Page No:
Date: / /

Javascript

Q.1) What is Javascript, list some features, advantages & disadvantages of it.

Ans-

Javascript is a progeramming language. many of these are related to the way, Javascript is often executed directly in a client's browser. Commonly utilized in web development. it was originally developed by Netscape as a way to feature dynamic & interactive elements on websites.

Javascript may be client-side scripting language, which suggests the ASCII text file is processed by client's browser instead of an online server. this can load webpage without communication with main server with help of Javascript.

* Advantages of Javascript :-

1) Regardless of where you host Javascript, it always gets executed on the client environment to save lots of bandwidth & make execution process fast.

2) The biggest advantage of Javascript having ~~the~~ the ability to support all modern browsers & produce an equivalent result.

3) It is not difficult to start working in Javascript. for this ~~this~~ reason, many of us prefer to start our adventure in the IT sector by learning this language.

4) It gives the power to make rich interfaces.

5) There are some ways to use Javascript through Node.js servers.

* Disadvantages of Javascript :-

1) This may be difficult to develop large applications, although you'll also use the typescript overlay.

2) The main problem in Javascript is that the code is always visible to everyone, anyone can view.

3) No matter what proportion fast Javascript interpreter, Javascript DOM is slow & can be a never-fast rendering with HTML.

4) Javascript usually interpreted differently by different browsers. this makes it somewhat complex to read & write cross-browser code.

5) This continuous conversion takes longer than the conversion of a number to an integer. this increases the time needed to run the scripts & reduces its speed.

Page No:
Date: / /

Q.2) Who developed Javascript & what was the first name of it.

Ans.

Javascript launched in may 1995 by Brendan Eich, who used to work at Netscape. Initially, it wasn't called Javascript, it was given the name Mocha.

The name mocha was choosen by Marc Andreessen, a Netscape founder. the name was changed to liveScript in September 1995. in the same year, December, it received a trademark license from Sun & the name Javascript came into the picture.

The general-purpose core of the language has been embedded in Netscape, internet explorer & other web browsers.

The ECMA-262 specification defined a standard version of the core Javascript language.

1) Javascript is lightweight, interpreted programming language.

2) Designed for creating network-centric application.

3) Complementary to & integrated with Java.

4) Complementary to & integrated with HTML.

5) open & cross-platform.

Page No:
Date: / /

Q.3) what are the key differences between Java & Javascript.

Ans. Java is an OOP programming language & it helps to create applications that function in a virtual machine or browser, while Javascript is an OOP scripting language, also the Javascript code runs on browsers only.

Java	Javascript
1) This is OOP or object-oriented language.	This is an object based scripting language.
2) A stand-alone language.	Not stand-alone, incorporated into HTML program for operation.
3) Strongly typed language is used, & data type of variable is decided before declaring or using it.	language utilised is loosely typed, so that user does not have to worry about the data type before the declaration.
4) code has to be compiled	The code is all text.
5) slightly more complex	Easier in comparison.
6) used to perform complex tasks	Complex tasks cannot be executed.

7)	Large amount of memory is required.	memory consumption is less.
8)	programs saved with ".Java" extension.	program saved with ".JS" extension.
9)	Stored in the client host machine under the "Byte" code.	Stored in host or client as "source" code.
10)	Compiled on the server before it is executed on the client side.	Javascript is interpreted on the client side.
11)	Is static & the code once written can be run on any computing platform.	Dynamic & is a cross-platform language.

Q.4) what are the different datatypes present in Javascript.

Ans.

Javascript provides different data types to hold different types of values. there are two types of data types in Javascript.

Javascript is a dynamic type language, means you don't need to specify type of the variable because it is dynamically used by Javascript engine. You need to use var here to specify the data type.

* Javascript primitive datatypes :- primitive data types are number, string, boolean, Null, infinity & symbol. non-primitive data types is object. the Javascript arrays & functions are also objects.

1) Number :- A number datatypes can be an integer, a floating point value, an exponential value, a 'NaN' or a 'infinity'.

E.g :-

```
var num7 = new Number(5);
Console.log(num1);
typeof(num1);
```

2) String :- The string datatypes in Javascript can be any group of characters enclosed by single or double quotes or by backticks.

