language

2. Enumerate the differences between Java and JavaScript?

Java is a complete programming language. In contrast, JavaScript is a coded program that can be introduced to HTML pages. These two languages are not at all inter-dependent and are designed for different intent. Java is an object-oriented programming (OOPS) or structured programming languages like C++ or C, whereas JavaScript is a client-side scripting language.

3. What are JavaScript Data Types?

Following are the JavaScript Data types:

- Number
- String
- Boolean
- Object
- Undefined

4. What is the use of isNaN function?

isNan function returns true if the argument is not a number; otherwise, it is false.

5. Which is faster between JavaScript and an ASP script?

JavaScript is faster. JavaScript is a client-side language,, and thus it does not need the assistance of the webserver to execute. On the other hand, ASP is a server-side language and hence is always slower than JavaScript. Javascript now is also a server-side language (nodejs).

6. What is negative Infinity?

Negative Infinity is a number in JavaScript which can be derived by dividing negative number by zero.

7. Is it possible to break JavaScript Code into several lines?

Breaking within a string statement can be done by using a backslash, '\,' at the end of the first line.

Example:

```
document. Write ("This is \a program,");
```

And if you change to a new line when not within a string statement, then javaScript ignores the break in the line.

Example:

```
var x=1, y=2,
z=
x+y;
```

The above code is perfectly fine, though not advisable as it hampers debugging.

8. Which company developed JavaScript?

Netscape is the software company that developed JavaScript.

9. What are undeclared and undefined variables?

Undeclared variables are those that do not exist in a program and are not declared. If the program tries to read the value of an undeclared variable, then a runtime error is encountered.

Undefined variables are those that are declared in the program but have not been given any value. If the program tries to read the value of an undefined variable, an undefined value is returned.

10. Write the code for adding new elements dynamically?

```
<html>
```

```
<head>
<title>t1</title>

<script type="text/javascript">
    function addNode () { var newP = document. createElement("p");
    var textNode = document.createTextNode(" This is a new text
node");
    newP.appendChild(textNode);
document.getElementById("firstP").appendChild(newP); }
</script> </head>
<body> <p id="firstP">firstP<p> </body>
</html>
```

11. What are global variables? How are these variable declared?

Global variables are available throughout the length of the code so that it has no scope. The var keyword is used to declare a local variable or object. If the var keyword is omitted, a global variable is declared.

Example:

// Declare a global: globalVariable = “Test”;

The problems faced by using global variables are the clash of variable names of local and global scope. Also, it is difficult to debug and test the code that relies on global variables.

12. What is a prompt box?

A prompt box is a box that allows the user to enter input by providing a text box. A label and box will be provided to enter the text or number.

13. What is ‘this’ keyword in JavaScript?

‘This’ keyword refers to the object from where it was called.

14. What is the working of timers in JavaScript?

Timers are used to execute a piece of code at a set time or repeat the code in a given interval. This is done by using the functions **setTimeout, setInterval,** and **clearInterval**.

The **setTimeout(function, delay)** function is used to start a timer that calls a particular function after the mentioned delay. The **setInterval(function, delay)** function repeatedly

executes the given function in the mentioned delay and only halts when canceled. The **clearInterval(id)** function instructs the timer to stop.

Timers are operated within a single thread, and thus events might queue up, waiting to be executed.

15. Which symbol is used for comments in Javascript?

// for Single line comments and

/* Multi

Line

Comment

*/

16. What is the difference between ViewState and SessionState?

- 'ViewState' is specific to a page in a session.
- 'SessionState' is specific to user-specific data that can be accessed across all web application pages.

17. What is === operator?

=== is called a strict equality operator, which returns true when the two operands have the same value without conversion.

18. How you can submit a form using JavaScript?

To submit a form using JavaScript use

```
document.form[0].submit();
document.form[0].submit();
```

19. Does JavaScript support automatic type conversion?

Yes, JavaScript does support automatic type conversion. It is the common way of type conversion used by JavaScript developers

20. How can the style/class of an element be changed?

It can be done in the following way:

```
document.getElementById("myText"). style. fontSize = "20";
```

or

```
document. getElementById ("myText"). className = "anyclass";
```

21. How to read and write a file using JavaScript?

There are two ways to read and write a file using JavaScript

- Using JavaScript extensions
- Using a web page and Active X objects

22. What are all the looping structures in JavaScript?

Following are looping structures in Javascript:

- For
- While
- Do-while loops

23. What is called Variable typing in Javascript?

Variable typing is used to assign a number to a variable. The same variable can be assigned to a string.

Example:

```
i = 10;
i = "string;"
```

This is called variable typing.

24. How can you convert the string of any base to an integer in JavaScript?

The parseInt() function is used to convert numbers between different bases. parseInt() takes the string to be converted as its first parameter. The second parameter is the base of the given string.

To convert 4F (or base 16) to integer, the code used will be –

```
parseInt ("4F", 16);
```

25. Difference between “==” and “===”?

“==” checks only for equality in value, whereas “===” is a stricter equality test and returns false if either the value or the type of the two variables are different.

26. What would be the result of 3+2+”7″?

Since 3 and 2 are integers, they will be added numerically. And since 7 is a string, its concatenation will be done. So the result would be 57.

27. How to detect the operating system on the client machine?

In order to detect the operating system on the client machine, the navigator. Platform string (property) should be used.

28. What do you mean by NULL in Javascript?

The NULL value is used to represent no value or no object. It implies no object or null string, no valid boolean value, no number, and no array object.

29. What is the function of the delete operator?

29. What is the function of the delete operator?

The delete keyword is used to delete the property as well as its value.

