HTML MODULE 03
INTRODUCTION TO JAVASCRIPT

Summer 2021 – Web Development using ASP .Net Core MVC

JAVASCRIPT REFERENCE

- Here is a JavaScript reference page. Please use it!
  - https://www.w3schools.com/jsref/default.asp
- Source:
  - https://www.w3schools.com/js/js_intro.asp
- "JavaScript is the world's most popular programming language."
- "JavaScript is the programming language of the Web."
- Why use JavaScript's for web pages?
  - HTML used to define the **content**
  - CSS used to specify the **layout**
  - JavaScript used to program the **behavior**

JavaScript Reference

The references describe the properties and methods of all JavaScript objects, along with examples.

Array	Boolean	Classes	Date
Error	Global	JSON	Math
Number	Operators	RegExp	Statements
String			

HTML DOM Reference

The references describe the properties and methods of each DOM object, along with examples.

Attributes	Document	Element	Events
Event Objects	HTMLCollection	Location	Navigator
Screen	Style	Window	

Web APIs

This references describes the most common Web APIs, along with examples.

Console	Geolocation	History	Storage

WHAT CAN YOU DO WITH JAVASCRIPT?

- Perform **computations**
- **Store information** in variables
- Create **conditional** and **repetitive** statements
- **Add/remove/change** content in a page/element
- **Change CSS styling** applied to elements
- Respond to various **events** (for example: click events, mouse move events, etc.)
- **Validate** forms
- **Alert** users

- Many other …

HOW TO USE JAVASCRIPT

- Inside an HTML page: one can insert JavaScript code between **<script>** and **</script>** tags.
  - Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_whereto
  - Example:

```html
<!DOCTYPE html>
<html>
<body>
<h1> Do you like this course? </h1>
<p id="demo">Undecided</p>
<button onclick="voteYes()"> Yes </button>
<button onclick="voteNo()"> No </button>

<script>
   function voteYes(){  document.getElementById("demo").innerHTML = "Yes";}
   function voteNo(){  document.getElementById("demo").innerHTML = "No";}
</script>

</body>
</html>
```

Do you like this course?

Undecided

Yes No

  - Those scripts can be placed in the <**body**> section, in the <**head**> section, or in both sections.
  - "Placing scripts at the bottom of the <body> element improves the display speed, because script interpretation slows down the display."

- Outside an HTML page: one can use an **external** .js file. Inside the HTML use something like: <script src="myScript.js"></script>
  - To use multiple JavaScript files, use several **script** tags.

- Source: https://www.w3schools.com/js/js_whereto.asp

JAVASCRIPT OUTPUT

- On your own, please read: https://www.w3schools.com/js/js_output.asp

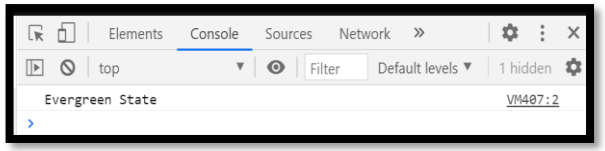
- In particular, one can use JavaScript to write into:
- an HTML element: by using innerHTML
  - Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_dom
    - document.getElementById("demo").innerHTML = "Evergreen State";

Evergreen State

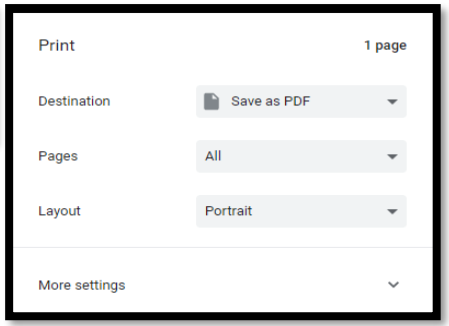
- the HTML output: by using document.write() ← you should only use this for testing
  - Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_write_over
    - document.write("Evergreen State");

- an alert box: by using window.alert()
  - Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_alert
    - ~~window.~~alert("Evergreen State");

- the browser console: by using console.log()
  - Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_console
    - console.log("Evergreen State");



- the printer: by using window.print()
  - Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_output_print



JAVASCRIPT STATEMENTS

- A **JavaScript program** is a list of programming instructions (called **statements**).
  - JavaScript programs are **executed** by a **web browser**.

