JavaScript reference sheet

JavaScript Basics

Adding internal JavaScript to HTML

<script type="text/javascript"> //JS code goes here </script>

External JS File

Adding external JavaScript to HTML

<script src="filename.js"></script>

Output

- Writing into an HTML element, using `innerHTML`.
- Writing into the HTML output using `document.write()`.
- Writing into an alert box, using `window.alert()`.
- Writing into the browser console, using `console.log()`.

Html tags in javascript

Console.log(“ <h1> I am header </h1> ”);

Variables

Var /let /const a=10;

Strings

var a=”This is a book”;

Method	Use
`a.length`	The `length` property returns the length of a string:
`a.slice(start, end)`	`slice()` extracts a part of a string and returns the extracted part in a new string.
`a.replace("book","pen");`	The `replace()` method replaces a specified value with another value in a string:
`a.toUpperCase/ toLowerCase ();`	To convert upper / lower case
`trim()`	To remove white spaces
`charAt(index)`	To find character at particular index
`Match()`	To match 2 strings
`a.search(“Term”)`	To search in string
`a.split(“\n”)`	To split string into sub string

Arrays

var fruit = ["element1", "element2", "element3"];

Method	Use
Concat	Joins two or more arrays together.
join()	Converts the array elements to a string.
Reverse()	This method reverses the order of the array elements.
Sort() cars.sort(function(a,b){return a - b}	Sorts the array elements in a specified manner.
toString()	Converts the array elements to a string.
Push()	Add new element at the end
Unshift()	Add new element at the start.
Pop()	Deletes the last element of the array.
Shift()	Delete element from the start.
indexOf()	Return index of character
Includes(“term to search”)	To search term in array
Some(“search at least one term”) every(“every item should be match with)	To match the array elements.

Objects data type

```
const person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

Accessing elements:

Person.firstName;

Person.lastName;

Var name=Person.fullName();

// Create Objects

```
const apples = {name: 'Apples'};

const bananas = {name: 'Bananas'};

const oranges = {name: 'Oranges'};
```

Using map function objects

```
// Create a new Map

const fruits = new Map();

                    // Add new Elements to the Map

fruits.set(apples, 500);

fruits.set(bananas, 300);

fruits.set(oranges, 200);

                    //Getting elements

fruits.get(apples);     // Returns 500
```

Create a Set

```
                    const letters = new Set();
```

Method	Use
new Set()	Creates a new Set object
add()	Adds a new element to the Set
clear()	Removes all elements from a Set
delete()	Removes an element specified by its value.
entries()	Returns an array of the values in a Set object
has()	Returns true if a value exists
forEach()	Invokes a callback for each element
keys()	Returns an array of the values in a Set object
values()	Same as keys()
Size	Returns the element count

Number Methods

JS math and number objects provide several constant and methods to perform

mathematical operations.

Method	Use
toExponential()	Converts a number to its exponential form.
toPrecision(number upto require answer). toPrecision(2)	Formats a number into a specified length.
toString()	Converts an object to a string

parseInt/parseFloat(“10”);	Convert String to integer/ float

Math Methods

Method	Use
Math.PI;	Return Pi value
Math.ceil(x); Math.floor(x);	Rounds a number upwards/ lower to the nearest integer, and returns the
exp(x)	Returns the value of E^x.
log(x)	Returns the logarithmic value of x
pow(x,y)	Returns the value of x to the power y.
random()	Returns a random number between 0 and 1.
sqrt(x)	Returns the square root of a number x
abs(x)	Returns the absolute value of x
cbrt(x)	Returns the cube root of x
cos(x) sin(x) tan(x)	Returns the cos/sin/tan of x
log(x)	Returns the log of value x
pow(whose power, how much power) pow(8,2)	Returns the power of x on y
round(x)	Returns the rounded value of x
cosh(x) sinh(x) tanh(x)	Returns the cosh /sinh /tanh of x
min / max(x , y ,z…)	Returns the maximum / minimum

Date methods

Var d=new Date();

String to date conversion. Var a=”jan-02-2021” ;

d.parseDate(a)

Method	Use
d.getFullYear()	Return full year
d.getMonth();	Return month from 0 to 11 (january=0)
d.getDate()	Returns the date from the date object
d.getDay()	Returns the day from the date object
getHours()	Returns the hours from the date object
getMinutes()	Returns the minutes from the date object

getSeconds()	Returns the seconds from the date object
getTime()	Returns the time from the date object
setYear(2020)	Sets year
setMonth(3)	Set month
setDay(2)	Set day

AddEventListener

