

Javascipt

Complete Reference



June 23, 2015

[Company name]

[Company address]

# Basics in the Javascript

## Javascript compatibility

To avoid javascript to execute in the older browser include

<!--

//-->

<script type=”text/javascript”>

<!--

document.write(“This won’t be visible in the javascript incompatible browsers”);

//-->

</script>

## Variables

Variables must be declared with the keyword var. variables are case sensitive.

## Variable naming

1. The variable name must begin with \_ or alphabets. Must not start with numbers or special characters

2. A variable name can contain only \_ or alphabets or numbers.

3. A variable name should not contain special characters.

## Types of Variable

Javascript can handle numbers, decimals, negative numbers, Boolean and string.

Number:

var a=123;

Decimals:

var b=23.343;

Negative Numbers:

var c=-2343.343;

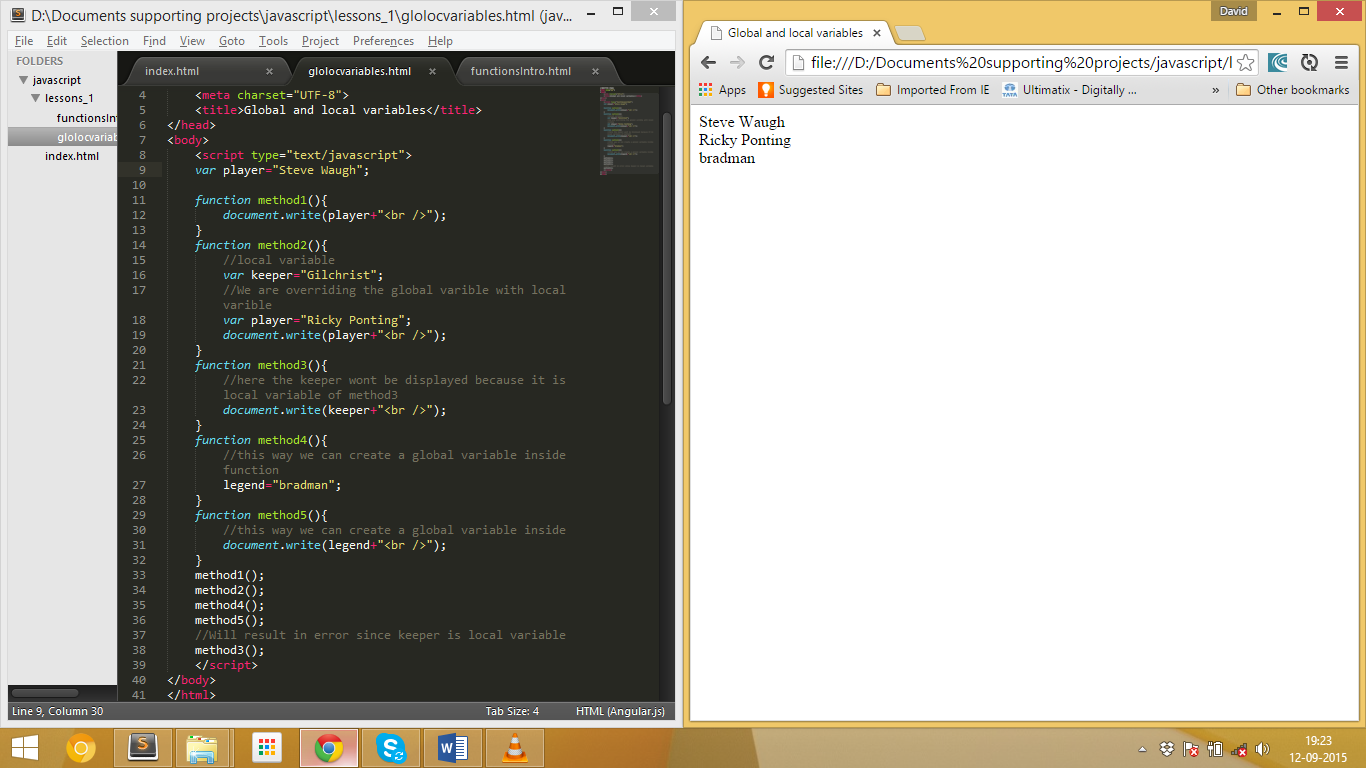
String: (the string must be enclosed with “)

var d=”Hi David”;

Boolean:

var bol=true;

## Global Variables and local variables



## Escape Characters

To include the Special symbols like “double quotes in the string. We need need to include the escape character \.

