JAVASCRIPT

JS is an object based programming language. Which is used to create a client side website. This is also used to make web pages dynamic. It is an interpreter language.

History

1995 Brendon Each (netscap programmer)
Mocha
Livescript
Javascript
ECMA Script (European Computer Manufacturing Association)

Advantages

Client side
Validation on browser
Large ecosystem
speed

Disadvantages

Less secure
No hardware support
Debugging Complexity

Types of Adding js

We can add js in html page by using <script> tag.

Internal

```
<script type=”text/javascript”> js code </script>
```

External

```
<script src=”path”  type=”text/javascript”></script>
```

Variables

Variables are container that contains multiples types of data. It is a memory location of specified value.

Types of variables

var – global variable, reassign, re-declare,
let – local variable, can reassign but can’t re-declare (Es 6)
const – immutable (we can’t change it’s value), can’t reassign and re-declare (Es 6)

```
rules of variable names –
        // var 1a = 20;
        var a = 20;
        var a1 = 20;
        // var a 1 = 20;
        var a_1 = 30;
        // var -a = 30;
        var _a = 20;
        // var @a = 20;
        var A = 40;
```

Data types -

Primitive (built-in types)

string – group of characters
number
Booleans – represents true and false value
Undefined
Null

Non-primitive (user defined types)

Array
Object

typeof

we can use typeof() method to get type of any object.

syntax - typeof(object)

Operators

Arithmetic operators

+	addition
-	subtraction
*	multiplication
/	division
%	modulus
++	increment
--	decrement

Comparison operators

==	equal to
!=	not equal to
===	equal to (check both value and data type)
!==	not equal to (check both value and data type)
>	greater than

<	less than
>=	greater than or equal to
<=	less than or equal to

Logical operators

And(&&) – return true if both statements are true.
Or(||) – returns true if one of the statement is true.
Not(!) – returns true when the given statement is false.

Assignment operators

=	a = 10
+=	a += 5 (a = a+5)
-=	a -= 5 (a = a-5)

Ternary operator (?)

Used for single line conditional statement.

```
                 True          False
Syntax -   (condition) ? expression1 : expression2
```

Input / Output

alert – function is used to alert user.
Syntax - alert('message');

confirm

used for take confirmation of user. Returns true if user click on ok else return false.
Syntax confirm(message);

document.write()

used to write directly on html document.
Syntax - document.write(object);

console.log()

write on browser console. Good for debugging.
Syntax - console.log(object);

prompt()

It is used to take input from user.
Syntax - prompt('message : string ' , value : any)

String concatenation

To concat two are more than two strings we can use (+) operator.

Syntax - ‘str1’ + ‘str2’

Conditional statements

- Use if to specify a block of code to be executed **if a specified condition is true**.
- Use else to specify a block of code to be executed **if the same condition is false**.

```
if (condition) {
            block of code
        }
else{
            block of code
        }
```

```
        var a = 30;
        var b = 50;
        var c = 40;
        if (a>b && a>c) {
            document.write('a is greater');
        }
        else if (b>a && b>c)  {
            document.write('b is greater');
        }
        else {
            document.write("c is greater");
        };
```

Nested if else

```
if (condition) {
            if (condition) {
                block of code
             }
             else{
             }
        }
else{
            block of code
        };
```

```
Ex :-
var n = prompt('Enter any number here ')
        if (n >= 0){
            if (n > 0){
                document.write('number is positive');
            }
            else{
                document.write('number is zero');
            }
        }
        else{
                document.write('number is negative');
            }
```

Loops

Loop control statements are used when the section of code can be executed a fixed number of times or until the specified condition not return true.

Types of loops –

While loop

For loop

While loops

Keep repeating any content until the specified condition not return False.

```
Syntax –
      Start
      While (condition/stop){
                  Block of code
                  steps
      }
Ex -
var s = 6; //start
        while (s > 0){  // stop
             document.write(s,'<br>');
             s--;
        }
```

Do while

Statement creates a loop that executes a specified statement at least once, regardless of the condition. It evaluates the condition after executing the statement.

Use - When you need to execute something at least once (e.g., user input validation).

```
Syntax –
do {
  // Code to be executed
} while (condition);
```

Break

used to break the loop according to conditions.

Continue

used to skip the specified condition and continue the loop.