```
element.addEventListener('event name like click', ()=>{
// Code to be executed when the event is fired
});
```

Mouse Events

Event	When fired
click	Fired when an element is clicked
oncontextmenu	Fired when an element is right-clicked
dblclick	Fired when an element is double-clicked
mouseenter	Fired when an element is entered by the mouse arrow
Mouseleave	Fired when an element is exited by the mouse arrow

Keyboard Events

Event	When fired
keydown	Fired when the user is pressing a key on the Keyboard.
keypress	Fired when the user presses the key on the keyboard.
keyup	Fired when the user releases a key on the keyboard.

Form Events

Event	When fired
select	Fired when the user select value from form
Focus	Fired when any field gain focus.
Blur	Fired when the focus goes out
change	Fired when the user change any value in the field of form.
Submit	When user click on submit button

Input	When any value is entered. It is similar like keypress.
Invalid	When any field is required and we submit it leaving empty, then this will show error.

Method	Use
Onload	Fired when page load
Onunload	Fired when page unload
onResize	Fired when user resize
onScroll	Fired when user scrolled

Event methods to add events

document.getElementById(*id*).onclick = function(){*code*}

element.addEventListener(“ event “, ()=>{
// Code to be executed when the event is fired
});

<Input type =”text” onclick=function()>

Window Methods

Methods that are available from the window object

Method	**Use**
alert()	Used to alert something on the screen
Blur()	The blur() method removes focus from the current window.
window.close()	The Window.close method use to close the current window.
window.open(" url ");	Opens a new window
window.scrollBy(100, 0);	Scroll 100px to the right
Window.scrollTo(100,0);	Scroll 100px to the down
stop()	Stops the further resource loading

Alerts

Alerts	What to do
window.confirm('Are you sure?')	Open a conformation dialog
Window.alert(“hi”)	Show alert message at top
Window.prompt(“what is your name”);	To take value from the user

Interval Methods to execute functions

Method	**Use**
setInterval(function_name, time in miliseconds);	Keeps executing code at a certain interval
Settimeout(function ,3000)	Executes the code after a certain interval of time
clearInterval(var)	Clears the setInterval. var is the value returned by setInterval call
clearTimeout(var)	Clears the setTimeout. var is the value returned by setTimeout call

Other way to use setinterval and setTimeout functions

```
Setinterval / setTimeout(() => {
// Code to be executed
}, 1000);
```

Get Elements of html

The browser creates a DOM (Document Object Model) whenever a web page is loaded, and with the help of HTML DOM, one can access and modify all the elements of the HTML document.

Method	Use
document.querySelector('css-selectors')	Selector to select first matching element
document.querySelectorAll('css-selectors', ...)	A selector to select all matching elements
document.getElementsByTagName('element-name')	Select elements by tag name
document.getElementsByClassName('class-name')	Select elements by class name
document.getElementById('id')	Select an element by its id
document.body	Getting body of page

document.anchors	Returns all <a> elements that have a name attribute
document.images	Returns all <img> elements
document.title	Returns the <title> element
document.forms	Returns all <form> elements

Creating / Deleting Elements

Method	Use
document.createElement('div')	Create a new element
document.createTextNode('some text')	Create a new text node
document.removeChild(“name”)	Remove child
document.appendChild(“child name”)	Add child at the end
insertBefore(who to insert, where to insert)	Insert before the element

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
const para = document.createElement("p");
const node = document.createTextNode("This is new.");
para.appendChild(node);

const element = document.getElementById("div1");
element.appendChild(para);
</script>
```

Insert before method example

```
const para = document.createElement("p");

const node = document.createTextNode("This is new.");

para.appendChild(node);


const element = document.getElementById("div1");

const child = document.getElementById("p1");

element.insertBefore(para,child);
```

Changing html

Method	Use
.innerHTML=” “;	Changing html
.innerText=” “;	Changing text
.value();	To get and set value (use in form)
.style.property=”value”; Document.getelementById(‘id’).style.backgroundColor=”red”;	Css changing
.setAttribute(attribute, value) document.getElementById("myImage").src = "landscape.jpg";	Attribute changing

