**JAVA SCRIPT**

* Java script was invented by Brendan Eich in 1995, and became an ECMA standard in 1997. ECMA script is the official name of the language.
* java script used for behavior of a web page
* if we write script in head its called **embedded script**
* if write in tag its called **inline script**
* inline-- inside html tag
* embedded-- inside head tag
* external-- outside of html doc
* One of many JavaScript html methods is getElementById(). To target id elements and classes.
* In HTML, JavaScript code must be inserted between <script> and </script>
* A JavaScript function is a block of JavaScript code that can be executed when called for. For example, a function can be called when an event occurs, like when the user clicks a button.
* You can pace any number of scripts in an html document. Scripts can be placed In the <body>, or in the <head> section of html page or in both.
* External java script advantages:
  + It separates html and code
  + It makes html and JavaScript easier to read and maintain
  + Cached JavaScript files can speed up page loads
* Java script 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()
* To access an html element JavaScript can use the document.getElementBy(Id) method.
* JS Syntax:
  + A computer program is a list of instructions to be executed by the computer
  + In a programming language these program instructions are called statements.
  + JavaScript is a programming language
  + JavaScript statements are separated by semicolons:
* JavaScript statements are composed of values, operators, expressions, keywords and comments.
  + The JavaScript syntax defines two types of values. Fixed values and variable values.
  + Fixed values are called literals. Variables are called variables.
    - The most important rules for writing fixed values are: numbers are written with or without decimals: 10.5 or 10
  + Strings are text written within double or single quotes: “John Doe”
  + Java script variables: in a programming language variables are used to store data values.
  + Java script uses the var keyword to declare variable. An **equal** sign is used to **assign** values to variables.
* Javasciript is a combination of values variables and operators which computes to value. The computation is called an evalution.
* The values can be of various types such as numbers and strings. For example “john” + “” “doe”, evaluates to “john doe”:
* Java script keywords are used to identify actions to be performed.. The var keyword tells the browser to create variables.
* Javascript **comments:**
  + Not all javascript statements are “executed”. Code after double slashes // or between /\* and \*/ is treated as a comment. Comments are ignored and will not be executed.
  + Var x = 5; //I will be executed.
  + //var x = 6; I will not be executed.
* In javaScript the first character must be a letter, an underscore or a dollar sign.
* Java script is **case sensitive.**
  + The variables lastName and lastname are two different variables.
  + Var lastname, lastName; lastName= ‘’doe”; lastname = ‘’peterson”;
  + JavaScript does not not interpret VAR ot Var as the keyword var.
  + === equal value and equal type,
* In JavaScript objects and functions are also variables. In JavaScript scope is the set of variables objects and functions you have access to.
* Javascript has function scope: the scope changes inside functions.
* Variables declared within a javascript function become local to the function. Local variables have local scope: they can only be accessed within the function.
* A variable declared outside a function become global. A global variable has global scope: all scripts and functions on a webpages can access it.
* Events: An html event can be something the browser does or something a user does. Here are some examples of html events.
  + An html web page has finished loading
  + An html input field was changed
  + An html button was clicked
  + Html allows event handler attributes with JavaScript code to do added to html elements.
  + Onchange, onclick, onmosueover, onmouseout, onkeydown, onload
* String Methods: But with javascript methods and properties are also available to primitive values, because javascript treats primitive values as objects when executing methods and properties.
  + The indexOf() method returns the index of the first occurrence of a specified text in a sting:
  + The lastIndexOf() method returns the index of the last occurrence of a specified text in a string.
  + Substring() method is similar to slice(). The difference is that substing() cannont accept negative indexes.
  + Substr() method is similar to slice. The difference is that the second parameter specifies the length of the extracted part.
  + The replace() method replaces a specified value with another value in a string:
  + A string is converted to upper case with toUpperCase():
  + toLowerCase()
  + The concat() method can be used instead of the plus operator. These two lines do the same
  + Extracting string characters
    - charAt() method returns the character at a specified index in a string.
    - chatCodeAt() method returns the Unicode of the character at a specified index in a string.
    - A string converted to array with **Split()**
* Javascript numbers are always 64 bit floating point, unlike many other programming languages js doesnot define different types of numbers like integers shot long floating-point
* Js numbers number are always stored as double precision floating point numbers following the international IEEE754 standard.
* This format stores numbers in 64 bits where the number us stored in bits 0 to 51 the exponent in bits (52 to 62) and the sign in bit 63
* NaN- Not a Number is a javascript reserved word indicating that a number is not a legal number. Trying to do arithmetic with a non-numeric string will result in NaN
* Note the difference between (x==y) and (x===y) comparing two javascript objects will always return false.
* Javascript counts months from 0 to 11 January is 0 December is 11
* Array: An array is a special variable, which can hold more than one value at a time. Var x = [“bmw”,”Porsche”,”audi”];
* Putting a comma after the last element is inconsistent across browsers. IE 8 and earlier will fail.
* The length property is always one more than the highest array index.
* Many programming languages support arrays with named indexes. Arrays with named indexes are called associative arrays. JS doesnot support arrays with named indexes.
* Arrays use numbered indexes.. objects use named indexes.
* How to I know if a variable is an array?? Using **typeOf**
* The instacneof operator returns true if an object is created by a given constructor.
* **Popping and Pushing Methods:** When you work with arrays it is easy to move elements and add new elements. This is what popping and pushing is:
  + Popping items out of an array or pushing items into an array.
* Shifting is equivalent to popping working on the first element instead of the last. The **shift()** method removes the first array element and “shifts” all other elements to a lower index.
* Using splice() to Remove elements.
* **Joining Arrays** The concat() method creates a new array by concatenating two arrays:
* **Conditional statement:** conditional statements are used to perform different actions based on different conditions. In javascript we have the following conditional statement:
  + Use if to speacify a block of code to be executed, if a specified condition if true.
  + Use else to specify a block of code to be executed if the same condition is false.
  + Use else if to specify a new condition to test if the first condition is false
  + Use switch to specify many alternative blocks of code to be executed.

