Table of Contents

[1. Java script fundamentals 2](#_Toc68104016)

[1.Data Types 2](#_Toc68104017)

[2. let, const and var 3](#_Toc68104018)

[3. Basic Operators 3](#_Toc68104019)

[4. Strings and Template Literals 4](#_Toc68104020)

[5. Type Conversion and Coercion 4](#_Toc68104021)

[5. Truthy and Falsy Values 4](#_Toc68104022)

[6. Equality Operators: == vs. === 5](#_Toc68104023)

[7. Switch case 6](#_Toc68104024)

[8. History of JAVASCRIPT 6](#_Toc68104025)

[9. Strict Mode 6](#_Toc68104026)

[10. Function declaration vs expressions. 7](#_Toc68104027)

[11. Array 8](#_Toc68104028)

[12. Array Operations 8](#_Toc68104029)

[13 Objects dot vs bracket notation 8](#_Toc68104030)

[2. Development Guide 9](#_Toc68104031)

[1 Software lists and setup 9](#_Toc68104032)

[3. Java script in browsers 10](#_Toc68104033)

[1. DOM 10](#_Toc68104034)

[2. Guess My Number! Game 11](#_Toc68104035)

[3. Open Overlay Modal (Manipulate CSS class from js) 11](#_Toc68104036)

[4. Capturing Keyboard events 12](#_Toc68104037)

[4. Pig Game 12](#_Toc68104038)

[4. How Java script works internally 13](#_Toc68104039)

[1. JavaScript re-defined 13](#_Toc68104040)

[2. JavaScript Engine 14](#_Toc68104041)

[3. Just-In Time Compilation of java script engine. 15](#_Toc68104042)

[4. JavaScript Runtime 16](#_Toc68104043)

[5. Execution context and call stack. 17](#_Toc68104044)

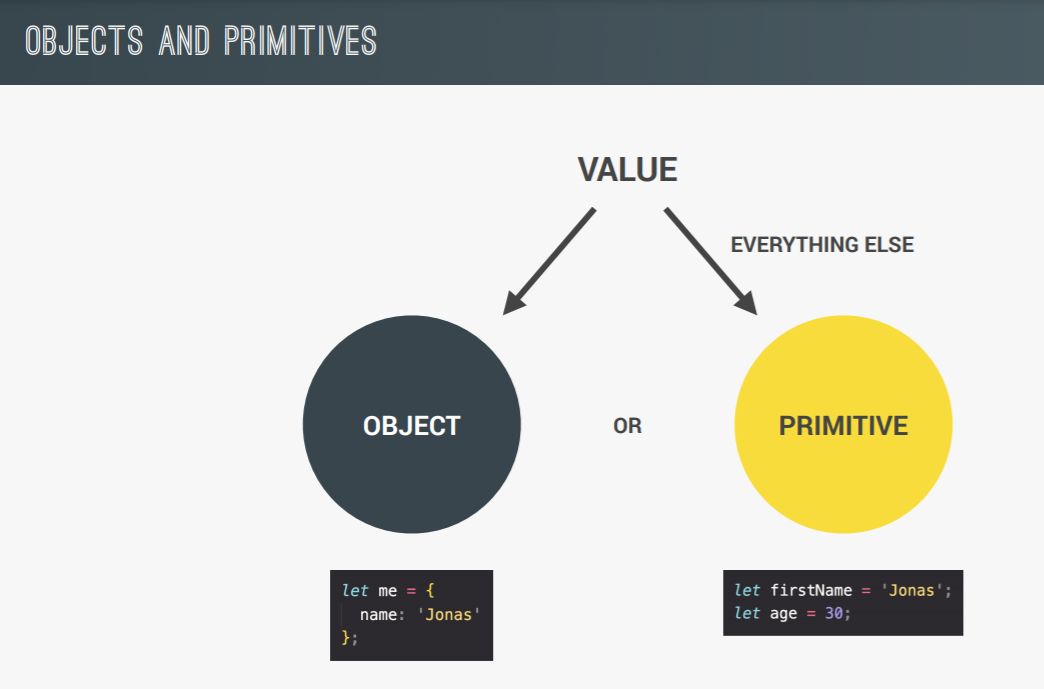
# Java script fundamentals

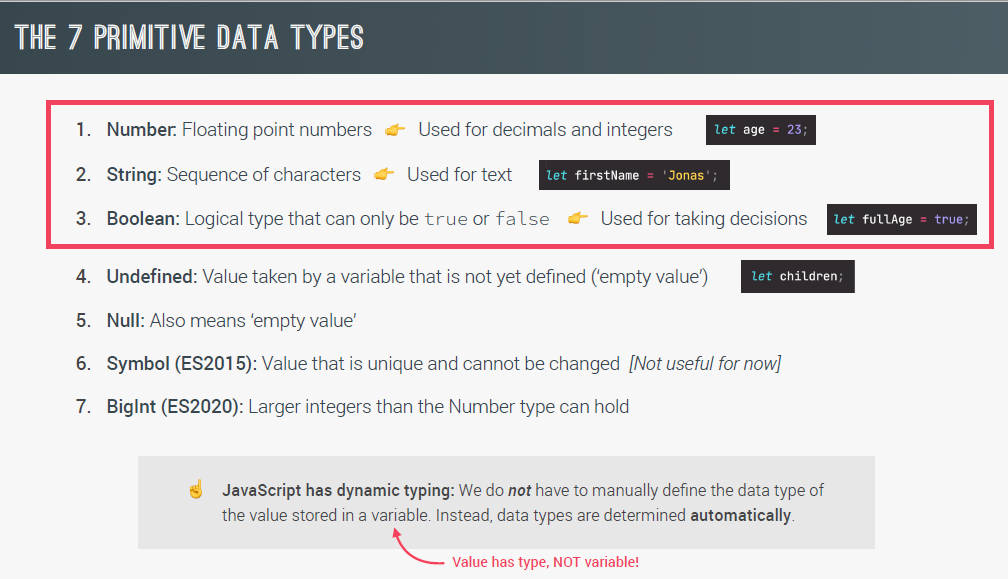
1. Java script is a high level, object oriented, multi-paradigm programming language.