For loop

Keep repeating any content until the specified condition not return False. And we can iterate any iterable object using for loop.

```
Syntax

        for (var i=0; i<=10; i++){
            document.write(i, ' <br>');
        };

Reverse with continue and break statements
for (var i=10; i>=0; i--){
            if (i <= 8 && i>=5){
                continue;
                // break;
            }
            document.write(i, ' <br>');
        };
```

Functions

Functions are sub program which is used to perform an action or task.

```
Syntax -
        function func_name(){
            document.write('block of code');
        };
        func_name();  // We need to call function to use it.

Function with argument
        function demo(a,b){
            document.write(a+b);
        };
        demo(10,50,80);
```

Arguments object

It create an array of given arguments.

```
Syntax -
        function demo(a,b){
         document.write(arguments[3], arguments[1]);
         document.write(arguments.length); // returns the length of arguments
        };
        demo(10,50,80,60,30);
```

Rest parameter (Es 6)

used to identify the rest arguments as an array.

```
Syntax -
        function demo(a,...arg){
            document.write(arg[2]);
        };
        demo(10,50,80,60,30);
```

Closure function

a function having access of parent function is called closure.

```
Syntax -
       function outer(){
            var a = ‘outer func var’;
            function inner(){
                var b = ‘inner func var’;
                document.write(a);
            }
            inner();
            document.write(b);
        }
        outer();
```

Arrow function (Es 6)

An arrow function is shorter syntax compare to normal function. Arrow function is similar to function expression but arrow is always anonymous function.

```
        var y = () =>{
            document.write(a);
        }
        y();
```

iifes (immediately invoked function expressions)

it is invoked immediately after its creation.

```
        (function (){
            document.write("this is iifes");
        })
        ();
```

Callback

callback is a function passed as an argument to another function. When the main function is done, it calls your callback.

1. *A **callback** is like a friend you call back later.* ☎
2. *It's a **function** that gets **invoked** after another function completes its task.*
3. *Think of it as a way to say, "Hey, do this when you're done!"*

```

Ex -
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2, myCallback) {
  let sum = num1 + num2;
  myCallback(sum);
}

myCalculator(5, 5, myDisplayer); // Display the sum
```

Hoisting

This is JS default behavior. JS by default moves all the declaration at the top of current scope. Means we can use hoisted variable before its declarations.
Only var supports hoisting.

```
      Writing time                              compile time

      x = 20 + 5;                               var x;
      document.write(x, '<br>');                x = 20 + 5;
      var x;                                    document.write(x, '<br>');
```

Array

Array is a collection of multiple elements. It is used to store single type multiple data in a single name or variable.
typeof() returns object as a type of array.
Arrays are represented by [].

```
Syntax –

var a = [20,40,50,60];
var b = new Array(20,40,50,60);

index – its represents the position of elements in array

 [20, 40, 50, 60]
  0   1   2    3

Accessing
a[2] // accessing


modification
arr[3] = 40;

delete specific element
delete a[2];


get the length of array
array.length
```

Methods of array

```
concat()- is used to marge two or more arrays.
Array1.concat(array2);



join()- is used to join the elements of an array into a string and return the
        string. Elements will be separated by specified string.
array.join(" - ");


reverse()- is used to reverse the existing arrays.
array.reverse();


slice()- is used to return the specified portion of arrays.
Syntax -
array.slice(start: number,  end: number);
ex. array.slice(2, 6);
```

splice()- is used to change the content of array. We can remove and add element in array.
Syntax -
array.splice(start: number, deleteCount: number, items: any);

pop()- is used to remove last element of array. And return the removed element.
Syntax - array.pop();

push()- is used to add one or more element at the end of existing array.
Syntax -
array.push(elements);

shift()- is used to remove first element of array. Its return removed element.
Syntax -
array.shift();

unshift()- is used to add one or more element at starting of existing array.
Syntax -
array.unshift(elements);

sort()- is used to sort array’s elements.
Syntax -
array.sort();

indexOf()- is returns the index number or position of specified element.
Syntax -
array.indexOf(element);
array.indexOf(element, start);

Array.isArray()- its check the passing argument is array or not.
Syntax -
Array.isArray(arr3);

for of – this statement is used to iterate over iterable object.

Syntax – *for(var i of arr){*
```
            Block of code
}
```

foreach() – method calls a function for each element of array.

Syntax – *arr.foreach(function)*

map() - The map() method creates a new array by applying a function to each element of an existing array.

Syntax:
```
      const newArray = arr.map((element, index, array) => {
            // Perform some transformation on the element
            return transformedValue;
      });
```
Ex-
```
      const numbers = [1, 2, 3, 4];
      const doubled = numbers.map(item => item * 2);
      // Result: [2, 4, 6, 8]
```

The filter() method creates a new array containing only elements that satisfy a specific condition.