1.SWITCH:

* The switch statement is used to perform different actions based on different conditions.

this are used to define several cases of the condition

SYNTAX:

switch(<condition>){

case<match the condition>:{

//logic

}

}

* + The switch expressions is evaluated once.
  + The value of the expression is compared with the values of each case
  + If there is a match the associated block is executed.

2.BREAK AND CONTINUE

Break:

there are used to break a loop when certain condition is met inside the loop we can use the break statement to stop the loop syntax.

break;

example:

var test =[1,4,454,88];

for(var i=0;i<test.length;i++){

//check if the value at that index of array test is 2

if(test[i] === 454){

break;

}

console.log(test[i]);

};

3.continue statement: it is mainly used for while and do while loop

syntax:

continue;

example:

var i = 0;

var n= 2;

while(i<10){

i++;

if(i ===5){

continue;

}

}

4.if else statement

* if basic idea behind this is to check for a condition.
* if that condition is true then perform some logic
* IF THAT CONDITION IS FALSE THEN PERFORM SOME DIFFERENT LOGIC

syntax:

if(<condition>){

//logic to be executed if <condition> is true

}

else{

//logic to be excited it <condition> is fase

}

\*\*else block is optional

\*\*<conditioin> should always return Boolean output

5.for

* this is standard looping keyword this is the mostly used looping mechanism
* to define a for statement we has three arguments to this for statement

syntax is:

for(<initialized variable>;<condition>;<incrementing/decrementing the initialized variable>){

//any logic

}

\*\* we usually use number for the initialized variables.

\*\*we need the condition to tell for statement to stop at one point where the condition fails

6.for(using the in statement)

* this is similar to the above looping statement but it is only used in objects

\*\*remember arrays are a type of objects so we can use this looping for arrays also

for(<new variable > in <object name>)

7.do while statements

* this is similar to for loop with certain syntax changes

syntax:

do{

//logic

<increment/decrement the initialized variable>

}while(<condition>)

example:

var i = 0 ;

do{

console.log(i);

}

while(i<10);

\*\*there is a slight difference between do while statements

* do while runs the logic before the condition check while run the condition first and then execute the logic

8.if else statement

* if basic idea behind this is to check for a condition.
* if that condition is true then perform some logic
* IF THAT CONDITION IS FALSE THEN PERFORM SOME DIFFERENT LOGIC

syntax:

if(<condition>){

//logic to be executed if <condition> is true

}

else{

//logic to be excited it <condition> is false

}

\*\*else block is optional

\*\*<condition> should always return Boolean output.