Getting values like we get in php through Associative array

```
        //It will get values from the form
const x = document.forms["form1"];   or
  let x = document.forms["form1"]["fname"].value;
        //It will get values of all p tag values
const x = document.QuerySelectorAll["p"];
let text = "";
        for (let i = 0; i < x.length ;i++) {
                //getting through index
          text += x.elements[i].value + "<br>";
                //getting using associative array
        x.elements['name that we have given in form like name=”fname”'].value
          text += x.elements['fname'].value + "<br>";
        }
document.getElementById("demo").innerHTML = text;
```

Animation using JS

```
function myMove() {
  let id = null;
  const elem = document.getElementById("animate");
  let pos = 0;
        clearInterval(id);
                id = setInterval(frame, 5);
  function frame() {
        if (pos == 350) {
                clearInterval(id);
        } else {
                pos++;
                elem.style.top = pos + 'px';
                elem.style.left = pos + 'px';
                }
        }
}
```

Navigating Between Nodes (node mean tags)

You can use the following node properties to navigate between nodes with JavaScript:

parentNode

childNodes[*nodenumber*]

firstChild

lastChild

nextSibling

previousSibling

we have to use nodeValue or nodeName property to get values of nodes

```
<h1 id="id01">My First Page</h1>
<p id="id02">   </p>

<script>
document.getElementById("id02").innerHTML = document.getElementById("id01").nodeName;
</script>
```

Promises

```
Let prom=new Promise(function(resolve,reject){
//code to execute
}
```

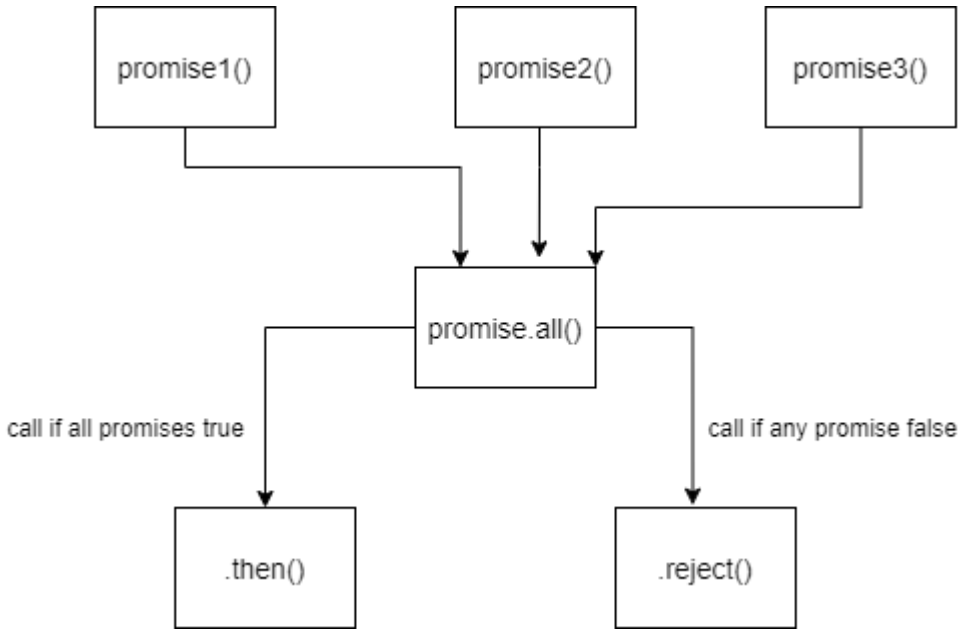
Resolve	Reject
.then	.catch
.then function will run when condition satisfied	.catch function will run in case of error

Example:

```
function prom(a, b) {
 return new Promise(function (resolve, reject) {
    let c = a / b;
    if (c > 0) {
        resolve(c);
    }
    else {
        reject("Answer is less then 0");
    }
 });
}
var first= prompt("enter first number ");
var second= prompt("enter Second number ");

 prom(first,second).then(function (result) {
          document.write("Answer is "+result.toPrecision(2));
 })
                    // Catch will call if rejected run
    .catch(function (err) {
        console.log(err);  });
```

Promise.all function



Async and await methods

Async message are similar like promises but here we don't have to make resolve and reject messages.

```
async function test(){
        document.write("A"+"<br>");
        document.write("B"+"<br>");
  // now it will wait for execution of script of outside of function
          await document.write("C"+"<br>");
        document.write(" D "+"<br>");
}
test();
document.write(" E "+"<br>");
```

BOM methods

The window.screen object can be written without the window prefix.

