Week5 Revisit

1. HTML, CSS
2. JavaScript Basic
3. Adv JavaScript
4. WebService, Types, Creation using SpringBoot
5. JavaLin

HTML – Is a Markup Language (Tag based language <html> </html> )

HTML is not case and space sensitive.

HTML tags are pre-defined.

HTML stands for HyperText Markup Language. It’s very simple and user friendly language.

No Compilation, No Interpretation, No Errors and No Debugging.

HTML will be rendered by browser’s using HTML rendering Engine.

HTML elements are combination of html tags with content. (opening tag+ content + closing tag)

Type of HTML tags 1) Simple tags 2) Self -closing tags (br,hr, input)

Types of HTML elements based on screen occupancy

1) Block level tags/Elements (It will always start in a new line, will occupy complete viewport (100%)

2) Inline elements/tags (It will not start from a new line, it will not occupy 100% viewport – span )

HTML is the language of Web/Internet

CSS – Cascading Style sheet (It’s the styling language of the web/internet)

Three ways of adding css to HTML

1) Inline (Highest preference will be given, will add a style attribute along with the html tag)

2)Internal (Medium preference will be given, will add a style tag in this method)

3) External (lowest preference, but this the recommended method. – External file will be linked by using link tag. A single css file can be used in the multiple html pages)

CSS uses different types of selector

1. Element or tag selector – Tag name will be used directly to apply the styles
2. Class selector (.) – Start with dot [This is the most recommended selector] class attribute will be used to apply style details
3. ID selector (#) – starts with # symbol. ID of the element should be unique. So it’s rarely used as a selector in CSS.
4. Universal Selector (\*) – To apply style for the entire page we use universal selector.

JavaScript -- It’s the programming Language of the Web/Internet

Javascript is a case sensitive, multi – paradigm, single threaded, event loop & object based programming language.

Object Oriented Lang – Paradigm

Procedure Oriented Lang – Paradigm

Functional Programming Lang – Paradigm - (Performing multiple operations in a single line – sorting, searching, filtering and finding the maximum value --- )

Scripting Language – Paradigm (For automation purpose , .bat, .cmd, .exe )

JavaScript is a dynamically typed/ loosely typed lang. (No need to declare the data type of variable upfront)

Official Name of JavaScript is ECMA Script (ES)

ECMA – Electronic Computer Manufacturing Association]

* 7 data types in JS [string, number, boolean, null, undefined, object, Symbol]
* Bigint is a special data type to store lengthy numbers. It ends with n. [Ex: 8736459837455n]
* Null Is not equal to 0 or “”
* Undefined means un-initialized variable

JavaScript Object creation (double quotes is compulsory for non-numeric values in the object properties.

var person1 = {

firstName: "John",

lastName: "Wilson",

age: 23,

};

person1.firstName == person1[“firstName”]

var person2 = new Object();

// Attach properties to person object

person2.firstName = "Ben";

person2["lastName"] = "Tennyson";

person2.age = 18;

for .. in loop is used to traverse properties of an Object

for .. of loop is used to tranverse through an array.

Important operators in javascript – typeof, instanceof (Will help to check whether the object belongs to a particular class/not.

Typeof () – Is used to find out the data type of a variable at run time.

var currentDate = new Date(); //current date

== vs === (Type Coercion)

5 == '5' // Returns true (== check values or same or not)

5 === '5' // Returns False ( === checks the values and also the data type, if both matched then only it returns true. Or else false.

**Type coercion**

Type coercion is the process of converting a value from one data type to another data type.

**Explicit type coercion** - We can explicitly convert the datatype of the variable. *For example:* Number('3'), String (123), Boolean(2)

**Implicit type coercion** - JavaScript is a loosely-typed language, values can also be converted between different types automatically. It usually happens when you apply operators to values of different types. *For example:* '3' \* '2', 2/’5', 123 + ''

## Truthy and Fasly in JavaScript

### Falsy value

In JavaScript, any expressions or value that results in boolean false value, are considered as Falsy. The falsy values/expressions in javascript are:

1. Obviously boolean false is false.
2. Any empty string will be evaluated to false.
3. Any undefined variable will be equal to false.
4. Any null variable will be equal to false.
5. Any numerical expression with result in NaN (not a number) will be equal to false.
6. Any numerical expression with result in zero will be equal to false.