Example:

<script type="text/javascript">

var name="\"Hi David \" how are are you?";

var name2='\"Hi David \" how are are you?';

document.write(name);

document.write(name2);

</script>

Output



# Functions

## Creating a function in javascript

The function can be created in two ways. The function can be created as variable or the function keyword followed by the function name.

1. Function as variable

var function1=function(){

document.write(“Hi david”);

}

1. Function followed by function keyword

function function2(){

document.write();

}

## Passing the arguments in the function

While calling the function it must match the function name no need to pass the exact number of arguments.

Example:

<body>

<script type="text/javascript">

var function1=function(abc,bcd){

document.write(abc+" how are you ? "+bcd);

}

</script>

<form>

<input type="button" value="click me" onclick="function1('HI')">

</form>

</body>

Output:



# Math Operators:

Like other programming language javascript has

Addition: +

Subtraction: -

Multiplication: \*

Division: /

Modulus: %

Increment and decrement operator: ++ and –

Assignment operator: =

Addition assignment: +=

Subtraction assignment: -=

Multiplication assignment: \*=

Division Assignment: /=

# Conditional Statements:

1. **If else**

if(condition){

}

else{

}

1. **Switch**

switch(condition){

case value1:

to do statements

break;

case value2:

to do statements

break;

default:

to do statements

}

1. **For**

for(x=0;x<5;x++){

}

1. **While loop**

while(condition){

}

1. **do while**

do{

}while(condition);

# Event Handlers

1. onClick:

This event is triggered when the left button in mouse is clicked. The keyword in **onClick**. We can call n number of function inside the **onClick. (i-e) onClick=”alert(‘ Hi David’);alert(‘second ‘);”**

**Example:**

<form>

<input type=”button” value =”click me” onClick=”alert(‘Thanks for clicking’);”/>

</form>

1. onMouseOver:

This event is triggered when the mouse is moved over the particular element. The keyword in **onMouseOver.**

**Example:**

<form>

<input type=”button” onMouseOver=”alert(‘The mouse is moved over me’);” value=”Move the mouse over me” />

</form>

1. onMouseOut:

When we move the mouse out of the particular element the onMouseOut event is triggered. The keyword is **onMouseOut**.

**Example:**

<form>

<input type=”button” value=”move the mouse out of me” onMouseOut=”alert(‘The mouse is moved out’);” />

</form>

1. onLoad:

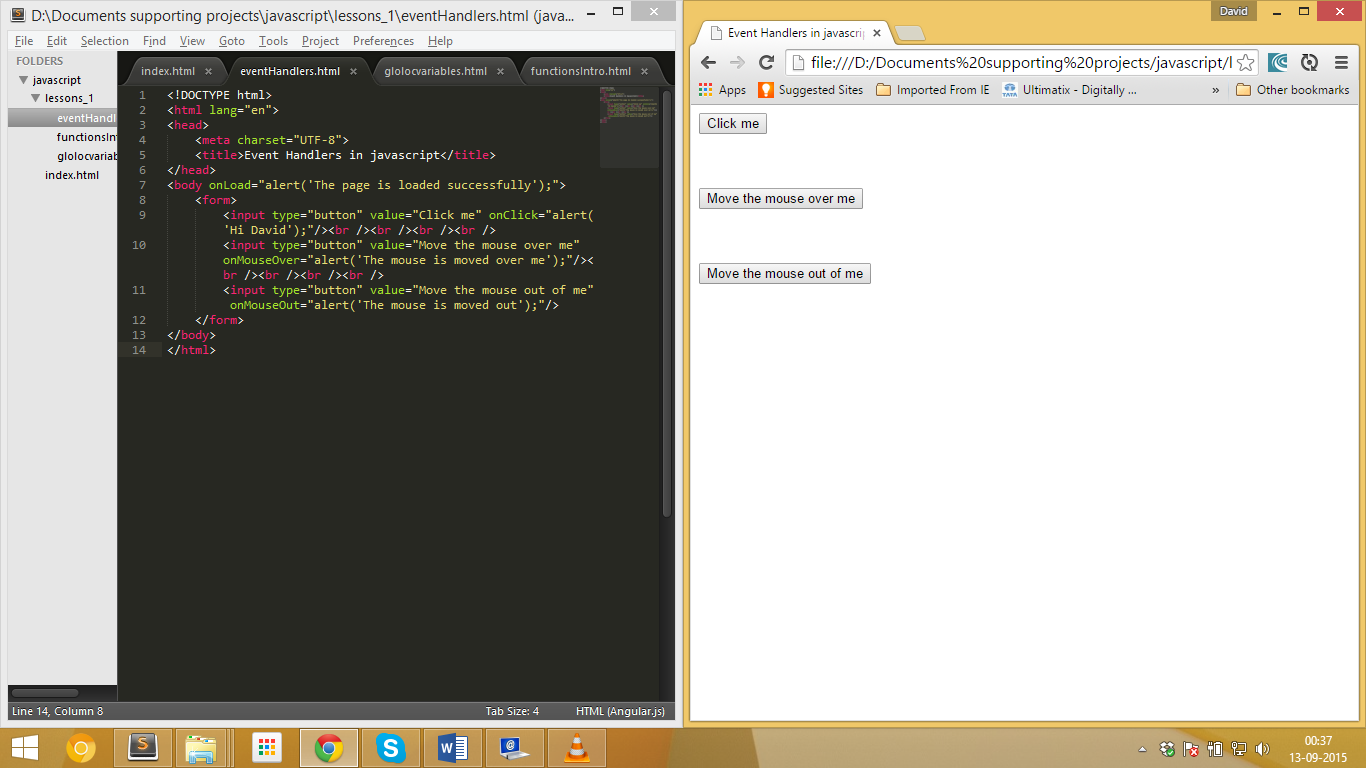
When the page is loaded completely this event is triggered. The keyword is **onLoad.**