E.g :- var str5 = new String("hello");
Console.log(str5);

3) Boolean :- The boolean datatypes has only two values, true & false. it is mostly used to check a logical condition. thus booleans are logical datatypes which can be used for comparison of two variables or to check a condition.

E.g :-

```
var a = 5;
var b = 6;
a <= b;
if (a>b){
  alert (a is a smaller number than b);
}
```

4) Undefined :- undefined datatype means a variable that is not defined. the variable is declared but doesn't contain any value.

E.g :- var a;
console.log(a);

5) Null :- The null is Javascript is datatype that is represented by only one value, the 'null' itself. a null value means no value.

E.g :-

```
var a = null;
console.log(a);
```

* Non-primitives datatype :- The 'object' is non-primitive datatypes in Javascript. arrays & functions in Javascript belong to the 'object' datatype.

1) object :- lets create an object literal. An object in Javascript contains key-values pairs in its address. when we refers to obj1, we are actually reffering to the address in memory which contains the value {a:5, b:6}; instead of the value {a:5, b:6};

E.g :-

```
var obj1 = {a:5, b:6};
console.log(obj1);
```

2) Array :- An array in Javascript is an object datatypes. An array contains more than one value with a numerical index, where the index starts from 0. thus it holds its value in a key-value pair.

E.g :-

```
arr1[0] = 4;
console.log(arr1);
typeof(arr1);
```

Q. 5) How to create an objects in Javascript.

Ans.

A Javascript object is an entity having state & behaviour. For e.g car, pen, bike etc.

Javascript is an object-based language. everything is an object in Javascript.

* Creating objects in Javascript

There are 3 ways to create objects.

1) By object literal.
2) By creating instance of object directly.
3) By using an object constructor.

1) Javascript object by object literal :-

The syntax of creating object using object literal is given below.

Syntax :- object =
{ property1 : value1, property2 : value2 ..... propertyN : value N };

E.g :-
```
<script>
emp = { id:102, name:"Nareesh", salary:40000 }
document.write (emp.id+" "+ emp.name + " " +emp.salary);
</script>
```

2) By creating instance of object :-

Syntax : var objectname = new Object();

The new keyword is used to create object.

E.g.
```
<script>
var emp = new Object();
emp.id = 101;
emp.name = "Ravi";
emp.salary = 10000;
document.write (emp.id+" "+emp.name + " " + emp.salary);
</script>
```

3) By using an object constructor :-

To create function with arguments. each argument value. can be assigned in the current object by using this keyword.

The this keyword refers to the current object.

E.g :-
```
<script>
function emp (id, name, salary){
this.id = id;
this.name = name;
this.salary = salary;
}
e = new emp (103, "Vim", 30000);
document.write (e.id+" "+e.name+" "+ e.salary);
</script>.
```

Page No:
Date: / /

Q.6) How to create an array in Javascript.

Ans.

Javascript array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in Javascript.

1) Javascript array literal :-

Syntax :- var arrayname = [value1, value2 ... valueN];

E.g :- <script>
var emp = ["Sono", "Kimel", "Ratan"];
for (i=0; i<emp.length; i++){
document.write (emp[i] + "
");
}
</script>

2) Javascript array directly (new keyword) :-

Syntax : var arrayname = new Array();

new keyword is used to create instance of array.

E.g :- <script>
var i;
var emp = new Array();
emp[0] = "Aman";
emp[1] = "Shrutika";
emp[2] = "peiyanshi";
for (i=0; i<emp.length; i++)
{
document.write (emp[i] + "
");
}
</script>

3) Javascript array constructor (new keyword) :-

To create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

E.g :- <script>
var emp = new Array ("Jai", "Raghav", "Smith");
for (i=0; i<emp.length; i++)
{
document.write (emp[i] + "
");
}
</script>

Page No:
Date: / /

Q.7) Difference between client side Javascript & server side Javascript.

Ans.

web-browsers execute client-side scripting. it is used when browsers have all code. web-servers are used to execute server-side scripting.

