**1. What is Java Script?**

Java Script is a client-side object-oriented scripting language commonly used to create dynamic and interactive elements on web pages. It is also used for input form validations on client side. It is program for behavior of webpages.

**2. What are different data types in JavaScript?**

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

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. It can hold any type of values such as numbers, strings etc. For example:

Primitive data types: Non-Primitive

String Objects var person = {firstName:"John", lastName:"Doe"};

Number Array var cars = ["Saab", "Volvo", "BMW"];

Boolean RegExp

Undefined

Null var a= null;

symbol

**3. What are undeclared and undefined variables?**

A variable is undeclared when it does not use the var keyword. It gets created on the global object (that is, the window), thus it operates in a different space as the declared variables.

Ex: carName = "Volvo";

A variable is undefined when it hasn’t been defined yet. If you call a variable or function without having created it yet the parser will give you a not defined error.

Ex: var carName;

**4. What are global variables? How are these variables declared and what are the problems associated with using them?**

A **JavaScript global variable** is declared outside the function or declared with window object. It can be accessed from any function.A variable that does not use the var keyword. It gets created on the global object (that is, the window), thus it operates in a different space as the declared variables.

They clutter up the global namespace and are slower to look up than local variables. First, having many global variables is not good practice because it's easy to forget you declared a variable somewhere and accidentally re-declare it somewhere else. If your first variable was local then you don't have a problem. If it was global, then it just got overwritten. This gets even worse when you get into implied global (e.g. when you say someVar = someValue without declaring someVar with the var keyword).Secondly, global variables take longer for Javascript to "find" than local variables. The difference in speed isn't huge, but it does exist.

**5. What are the different ways to apply javascript in HTML5?**

In HTML, JavaScript code must be inserted between <script> and </script> tags. You can place any number of scripts in an HTML document. Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both. External scripts are practical when the same code is used in many different web pages. JavaScript files have the file extension**.js**.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

<body> <script src = “ myscript.js”> </script> </body>

We can any number of external scripts by writing different scripts.

<body> <script src = “ myscript.js”> </script>

<script src = “ myscript2.js”> </script>

</body>

Web engines to run js Chrome v8, firefox gecko, mozilla checkra

Placing scripts in external files has some advantages:

* It separates HTML and code
* It makes HTML and JavaScript easier to read and maintain
* Cached JavaScript files can speed up page loads

**6. What is null in js?**

Null is primitive datatype in java, Null is where the thing is known to exist, but it's not known what the value is. The type of null variable is object.

Var a = null;

Value of a is null and typeof a is object.

7. **Explain window.onload and onDocumentReady?**

window.onload is fired when the entire page loads, including its content like images, css, scripts etc $(window).load();

ondocumentReady is fired when the DOM is ready which can prior to images or external files.

In general document.onload is executed before window.onload.

$(document).load();

**8. What are the ways to create the objects in java scripts?**

There are different ways to create new objects:

* Define and create a single object, using an object literal.

An object literal is a list of name:value pairs

var person = {firstName:"John", lastName:"Doe", age:50};

* Define and create a single object, with the keyword new.

var person = new Object();  
 person.firstName = "John";  
 person.lastName = "Doe";  
 person.age = 50;

* Define an object constructor, and then create objects of the constructed type.

Sometimes we like to have an "object type" that can be used to create many objects of one type. The standard way to create an "object type" is to use an object constructor function:

function person(first, last, age, eye) {  
     this.firstName = first;  
     this.lastName = last;  
     this.age = age;  
     this.eyeColor = eye;  
 }  
 var myFather = new person("John", "Doe", 50, "blue");  
 var myMother = new person("Sally", "Rally", 48, "green");

**9. How many ways we can create the arrays.**

Using new keyword: var cars = new Array("Saab", "Volvo", "BMW");

Using literal: var cars = [“Saab”, “Volvo”, “BMW”];

The easiest way to add a new element to an array is using the push method:

var fruits=["Banana", "Orange", "Apple", "Mango"];  
 fruits.push("Lemon");

New element can also be added to an array using the length property:

fruits[fruits.length] = "Lemon";

**10. What are prototypes and how to use them?**

The prototype is an object. All JavaScript objects inherit the properties and methods from their prototype. Objects created using an object literal, or with new Object(), inherit from a prototype called Object.prototype.

The standard way to create an object prototype is to use an object constructor function:

function Person(first, last, age, eyecolor) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age; }

With a constructor function, you can use the **new** keyword to create new objects from the same prototype:

var myFather = new Person("John", "Doe", 50, "blue");  
 var myMother = new Person("Sally", "Rally", 48, "green");

The prototype of a function is an object. Its main use is when a function is used as a constructor. The **prototype** object can be used to derive one object from another. For example, you can use the [Object.create](https://msdn.microsoft.com/en-us/library/ff925952(v=vs.94).aspx) function to derive a new object using the prototype of some other object.

**11. Callback and closures.**

Callback function, also known as a higher-order function, is passing one function as argument to another function and the callback function is called (or executed) inside the other Function. A callback function is essentially a pattern (an established solution to a common problem), and therefore, the use of a callback function is also known as a callback pattern.

A closure is an inner function that has access to the outer (enclosing) function’s variables—scope chain. The closure has three scope chains: it has access to its own scope (variables defined between its curly brackets), it has access to the outer function’s variables, and it has access to the global variables.

The inner function has access not only to the outer function’s variables, but also to the outer function’s parameters. Note that the inner function cannot call the outer function’s arguments object, however, even though it can call the outer function’s parameters directly. You create a closure by adding a function inside another function.

function add(){  
    var counter= 0;  
    function plus(){counter+= 1;}  
  plus();  
    return counter;   
}

add();

**Types of events in javascript**

**How do you manipulate HTML elements using JavaScript?**

The document object represents your web page. If you want to access any element in an HTML page, you always start with accessing the document object.

**Finding HTML Elements:**

document.getElementById(Id) document.getElementByTagName(name)

document.getElementByClassName()

document.querySelectorAll HTML elements by CSS selectors

document.form HTML elements by HTML object collections

document.links document.body document.images

**Changing the HTML Element:**

element.innerHtml = new html Content change the inner html of an element

element.attribute = new value change the attribute value of html element

element.setAttribute(attribute,value) change the attribute value of html element

element.style.property = new style; change the style of html element.

**JavaScript can "display" data in different ways:**

* 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()**.

**what is the difference between == and ===**

== is just comparing the two values, and if they are of different types, type conversion is done

=== compares the values and well as their types - so no type conversion will be done here.

## HTML DOM (Document Object Model):

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page. The **HTML DOM** model is constructed as a tree of **Objects**:

### **The HTML DOM Tree of Objects**



The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

* The HTML elements as **objects**
* The **properties** of all HTML elements
* The **methods** to access all HTML elements
* The **events** for all HTML elements

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

BOM:

The Browser Object Model (BOM) allows JavaScript to "talk to" the browser.

The **window** object is supported by all browsers. It represents the browser's window.

All global JavaScript objects, functions, and variables automatically become members of the window object.

Global variables are properties of the window object.

Global functions are methods of the window object.

Even the document object (of the HTML DOM) is a property of the window object:

window.document.getElementById("header");

same as

document.getElementById("header");