**Example:**

<body onLoad=”alert(‘The page is loaded successfully’);”>

<h1> hi david </h1>

</body>



1. onUnLoad:

Whenever the page is closed or it is refreshed the onUnLoad event is triggered. The keyword is **onUnload**. It is not supported in some of the browser(Ex: chrome).

**Example:**

<body onUnload=”alert(‘Thanks for visiting’);”>

</body>

# Objects

## Primitive and Objects

## Creating a new object

function car(color,speed){

this.color=color;

this.speed=speed;

}

var ferrari=new car(“red”,495);

## Object initializers

samsung={model:"S6 Edge",price:50000};

iphone={model:"6S",price:60000}

document.write("The latest model of samsung is "+samsung.model+" and its price is "+samsung.price+" which is less than iphone model "+iphone.model+"'s price "+iphone.price);

## Adding method in object

<script type="text/javascript">

function car(name,speed){

this.name=name;

this.speed=speed;

//one way of adding method in object

this.convertToKmph=function(){

return this.speed\*1.6;

}

//this is another way adding method in the object

this.nitrousPresent=checkNitrous;

}

function checkNitrous(){

var isPresent=false;

switch (this.name){

case "Ferrari":

isPresent=true;

break;

case "Lamborgini":

isPresent=true;

break;

default:

isPresent=false;

}

return isPresent;

}

var ferrari=new car("Ferrari",300);

var lamborgini=new car("Lamborgini",300);

var ford=new car("Ford",300);

document.write(ferrari.speed);

document.write(ferrari.convertToKmph());

document.write(ferrari.nitrousPresent());

document.write(ford.nitrousPresent());

</script>

## Arrays

The array stores the element of same data type in contiguous memory location.

Syntax:

var players=new Array(“Ricky Ponting”,”Adam Gilchrist”,”Glenn Mgrath”,”Brett Lee”);

<script type="text/javascript">

There are three ways to create arrays they are

**First way:** The elements are inserted into the array when initialization

var players=new Array("Adam Gilchrist","Ricky Ponting","Michael Clarke","Phil Hughes","Brett Lee","Glenn Mgrath");

document.write("Best Captain in the world \""+players[1]+"\"<br />");

**Second way:** The size of the array is declared first and the elements are inserted later

var footballPlayers=new Array(4);

footballPlayers[0]="Lionel Messi";

footballPlayers[1]="Andres Iniesta";

footballPlayers[2]="zinadane zidane";

footballPlayers[3]="Neymar";

document.write(footballPlayers[0]+" is the best player in world<br />");

**Third way:** No need to declare the size of the array can insert any number of elements later

var games=new Array();

games[0]="cricket";

games[1]="Football";

games[2]="kabadi";

games[3]="Hockey";

games[4]="swimming";

document.write(games[1]+" has lot of fan followers all over the world");

</script>

### Associative array

The associative array takes the key as string rather than number.