1. JavaScript rule and convention for variable names is same as java.

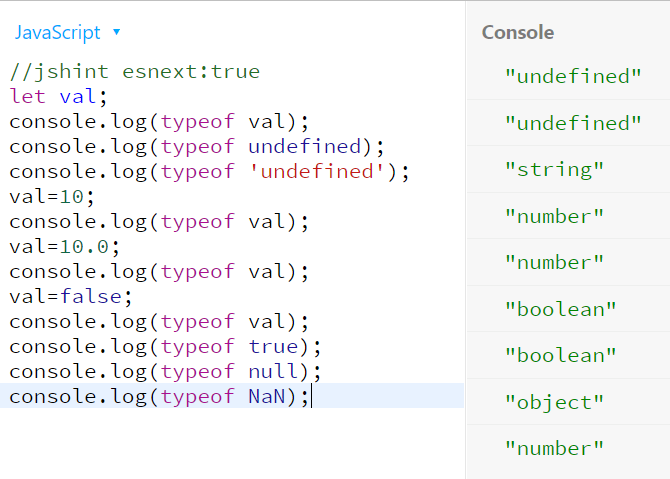
## 1.Data Types





1. In java script all the numeric are floating points. i.e. if(10===10.0) is true.
2. typeof will give the type of value particular variable hold.

Let val;



## 2. let, const and var

1. let and const are introduced in es6.



2. In javascript we can also create variable like this ->

newVar = 10;

console.log(newVar);

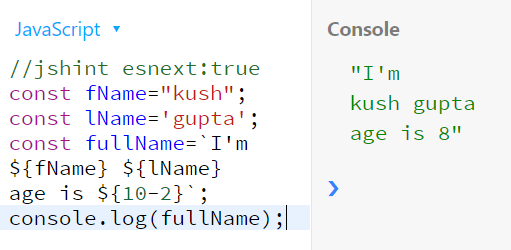
// it will create newVar variable and attach it to global object. Where as var is attached to functions scope only.

## 3. Basic Operators

1. 2\*3= 6, 2\*\*3=8 (it is just like pow), 10-8=2

2. plus operator can also be used to concat string.

## 4. Strings and Template Literals



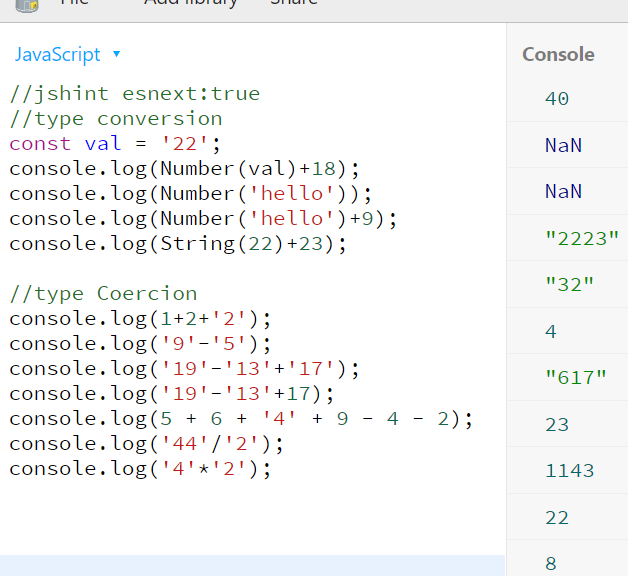
1. Template literals are introduced in es6 and provide below extra features –

1.1 can use multiline strings without need of writing \n\ for newline.

1.2 can use placeholders inside string ${} for actual value and can calculate expression also.

1.3 can use ‘ and “ also inside string without escape character.

## 5. Type Conversion and Coercion



1. type conversion is a way by which we can convert from one data type to other.

2. type coercion is way by which java script convert implicitly to some data type according to below rules ->

2.1 ‘+’ operator will perform mathematically add operation till numbers going from left to right. once string is found, number will be converted to string implicitly and concatenated.

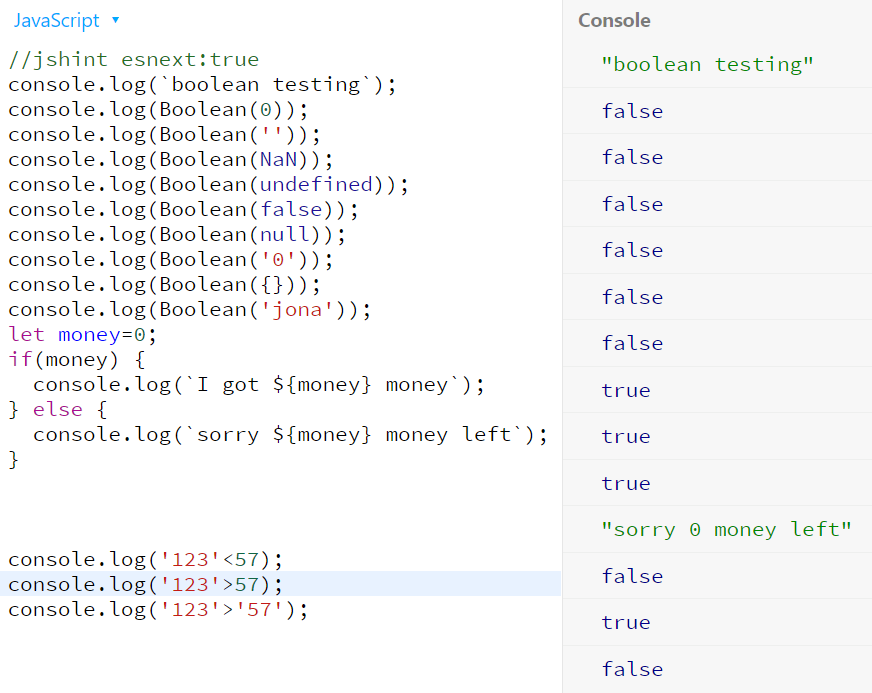
2.2 for ‘-‘ operator it will subtract normally. But for string like ‘22’-‘1’ it will convert string to number implicitly and give result in number i.e. 21.

2.3 same like minus operator, /,\* also works.

## 5. Truthy and Falsy Values

1. By default 0,’’,null,NaN,undefined and false are considered false values when used inside if block or logical operators.