\*\*to set a debugging breakpoint from the js file we use debugger in the code

9,forEach

this type of looping mainly happens indexes of an array

syntax:

<array>.forEach(function(value,index){

});

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

what is the difference between =, == , === in js?

= this means that we are setting a value to variable

== this means to check if the left hand side is equal to right hand side regardless of its type

=== this means to check if the left hand side is equal to right hand side both by data and by type

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

POPUPS:-

* alerts and confirm.

alert is a just debugging tool.

alert---

alert ("this is alert message")

confirm--

confirm ("this is confirm message")

var test=confirm("this is confirm message demo")

document.write(test);

prompt--

prompt ("enter your city")

var test = prompt("enter your city")

document.write("test");

===================================================================

Arrays:

a=[10,20,30] its an array.

to store multiple values.

array index starts from 0

we can write arrays in 3types.

a=[10,20]

alert(a(1));

var myarray = new Array(10,20,30)

alert(myarray[2]);

var test = [20,30,20]

alert(test[2]);

* array methods and concat and sort

var test= new Array(10,20,30);

alert(test.length); //output 3

* concat method to add more arrays in one variable or attach values

var test = new Array(10,20,30);

var test2 = test.concat(50,60,60);

alert(test2); //output is 10,20,30,50,60,60

* for sort--

var test = new Array(20,10,30);

test.sort(); //output is 10,20,30

===================================================================

JavaScript data types:

JS supports many data types.

data types are divided into two categories;

1.primitive data types:

* + these are the basic data types that you find in any language;
  + a) Number(integers and float - decimal number)
  + b) character (alphabets)
  + c) String( collection characters or alphabets is string. In short you can say a statement .
  + d)Boolean(this asks true or false)

2.object

3.functions

4.null

* + variable is defind but it has no value

5. undefined

* + it is variable is not defind

In JavaScript there are 5 different data types that can contain values:

* + - String
    - Number
    - Boolean
    - Object
      * Object
      * Date
      * Array
* And 2 data types that cannot contain values:
  + Null
  + Undefined
* The typeof Opertor : You can use the **typeof** operator to find the data type of a javascript variable.
* The data type of NaN is number
* The data type of an array is object
* The data type of a date is object
* The data type of null is object
* The data type of an undefined variable is **undefined**
* The data type pf a variable that has not been assigned a value is also **undefined\***
* You cannot use typeof to determine if a javascript object is an array.
* The constructor property returns the constructor function for all JavaScript variables.
* JS Regular Expressions: A regular expression is a sequence of characters that forms a search a pattern.
* A regular expression is a sequence of characters that forms a search pattern.
* Syntax:

/pattern/modifiers;

Var patt = /w3schools/I;

/w3schools/I is a regular expression.

W3schools si a pattern(to be used in a search) .

I is a modifier

===================================================================

Object:-

* these are special of data types where we can define our own set of datatypes objects take primitive data types inside them.
* objects are special type of data type.
* inside these objects we can call all the primitive data types.
* it gives us the power of creating our own structure of data types

syntax:

var x = {

<property> : <value of the property>

};

Example:

var x = {

firstName : 'srikar',

rollnumber: 25,

isPresent: false,

isSuspended:null,

y:{

lastName: 'kanth'

}

};

* now "X" is an object that holds four primitive data types.
* the "{" and "}" are called as Object-literals

there is a difference between variable and property

variable hold the entire object

\*\*don’t put comma to the last property, comma means there will be another property.

property is one entity of the object

}

updating properties. there are two ways of updating the properties of an object

1.dot notation

x.firstName = 'something',

2.property notation

x['firstName'] = 'something';

==================================================================

Functions: -

* functions are unique feature introduced in Js these take a value and then gives out a result.
* functions are totally different from other data types including objects.
* functions are used to perform some logic and then result out/return the result of that logic.
* the result can be any primitive data type or objects,

syntax:

function<name if the function>(){

return<statement>;

}

example:

function addition(){

var x = 1;

var y = x+1;

return y;

}

* functions allow us to create temporary variables inside them. this means that those variable live only inside that function.

example:

var x = function(value){

value + 1;

return value;

}

TYPES OF FUNCTIONS:

there are two types of functions;

1. named functions

2. anonymous functions.