	Client-side Scripting	Server-Side Scripting.
1)	Source code is visible to the user.	Source code is not visible to the user because its output of server-side is an HTML page.
2)	Its main function is to provide the requested output to the endusers.	Its primary function is to manipulate & provide access to the respective database as per the request.
3)	It usually depends on the browser and its version.	In this any server side technology can be used and it does not depend on the client.
4)	It runs on the user's computer. there are many advantages linked with this like faster.	It runs on the webserver.
5)	~~There~~ response times, a more interactive application.	The primary advantage is its ability to highly customize response requirement, access rights based on user.
6)	It does not provide security for data.	It provides more security for data.
7)	HTML, CSS & Javascript are used.	PHP, Python, Java, Ruby are used.
8)	No need of interaction with the server.	It is all about interacting with the servers.

Q.9) Difference between Javascript & VBscript.

Ans.

Javascript is lightweight & object oriented scripting language used to create dynamic HTML pages with interactive effects within a webpage.

VBscript is an active scripting language introduced by microsoft. It is developed based on visual Basic & is also called a visual Basic scripting language.

	Javascript	VBScript.
1)	It was developed by Netscape.	It was developed by Microsoft.
2)	It does job as a client-side scripting language	It does job as both server-side & client-side scripting language.
3)	It supports all web browsers & it is a default scripting language in mostly browsers.	It does not support all browsers, supports only internet explorer.
4)	It uses the same operator for various operations.	while it uses different operators for various operations.
5)	It uses curly braces for declaration of functions.	It uses function & end function for declaration of functions.
6)	Javascript file has the file extension ".js".	VBScript file has the file extensions ".vbs" or ".vba".
7)	It is simpler & easier to learn Javascript.	It is difficult for beginners to learn VBScript as compared to Javascript.

Page No:
Date: / /

Q.10) what is a name function in Javascript & how to define it.

Ans. A named function in Javascript is a function that has been defined with a name using the special syntax for declaring functions.

* define a named function in Javascript :-

To define a named function in Javascript we use the keyword function, give the function a name; & then write the function declaration.

```
function highfive()
{
    return "yeah!";
}
```

You can see that we start out by using the function keyword.

In this case the name is highfive():
The rest of code just returns string "yeah!" when the highfive named function called.

Page No:
Date: / /

Q.11) what is name can you assign an anonymous function to a variable & pass it..

Ans.

The meaning of the word 'anonymous' defines something that is unknown or has no identity. in Javascript, an anonymous function is that type of function that has no name or we can say which is without any name when we create an anonymous function, it is declared without any identifier. it is the difference between a normal function & an anonymous function.

* implementation of an anonymous function :-

```
syntax:- let x = function() {
              console.log("It is an anonymous function");
          }
```

The function is created for displaying message as its output. we have used the function keyword. which is used when we create any function in Javascript, & the function is assigned to a variable x using 'let'.

```
E.g:- function normale() {
          console.log('It is normal function');
      }
      normale();
```

Page No:
Date: / /

* Use of an anonymous functions :-

passing an anonymous function to other function as its argument.

E.g :-
```
setTimeout(function() {
    console.log('Execute later after 1 second')
}, 1000);
```

The function setTimeout() will output the anonymous function after 1 second. We have created an anonymous function & passed it to the setTimeout() as its argument. Inside it, when the code gets executed, it will print the statement after a second of execution time.

Page No:
Date: / /

Q.12) What are the scopes of variables in Javascript.

Ans:

The scope manages the availability of variables or we can also say that it determines the accessibility of variables.

* Types of scopes in Javascript :-

1) Block scope :- Earlier Javascript had only global scope & function scope. let & const are the two new important keyword that were introduced by the ES6 & these two keywords provide block scope in Javascript.

let keyword :-
```
{
    let x = 2;
}
x can not be used here.
{
    var x = 2;
}
x can be used here.
```

Variables declared with var keyword cannot have block scope & they can be declared inside {} block & can be accessed from outside the block.

2) Function Scope :- Javascript has function scope & each function creates a new scope. variables defined inside a function are not accessible from outside the function & variables declared with var, let & const are quite similar when declared inside a function.