Example

```
var student= {age:20, batch:"ABC"};
Delete student. age;
```

30. What is an undefined value in JavaScript?

Undefined value means the

- Variable used in the code doesn't exist
- Variable is not assigned to any value
- Property does not exist.

31. What are all the types of Pop up boxes available in JavaScript?

- Alert
- Confirm and
- Prompt

32. What is the use of Void (0)?

Void(0) is used to prevent the page from refreshing, and parameter "zero" is passed while calling.

Void(0) is used to call another method without refreshing the page.

33. How can a page be forced to load another page in JavaScript?

The following code has to be inserted to achieve the desired effect:

```
<script language="JavaScript" type="text/javascript" >
<!-- location. href="https://www.guru99.com/javascript-interview-
questions-answers.html"; //--></script>
```

34. What is the data type of variables in JavaScript?

All variables in JavaScript are object data types.

35. What is the difference between an alert box and a confirmation box?

An alert box displays only one button, which is the OK button.

But a Confirmation box displays two buttons, namely OK and cancel.

36. What are escape characters?

Escape characters (Backslash) is used when working with special characters like single quotes, double quotes, apostrophes, and ampersands. Place backslash before the characters to make it display.

Example:

```
document. write "I m a "good" boy."
document. write "I m a \"good\" boy."
```

37. What are JavaScript Cookies?

Cookies are the small test files stored in a computer, and they get created when the user visits the websites to store information that they need. Examples could be User Name details and shopping cart information from previous visits.

38. What a pop()method in JavaScript is?

The pop() method is similar to the shift() method, but the difference is that the Shift method works at the array's start. The pop() method takes the last element off of the given array and returns it. The array on which it is called is then altered.

Example:

```
var cloths = ["Shirt", "Pant", "TShirt"];
```

```
var cloths = [ Shirt , Pant , Jacket ];
cloths.pop();

//Now cloth becomes Shirt,Pant
```

39. Does JavaScript has concept level scope?

No. JavaScript does not have concept-level scope. The variable declared inside the function has scope inside the function.

40. What are the disadvantages of using innerHTML in JavaScript?

If you use innerHTML in JavaScript, the disadvantage is

- Content is replaced everywhere
- We cannot use it like “appending to innerHTML
- Even if you use +=like “innerHTML = innerHTML + ‘html'” still the old content is replaced by html
- The entire innerHTML content is re-parsed and builds into elements. Therefore, it’s much slower
- The innerHTML does not provide validation, and therefore we can potentially insert valid and broken HTML in the document and break it

41. What is break and continue statements?

Break statement exits from the current loop.

Continue statement continues with next statement of the loop.

42. What are the two basic groups of data types in JavaScript?

- They are as—Primitive
- Reference types

Primitive types are number and Boolean data types. Reference types are more complex types like strings and dates.

43. How can generic objects be created?

Generic objects can be created as:

```
var I = new object();
```

44. What is the use of a type of operator?

'Typeof' is an operator used to return a string description of the type of a variable.

45. Which keywords are used to handle exceptions?

Try… Catch—finally is used to handle exceptions in the JavaScript

```
Try{
    Code
}
Catch(exp){
    Code to throw an exception.
}
Finally{
    Code runs either it finishes successfully or after catch
}
```

46. Which keyword is used to print the text on the screen?

Document. Write ("Welcome") is used to print the text–Welcome on the screen.

47. What is the use of the blur function?

Blur function is used to remove the focus from the specified object.

48. What is variable typing?

Variable typing assigns a number to a variable and then assigns a string to the same variable. An example is as follows:

```
i= 8;
i="john";
```

49. How to find an operating system in the client machine using JavaScript?

The ‘Navigator. the app version is used to find the operating system’s name in the client machine.

50. What are the different types of errors in JavaScript?

There are three types of errors:

- **Load time errors**: Errors that come up when loading a web page, like improper syntax errors, are known as Load time errors and generate the errors dynamically.
- **Runtime errors**: Errors that come due to misuse of the command inside the HTML language.
- **Logical Errors**: These are the errors that occur due to the bad logic performed on a function with a different operation.

51. What is the use of the Push method in JavaScript?

The push method is used to add or append one or more elements to an Array end. Using this method, we can append multiple elements by passing multiple arguments.

52. What is the unshift method in JavaScript?

Unshift method is like the push method, which works at the beginning of the array. This method is used to prepend one or more elements to the beginning of the array.

53. What is the difference between JavaScript and Jscript?

Both are almost similar. Netscape and Jscript develop JavaScript was developed by Microsoft.

54. How are object properties assigned?

Properties are assigned to objects in the following way –

```
obj ["class"] = 12;
or
obj.class = 12;
```

55. What is the ‘Strict Mode in JavaScript, and how can it be enabled?

Strict Mode adds certain compulsions to JavaScript. Under the strict Mode, JavaScript shows errors for a piece of code, which did not show an error before, but might be problematic and potentially unsafe. Strict Mode also solves some mistakes that hamper the JavaScript engines from working efficiently.

Strict mode can be enabled by adding the string literal “use strict” above the file. This can be illustrated by the given example:

```
function myfunction() {
    "use strict;"
    var v = "This is a strict mode function";
}
```

56. What is the way to get the status of a CheckBox?

The status can be acquired as follows –

```
alert(document.getElementById('checkbox1').checked);
```

If the CheckBox is checked, this alert will return TRUE.

57. How can the OS of the client machine be detected?

The navigator. appVersion string can be used to detect the operating system on the client machine.

58. What is a window.onload and onDocumentReady?

The onload function is not run until all the information on the page is loaded. This leads to a substantial delay before any code is executed.

onDocumentReady loads the code just after the DOM is loaded. This allows early manipulation of the code.

59. How closures work in JavaScript?

The closure is a locally declared variable related to a function that stays in memory when it has returned.