2. if(undefined) or if(0) or if(null) will make code inside this if block not to be executed.



## 6. Equality Operators: == vs. ===

1. === check is strict checking which check data type and value both. Where as == is without strict check. == perform type coercion when needed to perform comparison.

2. !== and != are just not equal representation of not equals.

Example 🡪

let val='18';

if(val==18) {

console.log('== check');

}

if(val===18) {

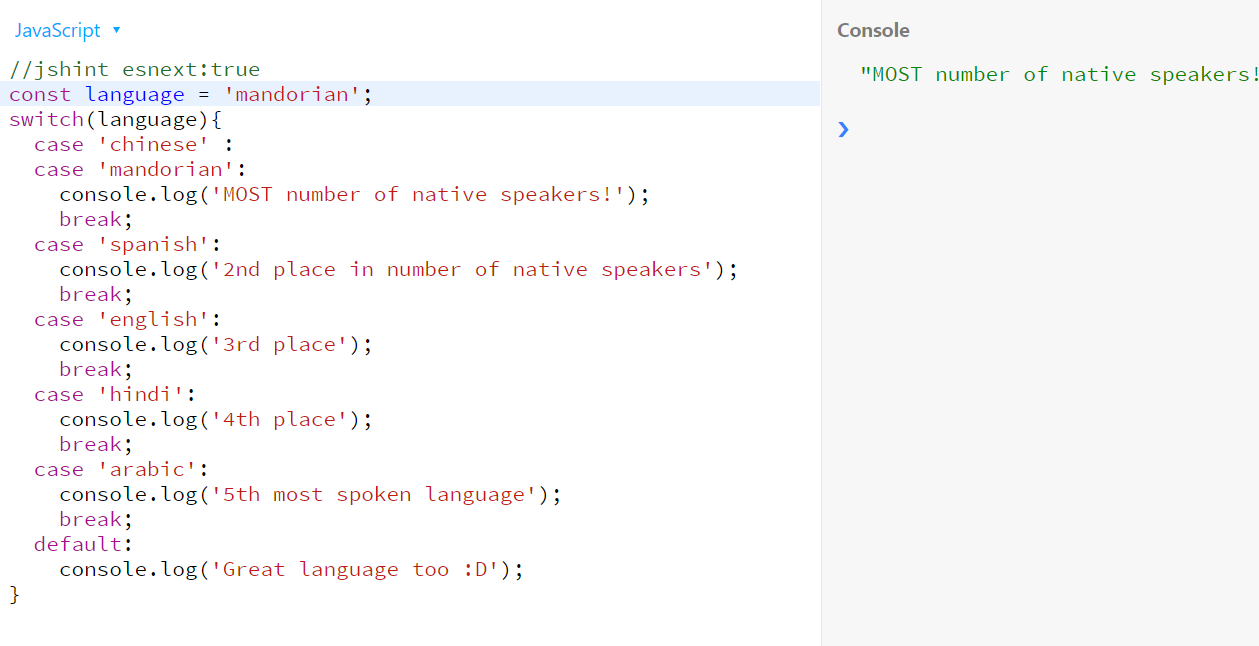
console.log('===check');

}

**Output ->**

== check

## 7. Switch case



## 8. History of JAVASCRIPT



## 9. Strict Mode

1. strict mode can be enabled by adding it at first line of java script file. Once enabled it will protect developer from some mistakes which javascript not complaint earlier.

Example 1->

const firstName = ‘kush’;

let value = 10;

if(firstName===’kush’) {

values=29;

}

Console.log(value); //output 10.

Above code works because js will create new variable values with 29. But with script mode enabled js will give error that values is not defined.

2. strict mode is enabled using below line

‘use strict’;

Example 2->

Strict mode restrict programmers to use reserved words to be used in code as a variable. Like interface, private, if etc. even though some of these reserved words are still not there in js and will be introduced in future.

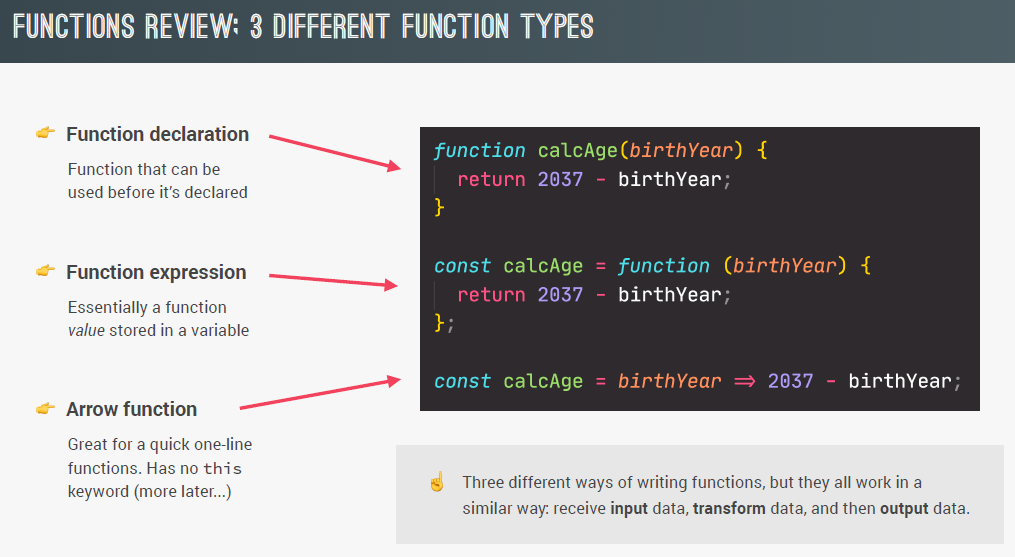
let private =’ss’; //give CT error if strict mode is on

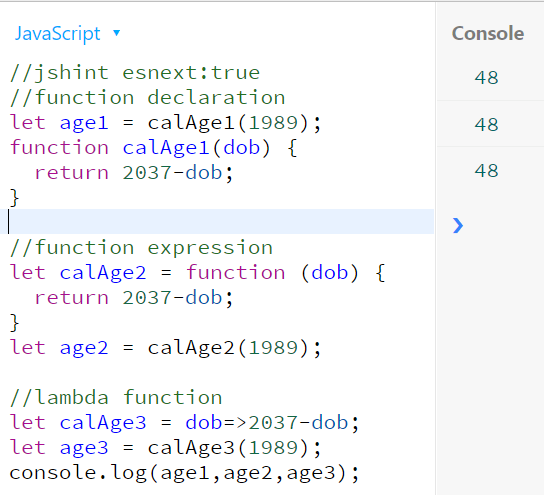
## 10. Function declaration vs expressions.