**Example:**

var asso\_array=new Array();

asso\_array["name"]="david";

asso\_array["age"]=24;

document.write("My name is "+asso\_array["name"]+" and my age is "+asso\_array["age"]);

## Array Methods

### Length

The length is used find the length of the array.

**Syntax:**

array\_name.length

**Example:**

var players=new Array("Adam Gilchrist","Ricky Ponting","Michael Clarke","Phil Hughes","Brett Lee","Glenn Mgrath");

document.write(players.length);

### Split

The split method is used to split the string into array. Split takes in a parameter. (the string is split based on that parameter.

**Syntax:**

Full\_String.split(“ “);

**Example:**

var fullString="Hi! How are you?"

var arr\_fullString=fullString.split(" ");

document.write(arr\_fullString[3]);

### Join

The join method is used to join the array into a string. Join method takes in a parameter. (array elements are joined based on that string default join char is **,**).

**Syntax:**

arr\_name.join(“ “)

**Example:**

var batsman=new Array("Burns","Finch","Smith","Bailey");

document.write(batsman.join(" ")+"<br />");

### Pop

The pop is used to take the last element out of the array.

**Explanation:**

1. Consider an array contains 6 elements:

var players=new Array("Adam Gilchrist","Ricky Ponting","Michael Clarke","Phil Hughes","Brett Lee","Glenn Mgrath");

1. Now the pop will pop the last element from the array.

document.write(players.pop());

### Concat

The concat method is used to concatenate two array.

**Syntax:**

first\_arr.concat(second\_arr);

**Example:**

var batsman=new Array("Burns","Finch","Smith","Bailey");

var bowler=new Array("Hastings","Starc","Cummins","Agar");

var team=batsman.concat(bowler);

### Reverse

The reverse method is used to reverse the elements present in the array.

**Syntax:**

array\_name.reverse();

**Example:**

var batsman=new Array("Burns","Finch","Smith","Bailey");

var bowler=new Array("Hastings","Starc","Cummins","Agar");

var team=batsman.concat(bowler);

document.write(**team.reverse()**+"<br />");

### Push

The push method is used to insert the element at the end of the array.

Push method can be used in three ways

1. To insert one element to the end of the array

var batsman=new Array("Burns","Finch","Smith","Bailey");

batsman.push(“Marsh”);

1. To insert N elements in to end of the array

var batsman=new Array("Burns","Finch","Smith","Bailey");

batsman.push(“Marsh”,”Maxwell”);

1. To insert an array to the end of an array

var batsman=new Array("Burns","Finch","Smith","Bailey");

var bowler=new Array("Hastings","Starc","Cummins","Agar");

batsman.push(bowler);

### Sort

The sort method is used to sort the elements in an array in the alphabetical order.

**Syntax:**

array\_name.sort();

# Dialog Boxes

## Alert

The alert is used to display the alert message as pop up window in browser.

**Syntax:**

alert(“aler\_message”);

**Example:**

alert(“Congratulations”);

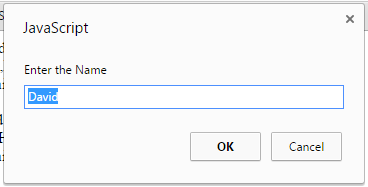
## Prompt

The prompt is like alert but it also has text box to get the user input.

prompt(“Enter the text to display”,”default value”);

Example:

var prompt\_var= prompt(“Enter the Name”,”David”);



# Math Objects

## Math Constants

1. PI Value

**Syntax:**

Math.PI

1. Euler value

**Syntax:**

Math.E

## Math methods

1. Square root:

**Syntax:**

Math.sqrt(number);

**Example:**

Math.sqrt(144);

# Date Objects

## Set Interval method:

The set Interval method two input parameter.

**Syntax:**

setInterval(“function\_name”,time\_interval\_in\_millisecond);

**function\_name**: The function will be executed after the specified millisecond

**time\_interval\_in\_millisecond**: the interval in which the function should be invoked.

## Date:

To get the current date and time use

var todayDate=new Date();

The current date and time will be stored in todayDate object.