- A **JavaScript statement** ends with a semicolon (just like in C#)
  - "Ending statements with semicolon is not required, but highly recommended."
  - var count = 0;
  - count = count+1;

- Use curly brackets {…} to define code blocks (statements that are executed together)
  - See an example here: https://www.w3schools.com/js/tryit.asp?filename=tryjs_statements_blocks

- There are also keywords (these are reserved words, they cannot be used for variable names)
  - Example: break, if, else, return, switch, var, …

- Source: https://www.w3schools.com/js/js_statements.asp

JAVASCRIPT COMMENTS (BRIEF)

- Just like you've seen in C#: single- and multi-line comments
- Single-line Comments: //this is a single-line comment
- Multi-line Comments /* this can be a multiline comment */
- Important note: JavaScript is case sensitive!
- Source: https://www.w3schools.com/js/js_comments.asp

JAVASCRIPT VARIABLES

- To **declare a variable**, use the **var** keyword. We use variables to store data values.
  - Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_variables
  - var year; //default value is **undefined** ← only two types of scope: **Global Scope** and **Function Scope**.
  - year = 2021; //**null** is different than **undefined**

- On your own, please read **JavaScript Identifiers** (what is allowed as variable names?)

- JavaScript has **dynamic types**, meaning variables can be used to hold different data types. The same variable can store integer, string,…
  - var carName = "Volvo";
  - carName = 2021;

- ES2015 introduced two important new JavaScript keywords: **let** and **const**. Use them when possible.
  - let year ; //default value is **undefined** ← also provide **Block Scope**
  - year = 2021;

- Some cases to consider:

```
var x = 2;

// Now x is 2

var x = 3;

// Now x is 3
```

```
let x = 2;        // Allowed
let x = 3;        // Not allowed

{
  let x = 4;    // Allowed
  let x = 5;    // Not allowed
}
```

```
var x = 10;
// Here x is 10
{
  var x = 2;
  // Here x is 2
}
// Here x is 2
```

```
var x = 10;
// Here x is 10
{
  let x = 2;
  // Here x is 2
}
// Here x is 10
```

- Sources:
  - https://www.w3schools.com/js/js_syntax.asp https://www.w3schools.com/js/js_datatypes.asp
  - https://www.w3schools.com/js/js_variables.asp
  - https://www.w3schools.com/js/js_let.asp See also: https://www.w3schools.com/js/js_const.asp

JAVASCRIPT OPERATORS

- Please review this as needed (most of them should be familiar to you from Python and/or C#)

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

undefined and **null** are equal in value but different in type

```
typeof undefined          // undefined
typeof null               // object

null === undefined        // false
null == undefined         // true
```

Operator	Description
&&	logical and
||	logical or
!	logical not

- Sources:
  - https://www.w3schools.com/js/js_operators.asp
  - https://www.w3schools.com/js/js_datatypes.asp

JAVASCRIPT FUNCTIONS

- A **function** is essentially a named reusable block of code, designed to do a particular task.
  - See an example here: https://www.w3schools.com/js/tryit.asp?filename=tryjs_functions

```
<script>
function max (num1, num2) {
  let answer = num2;
  if(num1>num2)
    answer= num1;
  return answer;
}
document.getElementById("demo").innerHTML = max(4, 2021);
</script>
```

- Above, note the **function** keyword used to define this function.
- What is the **name of the function** declared above?
- What are the **parameters** of the function above?
- Where do we **call/invoke** that function?
- What **arguments** did we pass to this function?
- What is the name of the **local variable** used in the code above?
- What does is mean for a function to **return** a value?

- Source: https://www.w3schools.com/js/js_functions.asp

JAVASCRIPT DATA TYPES

- On your own <u>you may want</u> to read in more depth the sources shown below.

- JavaScript has **primitive data types**:
  - string: let name="Alex"; ←one can use single or double quotes!
  - number: let pi=3.14159, y=name.length;
  - boolean: let isEven=true;
  - undefined: let x; //x is undefined

- JavaScript also has **complex data types**:
  - function: function f() {}
  - object: [1, 2, 3, 4] ← in JavaScript arrays are objects. Review **indexOf** …
    {name: 'Alex', major: "Computer Science", gpa: 2.85}

- Sources:
  - https://www.w3schools.com/js/js_datatypes.asp
  - https://www.w3schools.com/js/js_strings.asp
  - https://www.w3schools.com/js/js_string_methods.asp
  - https://www.w3schools.com/js/js_numbers.asp
  - https://www.w3schools.com/js/js_number_methods.asp
  - https://www.w3schools.com/js/js_arrays.asp
  - https://www.w3schools.com/js/js_array_methods.asp

JAVASCRIPT OBJECTS

- **Objects** are variables that can contain multiple values.
  - objects are **containers** for **named values** (called **properties**) and **methods**.

- Example:
  - let student1 = {name:"Alex", major:"Art", gpa:3.56, print: **function**() {return this.name + " " + this.major;} };
  - Above, **this** refers to the "owner" of the function.