1. function expressions also called anonymous function and they are assigned to a variable like any other values.

2. anonymous function can be invoked only after it is assigned to a variable otherwise js will give CT error. But normal function can be called in line 1 and in later line will have definition.

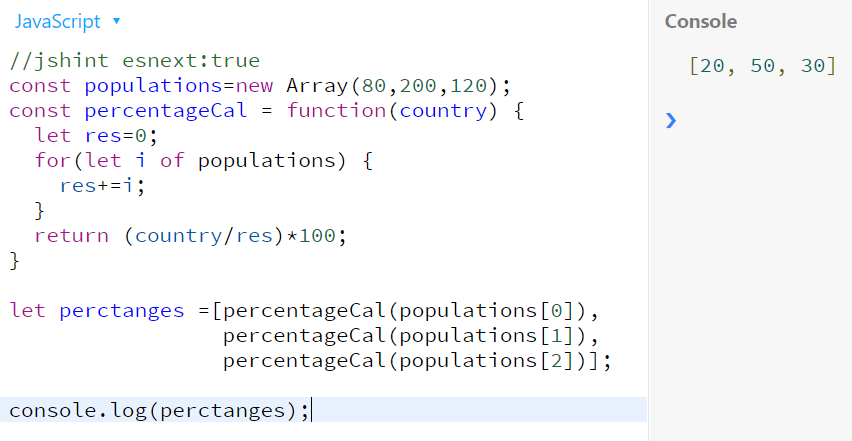
3. from es6 we can write lambda expression to represent function. It is just a simplified way of writing anonymous function. The difference b/w anonymous function and lamba is that lamba does not have this reference.

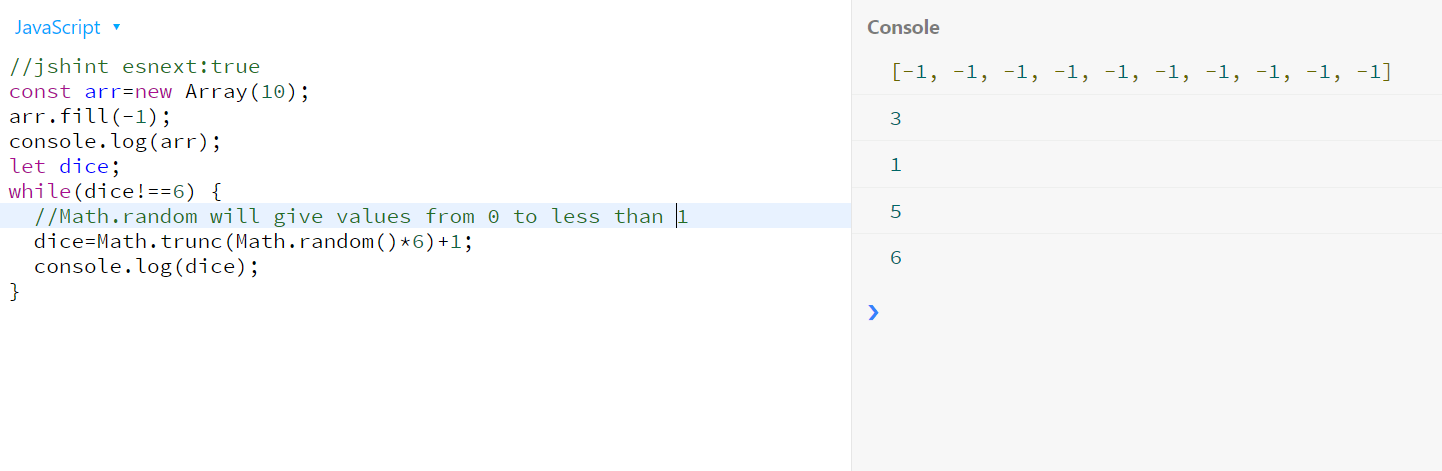




## 11. Array

1.there are two ways to create array. One using new operator and other using [] bracket. Always prefer [] bracket way as it will lead less confusion. As new Array(10,20) will create array with 2 elements. But new Array(10) will create array of size 10 with all 10 undefined values.