### Truthy value

In JavaScript, any expressions or value that results in boolean true value, are considered as Truthy. Any expression or value other than above listed falsy values – is considered truthy.

**Example:**

'Hello' // truthy

if(1){} // truthy

if(-1){} // truthy

new Boolean(false); // is truthy values because 'object' is alw

The main purpose of JavaScript is

DOM Manipulation (Adding/Removing HTML elements at run-time, Validation)

SCOPE – Is the visibility of a variable in a program.

Types

1. Global Scope – variables declared outside any functions/methods, classes (It can be accessed in the entire file, no restriction
2. Local Scope – Variable defined inside any functions, passed as an argument to the function, It can be accessed only with-in the scope.

# Variable Scope:

The Variable scope defines the lifetime and visibility of a variable. Each variable associated with a scope. The variable can be accessed only within its scope.

## Global Scope

Variables defined outside any function, block, or module have global scope. The global scope variables are accessed everywhere in the application.

Example:

var a = 'Hello World!'; // This is a global variable

function greeting() {

console.log(a);

}

greeting(); // Outputs 'Hello World!'

The global variable's lifetime is throughout the application.

Example:

var app = {}; // A global object

app.foo = 'Homer';

app.bar = 'Marge';

function func() {

console.log(app.foo);

}

func(); // Outputs 'Marge'

## Function Scope

The variable declared in a function is only visible inside that function. var is the keyword to define variable for a function-scope accessibility. These variables cannot be accessed or modified.

function func(){

if(true){

var fruit1 = 'apple'; //exist in function scope

const fruit2 = 'banana'; //exist in block scope

let fruit3 = 'strawberry'; //exist in block scope

}

console.log(fruit1);

console.log(fruit2); // results error - due to it exist in block scope

console.log(fruit3); // results error - due to it exist in block scope

}

foo();

JavaScript Hoisting : The declarations of variable will be automatically moved to top of the program.

In JavaScript, variable declarations made with var and function declarations made with the function keyword are **hoisted** - or moved - to the top of the scope in which they are declared when the JavaScript interpreter parses the code. This means that variables and functions can be used **before they are even declared** as shown below.

function example() {

// var a declaration hoisted here

a = 4;

var a;

a++;

console.log(a); // prints 5

}

// anotherExample declaration hoisted to here

anotherExample(); // no error thrown!

function anotherExample() {

console.log('it works!');

}

let, const keywords are used to create/declare a variable and constant (Block scoped variables)

Two types of creating arrays in javascript

1. Single/one Dimensional Arrays
2. Multi/two or more Dimensional Arrays

// array literal

let cheeses = ['bleu', 'cheddar', 'parmesan', 'brie']

// ["bleu", "cheddar", "parmesan", "brie"]

// new keyword – array object

let primes = new Array(2, 3, 5, 7, 11, 13)

// [2, 3, 5, 7, 11, 13]

Array.length property will give the size of the array

let array = [1, 2, 3]

console.log(array.length) // 3

console.log(array.\_\_proto\_\_) // []

Array Methods

1. Push 2) pop 3) Splice 4) filter 5)shift 6) for each

Template Literals – using backtick to represent a string. Template literals are strings that enclosed within the backtick character(`).

Fucntions – Self Invoking or IIFE (Immediately Invoked Function Expression)

function showMessage() {

alert( 'Hello everyone!' );

}

//function call

showMessage(); // outputs "Hello everyone!" in alert box.

In JavaScript, all function primitive arguments are passed by value. (won’t change original value)

In JavaScript, all function object arguments are passed by value (when changing the assignment using new object obj={‘pet’:’dog’}) & reference (when changing the value by obj.pet = ‘dog’).