E.g :-

var Keyword :-

```
function myfunction() {
    var firstName = "Naresh";
}
```

Const Keyword :-

```
function myfunction() {
    const firstName = "Keithar";
}
```

3) local Scope :- Variable declared inside a function become local to the function. local variables are created when a function starts & deleted when the function is executed.

E.g.

```
<!DOCTYPE html>
<html>
<head> </head>
<body>
<h2 style = "color:green">
Nik Jain
</h2>
<Sceipt>
function foo() {
var x = '1';
console.log ('inside function :',x);
}
foo();
console.log(x);
</Sceipt>
</body>
</html>
```

4) Global Scope :- variables declared globally have global scope & global variable can be accessed from anywhere in program. similar to function scope variables declared with var, let & const are quite similar when declared outside the block.

let Keyword :

```
let x = 2;
```

Const Keyword :

```
const x = 2;
```

var Keyword :

```
var x = 2;
```

E.g :-

```
<!DOCTYPE html>
<html>
<head> </head>
<body>
<h2 style = "color:green"> Nik Jain. </h2>
<sceipt>
var x = '1'
const y = '2'
let z = '3'
```

```
console.log(x);
console.log(y);
console.log(z);
    function getNo() {
        console.log(x);
        console.log(y);
        console.log(x);
    }
    getNo();
</script>
</body>
</html>
```

Q.13) Name some of the built in methods & values returned by them.

Ans:

A Javascript method is property containing a function definition. In other words, when the data stored on an object is a function we call that a method.

The method is what an object does.

Specific built-in objects have different built-in methods which we can use.

Some useful methods for Date, Math, String, Array & object objects.

1) Date :- Javascript date objects represent a single moment in time in a platform-independent format. Date objects contain a Number that represents milliseconds since 1 January 1970 UTC.

1) Date() :- when called as a function, returns a string representation of current date & time.

2) new Date() :- when called as a constructor returns a new Date object.

2) Math :- Math is built-in object that has properties & methods for mathematical constants & functions.

1) Math.round(num) :- returns the provided number rounded to the closest integer.

2) Math.floor(num) :- Rounds down to the previous integer.

3) Math.PI :- Not technically a method, but property! Handy if you need PI.

3) String :- The string object is used to represent & manipulate a sequence of characters. Strings are useful for holding data that can be represented in text form, & Javascript provides a number of useful string built-in methods.

E.g. console.log('hello'.toUpperCase());

1) String.length() :- Returns the length of string.

2) String.toUpperCase() :- Convert all characters to the uppercase.

3) String.toLowerCase() :- Convert all characters to the lowercase.

4) String.replace(searchFor, replaceWith) :- finds every instance of ~~each~~ search for substring & replaces it with the given new substring.

4) Array :- The simplest way to describe Arrays is that they list-like objects. Something super important about arrays is that they are indexed, meaning that you can access specific values by the index or the location they hold in the list.

E.g :- let fruits = ['Apple', 'orange', 'cherry'];
console.log(fruits.length);

1) array.push() :- Allows us to add items to the end of an array. push() changes, or mutates, the array.

Eg: const item = ['unit1', 'unit2'];
item.push('unit3', 'unit4');
console.log(item);

2) array.pop() :- Removes the last item of an array. It does not take any arguments, its simply removes the last element of the array & it returns the value of last element.

E.g :- const Nos = ['Nos1', 'Nos2'];
const removed = Nos.pop();
console.log(Nos);
console.log(removed);

Page No:
Date: / /

Q.14) List out the different ways an HTML element can be accessed in a Javascript code.

Ans.

Users need to manipulate the HTML element without changing the code of the HTML. in this scenario, users can use Javascript to change HTML elements without overwriting them.

From the DOM, users can access HTML elements in five different ways in Javascript.

1) Get Element By Id :- Generally, most developers use unique ids to the particular HTML element before accessing the HTML element with the id. users can use getElementById() method to access HTML element using the id.

Syntax :- document.getElementById('demo');

2) Get Element By className :- In Javascript, getElementByClassName() method is useful to access the HTML element using the className. developee can use single className multiple times in particulars HTML document.