To get the hours, minutes and seconds

todayDate.getHours();

todayDate.getMinutes();

todayDate.getSeconds();

# JSON JavaScript Object Notation

# Primitives and Objects

Not everything in javascript is objects.

The list of javascript primitives are

String

Number

Boolean

Undefined

Null

The list of javascript objects are

Object

Array

Function

Date

RegExp

Etc.,

The primitives does not have any methods or property

So, how a String has property.

For example:



Here the “david” has the length property How??

The javascript creates an object for the primitive string(“david”)

(I-e)



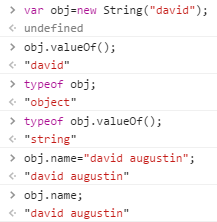
Then it destroys the string object and it is garbage collector;



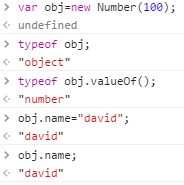


The only way to create an object from the primitive value is to use the constructor.

String Constructor:



Number Constructor:



We can’t able to assign the property to primitive variables.

# Factory functions:

Old way of creating objects in the javascript:

var car=new Object();

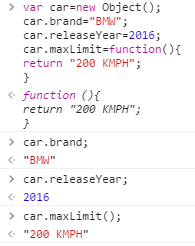
car.brand=”BMW”;

car.releaseYear=2016;

car.maxLimit=function(){

return “200 kmph”;

}



Javascript can execute single complex statement faster than the multiple simple statements;

var car={

brand:”BMW”,

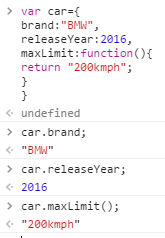
releaseYear:2016,

maxLimit:function(){

return “200 kmph”;

}

}



Factory function:

Factory function and constructor function are different.

Factory function is created by the following method.

var car=function(brand,releaseYear){

return{

brand:brand,

releaseYear:releaseYear,

maxSpeed:function(){

return "200 kmph";

}

};

};



# This and bind keyword

# JAVASCRIPT Patterns

## 14.1. Minimizing the global

### What is global variable

The variable which we create in the global scope is global variable.

Example:

var global\_var='placeholder';

There is another way we can create global variable in the JavaScript (implied global). The variable which we create inside the function without the **var** keyword will become **implied global**. When we use ‘use script’ we will get error for the implied global. Implied global is bad coding practice.

Example:

var global\_var='placeholder';

function func(){

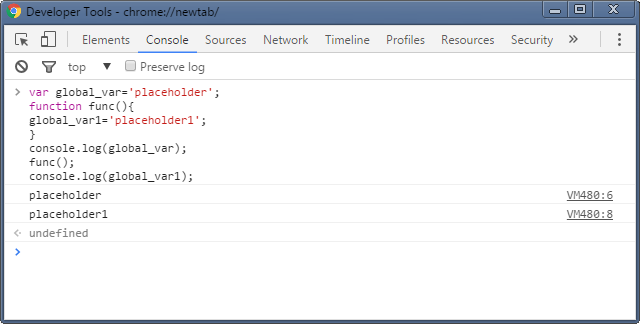
global\_var1='placeholder1'; //implicit global

}

console.log(global\_var);

func();

console.log(global\_var1);



Whereas the variable created in local scope is not available in the global scope.

Example:

var global\_var='placeholder';

function func(){

global\_var1='placeholder1';

var abce='some value';

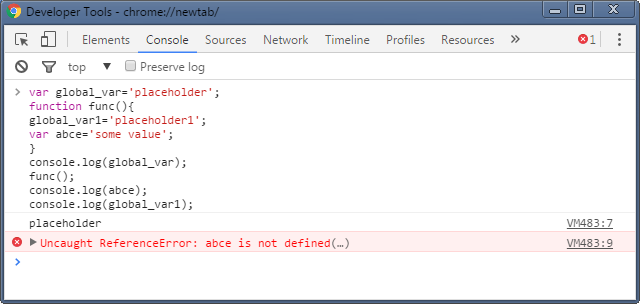
}

console.log(global\_var);

func();

console.log(abce);

console.log(global\_var1);



The Implied global are created by chained assignments also.

function func(){

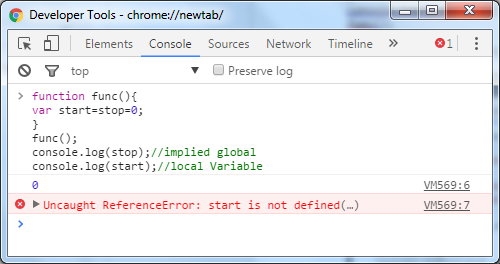
var start=stop=0;

}

func();

console.log(stop);//implied global

console.log(start);//local Variable



Here the chained assignment var start=stop=0; will be executed as var start= (stop=0); where the stop variable will become implied global.