For example:

```
function greet(message) {
    console.log(message);
}
function greeter(name, age) {

    return name + " says howdy!! He is " + age + " years old";
}
// Generate the message
var message = greeter("James", 23);
// Pass it explicitly to greet
greet(message);
This function can be better represented by using closures
function greeter(name, age) {
    var message = name + " says howdy!! He is " + age + " years old";
    return function greet() {
        console.log(message);
    };
}
// Generate the closure
var JamesGreeter = greeter("James", 23);
// Use the closure
JamesGreeter();
```

60. How can a value be appended to an array?

A value can be appended to an array in the given manner –

```
arr[arr.length] = value;
```

61. What is for-in loop in Javascript?

The for-in loop is used to loop through the properties of an object.

The syntax for the for-in loop is –

```
for (variable name in object){
    statement or block to execute
}
```

In each repetition, one property from the object is associated with the variable name. The loop is continued till all the properties of the object are depleted.

62. What are the important properties of an anonymous function in JavaScript?

A function that is declared without any named identifier is known as an anonymous function. In general, an anonymous function is inaccessible after its declaration.

Anonymous function declaration –

```
var anon = function() {
    alert('I am anonymous');
};
anon();
```

63. What is the difference between .call() and .apply()?

The function .call() and .apply() are very similar in their usage except a little difference. .call() is used when the number of the function's arguments are known to the programmer, as they have to be mentioned as arguments in the call statement. On the other hand, .apply() is used when the number is not known. The function .apply() expects the argument to be an array.

The basic difference between .call() and .apply() is in the way arguments are passed to the function. Their usage can be illustrated by the given example.

```
var someObject = {
myProperty : 'Foo',

myMethod : function(prefix, postfix) {
```

```
    alert(prefix + this.myProperty + postfix);
}

};
someObject.myMethod('<', '>'); // alerts '<Foo>'
var someOtherObject  = {

    myProperty : 'Bar.'

};
someObject.myMethod.call(someOtherObject, '<', '>'); // alerts
'<Bar>'

someObject.myMethod.apply(someOtherObject, ['<', '>']); // alerts
'<Bar>'
```

64. What is event bubbling?

JavaScript allows DOM elements to be nested inside each other. In such a case, if the handler of the child is clicked, the handler of the parent will also work as if it were clicked too.

65. Is JavaScript case sensitive? Give its example.

Yes, JavaScript is case-sensitive. For example, a function parseInt is not the same as the function Parseint.

66. What boolean operators can be used in JavaScript?

The ‘And’ Operator (&&), ‘Or’ Operator (||), and the ‘Not’ Operator (!) can be used in JavaScript.

*Operators are without the parenthesis.

67. How can a particular frame be targeted, from a hyperlink, in JavaScript?

This can be done by including the name of the required frame in the hyperlink using the ‘target’ attribute.

```
<a href="/newpage.htm" target="newframe">>New Page</a>
```

68. What is the role of break and continue statements?

The break statement is used to come out of the current loop. In contrast, the continue statement continues the current loop with a new recurrence.

69. Write the point of difference between a web garden and a web farm?

Both web-garden and web-farm are web hosting systems. The only difference is that web-garden is a setup that includes many processors in a single server. At the same time,web-farm is a larger setup that uses more than one server.

70. How are object properties assigned?

Assigning properties to objects is done in the same way as a value is assigned to a variable. For example, a form object's action value is assigned as 'submit' in the following manner – Document. form.action="submit"

71. What is the method for reading and writing a file in JavaScript?

This can be done by Using JavaScript extensions (runs from JavaScript Editor), for example, for the opening of a file –

```
fh = fopen(getScriptPath(), 0);
```

72. How are DOM utilized in JavaScript?

DOM stands for Document Object Model and is responsible for how various objects in a document interact with each other. DOM is required for developing web pages, which includes objects like paragraphs, links, etc. These objects can be operated to include actions like add or delete. DOM is also required to add extra capabilities to a web page. On top of that, the use of API gives an advantage over other existing models.

73. How are event handlers utilized in JavaScript?

Events are the actions that result from activities, such as clicking a link or filling a form by the user. An event handler is required to manage the proper execution of all these events. Event handlers are an extra attribute of the object. This attribute includes the event's name and the action taken if the event takes place.

74. What is the role of deferred scripts in JavaScript?

74. What is the role of deferred scripts in JavaScript?

The HTML code's parsing during page loading is paused by default until the script has not stopped executing. If the server is slow or the script is particularly heavy, then the web page is delayed.

While using Deferred, scripts delays execution of the script till the time the HTML parser is running. This reduces the loading time of web pages, and they get displayed faster.

75. What are the various functional components in JavaScript?

The different functional components in JavaScript are-

- **First-class functions:** Functions in JavaScript are utilized as first-class objects. This usually means that these functions can be passed as arguments to other functions, returned as values from other functions, assigned to variables, or can also be stored in data structures.
- **Nested functions:** The functions, which are defined inside other functions, are called Nested functions. They are called 'every time the main function is invoked.

76. Write about the errors shown in JavaScript?

JavaScript gives a message as if it encounters an error. The recognized errors are –

- **Load-time errors:** The errors shown at the time of the page loading are counted under Load-time errors. The use of improper syntax encounters these errors and is thus detected while the page is getting loaded.
- **Runtime errors:** This is the error that comes up while the program is running. For example, illegal operations cause the division of a number by zero or access a non-existent area of the memory.
- **Logic errors:** It is caused by syntactically correct code, which does not fulfill the required task—for example, an infinite loop.