Properties:

- screen.width
- screen.height
- screen.availWidth
- screen.availHeight
- screen.colorDepth
- screen.pixelDepth

Location Object

Method	Use
window.location.href	returns the href (URL) of the current page
window.location.hostname	returns the domain name of the web host
window.location.pathname	returns the path and filename of the current page
window.location.protocol	returns the web protocol used (http: or https:)
window.location.assign()	loads a new document

Example:

```
document.getElementById("demo").innerHTML =
"Page location is " + window.location.href;
```

Window navigator methods

Method	Use
Navigator.cookieEnabled	returns true if cookies are enabled, otherwise false:
appName	returns the application name of the browser
appCodeName	returns the application code name of the browser:
product	returns the product name of the browser engine:
platform	property returns the browser platform (operating system):
language	returns the browser's language:

Cookies

```
document.cookie = "username=John Doe";
document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";
```

```
function setCookie(cname, cvalue, exdays) {
  const d = new Date();
  d.setTime(d.getTime() + (exdays * 24 * 60 * 60 * 1000));
  let expires = "expires="+d.toUTCString();
  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";
}

function getCookie(cname) {
  let name = cname + "=";
  let ca = document.cookie.split(';');
  for(let i = 0; i < ca.length; i++) {
    let c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length, c.length);
    }
  }
  return "";
}

function checkCookie() {
  let user = getCookie("username");
  if (user != "") {
    alert("Welcome again " + user);
  } else {
    user = prompt("Please enter your name:", "");
    if (user != "" && user != null) {
      setCookie("username", user, 365);
    }
  }
}
```

Web API's

Form Api

Method	Use
checkValidity()	Returns true if an input element contains valid data.
setCustomValidity()	Sets the validationMessage property of an input element.

Example:

```
        function myFunction() {
  const inpObj = document.getElementById("id1");
  if (!inpObj.checkValidity()) {
    document.getElementById("demo").innerHTML = inpObj.validationMessage;
  }
}
```

History Api

Method	Use
back()	Loads the previous URL in the history list
forword()	Loads the next URL in the history list
go(1 or 2 or -1 -2)	Loads a specific URL from the history list

Fetch API

Use to fetch data from api's. It uses promises concept.

```
Async fetch(url).then(function(result){

Return result.json();

}).then(function(data){
Console.log(data);
}).catch(err){
Document.write(err);
}

async function getText(file) {
  let x = await fetch(file);
  let y = await x.text();   or
let y=await x.json();
  myDisplay(y);
}
```

Geo location API

navigator.getCurrentPosition()
this method return an object on success

Property	**Returns**
coords.latitude	The latitude as a decimal number (always returned)

coords.longitude	The longitude as a decimal number (always returned)
coords.accuracy	The accuracy of position (always returned)
coords.altitude	The altitude in meters above the mean sea level (returned if available)
coords.altitudeAccuracy	The altitude accuracy of position (returned if available)
coords.heading	The heading as degrees clockwise from North (returned if available)
coords.speed	The speed in meters per second (returned if available)
Timestamp	The date/time of the response (returned if available)
watchPosition()	Work like gps

```
const x = document.getElementById("demo");
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.watchPosition(showPosition);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}
function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
```

JSON

It is use When we have to get value from api's;
Syntax:

```
{
        Key : value,
}
const obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');
```

Method	Use

JSON.Parse(obj)	To convert into json we can use **parse** method
obj.name; or obj.[“name”];	Getting JSON object values
JSON.stringify()	Convert json to string

```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);

let text = "";
for (let x in myObj) {
  text += x + ", ";
}
```

Array in JSON

```
{
"name":"John",
"age":30,
"cars":["Ford", "BMW", "Fiat"]
}
for (let i in myObj.cars) {
  x += myObj.cars[i];
}
```

Module in js

Using this concept we can use one file variable, functions in other file

For all below work we have to make script tage type=”module”

What id doing	File 1	File 2
Getting file 1 name into file 2	Export var name=”zain”;	Import name from “./file2.js”

Getting sum function in file 2	Export function sum(){
	
	}
		Import sum from “./file2.js”

Getting class of file 1 in file 2	Export class classname{
	
	}
		Import className form “./file1.js”

Getting name and sum in file 2	Export { name,sum };	Import { name, sum } from “./ file1.js”
Getting all data in file 2	Export {name, sum , className}	Import * as data from “./file1.ja”

Conditional Statements

Else-if Statement

```
if (condition1) {
// block of code to be executed if condition1 is true
} else if (condition2) {
// code to be executed
} else {
// block of code to be executed
}
```