- To access those values/methods one can use either syntax shown below:
  - student1.major
  - student1["major"]

  - student1.print()

- Source: https://www.w3schools.com/js/js_objects.asp

JAVASCRIPT CLASS

- Skipped …

- A JavaScript **class** is a template for creating JavaScript objects.

```
<p id="demo"></p>

<script>
  class Student {
    constructor(name, major, gpa) {
      this.name = name;
      this.major = major;
      this.gpa = gpa;
    }

    print()
    {
      return  this.name + " " + this.major;
    }
  }

  student1 = new Student("Alex", "CS", 3.56);

  document.getElementById("demo").innerHTML = student1.print();
</script>
```

- Source: https://www.w3schools.com/js/js_classes.asp

JAVASCRIPT JSON

- JSON stands for **J**ava**S**cript **O**bject **N**otation
  - Since the JSON format is text only, it can be written in any programming language.
  - JSON is particularly useful to for **storing** and **transporting** data from a server to a webpage (also see Web API …).

- JSON Syntax Rules:
  - The data is specified using **name/value** pairs
  - These are separated using **commas**
  - **Curly braces** hold objects
  - **Square brackets** hold arrays.

← JSON names require **double quotes** (for both, **names** and **values**!)

- Examples:
  - var student1 = ' {
    "name":"Alex",
    "major":"Computer Science",
    "grades":["A-", "B+","A-"]
    } ';

```
<p id="demo"></p>

<script>
  let student1 = '{  "name":"Alex",                  '+
                 '   "major":"Computer Science",     '+
                 '   "grades":["A-", "B+", "A-"]   }';


  obj = JSON.parse(student1);
  document.getElementById("demo").innerHTML =  obj.name + " " + obj.major + " " + obj.grades[1];
</script>
```

Alex Computer Science B+

- Use the JSON.parse() method to **convert a (JSON) string** into a **JavaScript object**:
- Use the JSON.stringify() method to **convert** a **JavaScript object** into a **(JSON) string**:

```
<p id="demo"></p>

<script>
  let student1 = {
      name:"Alex",
      major:"Art",
      gpa:3.56,
      print: function() {return  this.name + " " + this.major;}
    };


  document.getElementById("demo").innerHTML =  JSON.stringify(student1);
</script>
```

{"name":"Alex","major":"Art","gpa":3.56}

- Source: https://www.w3schools.com/js/js_json.asp

JAVASCRIPT JSON

Find out more about JSON in here: https://www.w3schools.com/js/js_json_intro.asp

Run »

```
<!DOCTYPE html>
<html>
<body>

<h2>Create Object from JSON String</h2>

<p id="demo"></p>

<script>
  let student1 = {
      name:"Alex",
      major:"Art",
      gpa:3.56,
      grades: ["A", "B+", "A-"]
    };


  document.getElementById("demo").innerHTML =  JSON.stringify(student1);
</script>

</body>
</html>
```

Create Object from JSON String

{"name":"Alex","major":"Art","gpa":3.56,"grades":["A","B+","A-"]}

CONDITIONAL (IF/ELSE) STATEMENTS

- This part is very similar to what you've seen in C#:

- What does the following code do?

```
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

- See the following example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_randomlink
  - Where would you use such an approach?

- Source: https://www.w3schools.com/js/js_if_else.asp

SWITCH STATEMENTS

- This part is somewhat similar to what you've seen in C#:
- What does the following code do?

```
switch (new Date().getDay()) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
     day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
}
```

- Source: https://www.w3schools.com/js/js_switch.asp

LOOPS

- JavaScript supports various types of loops, including:
- for - loops through a block of code a **number of times**
  - Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_loop_for_ex

```
<p id="demo">
</p>

<script>
var text = "";
var i;
for (i = 0; i < 5; i++) {
  text += i +"^2 =" + i * i + "<br>";
}
document.getElementById("demo").innerHTML = text;
</script>
```

0^2 =0
1^2 =1
2^2 =4
3^2 =9
4^2 =16

- for/in – iterates through the **properties** of an object
  - Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_object_for_in

```
<p id="demo">
</p>

<script>
var txt = "";
let student1 = {
      name:"Alex",
      major:"Art",
      gpa:3.56,
      grades: ["A", "B+", "A-"]
    };

for (var x in student1) {
  txt += student1[x] + "<br>";
}
document.getElementById("demo").innerHTML = txt;
</script>
```

Alex
Art
3.56
A,B+,A-

- while - loops through a block of code **as long as** a specified condition remains true. Pre check.
  - Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_while

```
<p id="demo">
</p>

<script>
var text = ""
var i = 20;

while (i < 10) {
  text += i+"^2 = " + i*i + "<br>";
  i++;
}


document.getElementById("demo").innerHTML = text;
</script>
```

- do/while - loops through a block of code **as long as** a specified condition is true. Post check.
  - Example: https://www.w3schools.com/js/tryit.asp?filename=tryjs_dowhile