77. What are Screen objects?

Screen objects are used to read the information from the client's screen. The properties of screen objects are –

- AvailHeight: Gives the height of the client's screen
- AvailWidth: Gives the width of the client's screen
- ColorDepth: Gives the bit depth of images on the client's screen

- ColorDepth: Gives the bit depth of images on the client's screen
- Height: Gives the total height of the client's screen, including the taskbar
- Width: Gives the total width of the client's screen, including the taskbar

78. What is the unshift() method?

This method is functional at the starting of the array, unlike the push(). It adds the desired number of elements to the top of an array. For example –

```
var name = [ "john" ];
name.unshift( "charlie" );
name.unshift( "joseph", "Jane" );
console.log(name);
```

The output is shown below:

```
[" joseph ","," Jane ,", " charlie ", " john "]
```

79. What is unescape() and escape() functions?

The escape () function is responsible for coding a string to transfer the information from one computer to the other across a network.

For Example:

```
<script>
document.write(escape("Hello? How are you!"));
</script>
```

Output: Hello%3F%20How%20are%20you%21

The unescape() function is very important as it decodes the coded string.

It works in the following way. For example:

```
<script>
```

```
    document.write(unescape("Hello%3F%20How%20are%20you%21"));
</script>
```

Output: Hello? How are you!

80. What are the decodeURI() and encodeURI()?

EncodeURl() is used to convert URL into their hex coding. And DecodeURI() is used to convert the encoded URL back to normal.

```
<script>
   var uri="my test.asp?name=ståle&car=saab";

   document.write(encodeURI(uri)+ "<br>");

    document.write(decodeURI(uri));
</script>
```

Output –

my%20test.asp?name=st%C3%A5le&car=saab

my test.asp?name=ståle&car=saab

81. Why you should not use innerHTML in JavaScript?

innerHTML content is refreshed every time and thus is slower. There is no scope for validation in innerHTML. Therefore, it is easier to insert rogue code in the document and make the web page unstable.

82. What does the following statement declare?

```
var myArray = [[[]]];
```

It declares a three-dimensional array.

83. How are JavaScript and ECMA Script related?

83. How are JavaScript and ECMA Script related?

ECMA Script is like rules and guidelines, while Javascript is a scripting language used for web development.

84. What is namespacing in JavaScript, and how is it used?

Namespacing is used for grouping the desired functions, variables, etc., under a unique name. It is a name that has been attached to the desired functions, objects, and properties. This improves modularity in the coding and enables code reuse.

85. How can JavaScript codes be hidden from old browsers that do not support JavaScript?

For hiding JavaScript codes from old browsers:

Add “<!–” without the quotes in the code just after the <script> tag.

Add “//–>” without the quotes in the code just before the <script> tag.

Old browsers will now treat this JavaScript code as a long HTML comment. While a browser that supports JavaScript will take the “<!–” and “//–>” as one-line comments.

86. How to use Loop in JavaScript?

Loops are useful when you repeatedly execute the same lines of code a specific number of times or as long as a specific condition is true. Suppose you want to type a ‘Hello’ message 100 times on your webpage. Of course, you will have to copy and paste the same line 100 times. Instead, if you use loops, you can complete this task in just 3 or 4 lines.

87.How to use Loops in Javascript?

There are mainly four types of loops in JavaScript.

for loop

for/in a loop (explained later)

while loop

do…while loop

for loop

for loop

Syntax:

```
for(statement1; statement2; statment3)

{

lines of code to be executed

}
```

1. Statement1 is executed first, even before executing the looping code. So, this statement is normally used to assign values to variables used inside the loop.
2. The statement2 is the condition to execute the loop.
3. The statement3 is executed every time after the looping code is executed.

```
<html>
<head>
    <script type="text/javascript">
        var students = new Array("John", "Ann", "Aaron", "Edwin",
"Elizabeth");
        document.write("<b>Using for loops </b><br />");
        for (i=0;i<students.length;i++)
        {
        document.write(students[i] + "<br />");
        }
    </script>
</head>
<body>
</body>
</html>
```

while loop

Syntax:

```
while(condition)
```

```
while(condition)

{

lines of code to be executed

}
```

The “while loop” is executed as long as the specified condition is true. Inside the while loop, you should include the statement that will end the loop at some point in time. Otherwise, your loop will never end, and your browser may crash.

do…while loop

Syntax:

```
<pre>
do

{

block of code to be executed

} while (condition)
```

The do…while loop is very similar to the while loop. The only difference is that in do… while loop, the block of code gets executed once even before checking the condition.

Example:

```
<html>
<head>
    <script type="text/javascript">
        document.write("<b>Using while loops </b><br />");
        var i = 0, j = 1, k;
        document.write("Fibonacci series less than 40<br />");
        while(i<40)
        {
            document.write(i + "<br />");
```

```
            k = i+j;

            i = j;
            j = k;
        }
    </script>
</head>
<body>
</body>
</html>
```

88. What are the important JavaScript Array Method explain with example?

JavaScript Array Methods

The Array object has many properties and methods which help developers to handle arrays easily and efficiently. You can get the value of a property by specifying arrayname.property and the output of a method by specifying arrayname.method().