## 12. Array Operations

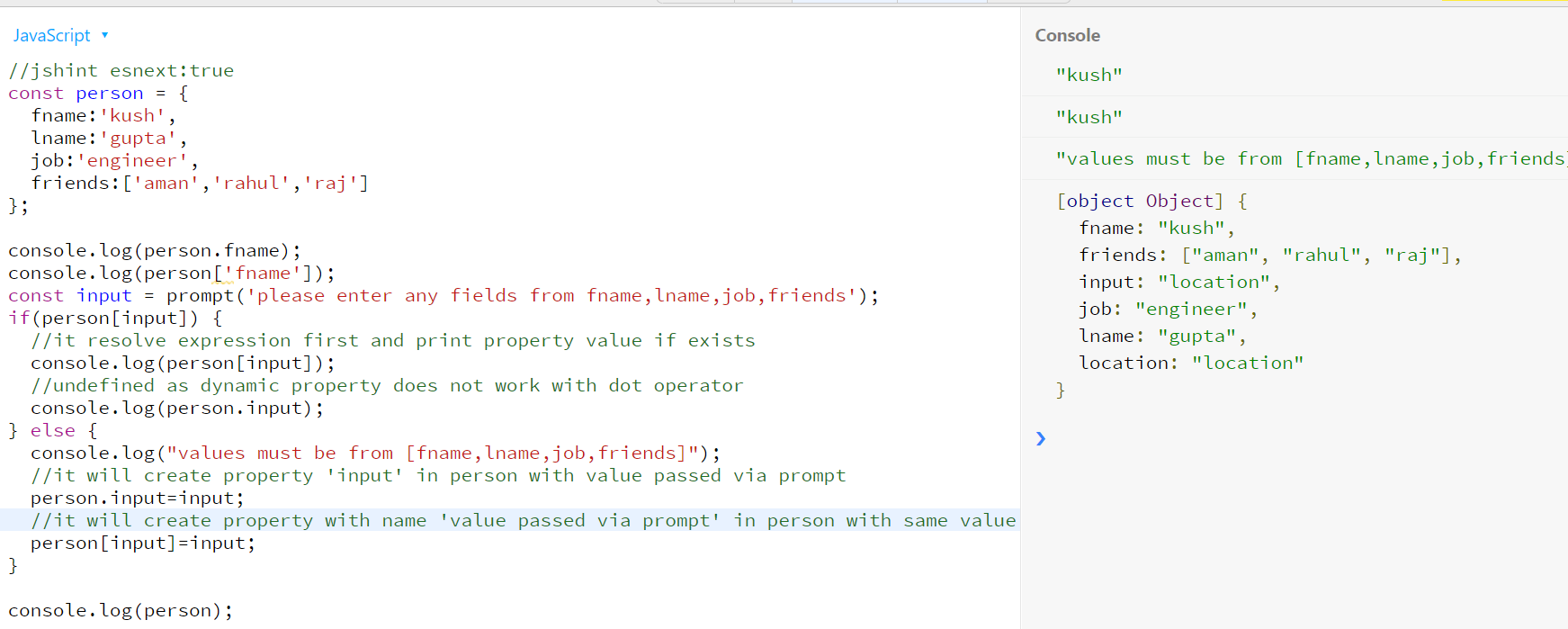


## 13 Objects dot vs bracket notation

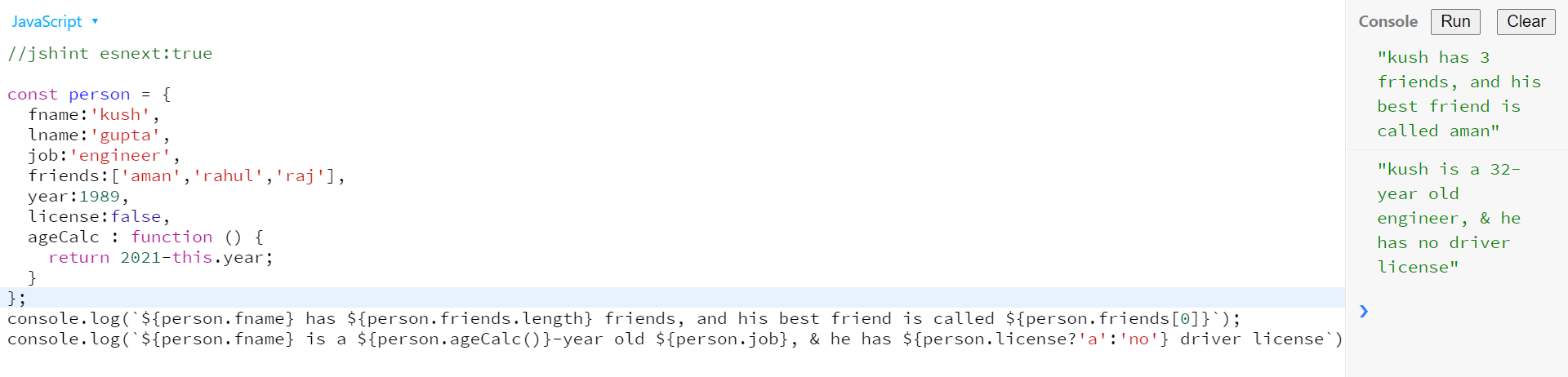
1. in java script objects contains key value pairs where key is always string and value can be anything.

2.every property value can be accessed either by dot or by bracket notation. Only difference is dot operator take static field. If we use field which does not exists both way give undefined value.

In below example input from prompt passed is location.



Example 2 with function as a value of a property.



Example 3 –



# Development Guide

## 1 Software lists and setup

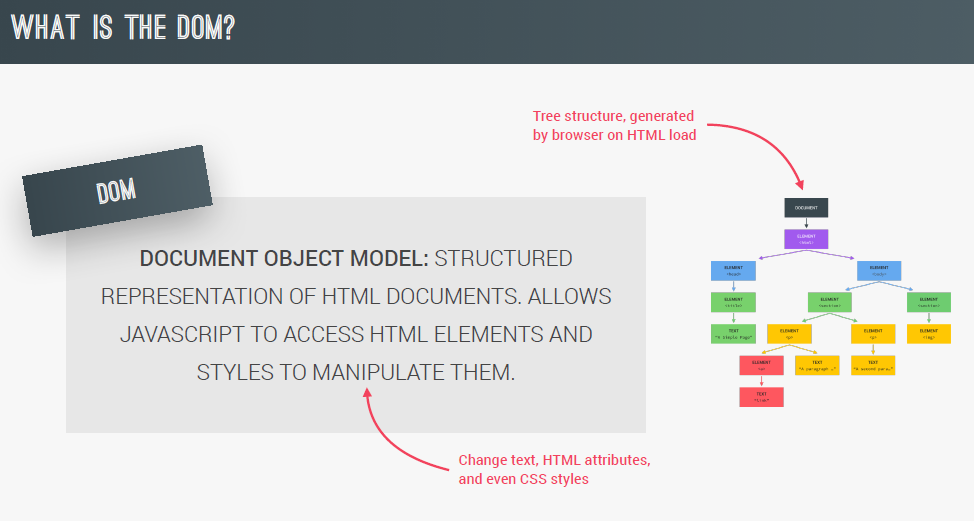
* Download and install vs code and node js.
* Go to terminal and check node -v for version.
* Type – npm install live-server -g
* Now in vs code create any html file and script file.
* Now goto path of above file via terminal and Type –> live-server
* It will launch application in browser with url -> <http://127.0.0.1:58442/>
* Further Changes will be hot deployed.

