run the functionname function when the event occurs.

* **Document.addEventListener(‘DOMContentLoaded’, function (){})** – will run function() when the page is fully loaded. This prevents errors where js references HTML objects that haven’t been loaded in yet.
* **document.addEventListener(‘click’, event => { const element = event.target; })** allows the listener to look out for a click and deal with the target of this event saved as element.

**elem = document.createElement(“div”) –** will create a new div element

* **elem.innerHTML** – allows us to access the inner code of the element
* **elem**.**innerText**- returns the text contained by an element and all its children elements
* **elem.appendChild(elemchild)** – will append another element to elem

**document.getElementById(“idname”)** – will find the element with id of idname

EVENT HANDLER

* **Handle event in HTML** e.g. <button onclick=”function()”>Click me</buton>
* **Handle event in JS** e.g. document.querySelector(‘button’).onclick = function;
  + **referencing a .js file in HTML :** add **<script src = "filename.js"></script>** in the title section

**placing a variable in string with `**the value of var is ${var} **`**

SPECIAL HTML INTERACTIVITY

**data-variable** specified in an HTML object can be obtain in .js via dataset.variable when referercing that object.

**document.createElement()** command types in .js will create a given element in HTML

**element.parentElement** will return the element inside which the specified element is

ELEMENTS

**object** – an object is any entity in JS that has a collection of name/value pairs.

* var book = { topic : “JavaScript”, fat : true}
* book.topic and book[“topic”] will both access the value “JavaScript”
* arrays can hold objects, e.g. [book, book]

React

Packages: React, ReactDOM, Babel