- **length property** –> If you want to know the number of elements in an array, you can use the length property.
- **prototype property** –> If you want to add new properties and methods, you can use the prototype property.
- **reverse method** –> You can reverse the order of items in an array using a reverse method.
- **sort method –>** You can sort the items in an array using sort method.
- **pop method** –> You can remove the last item of an array using a pop method.
- **shift method** –> You can remove the first item of an array using shift method.
- **push method** –> You can add a value as the last item of the array.

```
<html>
<head>
    <title>Arrays!!!</title>
    <script type="text/javascript">
        var students = new Array("John", "Ann", "Aaron", "Edwin",
"Elizabeth");
        Array.prototype.displayItems=function(){
            for (i=0;i<this.length;i++){
                document.write(this[i] + "<br />");
            }
```

```
        }
        document.write("students array<br />");
        students.displayItems();
        document.write("<br />The number of items in students array
is " + students.length + "<br />");
        document.write("<br />The SORTED students array<br />");
        students.sort();
        students.displayItems();
        document.write("<br />The REVERSED students array<br />");
        students.reverse();
        students.displayItems();
        document.write("<br />THE students array after REMOVING the
LAST item<br />");
        students.pop();
        students.displayItems();
                document.write("<br />THE students array after
PUSH<br />");
        students.push("New Stuff");
        students.displayItems();
    </script>
</head>
<body>
</body>
</html>
```

89. What is OOPS Concept in JavaScript?

Many times, variables or arrays are not sufficient to simulate real-life situations. JavaScript allows you to create objects that act like real-life objects. A student or a home can be an object that has many unique characteristics of its own. You can create properties and methods for your objects to make programming easier. If your object is a student, it will have properties like the first name, last name, id, etc., and methods like calculating rank, change address, etc. If your object is a home, it will have properties like a number of rooms, paint color, location, etc. The methods like to calculate area, change owner, etc.

How to Create an Object

You can create an object like this:

```
var objName = new                Object();
```

```
var objName = new                    Object();
objName.property1 = value1;

objName.property2 = value2;
objName.method1 = function()
{
line of code
}
```

OR

```
var objName= {property1:value1, property2:value2, method1: function()

{ lines of code} };
```

90. What is Loop Though the Properties of an Object?

The for/in a loop is usually used to loop through the properties of an object. You can give any name for the variable, but the object's name should be the same as an already existing object you need to loop through.

Syntax:

```
for (variablename in objectname)

{

lines of code to be executed

}
```

Example:

```
<html>
<head>
    <script type="text/javascript">
        var employee={first:"John", last:"Doe",
department:"Accounts"};
```

```
                var             details = "";

                document.write("<b>Using for/in loops </b><br />");
        for (var x in employee)
        {
                details = x + ": " + employee[x];
                document.write(details + "<br />");
        }
    </script>
</head>
<body>
</body>
</html>
```

91. What is JavaScript Unit Testing, and what are the challenges in JavaScript Unit Testing?

JavaScript Unit Testing is a testing method in which JavaScript tests code written for a web page or web application module. It is combined with HTML as an inline event handler and executed in the browser to test if all functionalities work fine. These unit tests are then organized in the test suite.

Every suite contains several tests designed to be executed for a separate module. Most importantly, they don't conflict with any other module and run with fewer dependencies on each other (some critical situations may cause dependencies).

Challenges of JavaScript Unit Testing:

Here are important challenges of JavaScript Unit Testing:

- Many other languages support unit testing in browsers, in the stable as well as in runtime environment, but JavaScript can not
- You can understand some system actions with other languages, but this is not the case with JavaScript
- Some JavaScript are written for a web application that may have multiple dependencies.
- JavaScript is good to use in combination with HTML and CSS rather than on the web
- Difficulties with page rendering and DOM manipulation
- Sometimes you find an error message on your screen regarding 'Unable to load example.js' or any other JavaScript error regarding version control. These

vulnerabilities come under Unit Testing JavaScript

Solutions of JavaScript Unit Testing:

To avoid such issues, what you can do is;

- Do not use global variables.
- Do not manipulate predefined objects.
- Design core functionalities based on the library.
- Try to create small pieces of functionalities with lesser dependencies.

92. What are some important JavaScript Unit Testing Frameworks?

Following is a curated list of popular JavaScript Unit Testing Frameworks and Tools that are widely used :

Unit.js: It is known as an open-source assertion library running on browser and Node.js. It is extremely compatible with other JavaScript Unit Testing frameworks like Mocha, Karma, Jasmine, QUnit, Protractor, etc. Provides the full documented API of assertion list.

QUnit: It is used for both client-side and server-side JavaScript Unit Testing. This Free JavaScript testing framework is used for jQuery projects. It follows Common JS unit testing Specification for unit testing in JavaScript. It supports the Node Long-term Support Schedule.

Jasmine: Jasmine is the behavior-driven development framework to unit test JavaScript. It is used for testing both synchronous and asynchronous JavaScript codes. It does not require DOM and comes with an easy syntax that can be written for any test.

Karma: Karma is an open-source productive testing environment. Easy workflow control running on the command line. Offers the freedom to write the tests with Jasmine, Mocha, and QUnit. You can run the test on real devices with easy debugging.

Mocha: Mocha runs on Node.js and in the browser. Mocha performs asynchronous testing more simply. Provides accuracy and flexibility in reporting. Provides tremendous support of rich features such as test-specific timeouts, JavaScript APIs.

Jest: Facebook uses jest so far to test all the JavaScript code. It provides the 'zero-configuration testing experience. Supports independent and non-interrupting running tests without any conflict. Do not require any other setup configuration and libraries.

AVA: AVA is a simple JavaScript Unit Testing Framework. Tests are being run in parallel and serially. Parallel tests run without interrupting each other. This testing framework supports asynchronous testing as well. AVA uses subprocesses to run the unit test JavaScript.

93. What is QuickSort Algorithm in JavaScript?

Quick Sort algorithm follows Divide and Conquer approach. It divides elements into smaller parts based on some conditions and performing the sort of operations on those divided smaller parts.

Quick Sort algorithm is one of the most used and popular algorithms in any programming language. If you are a JavaScript developer, you might have heard of sort() which is already available in JavaScript. Then, you might have been thinking about what the need for this Quick Sort algorithm is. To understand this, first, we need what is sorting and what is the default sorting in JavaScript.