2 Debug

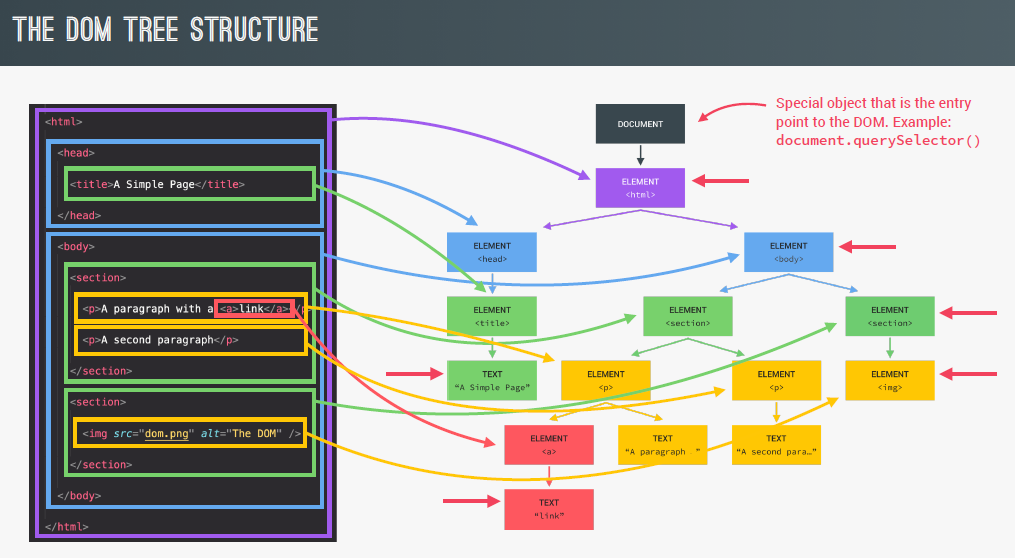
* console.log -> log with normal text
* console.warn -> log with warn text
* console.error -> log with error text
* console.table -> if logged with object. It will display in console in tabular form key,value fields.

# Java script in browsers

## 1. DOM



1. Tree structure root node is document. And by using document we can extract all other child nodes. example by document.querySelector(“#id”);
2. Html is the first child of document. As usually html page starts with html tag. And rest of the tree resembles the content inside the html page.
3. By using dom we can access html content inside js and can modify content according to need.



## 2. Guess My Number! Game

1. Computer will randomly select a Number from 1 to 21 and then ask the user to guess it in maximum 20 attempts.

2. once user enter the number which exceeds the random number, message will be displayed that It’s too high else too low.

3. On tight guess it will current score. And if it is higher then prev max it will be the max score.

4. user can play again by clicking again button.