```
<p id="demo">
</p>

<script>
var text = ""
var i = 20;

do {
  text += i+"^2 = " + i*i + "<br>";
  i++;
}while (i < 10);


document.getElementById("demo").innerHTML = text;
</script>
```

20^2 = 400

- Sources:
  - https://www.w3schools.com/js/js_loop_for.asp
  - https://www.w3schools.com/js/js_loop_forin.asp
  - https://www.w3schools.com/js/js_loop_while.asp

LOOPS

- **for/in** statement loops through the **properties of an Object**
- **for/of** statement loops through the **values of an iterable object.**
  - Examples of **iterable** data structures: Arrays, Strings, Maps
  - See an example in here: https://www.w3schools.com/js/tryit.asp?filename=tryjs_object_for_of2

```
<p id="demo"></p>

<script>
let language = "Evergreen State";
let text = "";

for (let x of language) {
  text += x + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>
```

E
v
e
r
g
r
e
e
n

S
t
a
t
e

```
<p id="demo"></p>

<script>
let language = "Evergreen State";
let text = "";

for (let x in language) {
  text += x + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>
```

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14

```
<p id="demo"></p>

<script>
let language = "Evergreen State";
let text = "";

for (let x in language) {
  text += language[x] + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>
```

E
v
e
r
g
r
e
e
n

S
t
a
t
e

- Sources:
  - https://www.w3schools.com/js/js_loop_forin.asp

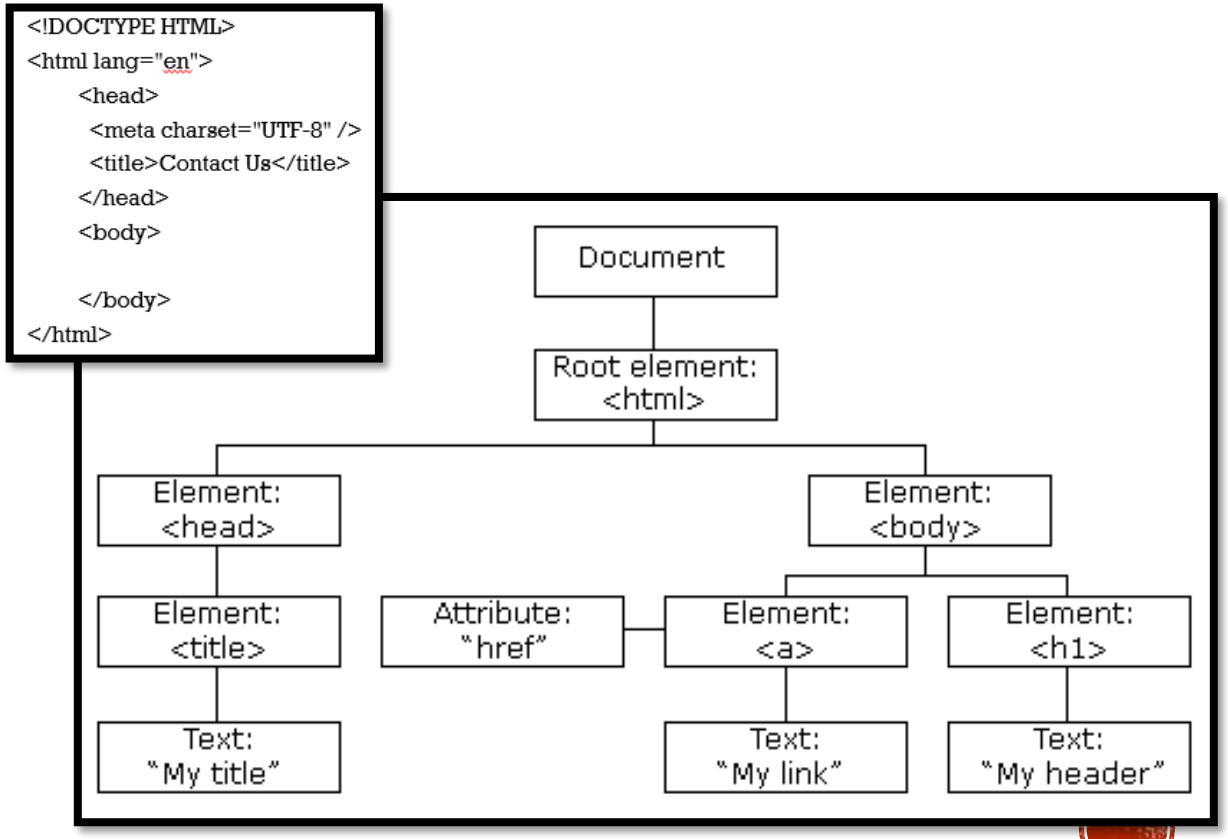
IN-CLASS DEMO

Demonstration: Creating a Simple JavaScript File that Defines Variables, Array and Functions

- Source/Steps
- https://github.com/MicrosoftLearning/20480-Programming-in-HTML5-with-JavaScript-and-CSS3/blob/master/Instructions/20480C_MOD03_DEMO.md

HTML DOM (DOCUMENT OBJECT MODEL)

- When a web page is **loaded**, the browser creates a **Document Object Model** of the page.
  - The **DOM** represents a document with a **logical tree**.
  - The DOM allows us to programmatically (say using JavaScript) modify a document's structure, style, or content.

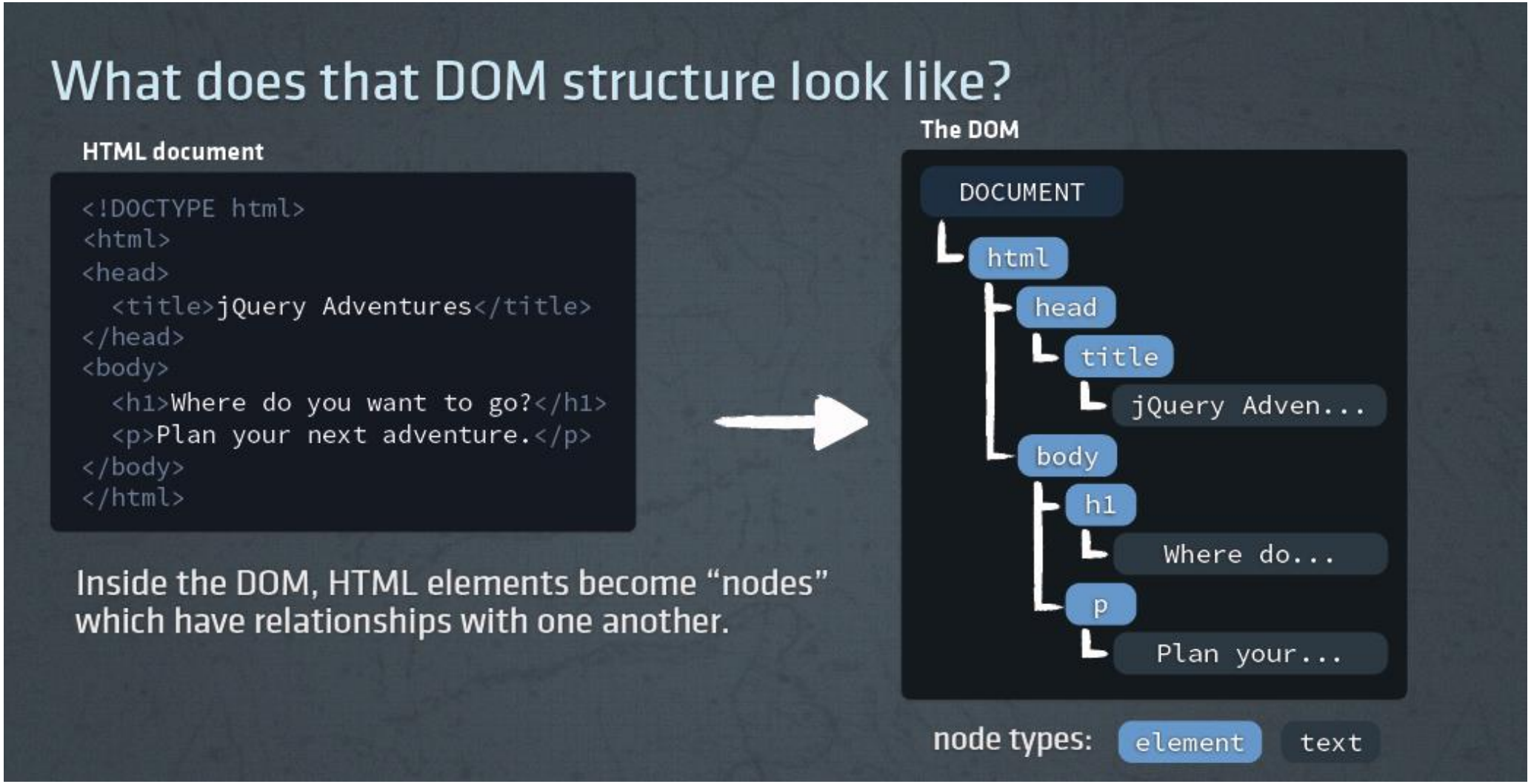


- Using this DOM, JavaScript can be used to:
  - add/change/remove **HTML elements** in a page
  - add/change/remove **HTML attributes** in a page
  - change the **CSS styles** in a page