Named functions: -

* this will have a name associated to them
* named functions must return something. whereas anonymous functions may or may not return any thing.

anonymous functions are also called ad CALLBACK FUNCTIONS

//function definitions: this means define a function.

function addition(){

var x = 1;

return (x+1);

};

//function calling: this means to call a function

addition();

var addition = 3;

console.log(addition);

console.log(addition());

===================================================================

SCOPE IN JS

* this simply means the life of the variable, if a variable is created inside a function then the scope of the variable is within that function, //local variable
* if a variable is created outside of function then the scope of that variable is within that entire file. // global variable

var global = 100; // gloabl variable

function test(){

var local =55;

};// local variable

* since life of the global variables is throughout the life of the app js does not know how much memory needs to allocate to that so because of that we always avoid global variables
* global variable slow our application
* if a variable is created inside a function then the scope of the variable is within that function if a variable is created outside a function then the scope of that variable is within that entire js file.
* since life of the global variables is through the life of the app js doesnt know how much memory it needs to allocate to that so because of that we always avoid global variables!!!

===================================================================

IIFE construct:

function initialize(){

var x = 10;

};

initialize();

* these are used to avoid memory leaks from global variables.

\*\*\*always have iife construct in your js. even if its only one line write it inside the iife construct.

===================================================================

Classes:

* objects have fixed properties and fixed values when we define them. so when we want to construct a dynamically changing object we end up using expression
* classes are nothing but functions that are converted to objects. by using this
* objects are collection of property
* functions is a set of statements that performs a test

difference between classes and objects:

* classes are functions converted to objects
* to convert a function to a class we used **'this'** key word inside the function.
* to inherit the properties of this class, we use the keyword 'new'
* class does not support two way bridge. they can only be cloned.

syntax:

function() fullname{

this.firstName= 'krishna';

this.lastName= 'kanth';

}

syntax for inheritance

var person = new fullName();

===================================================================

JS CLOSURES

* if we have one function called outerFunc inside which we create a variable called 'global' now we create a function called innerFun inside the outerFunc. this innerFunc has a same variable declaration. Then the call inside that innterFunc will always invoke the newly created variable insdie the innerFunc
* Definition:

lets say we have one function inside another function then we can variables created in the outer function inside the inner function.

is the above example js is going look for test inside the innerMost()

if it cannot find it it will move one immediate parent level up and will look for definition of test in inner90 if it still connot find it it will move one more level up and look for definition of test in outser(0 once it finds the definition it will take that if not the moves one level up and if it still cannont find the definition it will through an exception error saying

===================================================================

DOM: document object model

what is DOM?

* whenever we connect js with html and open that html in the browser then that browser is converted into ab object, there an object called WINDOW which gets constructed.

this window object holds all the document

\*\*from js prospective the entire html is called as 'document'

\*\*window object is the parent of all this window object is basiclally our browser inside java script each html now becomes a document as a property inside this window object

* hence to target an element we simply say either window.document.getElementById

document.querySlector which used to trigger elements

-------------------------------------------------------------------------------------------------------------

SELECTING AN ELEMENT FROM DOM

* we can make use of predefined properties of the document object. some of them are

1.getElementById() -> this targets only unique IDs

2.getElementsByClassName() -> this targets all the elements with the same class names

3.getElementByTagName() -> this targets all the elements with the specified tag names

4.querySelector() -> this is a special superset of all the selectors. we use these to target absolutely anything in the html document.

5.querySelectorAll() --> this is similar to querySelector() but it returns all the elements as an array.