Switch Statement

Switch case statement in JavaScript

```
switch(expression) {
case x:
// code block
break;
case y:
// code block
break;
default:
// code block
}
```

Iterative Statements (Loops)

For Loop

For loop syntax in javascript

```
for (var i=1; i<10; i++) {
// code block to be executed
}
```

While Loop

```
while (condition) {
// code block to be executed
}
```

Do While Loop

```
do {
// run this code in block
i++;
} while (condition);
```

try and catch

Try the code block and execute catch when err is thrown

```
try {
  Block of code to try
}
catch(err) {
  Block of code to handle errors
}
finally {
  Block of code to be executed regardless of the try / catch result
}
```

Reguler expression

Expression	Definition
[abc]	Find any of the characters between the brackets
[0-9]	Find any of the digits between the brackets
(x|y)	Find any of the alternatives separated with |

Metacharacter	Description
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b

Quantifier	Occurrence
[ ]?	0 or 1 time
[ ]*	0 to many times
[ ]+	1 to many times
{minimum , maximum}	Minimum time must maximum limit provided.

```
const pattern = /e/;


pattern.test("The best things in life are free!");
```

Class

```
class ClassName {
  constructor() { ..

. }

function_1() { ..

. }
  function_2() { .

.. }
 }

class car {
  constructor(name,year) { ..

this.name=name;

this.year=year;
```

```
}   //  below I have not written function with function name age it is allowed in JS.

age() {

return name+year;
 }
```

Inheritence

It is done with extend keyword

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }
  present() {
    return 'I have a ' + this.carname;
  }
}

class Model extends Car {
  constructor(brand, mod) {
    super(brand);
    this.model = mod;
  }
  show() {
    return this.present() + ', it is a ' + this.model;
  }
}

let myCar = new Model("Ford", "Mustang");
document.getElementById("demo").innerHTML = myCar.show();
```

Getter / Setters

```
 class Car {
  constructor(brand) {
    this.carname = brand;
  }
  get cnam() {
    return this.carname;
  }
  set cnam(x) {
    this.carname = x;
  }
}

let myCar = new Car("Ford");

document.getElementById("demo").innerHTML = myCar.cnam;
```

Static functions

We can call it without making object of that class.

```
static hello() {
```

```
  return "Hello!!";
}
```

Functions outside classes

```
function nameOfFunction () {
// function body
}
```

Arrow function

```
hello = ( parameters ) => {
  return "Hello World!";
}
```

Function with parameters and returning something.

```
function product(a, b) {
  return a * b;
}
```

Advance JavaScript

Symbol variable

```
Var id1=Symbol(12);     Var id2=Symbol(12);
```

It is unique for every id. If we match it like id1==id2 , it returns false.

```
Console.log(id1.desciption);    or console.log(id1.toStrring());
```

Symbol in objects.

```
Let user ={
Name: “zain”,
[age]:22  // it is symbol
}     // accessing symbol
Console.log(user[age]);
```

Symbol can not use in loops and also while converting string to json, it also skip symbol.

For to remove it, convert symbol into string first using description method.

Iterators

They are use to get control at every value of array

```
  var x=["apple","banana","grapes"];
      let y=x[Symbol.iterator]();
let rs=y.next();
   while(!rs.done){
      document.write(rs.value);
      rs=y.next();
   }
```

Generators

They are use to get control of all statements of functions

```
function *gen(){
    console.log("A")
  yield 'yield 1';
  console.log("B");
  yield 'y2';
  }
  let g=gen();
  console.log(g.next());
  console.log(g.next());
```

Strict mode

It check weather there is any syntax error or not. To use this write:

“use strict”;

now this will check all syntax errors and show us.

Like a=10; here declaration error of not writing var or let but browser will ignore it if we are not using strict mode

Ajax

The keystone of AJAX is the XMLHttpRequest object.

Steps:

1. Create an XMLHttpRequest object
2. Define a callback function
3. Open the XMLHttpRequest object
4. Send a Request to a server

States

0: request not initialized
1: server connection established
2: request received
3: processing request
4: request finished and response is ready.

Statuses
200: "OK"
403: "Forbidden"
404: "Page not found"

```
Variable=new XMLHttpRequest();

xhttp.open("GET", "ajax_info.txt",true);

xhttp.send();
```

XMLHttpRequest object properties.

Property	Description
onload	Defines a function to be called when the request is recived (loaded)
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

readyState executed when the readyState changes.

```
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt");
  xhttp.send();
```

```
}
```