  - react to/create new **HTML events** in a page
  - validate a page

- Sources:
  - https://www.w3schools.com/js/js_htmldom.asp
  - https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

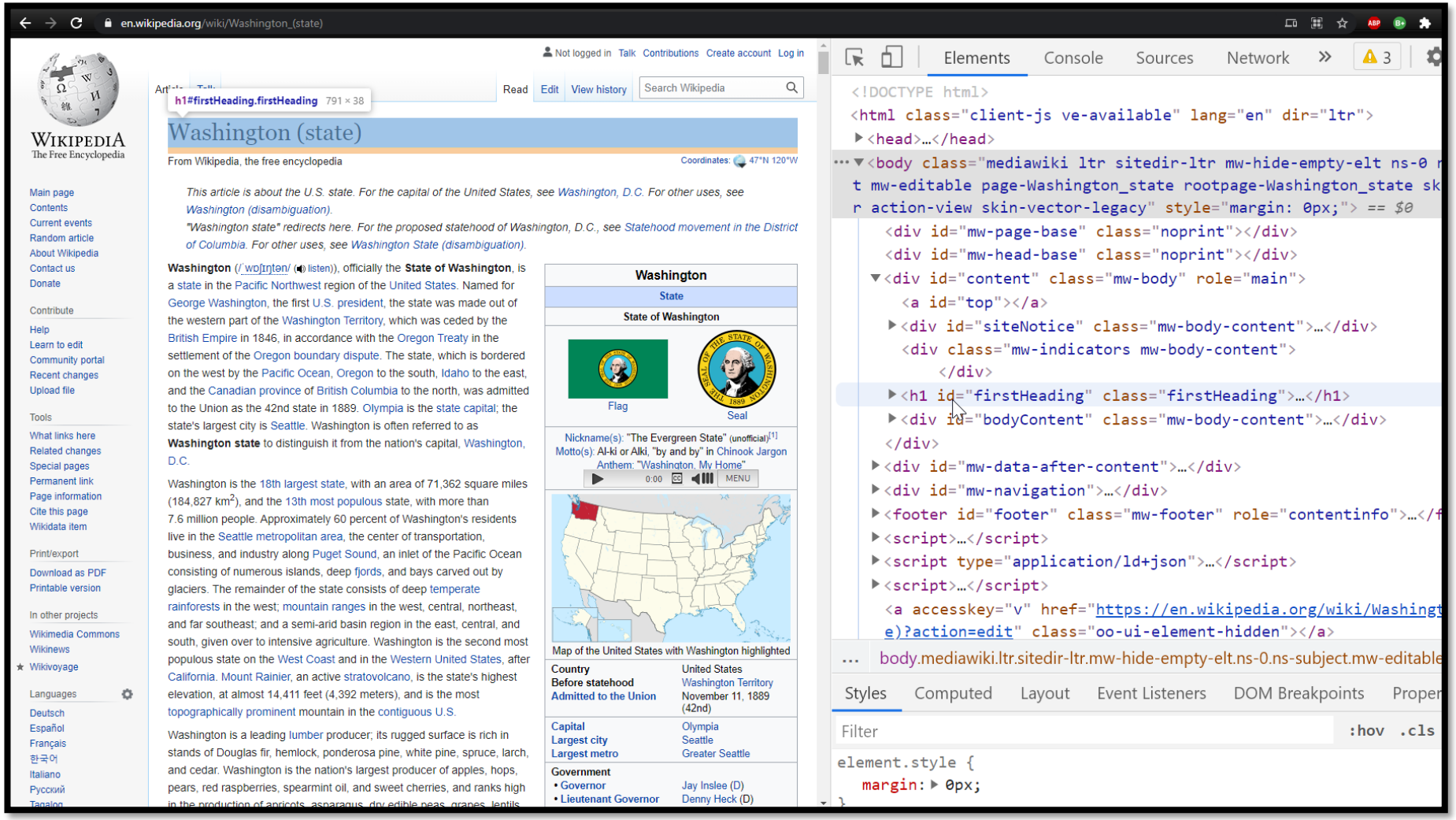
HTML DOM (DOCUMENT OBJECT MODEL)

- Here is another example.
- Source: https://javascript.plainenglish.io/the-dom-of-javascript-848506ebf386



HTML DOM (DOCUMENT OBJECT MODEL)

- Here is yet another example.
- Source: https://en.wikipedia.org/wiki/Washington_(state)



DOM METHODS AND PROPERTIES

- In the DOM, all **HTML elements** are defined as **objects**.

- The **DOM Programming Interface** consists of **properties** and **methods** of each object.
  - **Methods** are **actions** that you **call** to do something
  - **Properties** are **values** that you can **get** (read) or **set** (replace)

- Example:
  - https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_method
  - **getElementById** is a **method** used to find an HTML element
  - **innerHTML** is a **property** used to get or set the content of an HTML elements.
  - **document** **object** represents your web page.