To reduce the implied global by the chained assignments declare all the variables in beginning.

function func(){

var first,second;

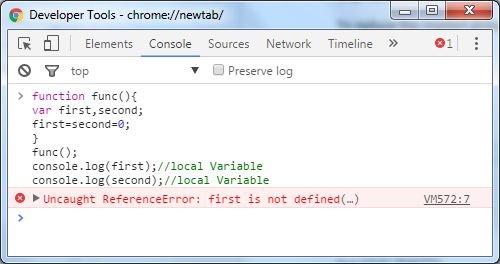
first=second=0;

}

func();

console.log(first);//local Variable

console.log(second);//local Variable



Every global variable we create becomes the property of the global object.

In browsers, there is a property called **window** that points to the global object itself. We can access the global variable with this window property.

Example:

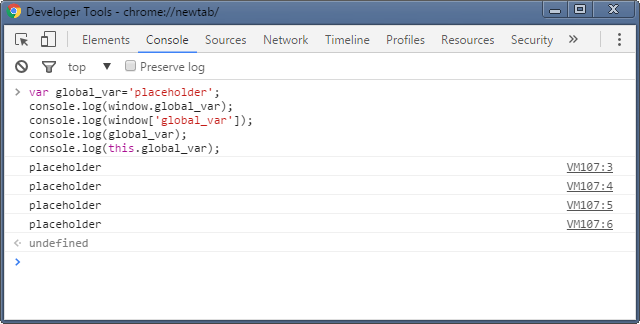
var global\_var='placeholder';

console.log(window.global\_var);

console.log(window['global\_var']);

console.log(global\_var);

console.log(this.global\_var);



### 14.1.2. Why we need to minimize the global variable?

There are scenarios when two different part of the same application can use same global variable. This will result in unexpected behaviour since one variable may be overridden by the other global variable.

For example:

Consider we are writing a script with the global variable render. There is script for advertisement which uses a global variable name render. When both scripts are placed in the same application one of render will be overridden which will result in anomaly behaviour.

### Difference between global, implied global

The implied global and global has one difference (i-2) implied global can be deleted (irrespective of its scope) whereas the global variable cannot be deleted.

var global\_var="placeholder";

global\_implied\_gs="global scope implied global";

function func(){

global\_implied\_ls="local scope implied global";

var local\_var='local scoper';

}

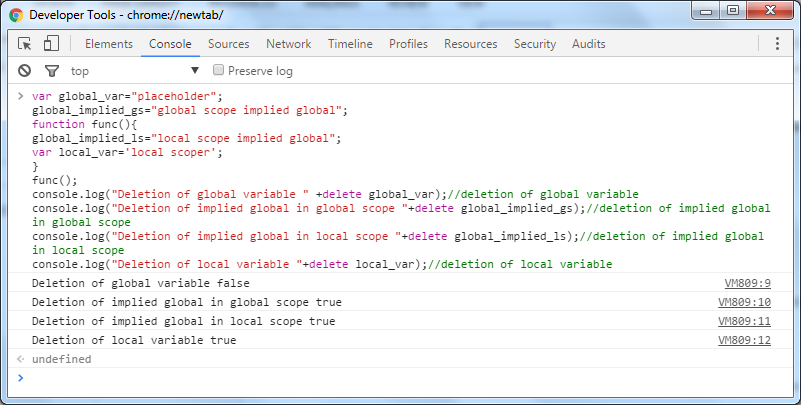
func();

console.log("Deletion of global variable " +delete global\_var);//deletion of global variable

console.log("Deletion of implied global in global scope "+delete global\_implied\_gs);//deletion of implied global in global scope

console.log("Deletion of implied global in local scope "+delete global\_implied\_ls);//deletion of implied global in local scope

console.log("Deletion of local variable "+delete local\_var);//deletion of local variable



## Using single var declaration

### 14.2.1 Benefits of using single var

* To overcome the hoisting problem.
* Less code so useful during minification of scripts

### 14.2.2. Hoisting:

JavaScript enables you to have multiple var statements anywhere in a function, and they all act as if the variables were declared at the top of the function. This behavior is known as hoisting.