Quicksort follows the **Divide-and-Conquer** algorithm. It divides elements into smaller parts based on some conditions and performs the sort operations on those divided smaller parts. Hence, it works well for large datasets. So, here are the steps of how Quicksort works in simple words.

1. First, select an element that is to be called the **pivot** element.
2. Next, compare all array elements with the selected pivot element and arrange them so that elements less than the pivot element are left. Greater than pivot is to its right.
3. Finally, perform the same operations on the left and right side elements to the pivot element.

So, that is the basic outline of Quicksort. Here are the steps which need to be followed one by one to perform Quicksort.

94.How does QuickSort Work

Step 1) First, find the **“pivot”** element in the array.

Step 2) Start the left pointer at the first element of the array.

Step 3) Start the right pointer at the last element of the array.

Step 4) Compare the element pointing with the left pointer, and if it is less than the pivot element, then move the left pointer to the right (add 1 to the left index). Continue this

until the left side element is greater than or equal to the pivot element.

Step 5) Compare the element pointing with the right pointer. If it is greater than the pivot element, move the right pointer to the left (subtract 1 to the right index). Continue this until the right-side element is less than or equal to the pivot element.

Step 6) Check if the left pointer is less than or equal to a right pointer, then saw the elements in these pointers' locations.

Step 7) Increment the left pointer and decrement the right pointer.

Step 8) If the left pointer index is still less than the right pointer's index, repeat the process; else, return the left pointer's index.

So, let us see these steps with an example. Let us consider an array of elements which we need to sort is [5,3,7,6,2,9].

Here are the steps to perform Quick sort that is being shown with an example [5,3,7,6,2,9].

STEP 1) Determine pivot as a middle element. So, **7** is the pivot element.

STEP 2) Start left and right pointers as first and last elements of the array, respectively. The left pointer points to 5 at index 0, and the right pointer points to **9** at index 5.

STEP 3) Compare the left pointer element with the pivot element, since 5 < 6 shift left pointer to the right to index 1.

STEP 4) Now, still 3 <6, so shift the left pointer to one more index to the right. Now 7 > 6 stops incrementing the left pointer, and now the left pointer is index 2.

STEP 5) Now, compare the value at the right pointer with the pivot element. Since 9 > 6, move the right pointer to the left. Now, as 2 < 6, stop moving the right pointer.

STEP 6) Swap both values present at left and right pointers with each other.

STEP 7) Move both pointers one more step.

STEP 8) Since 6 = 6, move pointers to one more step and stop as the left pointer crosses the right pointer and returns the left pointer's index.

Here, based on the above approach, we need to write code for swapping elements and partitioning the array as mentioned in the above steps.

Example:

```
var items = [5,3,7,6,2,9];
function swap(items, leftIndex, rightIndex){
    var temp = items[leftIndex];
    items[leftIndex] = items[rightIndex];
    items[rightIndex] = temp;
}
function: partition(items, left, right) {
    var pivot   = items[Math.floor((right + left) / 2)], //middle
element
        i       = left, //left pointer
        j       = right; //right pointer
    while (i <= j) {
        while (items[i] < pivot) {
            i++;
        }
        while (items[j] > pivot) {
            j--;
        }
        if (i <= j) {
```

```
            swap(items, i, j); //sawpping two elements
            i++;
            j--;
        }
    }
    return i;
}

function quickSort(items, left, right) {
    var index;
    if (items.length > 1) {
        index = partition(items, left, right); //index returned from
partition
        if (left < index - 1) { //more elements on the left side of
the pivot
            quickSort(items, left index - 1);
        }
        if (index < right) { //more elements on the right side of the
pivot
            quickSort(items, index, right);
        }
    }
    return items;
}
// first call to quick sort
var sortedArray = quickSort(items, 0, items.length - 1);
console.log(sortedArray); //prints [2,3,5,6,7,9]
```

94. What is DOM in JavaScript?

JavaScript can access all the elements in a web page using the Document Object Model (DOM). The web browser creates a DOM of the webpage when the page is loaded.

95. How to use DOM and Events?

Using DOM, JavaScript can perform multiple tasks. It can create new elements and attributes, change the existing elements and attributes and even remove existing elements and attributes. JavaScript can also react to existing events and create new events in the page.

1. getElementById, innerHTML Example
2. getElementById: To access elements and attributes whose id is set

2. getElementById: To access elements and attributes whose id is set.
3. innerHTML: To access the content of an element.

```
<html>
<head>
    <title>DOM!!!</title>
</head>
<body>
  <h3 id="one">Welcome</h3>
  <p>This is the welcome message.</p>
  <h3>Technology</h3>
  <p>This is the technology section.</p>
  <script type="text/javascript">
        var text = document.getElementById("one").innerHTML;
        alert("The first heading is " + text);
  </script>
</body>
</html>
```

2.getElementsByTagName Example

getElementsByTagName: To access elements and attributes using tag name. This method will return an array of all the items with the same tag name.

```
<html>

<head>

    <title>DOM!!!</title>

</head>

<body>

  <h3>Welcome</h3>

  <p>This is the welcome message.</p>

  <h3>Technology</h3>

  <p id="second">This is the technology section.</p>
```

```
  <script type="text/javascript">


    var paragraphs = document.getElementsByTagName("p");

    alert("Content in the second paragraph is " +
paragraphs[1].innerHTML);

    document.getElementById("second").innerHTML = "The orginal
message is changed.";

  </script>

</body>

</html>
```

Event handler Example

1. createElement: To create new element
2. removeChild: Remove an element
3. you can add an **event handler** to a particular element like this

```
    document.getElementById(id).onclick=function()
        {
            lines of code to be executed
        }
```

OR

```
document.getElementById(id).addEventListener("click", functionname)
```

Example:

```
<html>
<head>
    <title>DOM!!!</title>
```

```
    <title>DOM!!!</title>
</head>

<body>
  <input type="button" id="btnClick" value="Click Me!!" />
  <script type="text/javascript">
    document.getElementById("btnClick").addEventListener("click",
clicked);
    function clicked()
    {
            alert("You clicked me!!!");
    }
  </script>
</body>
</html>
```

96. What is External JavaScript?

You plan to display the current date and time on all your web pages. Suppose you wrote the code and copied it in to all your web pages (say 100). But later, you want to change the format in which the date or time is displayed. In this case, you will have to make changes to all the 100 web pages. This will be a very time-consuming and difficult task.