Syntax :- document.getElementByClassName (element-classnames);

Page No:
Date: / /

3) Get element by Name :- In Javascript, getElementByName() method is usefull to access the HTML elements using the name. Here, the name suggests the name attribute of the HTML element. users can get the length of the collection using built-in length method.

Syntax :- document.getElementByName (element-name);

4) Get elementBy TagName :- In Javascript, getElement By TagName() methods is useful to access the HTML elements using the tag Name. This method is the same as the getElementsByTagName method.

Syntax :- document.getElementByTagName (Tag-name);

5) Get element by css selector :- users can select the HTML elements using the different CSS selectors such as class, id, & tag name at a single time. HTML elements can be retrieval using CSS selectors in two ways. the querySelector() method returns the first element that matches the particular CSS selector. The querySelectorAll() method returns all elements that matches the particulars CSS selector.

Syntax :- document.querySelector (selectors);
document.querySelectorAll (selectors);

Page No:
Date: / /

Q.15) In how many ways Javascript code can be involved in HTML file.

Ans-

There are following three ways in which users can add Javascript to HTML pages.

1) Embededding code :- To add the Javascript code into the HTML pages, we can use the <sceipt> ... </sceipt> tag of the HTML that wrap around Javascript code inside the HTML program. users can also define Javascript code in the <body> tag or <head> tag because it completely depends on the structure of the webpage that the users use.

```
E.g :- <!DOCTYPE html>
       <html>
       <head></head>
       <title> title </title>
       <sceipt>
          document.write("welcome");
       </sceipt>
       </head>
       <body>
       <p> In this e.g we saw how to add
        JS in head </p>
       </body>
       </html>
```

Page No:
Date: / /

2) Inline code :- Generally, this method is used when we have to call a function in the HTML event attributes there are many cases in which we have to add Javascript code directly eg. onMouse, event, onclick etc.

```
E.g. <!DOCTYPE html>
     <html>
     <head>
     <title> title </title>
     </head>
     <body>
     <p> in this e.g. we saw how to use
       inline JS </p>
     <p> <a href="#" onclick="alert('welcome');
          "> click me </a>
     </p>
     </body>
     </html>
```

3) External file :- we can also create a separate file to hold the code of Javascript with the (.JS) extension & later incorporate / include it into our HTML document using the src attribute of the <sceipt> tag it. becomes very helpful if we want to use same code in multiple HTML document.

Page No:
Date: / /

E.g :- <html>
<meta charset = "utf-8"> <head>
<title> including a External JS </title>
</head>

Page No:
Date: / /

Q. 16) what is an event bubbling in Javascript.

Ans.

Event Bubbling is concept in the DOM (Document object Model). It happens when an element receives an event, & that event bubbles up to its parent & ancestor element in the DOM tree until it gets to the root element.

HTML :- <body>

<button> click me! </button>

</body>

CSS :- body {
padding : 20px;
background-color : pink;
}
div {
padding : 20px;
background-color : green;
width : max-content;
}
span {
display : block;
padding : 20px;
background-color : blue;
}

Page No:
Date: / /

* How to handle Events that bubble :-

The "Event Bubbling" behaviour makes it possible for you to handle an event in a parent element instead of the actual element of the actual element that received the event. let's create some event listeners & handlers :

```
const body = document.getElementByTagName
  ("body")[0]
const div = document.getElementByTagName
  ("div")[0]
const span = document.getElementByTagName
  ("span")[0]
const button = document.getElementByTagName
  ("button")[0]

body.addEventListener('click', () => {
  console.log("body was clicked");
})

div.addEventListener('click', () => {
  console.log("div was clicked")
})

span.addEventListener('click', () => {
  console.log("span was clicked")
})

button.addEventListener('click', () => {
  console.log("button was clicked")
})
```

Page No:
Date: / /

8.17) what are Javascript events, explain.

Ans:

Javascript interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

1) onclick Event :- This is most frequently used event type which occurs when a user clicks the left button of his mouse.