For example:

var name="David";

function func(){

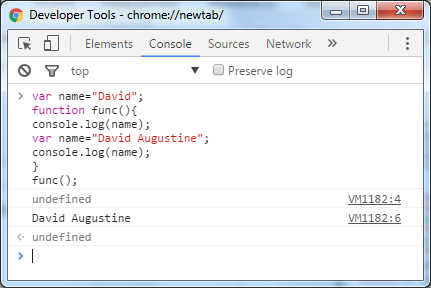
console.log(name);

var name="David Augustine";

console.log(name);

}

func();



The output we will expect is David and David Augustine. But the output is undefined and David Augustine. This is because the variable name which we declared inside the function func will act as it was defined at the top of the function.

The above code will be similar to the below code.

var name="David";

function func(){

var name;

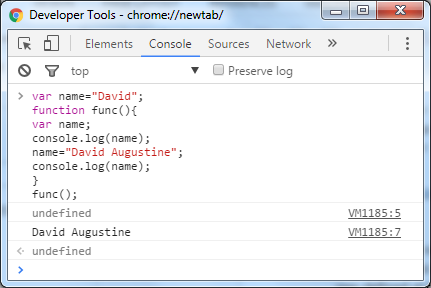
console.log(name);

name="David Augustine";

console.log(name);

}

func();



This example will make it more clear.

var name="David";

function func(){

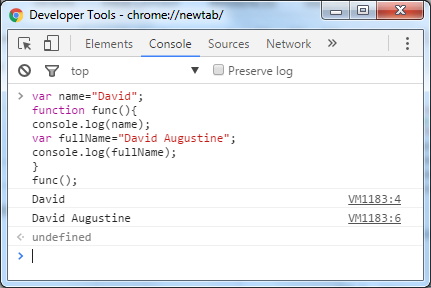
console.log(name);

var fullName="David Augustine";

console.log(fullName);

}

func();



## Optimizing the conditions in the “for” loop:

The for loop below will compute the length of the array each and every time. Instead of writing the for loop like below

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

//some logic

}

We can write like below, It will improve the performance because the length will be computed only and it is saved in the max variable and this value will be used every time.

for(var i=0,max=myArray.length;i<max;i++){

//some logic

}

var myArray = ['one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten','one','two','three','four','five','six','seven','eight','nine','ten'];

var t0 = performance.now();

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

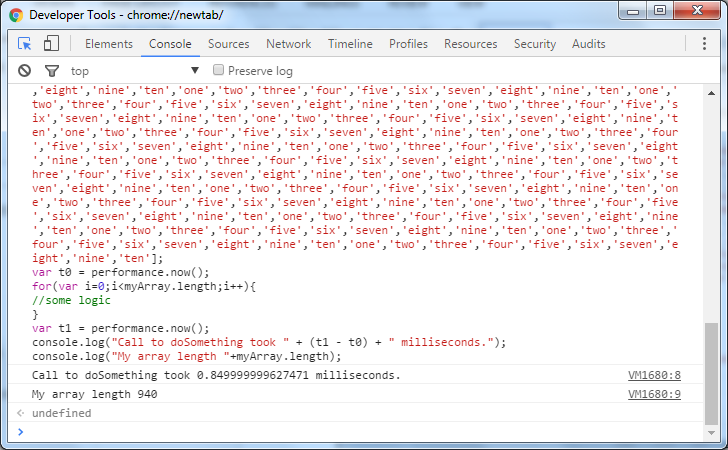
//some logic

}

var t1 = performance.now();

console.log("Call to doSomething took " + (t1 - t0) + " milliseconds.");

console.log("My array length "+myArray.length);



var t0 = performance.now();