-------------------------------------------------------------------------------------------------------------

DOM EVENTS:

* All the actions that happen on the html is considered as EVENTS in JS

There are some standard DOM events:

1. onload

this is triggered when the window is loaded

syntax:

<body onload="myTest()">

<p>this is a para</p>

</body>

<script>

(function(window){

window.myTest = function(){

console.log('test');

}

})(window);

</script>

2. onclick

this is triggered when the user clicks on an element

<div onclick="myClick()"></div>

<script>

(function(window){

window.myClick = function(){

console.log('I WAS CLICKED');

}

})(window);

</script>

3.onsubmit

this is triggered when we submit a form

4.onchange

this is triggered when the value of an element

5.onfocus

this is triggered when the element is in focus of an element

6.onblur

this is triggered when the element is focused out

\*\* every element has a default action/event. for example, the anchor tag has click navigation as default event

to stop the default action, we can make use of preventDefault callback

===================================================================

Event target

this are used to captured the events from the js side rather then html

these are 3 methods:

1.addEventListner():

This method is used to listen for an event

2.removeEventListener

3.dispatchEvent

===================================================================

Using string methods:

In JavaScript regular expressions are often used with the two string methods: search() and replace()

* The search() method uses an expression to search for a match and returns the position of the match.
* The replace() method returns a modified string where pattern is replaced.

Use a case insensitive regular expression to replace Microsoft with w3schools in a string

Var str = “visit microsoft!”;

Var res = str.replace(/Microsoft/I, “w3schools”); //visit w3schools!

I = perform case-insensitive matching

g = perform a global match

m = Perform multiline matching

Using test() :

The test() method is a RegExp expression method. It serachs a sting for a pattern and returns true or false depending on the result.

Var patt = /e/;

Patt.test(“The best things in life are free”); // true

Using exec():

The exec() method is a RegExp expression method. It searches a string for a specified pattern and returns the found text.

/e/.exec(“The best things in life are free!”);

E

JS ERRORS

The try statements lets you test a block of code for errors.

The catch statements lets you handle the error.

The throw statement lets you create custom errors.

The finally statement lets you execute code after try and catch regardless of the result.

try {  
    Block of code to try}  
catch(err) {  
    Block of code to handle errors}

JS Debugging:

* Console.log()
* Debugger
  + - The debugger keyword stops the exection of js and calls the debugging function.
    - var x = 15 \* 5;  
      debugger;  
      document.getElementbyId("demo").innerHTML = x;
* Use {} instead of new Object()
* Use "" instead of new String()
* Use 0 instead of new Number()
* Use false instead of new Boolean()
* Use [] instead of new Array()
* Use /()/ instead of new RegExp()
* Use function (){} instead of new Function()

var x1 = {};           // new object  
var x2 = "";           // new primitive string  
var x3 = 0;            // new primitive number  
var x4 = false;        // new primitive boolean  
var x5 = [];           // new array object  
var x6 = /()/;         // new regexp object  
var x7 = function(){}; // new function object

ECMA-262 is the official name of the standara. Ecmascript is the official name of the language.

|  |  |  |
| --- | --- | --- |
| Year | Name | Description |
| 1997 | ECMAScript1 | First edition |
| 1998 | ECMAScript2 | Editorial changes only |
| 1999 | ECMAScript3 | Added regular expressions added try/catch |
|  | ECMAScript4 | Was never released |
| 2009 | ECMAScript5 | Added “strict mode”  Added JSON support |
| 2011 | ECMAScript5.1 | Editorial changes |
| 2015 | ECMAScript6 | Added classes and modules |
| 2016 | ECMAScript7 | Added exponential operator(\*\*) added array.prototype.includes. |

***JSON***

* Json in a format for storing and transporting data.
* Json is often used when data is sent from a server to a webpage.
* JSON stands for java script object notation
* Json is lightweight data interchange format
* Json is language independent
* Json is self-describing and easy to understand

\*\* The json syntax is derived from js object notation syntax but the json format is text only. Code for reading and generating json data can be written in any programming language .

This json syntax defines an employee object an array of 3 employee records:

{  
"employees":[  
    {"firstName":"John", "lastName":"Doe"},   
    {"firstName":"Anna", "lastName":"Smith"},  
    {"firstName":"Peter", "lastName":"Jones"}  
]  
}

Json syntax rules:

* Data is in name/value pairs
* Data is separated by commas
* Curly braces hold objects
* Square brackets hold arrays

var x = {

firstname: 'krishna';

callback: function(){

return 'kanth';

},

rollnumber : 55

};

===================================================================

1. **How can you declare a class in Javascript?**

In javascript there's no classes like in Java, what we actually call

a class is in reality a function simulating a class behavior. For being so flexible, there are many ways to create a class in javascript

below you'll find 3 ways of doing that.