Adding JavaScript to a HTML page

1. Inline (using some events)
2. Internal (using script tag)
3. External (Recommended approach – external js file will be linked by script tag with src attribute)

The Scripting Engine of the browser is responsible for running the javascript code.

Types of functions in JavaScript

1. Simple/Normal functions (with function keyword and a name) –

function showMessage() {

alert( 'Hello everyone!' );

}

1. IIFE (Immediately Invoked function Expression /Self-Invoked Functions) –

(function showMessage() {

alert( 'Hello everyone!' );

})();

1. Anonymous/Nameless function [functional Expression = assigning a function to a variable]

var anon = function() {

alert('I am anonymous');

};

var prd = function (a, b) {

return a \* b;

};

anon();

alert("prd = " + prd(2,4));

1. Callback Function [This functions can be called automatically by a asynchronous javascipt code after completing the process -- A callback function is a function that gets executed after another function completes the execution ]

function funcOne(x) { alert("x = " + x); }

function funcTwo(y, callback) {

callback(y);

}

funcTwo(2, funcOne);

callback function with anonymous function

function funcTwo(y, callback) {

callback(y);

callback(y);

}

functionTwo(10, function(x) { alert("x = " + x); })

1. Closure (It  remembers and accesses the variables and arguments of its outer function even after the function return.)

function greeting() {

var message = 'Hi';

function sayHi() {

console.log(message);

}

return sayHi;

}

let hi = greeting();

hi(); // prints "hi" in the console.

## Hoisting of Functions and Variables

Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their scope before code execution.

<script>

//line 1

x = 1;

document.getElementById("p1").innerHTML = x ;

var x;

</script>

JavaScript Hositing only moves the variable declaration to the top, not the variables that are declared and initialized in a single line.

Example:

alert('x = ' + x); // displays x = undefined

var x = 1;

**NOTE:** Since hoisting is only possible with the declaration but not the initialization, var x = 1; not moved to the top of their scope.

1. Arrow Function (anonymous asynchronous function)

One-line arrow function

var hello = () => "Hello World!";

hello(); // output: "Hello World!"

This keyword usage

* This refers the global object [Window] when assigning to a variable

function myFunction() {

var x =this;

return x;

}

X refers to Global Object (Window)

* In Strict mode “this” refers to undefined while return from a method.

"use strict";

var x = this; //Here, this refers to the Global object [object Window]

"use strict";

function myFunction() {

return this; //Here, this is undefined.

}

* Referring as an attribute value “this” refers to the current html element -- **this in Event Handlers**: refers to the HTML element that received the event

<button onclick="this.style.backgroundColor= 'green'">

Click Me!

</button>

let people = function(name, age) {

this.name = name;

this.age = age;

this.displayInfo = function() {

document.write(this.name + " is " + this.age + " years old");

}

}

* **The call() and apply() method**: allows us to write a method that can be used on different objects. Here, person1 object writes its fullName function on person2 object using call() method

Example:

var person1 = {

fullName: function() {

return this.firstName + " " + this.lastName;

}

}

var person2 = {

firstName:"John",

lastName: "Wilson",

}

document.write("Hello, "+ person1.fullName.call(person2));

DOM – Tree Like representation of the HTML page by the browser.

DOM manipulation will happen with the help of this DOM.

DOM stands for Document Object Model. - Each HTML page will be represented as a document object – It’s a tree kind of data structure with one root (html), two child(head & body) and many grand children (p, title, table, form)

DOM methods like “getElementById(“id”), getElementsByName(“name”), getElementsByTagName(“tagName”), querySelectorAll(),

DOM Manipulation

1. Create element
2. Replace element
3. Remove
4. Append a Node
5. InsertBefore

EVENTS – User Interaction with the web page

Types of Events – Page related Events (onpageload, onpageunload)

Keyboard related Events (onkeypress, onkeyup, onfocus, onblur, onkeydown)

Mouse related events (onclick, onsingleclick, ondoubleclick)

EventHandler = EventListner