```
<!DOCTYPE html>
<html>
<body>

<h2>My First Page</h2>

<p id="demo"></p>

<script>
    document.getElementById("demo").innerHTML = "Evergreen State";
</script>

</body>
</html>
```

- Source:
  - https://www.w3schools.com/js/js_htmldom_methods.asp
  - https://www.w3schools.com/js/js_htmldom_document.asp

FINDING HTML ELEMENTS

- To access an element in an HTML page, you always start by accessing the **document** object.
  - For collections, you'll need to loop through … https://www.w3schools.com/js/js_htmldom_collections.asp

- Then one can use the following **methods**:
  - document.getElementById(id) ← finds and returns an element, using an element id
  - document.getElementsByName(name) ← finds and returns a collection of elements, using a "name"
  - document.getElementsByTagName(name) ← finds and returns a collection of elements, using a tag name
  - document.getElementsByClassName(name) ← finds and returns a collection of elements, using a class name
  - document.querySelectorAll(selector) ← finds and returns a collection of elements, using a CSS selector

- One can also use the following **properties**:
  - See the sources below for a more complete list!
  - document.body ← returns the <**body**> element
  - document.cookie ← returns the document's **cookie**
  - document.forms ← returns a collection with all <**form**> elements (see here an example)
  - document.head ← returns the <**head**> element
  - document.images ← returns a collection of all <**img**> elements
  - document.links ← returns a collection of all <**area**> and <**a**> elements that have a **href** attribute (see here an example)
  - document.scripts ← returns a collection of all <**script**> elements
  - document.title ← returns the <**title**> element (see here an example)

- Sources:
  - https://www.w3schools.com/js/js_htmldom_document.asp
  - https://www.w3schools.com/js/js_htmldom_elements.asp
  - https://stackoverflow.com/questions/16664205/what-is-the-difference-between-getelementsbytagname-and-getelementsbyname-in-jav

CHANGING HTML ELEMENTS & CSS

- To **modify the content** of an HTML element, use the **innerHTML** property.
  - For example: document.getElementById("par1").innerHTML = "Paragraph contents changed!";

- To **modify the value of an attribute** for an HTML element, use the **.attribute** syntax.
  - For example: document.getElementById("img1").src = "OlympiaLogo.jpg";
  - One can also use the **setAttribute** method:
    document.getElementById("img1").setAttribute("src", "OlympiaLogo.jpg")

- To **modify the style** of an HTML element, use the **.style.property** syntax.
  - For example: document.getElementById("id01").style.color = 'red';

- Examples:
  - https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_image
  - https://www.w3schools.com/js/tryit.asp?filename=tryjs_change_style
  - https://www.w3schools.com/js/tryit.asp?filename=tryjs_dom_color2

- Sources:
  - https://www.w3schools.com/js/js_htmldom_document.asp
  - https://www.w3schools.com/js/js_htmldom_html.asp
  - https://www.w3schools.com/js/js_htmldom_css.asp

ADDING EVENT HANDLERS

- Sources:
  - https://www.w3schools.com/jsref/dom_obj_event.asp
  - https://www.w3schools.com/js/js_events.asp

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

change	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)
click	The event occurs when the user clicks on an element
contextmenu	The event occurs when the user right-clicks on an element to open a context menu
copy	The event occurs when the user copies the content of an element
cut	The event occurs when the user cuts the content of an element
dblclick	The event occurs when the user double-clicks on an element
drag	The event occurs when an element is being dragged
dragend	The event occurs when the user has finished dragging an element
dragenter	The event occurs when the dragged element enters the drop target
dragleave	The event occurs when the dragged element leaves the drop target
dragover	The event occurs when the dragged element is over the drop target
dragstart	The event occurs when the user starts to drag an element
drop	The event occurs when the dragged element is dropped on the drop target
durationchange	The event occurs when the duration of the media is changed
ended	The event occurs when the media has reach the end (useful for messages like "thanks for listening")
error	The event occurs when an error occurs while loading an external file
focus	The event occurs when an element gets focus
focusin	The event occurs when an element is about to get focus
focusout	The event occurs when an element is about to lose focus
fullscreenchange	The event occurs when an element is displayed in fullscreen mode
fullscreenerror	The event occurs when an element can not be displayed in fullscreen mode
hashchange	The event occurs when there has been changes to the anchor part of a URL
input	The event occurs when an element gets user input
invalid	The event occurs when an element is invalid
keydown	The event occurs when the user is pressing a key
keypress	The event occurs when the user presses a key
keyup	The event occurs when the user releases a key
load	The event occurs when an object has loaded
loadeddata	The event occurs when media data is loaded
loadedmetadata	The event occurs when meta data (like dimensions and duration) are loaded
loadstart	The event occurs when the browser starts looking for the specified media
message	The event occurs when a message is received through the event source
mousedown	The event occurs when the user presses a mouse button over an element
mouseenter	The event occurs when the pointer is moved onto an element
mouseleave	The event occurs when the pointer is moved out of an element

ADDING EVENT HANDLERS

- Use syntax similar to:
- element.onclick = functionToCall/EventHandler
  - Example
  - Example ← must show in class
  - Example
  - Example
  - Example