So, save the JavaScript code in a new file with the extension .js. Then, add a line of code in all your web pages to point to your .js file like this:

```
<script type="text/javascript," src="/currentdetails.js,">
```

Note: It is assumed that the .js file and all your web pages are in the same folder. If the external.js file is in a different folder, you need to specify your file's full path in the src attribute.

Example:

```
var currentDate = new Date();
var day = currentDate.getDate();
Var month = currentDate.getMonth() + 1;
var monthName;
var hours = currentDate.getHours();
var mins = currentDate.getMinutes();
```

```
var mins = currentDate.getMinutes();
var secs = currentDate.getSeconds();

var strToAppend;
It (hours >12 )
{
    hours1 = "0" + (hours - 12);
strToAppend = "PM";
}
else if (hours <12)
{
    hours1 = "0" + hours;
    strToAppend = "AM";
}
else
{
    hours1 = hours;
    strToAppend = "PM";
}
if(mins<10)
mins = "0" + mins;
if (secs<10)
    secs = "0" + secs;
switch (month)
{
    case 1:
        monthName = "January";
        break;
    case 2:
        monthName = "February";
        break;
    case 3:
        monthName = "March";
        break;
    case 4:
        monthName = "April";
        break;
    case 5:
        monthName = "May";
        break;
    case 6:
        monthName = "June";
        break;
    case 7:
        monthName = "July";
```

```
            monthName = "July";
            break;
        case 8:
            monthName = "August";
            break;
        case 9:
            monthName = "September";
            break;
        case 10:
            monthName = "October";
            break;
        case 11:
            monthName = "November";
            break;
        case 12:
            monthName = "December";
            break;
    }

    var year = currentDate.getFullYear();
    var myString;
    myString = "Today is " + day +  " - " + monthName + " - " + year + ".
    <br />Current time is " + hours1 + ":" + mins + ":" + secs + " " +
    strToAppend + ".";
    document.write(myString);
```

97. When to Use Internal and External JavaScript Code?

Suppose you have only a few lines of code that is specific to a particular webpage. In that case, it is better to keep your JavaScript code internal within your HTML document.

On the other hand, if your JavaScript code is used in many web pages, you should consider keeping your code in a separate file. If you wish to make some changes to your code, you have to change only one file, making code maintenance easy. If your code is too long, it is better to keep it in a separate file. This helps in easy debugging.

98. What are Cookies in JavaScript?

A cookie is a piece of data stored on your computer to be accessed by your browser. You also might have enjoyed the benefits of cookies knowingly or unknowingly. Have you ever saved your Facebook password so that you do not have to type it every time you try

to login? If yes, then you are using cookies. Cookies are saved as key/value pairs.

Javascript Set-Cookie:

You can create cookies using document. cookie property like this.

```
document.cookie = "cookiename=cookievalue"
```

You can even add an expiry date to your Cookie to remove the particular Cookie from the computer on the specified date. The expiry date should be set in the UTC/GMT format. If you do not set the expiry date, the cookie will be removed when the user closes the browser.

```
document.cookie = "cookiename=cookievalue; expires= Thu, 21 Aug 2014
20:00:00 UTC"
```

You can also set the domain and path to specify which domain and to which directories in the specific domain the Cookie belongs to. By default, a cookie belongs to the page that sets the Cookie.

```
document.cookie = "cookiename=cookievalue; expires= Thu, 21 Aug 2014
20:00:00 UTC; path=/
```

//create a cookie with a domain to the current page and a path to the entire domain.

JavaScript get Cookie

You can access the Cookie like this, which will return all the cookies saved for the current domain.

```
var x =  document.cookie
```

JavaScript Delete Cookie

To delete a cookie, you just need to set the cookie's value to empty and set the value of expires to a passed date.

Example:

```
<html>
<head>
    <title>Cookie!!!</title>
    <script type="text/javascript">
        function createCookie(cookieName,cookieValue,daysToExpire)
        {
          var date = new Date();
          date.setTime(date.getTime()+(daysToExpire*24*60*60*1000));
          document.cookie = cookieName + "=" + cookieValue + ";
expires=" + date.toGMTString();
        }
        function accessCookie(cookieName)
        {
          var name = cookieName + "=";
          var allCookieArray = document.cookie.split(';');
          for(var i=0; i<allCookieArray.length; i++)
          {
            var temp = allCookieArray[i].trim();
            if (temp.indexOf(name)==0)
            return temp.substring(name.length,temp.length);
             }
            return "";
        }
        function checkCookie()
        {
          var user = accessCookie("testCookie");
          if (user!="")
            alert("Welcome Back " + user + "!!!");
          else
          {
            user = prompt("Please enter your name");
            num = prompt("How many days you want to store your name
on your computer?");
            It (user!="" && user!=null)
            {
            createCookie("testCookie", user, num);
```

```
            createCookie("testCookie", user, num);
            }

          }
        }
    </script>
</head>
<body onload="checkCookie()"></body>
</html>
```

99. Give an example of JavaScript Multiplication Table

Here, are example of simple multiplication table asking the user the number of rows and columns he wants.