for(var i=0,max=myArray.length;i<max;i++){

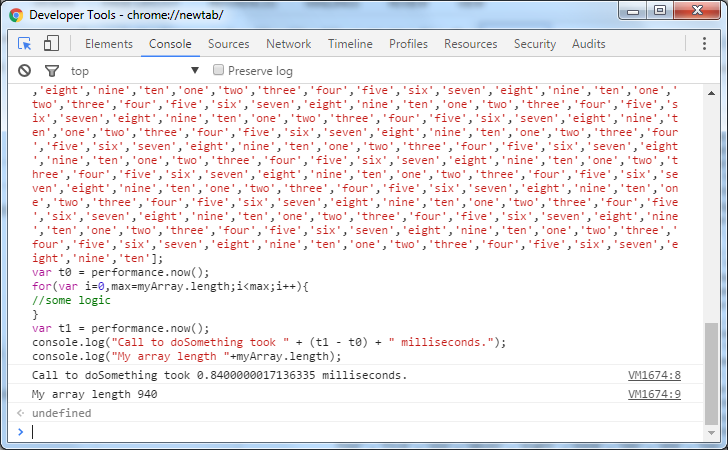
//some logic

}

var t1 = performance.now();

console.log("Call to doSomething took " + (t1 - t0) + " milliseconds.");

console.log("My array length "+myArray.length);



## For-in loops

For – in loops are used in while working with object and for loops are used while working with the array.

While using the for-in loop to iterate the object it is essential to include the **hasOwnProperty** filter to exclude the properties that came down the prototype chain.

//creating an object named ferrari

var ferrari={

speed:'250kmph',

color:'green',

cost:'1 million'

}

//include the manufacture date for all the product

if(typeof Object.prototype.manufactureDate==="undefined"){

Object.prototype.manufactureDate=new Date();

}

//creating an object named mercedes

var mercedes={

speed:'175 kmph',

color:'white',

cost:'1 million'

}

console.log('Printing the properties in the mercedes object');

for(var i in mercedes){

console.log(mercedes[i]);

}

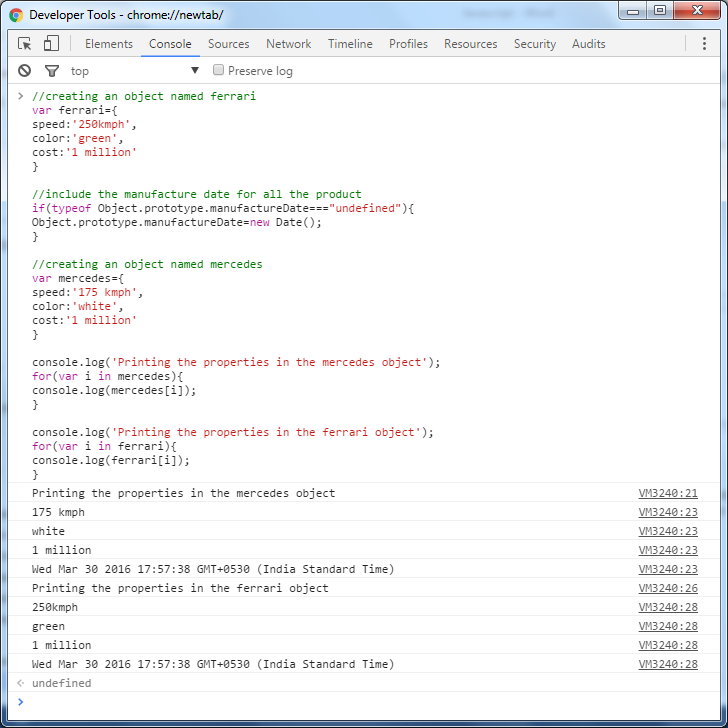
console.log('Printing the properties in the ferrari object');

for(var i in ferrari){

console.log(ferrari[i]);

}

In the above code we are not including the **hasOwnProperty()** filter so we will get all the property (including the manufacture Date which is created by prototype chain).



//creating an object named ferrari

var ferrari={

speed:'250kmph',

color:'green',

cost:'1 million'

}

//include the manufacture date for all the product

if(typeof Object.prototype.manufactureDate==="undefined"){

Object.prototype.manufactureDate=new Date();

}

//creating an object named mercedes

var mercedes={

speed:'175 kmph',

color:'white',

cost:'1 million'

}

console.log('Printing the properties in the mercedes object');

for(var i in mercedes){

if(mercedes.hasOwnProperty(i)){

console.log(mercedes[i]);

}

}

console.log('Printing the properties in the ferrari object');

for(var i in ferrari){

if(ferrari.hasOwnProperty(i)){

console.log(ferrari[i]);

}

}