```
<button id="myBtn">Click Me!</button>

<p id="demo"></p>

<script>
document.getElementById("myBtn").onclick = displayText;

function displayText() {
  document.getElementById("demo").innerHTML = "You clicked on "+ Date();
}
</script>
```

```
<button id="myBtn">Click Me!</button>

<p id="demo"></p>

<script>
document.getElementById("myBtn").onclick = function displayText() {
                document.getElementById("demo").innerHTML = "You clicked on "+ Date();
            }
</script>
```

- element.addEventListener('click', functionToCall)

```
<button id="myBtn">Click Me!</button>

<p id="demo"></p>

<script>
document.getElementById("myBtn").addEventListener("click", displayText);

function displayText() {
  document.getElementById("demo").innerHTML = "You clicked on "+ Date();
}
</script>
```

- Sources:
  - https://www.w3schools.com/js/js_htmldom_document.asp
  - https://www.w3schools.com/js/js_htmldom_events.asp
  - https://www.w3schools.com/js/js_htmldom_eventlistener.asp

ADDING HTML ELEMENTS

- To **add a new element** to the HTML DOM:
  - First, you must create the element (element node) ← use the **createElement** method
  - Then you append it to an existing element. ← use the **appendChild** method

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
  <button onClick = "appendNewParagraph()"> Append Paragraph </button>
  <button onClick = "appendTopParagraph()"> Add Top Paragraph </button>
  <button onClick = "removeP1()"> Remove p1 </button>
  <button onClick = "removeP2()"> Remove p2 </button>

</div>

<script>
function appendNewParagraph(){
    var newPara = document.createElement("p");
    newPara.innerHTML = "New Paragraph created";

    var element = document.getElementById("div1");
    element.appendChild(newPara);
}

function appendTopParagraph(){
    var newPara = document.createElement("p");
    newPara.innerHTML = "New Top Paragraph created";

    var element = document.getElementById("div1");
    element.insertBefore(newPara, element.firstChild);
}

function removeP1(){
    document.getElementById("p1").remove();
}

function removeP2(){
    var child = document.getElementById("p2");
    child.parentNode.removeChild(child);
}
```

- To **remove an existing element**:
  - Use the **remove** method
  - Alternatively, one can use the **removeChild** method

- Example

- See here how to navigate between nodes.

- Sources:
  - https://www.w3schools.com/js/js_htmldom_document.asp
  - https://www.w3schools.com/js/js_htmldom_navigation.asp
  - https://www.w3schools.com/js/js_htmldom_nodes.asp

ANOTHER EXAMPLE … IF TIME

- Let's go over the following example:
- What does it do?

```
<!DOCTYPE html>
<html>
<body>

<h1> Type in some text <h1>
<p id='display'></p>


<script>
document.addEventListener("keypress", displayText);

function displayText() {
  var pDisplay = document.getElementById("display");
  pDisplay.innerHTML = pDisplay.innerHTML + String.fromCharCode(window.event.keyCode);
}
</script>

</body>
</html>
```

- Can you improve it? (say allow backspaces?)

IN-CLASS DEMO

Demonstration: Manipulating the DOM with JavaScript

- Source/Steps
- https://github.com/MicrosoftLearning/20480-Programming-in-HTML5-with-JavaScript-and-CSS3/blob/master/Instructions/20480C_MOD03_DEMO.md#demonstration-manipulating-the-dom-with-javascript

LAB/HOMEWORK: DISPLAYING DATA AND HANDLING EVENTS BY USING JAVASCRIPT

- **Module 03**
  - Exercise 1: Displaying Data Programmatically
  - Exercise 2: Handling Events

- You will find the **high-level** steps on the following page:
  https://github.com/MicrosoftLearning/20480-Programming-in-HTML5-with-JavaScript-and-CSS3/blob/master/Instructions/20480C_MOD03_LAB_MANUAL.md
- You will find the **detailed** steps on the following page:
  https://github.com/MicrosoftLearning/20480-Programming-in-HTML5-with-JavaScript-and-CSS3/blob/master/Instructions/20480C_MOD03_LAK.md

- For your homework submit one zipped folder with your complete solution.