E.g :-

```
<html>
  <head>
    <script type="text/javascript">
      function sayHello(){
        alert("Hello world")
      }
    </script>
  </head>
  <body>
    <p> click the following button </p>
    <form>
      <input type="button" onclick="sayHello()"
      value="sayHello" />
    </form>
  </body>
</html>
```

2) onsubmit Event :- onsubmit is an event that occures when you try to submit a form. You can put your form validation against this event type.

Eg:-
```
<html>
<head>
<sceipt type = "text/Javasceipt">
function validation () {
all validation goes here
ceturn either true or
false
}
</sceipt> </head>
<body>
<form method = "POST" action = "1.cgi"
onsubmit = "ceturn validate ()">
<input type = "submit" value = "
submit" />
</form>
</body>
</html>
```

3) onmouseover & onmouseout :- These two event types will help you create nice effects with images or even with text as well. the onmouseover event triggers when you bring your mouse over any element. & the onmouseout triggers when you move your mouse out from that element.

Eg:-
```
<html>
<head>
<sceipt type = "text/Javasceipt">
function over () {
document.write ("mouse over");
}
function out () {
document.write ("mouse out");
}
</sceipt>
</head>
<body>
<p> Being your mouse inside the
division to see the result </p>
<div onmouseover = "over ()"
onmouseout = "out ()">
<h2> This is inside the division
</h2>
</div>
</body>
</html>
```

Page No:
Date: / /

~~p.17~~

1) onabort :- Triggers on an abort event.

2) onblur :- Triggers when the window loses focus.

3) onchange :- Triggers when an element changes

4) onclick :- Triggers on a mouse click.

5) ondrag :- Triggers when an element is dragged.

6) ondrop :- Triggers when dragged element is being dropped.

7) onfocus :- Triggers when the window gets focus.

8) onkeyup :- Triggers when a key is released.

Page No:
Date: / /

p.18) find the grade for input marks.

Ans.

index.html :-

```
<!DOCTYPE html>
<html>
<head>
<title> Calculate grade from input marks </title>
</head>
<body>
<div class = "container">
<h1> Student grade system </h1>
<div class = "screen-body-item">
<div class = "app">
<div class = "from-group">
<input type = "text" class = "from-control" placeholder = "Chemistry"
id = "Chemistry" />
</div>
<div class = "from-group">
<input type = "text" class = "from-control" placeholder = "Maths"
id = "Maths" />
</div>
<div class = "from-control">
<input type = "text" class = "from-control"
placeholder = "physics" id = "physics" />
</div>
<div>
<input type = "button" value = "Show percentage" class = "from-button"
onclick = "calculate()" />
</div>
</div>
</div>
</div>
```

```
<div class = "form-group showdata">
  <P id = "showdata"></P>
</div>
</div>
  <script src = "main.js"></script>
</body>
</html>
```

main.js :-

```
const calculate = () => {
  let chemistry = document.querySelector
  ("#chemistry").value;
  let Maths = document.querySelector
  ("#Maths").value;
  let physics = document.querySelector
  ("#physics").value;
  let grades = "";

  let totalgrades =
    parseFloat(chemistry) +
    parseFloat(Maths) +
    parseFloat(physics);

  let percentage = (totalgrades/300) * 100;
  if (percentage <= 100 && percentage >= 80)
  {
    grades = "A";
  } else if (percentage <= 79 &&
    percentage >= 60) {
      grade = "B";
    } else if (percentage <= 59 &&
    percentage >= 40) {
      grades = "C";
    } else {
      grade = "F";
    }

    if (chemistry == "" || Maths == "" ||
      physics == "") {
        document.querySelector("#showdata")
      .innerHTML = "Please enter all the
        fields";
      } else {
        if (percentage >= 39.5) {
          document.querySelector(
          "#showdata").innerHTML =
        `out of 300 your total is
        ${totalgrades} & percentage is
        ${percentage}%. <br>
        your grade is ${grades}.
        you are pass.`;
        } else {
          document.querySelector(
            "#showdata"
          ).innerHTML =
            `out of 300 your total is
          ${totalgrades} and percentage is
          ${percentage}%. <br>
          your grade is ${grades}. you are
            fail.`;
          }
      }
};
```