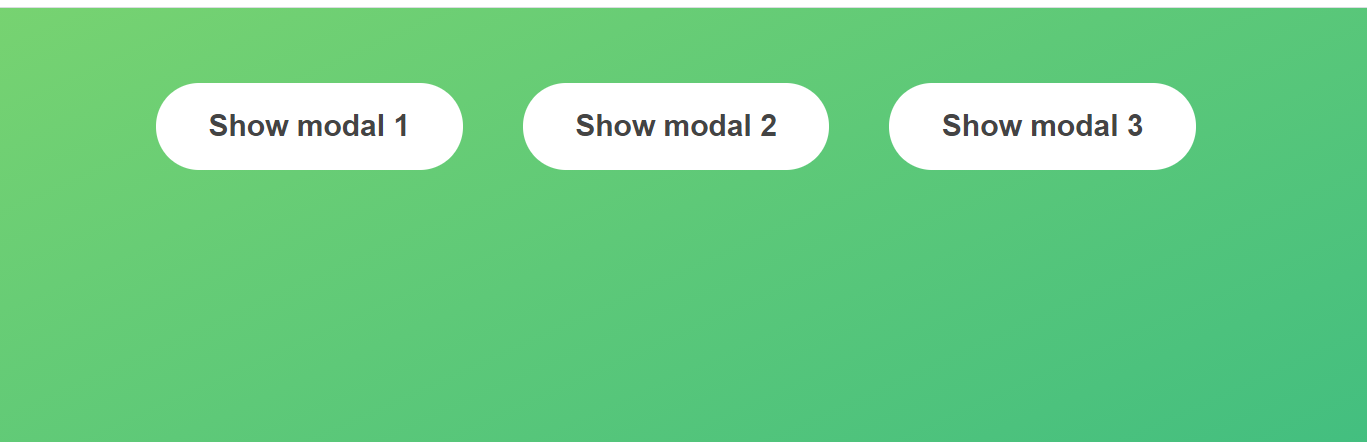
Git utl –

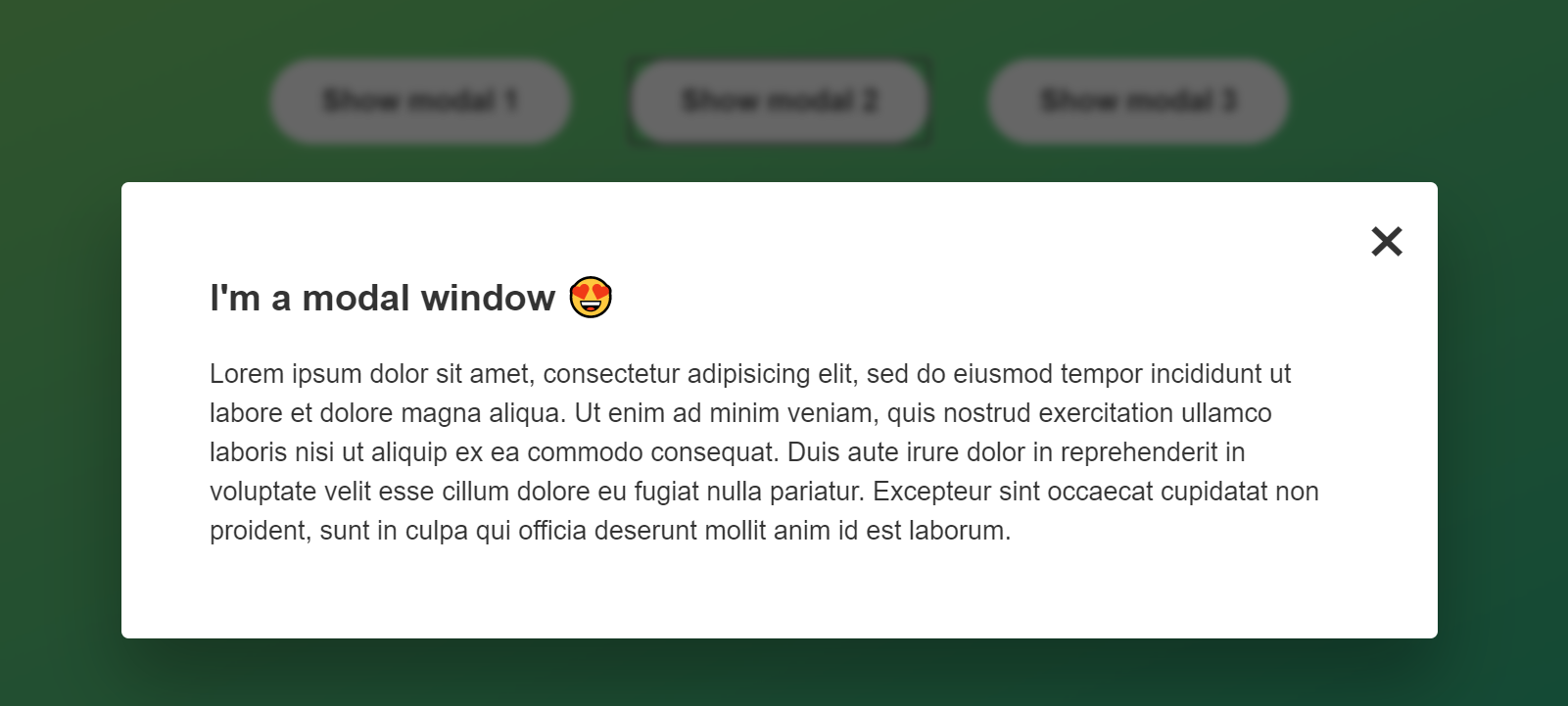
<https://github.com/kushguptacse/JavaScript/tree/main/guessMyNumber>

## 3. Open Overlay Modal (Manipulate CSS class from js)

1. create an application on which on click of button modal overlay open and on close button of overlay it got closed. Also on click on esc button press overlay closed. And on click of outside of popup it get closed.

2. <https://github.com/kushguptacse/JavaScript/tree/main/model>





## 4. Capturing Keyboard events

1. To close the overlay on click of Escape button. We need to attach eventLisetener with keyDown event to document object.

*const* closeModel = ()=> {

    modal.classList.add('hidden');

    overlay.classList.add('hidden');

};

document.addEventListener('keydown',(*event*)=>{

    if(event.key ==='Escape' && !modal.classList.contains('hidden')) {

        closeModel();

    }

});

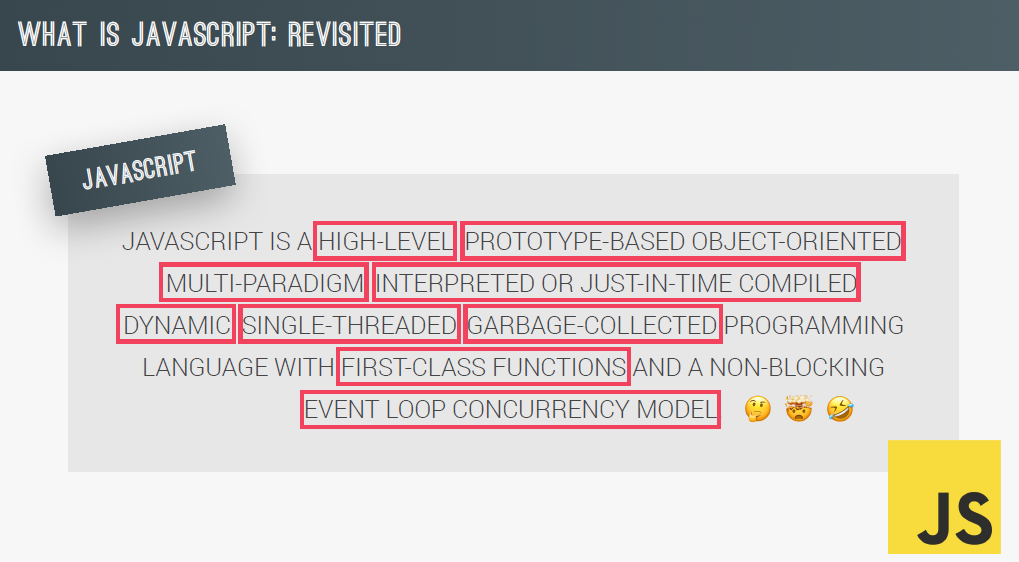
## 4. Pig Game

1. two player game. For detail flow chart can be viewed in java script git repository.

Checkout flow diagram and code from git.

# How Java script works internally

## 1. JavaScript re-defined



1. High Level -> It manages memory management by it owns. Programmer don’t need to allocate or deallocate memory.

2. Garbage-Collected -> automatically collects garbage to clean memory.

3. Interpreted or Just-In Time compilation -> it happens inside javascript engine.

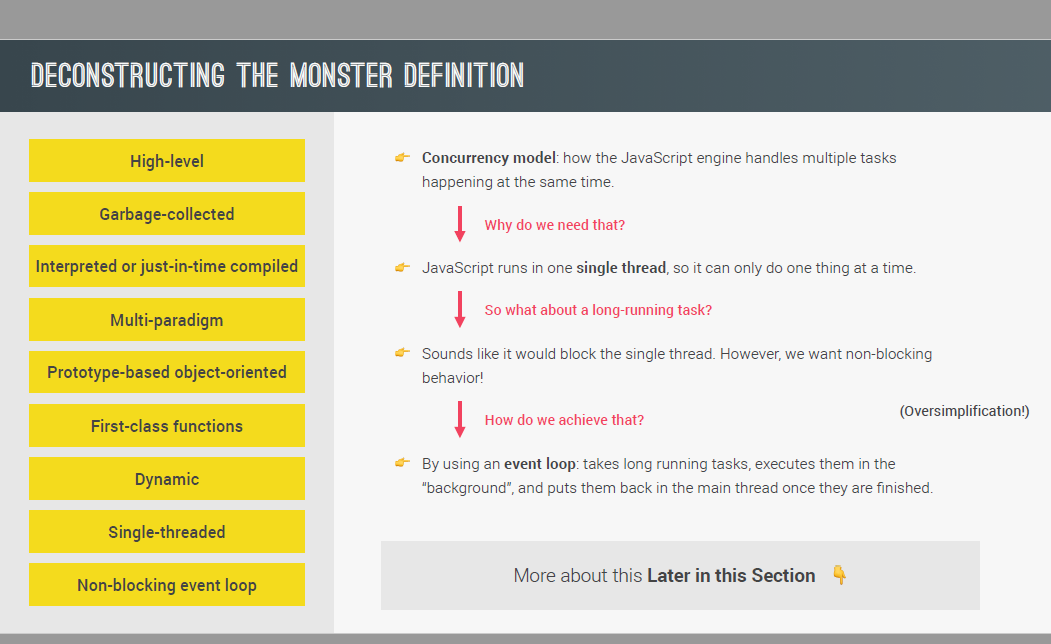
4. Multi-Paradigm -> support different style of programming. Like it supports all -> procedural , functional, object oriented.