Class using function as a constructor:

Person(name) {

this.name = name;

}

// Creating an object

var person = new Person("Rafael");

person.name; // "Rafael"

It's very important to notice that you have to use the keyword new when creating a new instance of that class otherwise you will have logical problems regarding the this will reference window object.

2. **Difference between == and ===.**

This is pretty simple but at the same time some people never came across a triple equals or never wondered what's the difference.

Double equals == is used to compare the value of two operands:

"2" == 2; // true

2 == 2; // true

Triple equals === is used to compare the value AND type of two

operands:

"2" === 2; // false

2 === 2; // true

3. **Difference between null and undefined**

This can be tricky and the best way to keep in your head is to memories because if you try to relate javascript null to other languages, it will get more confusing.

In javascript, null is an object with no value and undefined is a type.

typeof null; // "object"

typeof undefined; // "undefined"

var a;

var b = null;

a == b; // "true" because their values are the same

a === b; // "false". they have different types

4.**Have you already used MVC before? What you like/dislike about it?**

As the UI gets more and more complex we need some good ways to keep it more and more maintainable and reusable, and Some MVC frameworks for javascript have been

widely adopted lately and it's a good plus if you have already used before and knows what's the benefits of them. The most famous MVC frameworks are backbone.js and

angular.js, it's hard to not hear about them.

There are many advantages in using these frameworks, I can point out some of them:

Organization: Forces your webapp to follow a well structured pattern;

Maintainable: With organization comes an easy to maintain code;

UI Binding: Some frameworks allow you to do that. So everytime your model changes, the view reflects it and vice-versa;

Decoupled client: MVC frameworks like backbone.js incentivise you to use REST API's though their urlRoot attribute in their Models;

Reusable components: Create reusable visual components;

Single-page apps: Build single-page apps with Ajax requests;

Friendly URL's: Native support for client-side url mapping;

**What are global variables? How are they declared? What are the problems with using globals?**

Global variables are available throughout your code: that is, the variables have no scope. Local variables scope, on the other hand, is restricted to where it is declared (like within a function). The var keyword is used to declare a local variable or object, while omitting the var keyword creates a global variable. Most JavaScript developers avoid global. One reason why is they're averse to naming conflicts between local and globals, Also, code that depends on globals can be difficult to maintain and test.

// Declare a local variable

var localVariable = "TechRepublic"

// Declare a global

globalVariable = "CNet"

## ****How do you organize your JavaScript code?****

The key concept here is to get an idea of how the candidate maintains and designs code. Do they design code that is specific to an application with no possible reuse? Do they use class inheritance or the module pattern to build reusable code? These approaches allow multiple developers to work on a project without stepping on their coworkers' toes. In addition, testing modular code or classes is easier to approach than a jumbled mess of code thrown together (look around the Web, and you'll find plenty of examples).

## ****What are JavaScript types?****

Unlike Java or C#, JavaScript is a loosely-typed language (some call this weakly typed); this means that no type declarations are required when variables are created. Strings and numbers can be intermixed with no worries. JavaScript is smart, so it easily determines what the type should be. The types supported in JavaScript are: Number, String, Boolean, Function, Object, Null, and Undefined.

var fName = "Mary";   //Declare a String

var total = 100.32;    //Declare a number

var fName = new String; //Another way to declare a string

fName = "Mary";

var total = new Number;

var isIt = new Boolean;

var names = new Array;

var car = new Object;

## ****What is the difference between undefined and null?****

The value of a variable with no value is undefined (i.e., it has not been initialized). Variables can be emptied by setting their value to null. You can test for each using the === (three equal signs) or == (two equal signs) for comparison checking. The big difference is the latter uses coercion, which can have some [odd results](http://ecma-international.org/ecma-262/5.1/#sec-11.9.3) -- it returns true for a null or undefined comparison if they are either.

if (nullExample === null) { // executes this block only if null }

if (undExample ===Undefined) { // executes this block only if Undefined }

if (bothExampe == null) { // executes this block if Undefined or null }

You can be more exact with a comparison by using the typeof to return an object's type.