Example:

```
<html>
<head>
  <title>Multiplication Table</title>
  <script type="text/javascript">
    var rows = prompt("How many rows for your multiplication
table?");
    var cols = prompt("How many columns for your multiplication
table?");
    if(rows == "" || rows == null)
            rows = 10;
    if(cols== "" || cols== null)
            cols = 10;
    createTable(rows, cols);
    function createTable(rows, cols)
    {
      var j=1;
      var output = "<table border='1' width='500'
cellspacing='0'cellpadding='5'>";
      for(i=1;i<=rows;i++)
      {
        output = output + "<tr>";
        while(j<=cols)
        {
            output = output + "<td>" + i*j + "</td>";
             j = j+1;
```

```
            }
             output = output + "</tr>";
             j = 1;
    }
    output = output + "</table>";
    document.write(output);
    }
  </script>
</head>
<body>
</body>
</html>
```

100. Explain Popup Message using event with example

Display a simple message “Welcome!!!” on your demo webpage and when the user hovers over the message, a popup should be displayed with a message “Welcome to my WebPage!!!”.

Example:

```
<html>
    <head>

    <title>Event!!!</title>

    <script type="text/javascript">

    function trigger()

    {

    document.getElementById("hover").addEventListener("mouseover",
popup);

    function popup()

    {

    alert("Welcome to my WebPage!!!");
```

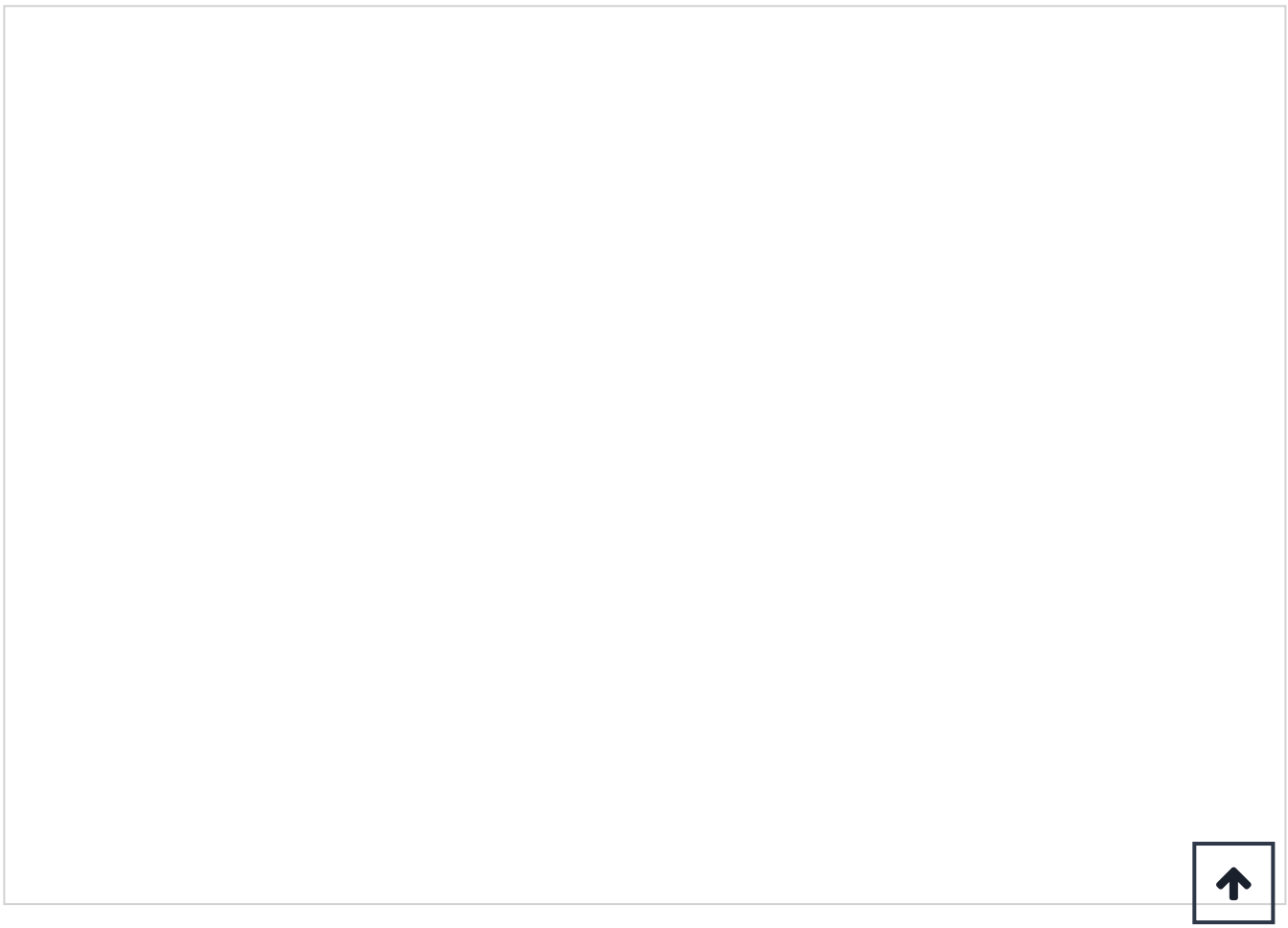
```
}

}

</script>

<style>

p{
      font-size:50px;
      position: fixed;
      left: 550px;
      top: 300px;
}
</style>
</head>
<body  onload="trigger();">
<p id="hover">Welcome!!!</p>
</body>
</html>
```



You Might Like: