**Javascript**

To print in web browser:- **alert(“Hello World”)**

To print in console:- **Console.log(“Hello World”)**

To give input in browser:- **prompt(“Enter your name: ”)**

**25th March 2019**

JS is the only language for web browser programming.

We can create applications for desktop and Mobile.

For Desktop :- We can write JS program using **ELECTRON**

For Mobile/Tablet :- We can write JS program using **CORDOVA**

For Server :- We can write JS program using **NodeJS**

**JS versions & History:-**

1995 > JS created for Netescape program (First browser)

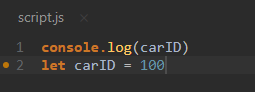
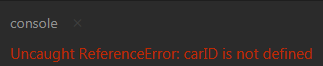
1997 > Standardize (By ECMA)

1993 > ECMA script 3

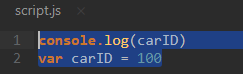
2009 > ECMA script 5 (ES 5)

2015 > ECMA script 15 (ES 6)

**Let & var for variable declaration:-**

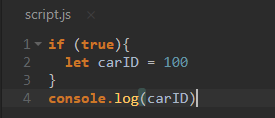
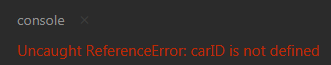
 

In case of var

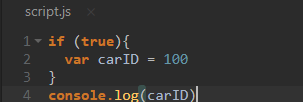
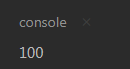
 No error

(Its best practice to use let over var)

“let is scope blocking”

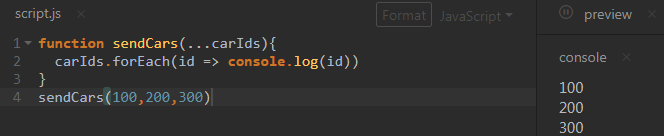
 

But in case var we can access the carID prefectly

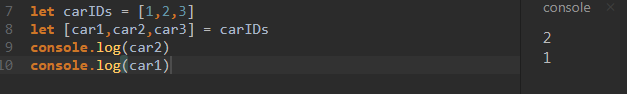
**REST parameters:-**

It allows a function to store multiple arguments in a array.Its same as args and kwargs in python

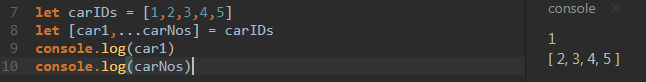


(REST parameter is always used as last parameter of the argument list in function)

**Destructuring the Arrays:-**

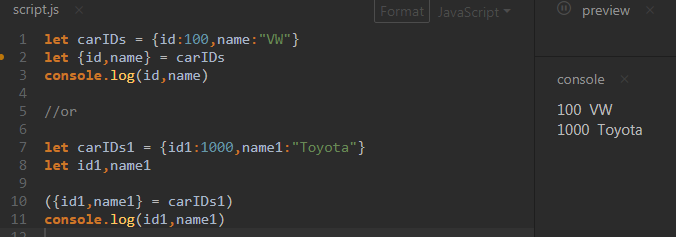


And



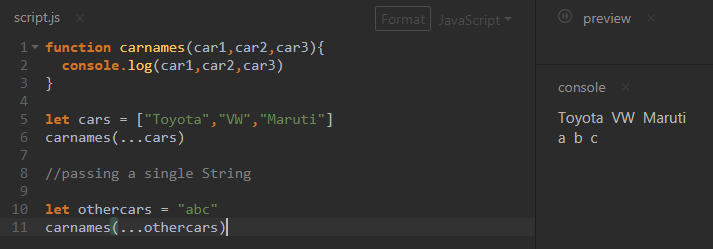
**Destructuring the Objects:-**

Its same as Array destructuring except it takes key pair (Dectinary in Python)

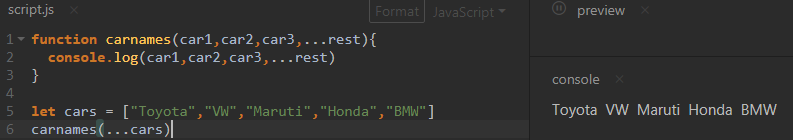


**Spread Syntax:-**

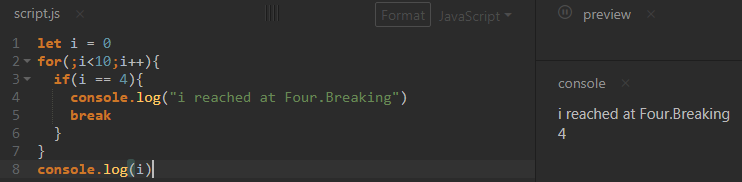
Take an array and spread out its elements.Its just oppsite of REST parameters



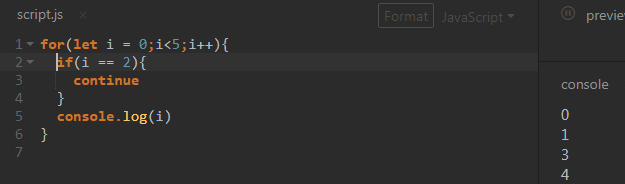
Using Spread and Rest together



**Controlling Loops:-**



When Continue block hit ,it will not run anything except i++



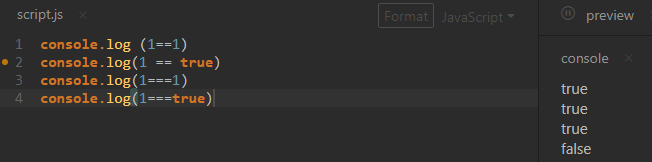
**Operators:-**

**Equality operator :-**

If (var1 == var2) {} // Here js engine wil try to convert the type of var1 to make same as var2

If(var1 === var2) {} //Here user has to give both vars indentical to each other.Strict equality

Note:- Best practice is to use ===

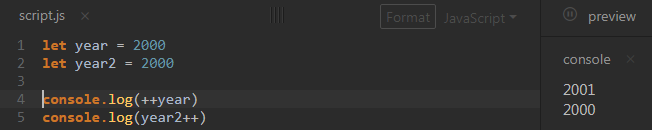


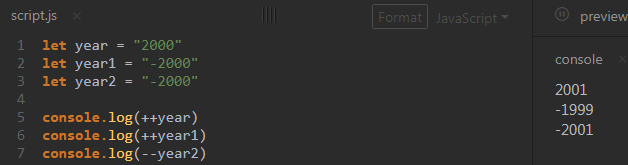
**Unary Operator:-**

++var = First increment then use

Var ++ = First use then increament

Uninary operator opeartes on only single variable



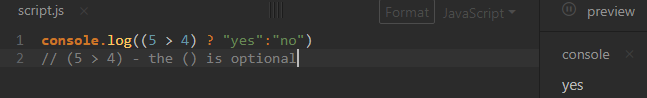


**Relational Operaotors:-**

All upper case values considered less than lowercase values

Zoo < alpha //true

**Conditional Operaror:-**



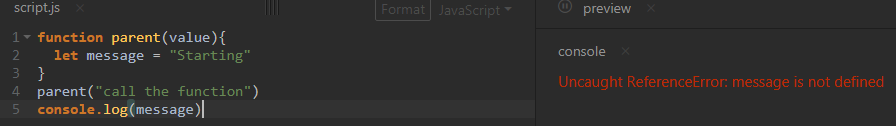
**Operation Precedence:-**

Refer the table here

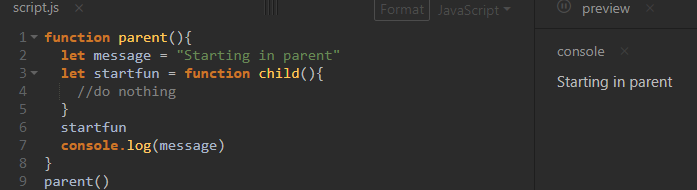
<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence>

**Function & Scope:-**

Variables that can be accessible inside and outside the functions.



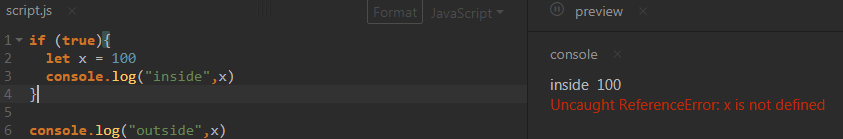
Here as soon as function excuted the variables inside removed hence meesage is not printing.



Here although startfun is completed but still parent function is still running hence we got console.log output.

**Block Scope:-**

Variables present inside the {} can not be accesses from outside.

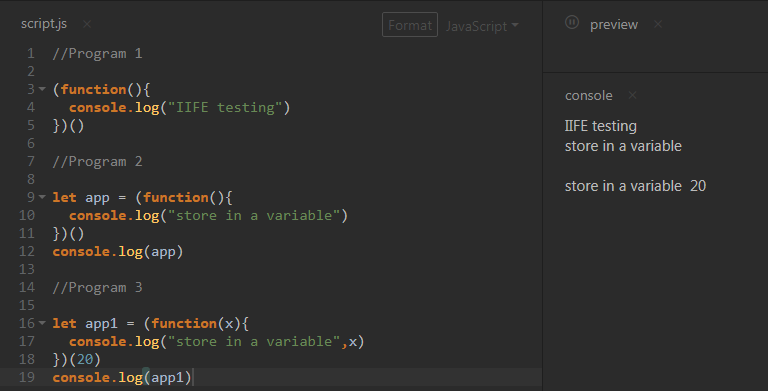


Note:- var keyword don’t follow the Block Scope

**IIFE’s:-**

Immediately Invokes Function Expression

This is used for managing the code and treat portions of the code isolate like modules and functions.



Program 1 shows we are creating a function without any name (just key word) and calling it as soon as it over using **( )**

This is called immediate invocation of the Function expression

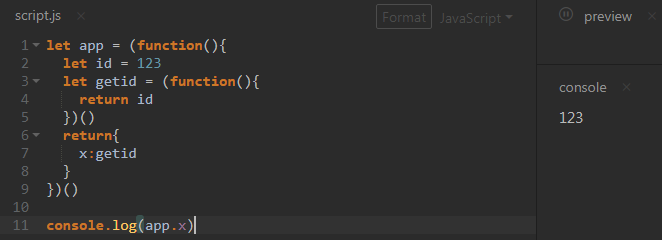
Program 2 shows we can use the Function expression and store the result in a variable.

Program 3 shows the same but here we are passing the argument while invoking the function.

The real use of IIFE will be realized when we use the Closure in IIFE.

**Closures:-**

When we run the function all the code statements inside its scope executed and become unavailable once function execution finished but sometimes we want these values to be hang around and we can do that using Closure in IIFE pattern.



**Return {**

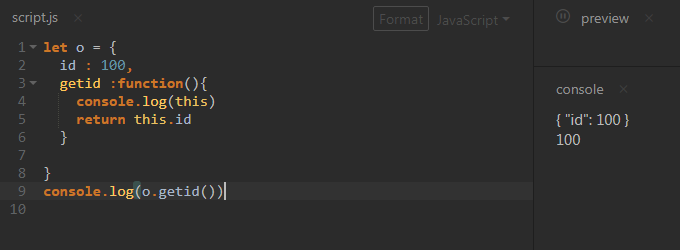
**X:getid**

**}**

Is called Closure bcz x:getid calles the getid function and when getid function executes its search for id variable when its not found in getid then it search it in parent function ,that way the entire code runs and holds the value .

**The “this” key word:**

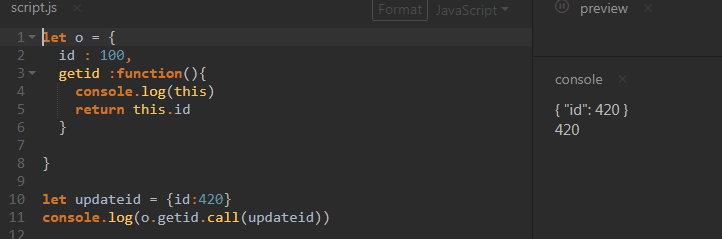
Every function is always associated with the “this” keyword.



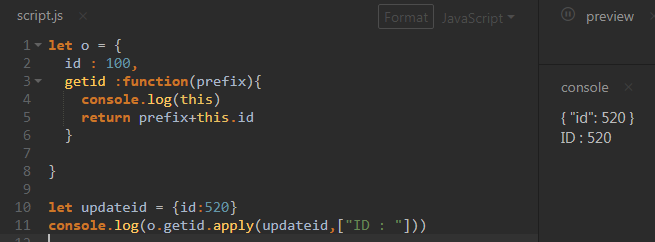
**Call & Apply:**

There are two ways we can call a function in JS. These call a function and change the “this” value.

Call:



Apply:

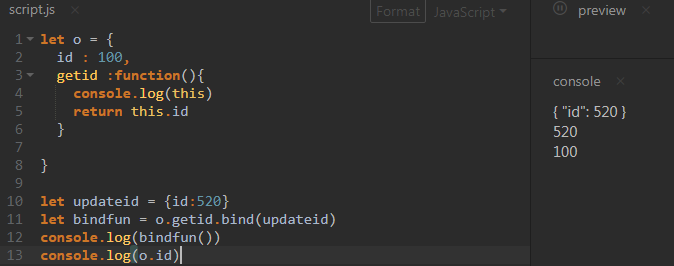


The only difference between Call & Apply is in Apply we are passing the argument to getid function.

**The argument always be an array.**

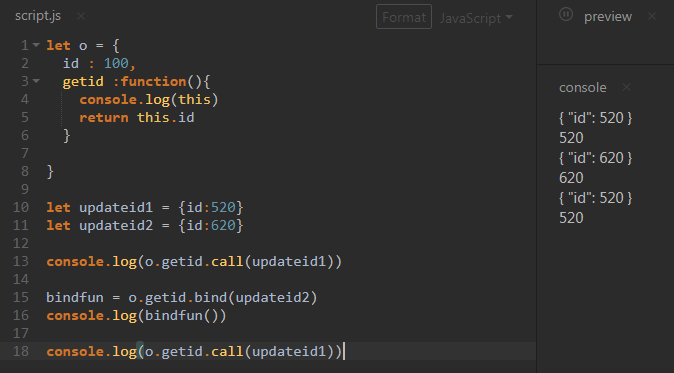
**Bind:**

Call & Apply made the changes in original functions but Bind make a copy of the function and then apply the change on copy function.

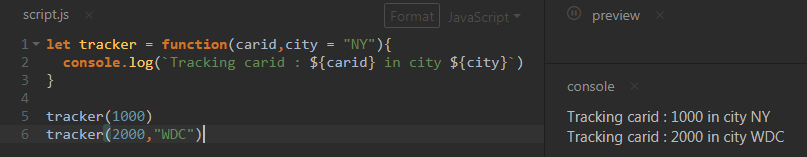


Testing with call and bind.

Call changed the value to 520 and bind change the value to 620 in copy hence while we run again call still we can see the old value as 520



**Default Parameter**

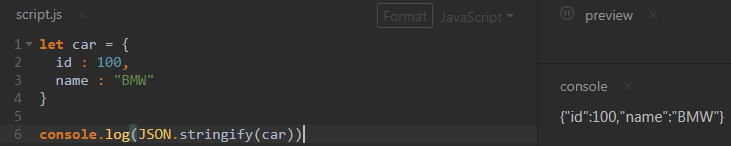


If we are passing the value for fixed argument then the value will be override.

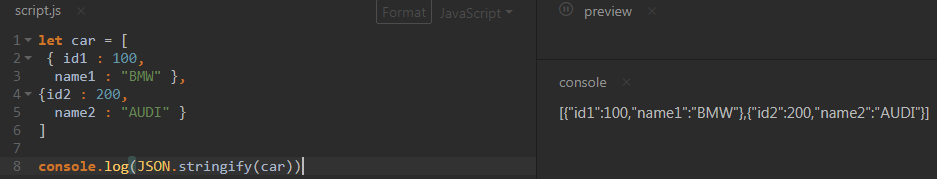
(Added in ECMA 2015 only)

**JSON:-**

To transmit js object over the wire.



Convert Array to JSON



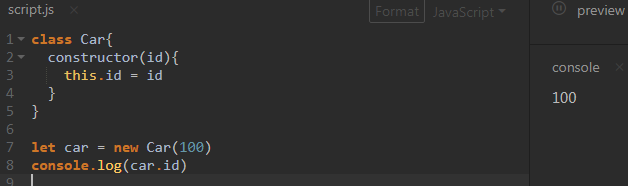
**Array Iteration:**



The last example “find” :- code will run until it finds the id > 100,once it found ,the script will

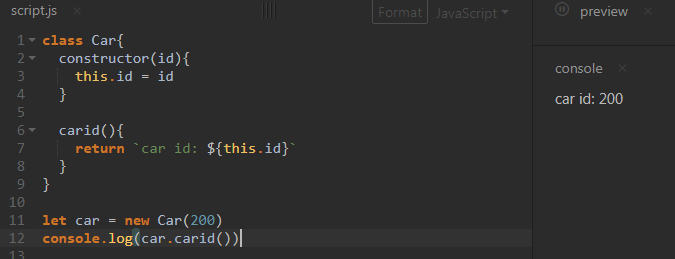
**Class & Modules:-**

Constructor is a function that get executed when an instance of an object got created.



We have to always use **this** inside class.

**Methods:-**



Methods are the functions inside the class. Here we don’t need to use the keyword **“function”**

**Inheritance:-**



Parent class is not affected by the child class value change.

**Programming the BOM & DOM:-**

These are vast topics.

BOM :- Browser Object Module :- That allows to access the browser functionalities

DOM :- Document Object Module:- Change the actual webpage

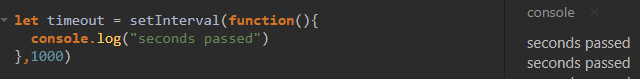
**The Window Object:**

Window is the global object in JS.we can access it from anywhere.



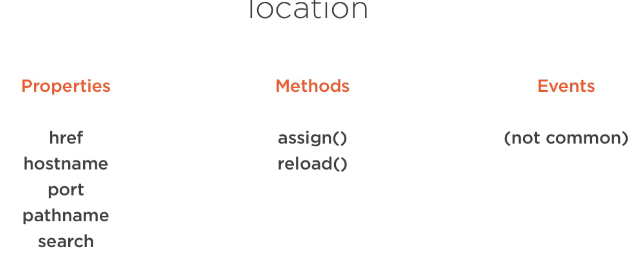
Timers:-

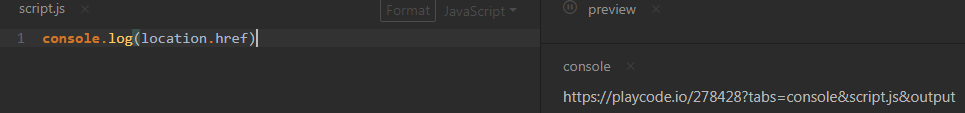
They fire asynchronously. Used in animation, videogames, handle things with user interface etc



It will keep on printing every one seconds.

**The Location Object (Its part of BOM)**





Href is showing the current web address

**The Document Object**



**Mozilla developer network for DOM and BOM details**

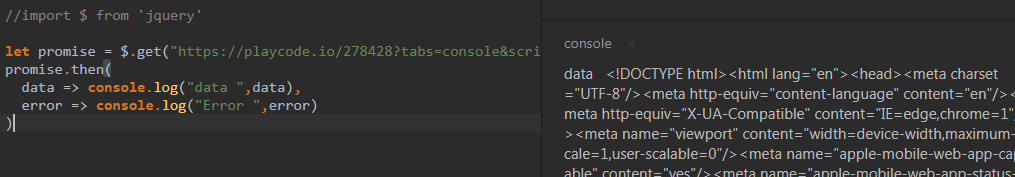
**Data Access using HTTP**

**XHR :- Xml Http Request**

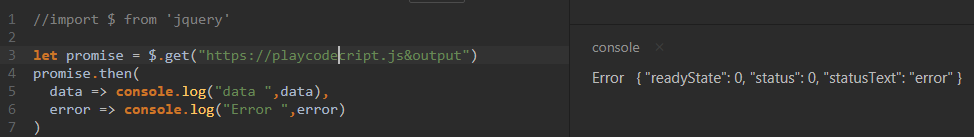
Its no longer in use after arrival of Jquery.XHR is difficult .You have to define both readyState and statusText for get call .

**JQUERY:-**

First we need to install the Jquery module(Here we are not importing as the online editor is already have it.)



When we have wrong api



Promise function is handling both success and error responses.

**FORMS:-**

When an user fill the form and submit ,it first goes to a jvascript which validate that form information before sendg it to server.The submit event send to JS.

This is happened due to DOM

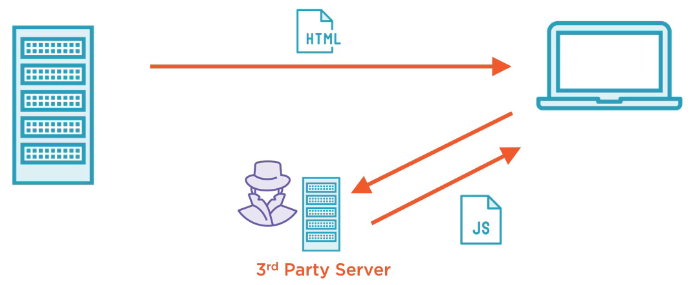
**Security & Building for Production:-**

Once our application build is finish we have to secure it and we also need to create a build pack for production deployment.

**Chrome developer tools:-**

1. Don’t store passwords, secrets or sensitive information in code
2. Don’t use global variables as they are visible in browser
3. Hackers can read your JS code and data send to your browser

**Cross Site Scripting Attack (XSS)**



Sometimes we need our JS files from 3rd party server and there is high chances Hackers can change those scripts.

Adressing XSS:-

CSP :- Content Security Policy(Use HTTP Header)

CORS :- Cross Origin Resource Sharing

These setup required server sides.

**-: Going Intermediate Level :-**

**DOM (Document Object Modules)**

DOM allows to change the style and structure of HTML pages.It represent the internal of the page how browser sees it and allow developers to change it using javascript.

P.S.:-If you’d like to have a look at the DOM for a page, open up the developer tools in your browser and look for the “elements” pane. It’s a great insight into how the browser thinks, and in most browsers you can remove and modify elements directly.

Document: - The entire thing. The html file which loaded is called the document.

Object: - Everything in DOM is an object.An html file will have tags, heads etc .al these called objects.

Modules: - JS provided lots of modules to interact with these objects .These are called Document modules.

**Events & Callbacks:-**

Most of the browser programs are event driven. When a page loads or user clicks on the page an event ocure.So the code behind the page wait for the event and when an event occur it **Listens** first and supply a function to respond that event .This process called **callback**.

To do any task with browser page we have to indentify the elements of the page and we can get the element by calling its ID,TAGS,CLASS name or CSS selector.JS provided document modules for these.

**AJAX:-**

To retrieve new content for a page without reloading the page a technique used which was developed by Microsoft for its MS access application called XHR (XML HTTP request) .The webapps who do this called AJAX apps .AJAX stands for Asynchronous Javascript and XML.

Like your Gmail, Facebook you are getting notification and new emails without loading the entire page.

XML HTTP REQUEST :- XHR

How does a XHR looks like

var req = new XMLHttpRequest()

req.onload = function(event){//something}

req.open('get', 'some-file.txt', true)

req.send()

First we are creating a XMLHttpRequest.Then we are creating a callback function which will take the event as its first argument when the event occure.Then we are doing the get which is our event.

The third is a boolean specifying whether the request is asynchronous - here we have it true, so the XMLHttpRequestis fired off and then code execution continues until a response from the server causes the onload callback to be fired.

The asynchronous parameter defaults to false - if it’s false, execution of the code will pause at this line until the data is retrieved and the request is called synchronous. Synchronous XMLHttpRequests are not used often as a request to a server can, potentially, take an eternity. Which is a long time for the browser to be doing nothing.

**SCOPE:-**

Global variables — that is, variables that can be read and modified anywhere in your application — are not good because they can expose security issues and make code much harder to maintain.

Remember that code is read much more than it’s written. When reading code, if you can’t determine where a variable came from and what its potential values are, there’s a problem.

So it’s best to limit the scope of a variable as much as possible, making it visible to as few parts of your code as possible.

**JQUESRY:-**

Jquery is popular DOM library.

Ususally the the DOM modules provided are vary among the browsers so it become messy and complicated sometimes.To easy off this sitiation there are powerful DOM libraries created and Jquesry is one of them.It almost unified the DOM commands to intercat among all the brosers.

Jquery has very distinctive syntax and all based around the $ symbol.

$('.btn').click(function () {

// do something

});