Syntax:
```
      const newArray = arr.filter((element, index, array) => {
            // Return true or false based on a condition
            return condition;
      });
```

Ex -
```
      const numbers = [1, 2, 3, 4];
      const evens = numbers.filter(item => item % 2 === 0);
      // Result: [2, 4]
```

Filtering students with grades >= 90:
```
const students = [
  { name: 'Quincy', grade: 96 },
  { name: 'Jason', grade: 84 },
  // ... other student objects ...
];
const topStudents = students.filter(student => student.grade >= 90);
// Result: [{ name: 'Quincy', grade: 96 }, ...]
```

The reduce() method reduces an array to a single value by applying a reducer function.

Syntax:
```
const result = arr.reduce((accumulator, currentValue, index, array) => {
  // Perform some operation using accumulator and currentValue
  return updatedAccumulator;
}, initialValue)
```
Ex -
```
const numbers = [1, 2, 3, 4];
const sum = numbers.reduce((acc, curr) => acc + curr, 0);
// Result: 10
```

The spread operator, denoted by three dots (...), you can use it to **copy all or parts of an existing array into another array. It works with objects too.**

Ex -

```
const numbersOne = [1, 2, 3];
const numbersTwo = [4, 5, 6];
const numbersCombined = [...numbersOne, ...numbersTwo];
// Result: [1, 2, 3, 4, 5, 6]
```

String methods

charAt(pos: number) – returns the element of given position or index number.
Syntax – *str.charAt(4)*

toUpperCase() – its convert string into uppercase
Syntax – *str.toUpperCase()*

toLowerCase() – its convert string into lowercase
Syntax – *str.toLowerCase()*

trim()- is used to remove white space from both side of string (start & end).
Syntax – *str.trim()*

replace(old, new)- is used to replace the string.
Syntax – *str.replace('hii', 'hello')*

split(“ ”)- is used to separate the string and its return a new array.
Syntax – *str.split(' ')*

search(“element”)- is used to search the position of specified string.
Syntax – *str.search('c')*

Template literal

```
    Var a = 10;
    document.write(`xthis is ${a}`)
```

DATE object

Date object provides us method to access and modify date and time.
To create Date object we have to use *new Date()* constructor.

```
Syntax –  var d = new Date();
        d.setDate(15);
        d.setFullYear(2024,1, 17)
        d.setHours(5, 56, 16)
        d.setMinutes(45)
        d.setMonth(4)
        d.setSeconds(3)

        d.getDate()
```

Friday 30-Feb-2024

switch statement

check several possible constant for an expression.

Syntax –

```
        switch (key) {
            case value:
                   document.write()
                break;

            default:
                break;
        }
```

DOM – DOCUMENT OBJECT MODEL

Dom represent a document model with logical tree.
Using dom we can create, navigate, access and modify the document tree.

ELEMENTS SELECTOR -

```
      document.getElementById('');
      document.getElementsByClassName('');
      document.getElemensByTagName('*');
```

parentElement – its return the parent Element of specified element. Its ignore text, comment and white space.

.*firstChild* – returns the first child of specified element and it's include white space, text and comment.

.*firstElementChild* – returns the first child element of specified element and it doesn't include white space, text and comment.

.*lastChild* – returns the last child of specified element and it's include white space, text and comment.

.lastElementChild – returns the last child element of specified element and it doesn't include white space, text and comment.

.childNodes – returns the collection of child nodes of given element and it includes white space, text and comment. The first child will be assign on 0 index.

.children – returns the collection of child nodes of given element and it does not includes white space, text and comment. The first child will be assign on 0 index.

.createElement

Is used to create a specified html element by using their tag names.

document.createElement('tag_name');

.createComment

Is used to create an html comment.

document.createComment('ur comment goes here');

.createTextNode

Is used to create an html Text.

document. createTextNode('ur text goes here');

The **innerText** property returns the plain text content of an element, excluding any HTML tags.

The **innerHTML** property returns the full content of an element, including any HTML tags.
It interprets and formats the content as HTML.
example:

```
const myElement = document.getElementById('myElement');
const htmlContent = myElement.innerHTML; // Returns HTML content
```

Insert Element Node

.appendChild

- is used to append(add) specific node at the end of children elements.

Syntax –

target.appendChild(node)

Inserting Before an Existing Node:

Use **element.insertBefore(newNode, existingNode)** to insert an element before an existing node.

Example:

```
      const element = document.getElementById("div1");
      const child = document.getElementById("p1");
      element.insertBefore(para, child);
```

Remove Element Node

Removing Child Nodes:

If you want to remove a specific child node from its parent, use removeChild():

```
const parentElement = document.getElementById("parent");
const childToRemove = document.getElementById("child");
parentElement.removeChild(childToRemove);
```

Using remove() Method:

The remove() method directly removes an element from the DOM.
Example:

```
const myElement = document.getElementById("myElement");
myElement.remove();

const elementToRemove = document.querySelector(".myClass");
elementToRemove.remove();
```