If (typeof variable ==="undefined")  { // executes this block of if undefined }

## ****What is JavaScript's this keyword?****

JavaScript's this keyword normally refers to the object that owns the method, but it depends on how a function is called. Basically, it points to the currently in scope object that owns where you are in the code. When working within a Web page, this usually refers to the Window object. If you are in an object created with the new keyword, the this keyword refers to the object being created. When working with event handlers, JavaScript's this keyword will point to the object that generated the event.

## ****What is event bubbling?****

Event bubbling describes the behavior of events in child and parent nodes in the Document Object Model (DOM); that is, all child node events are automatically passed to its parent nodes. The benefit of this method is speed, because the code only needs to traverse the DOM tree once. This is useful when you want to place more than one event listener on a DOM element since you can put just one listener on all of the elements, thus code simplicity and reduction. One application of this is the creation of one event listener on a page's body element to respond to any click event that occurs within the page's body.

## ****Do you have a JavaScript framework preference? What are your thoughts on using frameworks?****

This open-ended question has the potential to spawn a good conversation. There are the vastly popular frameworks like [jQuery](http://www.techrepublic.com/article/simplify-javascript-development-with-jquery/6187023), although you might be surprised when the person tells you about the framework they developed or even played the contributor role.

The answer to the second question gives you an idea of the candidate's feelings about open source (well, that is the way I see it). There are so many open source options available today ([Knockout](http://knockoutjs.com/), [postal.js](http://www.techrepublic.com/blog/programming-and-development/utilize-javascript-messaging-with-postaljs/5321), [jQuery](http://www.techrepublic.com/blog/webmaster/how-to-get-started-with-jquery/1415), etc.), and a developer's time is very valuable, so why reinvent the wheel? These open source options provide robust code that has been thoroughly tested by an army of developers. From my perspective, I want developers who will use whatever is available to meet a project's demands. Plus, the interviewee might introduce you to something you've never used.

## ****How are errors gracefully handled in JavaScript?****

Exceptions that occur at runtime can be handled via try/catch/finally blocks; this allows you to avoid those unfriendly error messages. The finally block is optional, as the bare minimum to use is try/catch. Basically, you try to run code (in the try block between the braces), and execution is transferred to the catch block of code when/if runtime errors occur. When the try/catch block is finally done, code execution transfers to the finally code block. This is the same way it works in other languages like C# and Java.

try {

// do something

} catch (e) {

// do something with the exception

} finally {

// This code block always executes whether there is an exception or not.

}

You can give bonus points to any candidate who discusses the onerror event handler tied to the Window object in the browser -- this allows it to monitor all errors on a page. This allows you to properly handle code syntax errors and runtime exceptions.

## ****Can you explain how inheritance works in JavaScript?****

This subject confuses many developers, and I would expect a candidate to stammer on this question or throw up their hands and say "can anybody?" Instead of trying to explain, take a look at this [overview on the Mozilla Developer Network](https://developer.mozilla.org/en-US/docs/JavaScript/Guide/Details_of_the_Object_Model).

## ****How do JavaScript timers work? What is a drawback of JavaScript timers?****

Timers allow you to execute code at a set time or repeatedly using an interval. This is accomplished with the setTimeout, setInterval, and clearInterval functions. The setTimeout(function, delay) function initiates a timer that calls a specific function after the delay; it returns an id value that can be used to access it later. The setInterval(function, delay) function is similar to the setTimeout function except that it executes repeatedly on the delay and only stops when cancelled. The clearInterval(id) function is used to stop a timer. Timers can be tricky to use since they operate within a single thread, thus events queue up waiting to execute.

6)What is the difference between javascript and jscript?

* Both are almost similar. Java script is developed by Netscape and jscript was developed by Microsoft.

7)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.

8) write about the errors shown in javascript?

* Load-time errors:
* Run-time errors:
* Logic errors: it is caused by the use of syntactically correct code, which does not ful fill the required task

9) Does Javascript support automatic type conversion. If yes give example?

* Yes js support automatic type conversion. You should take advantage of it it is most common way of type conversion used by javascript developers.

Ex:-

var s = '5';

var a = s\*1;

var b = +s;

typeof(s); *//"string"*

typeof(a); *//"number"*

typeof(b); *//"number"*