5. Prototype-based Object Oriented -> every thing in JS is objects except primitive values. To achive inheritance prototype is used.

6. First-class function -> it means function are treated like normal variables. i.e. can be passed as argument or returned from function.

7. Dynamic language-> variables data types association are dynamic and can also be changed.

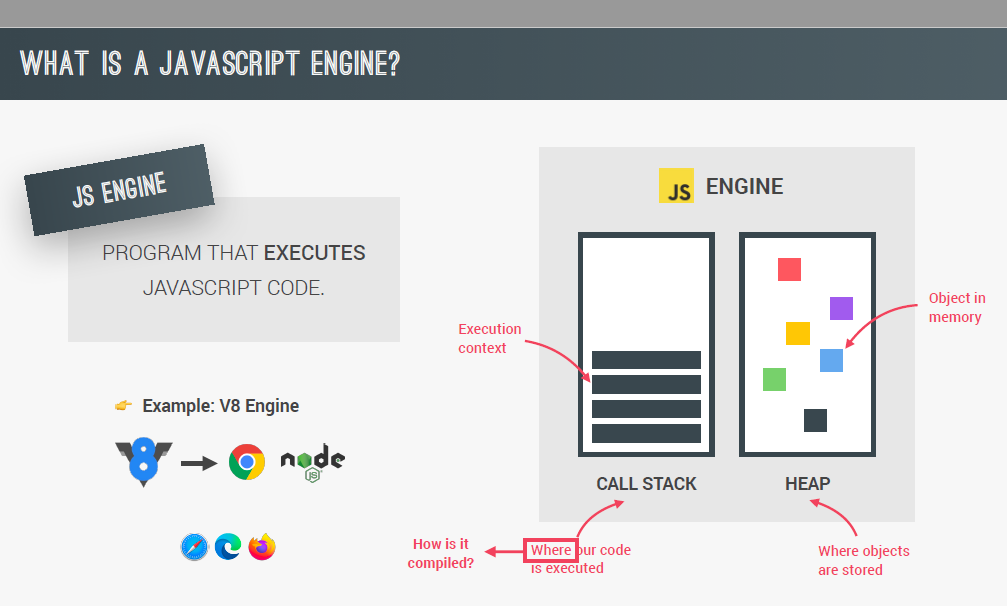
8. single Thread and Non-blocking event loop ->



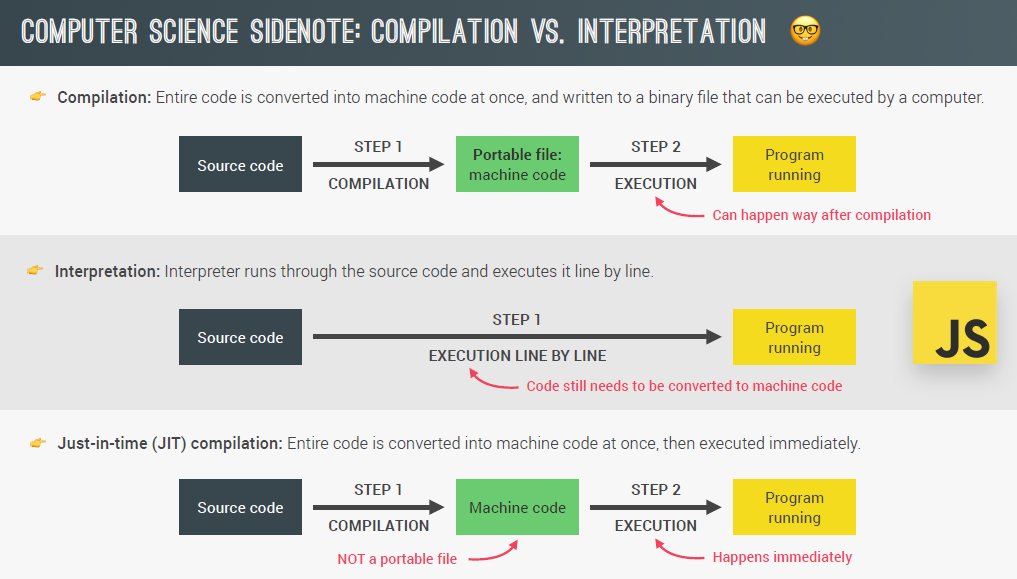
## 2. JavaScript Engine

1. JS Engine is a program that executes java script code. All browser contains this engine. One of the popular js Engine is V8 engine created by google. It is used in chrome and also nodejs. That is same engine can be used to built js app inside or outside the browser.

2. JS Engine contains two components Call Stack and Heap.



3. as we know machine understands only 0/1. So we need to convert javascript code into machine code so that it can be executed by processor. There are three approaches for that – compilation, interpretation, and Just-in time compilation.



4. earlier java script is Interpreted. i.e. source code converted into machine code and then executed directly line by line. Which is slow as the whole process is repeated line by line.

5. now modern java script language uses JIT compilation. i.e. first entire source code is converted into machine code. And then executed immediately. It is different with normal compilation in a way that no portable binary file is generated in jit compilation.

## 3. Just-In Time Compilation of java script engine.

1. first the entire js code is parsed and converted into a data structure called abstract syntax tree (AST).

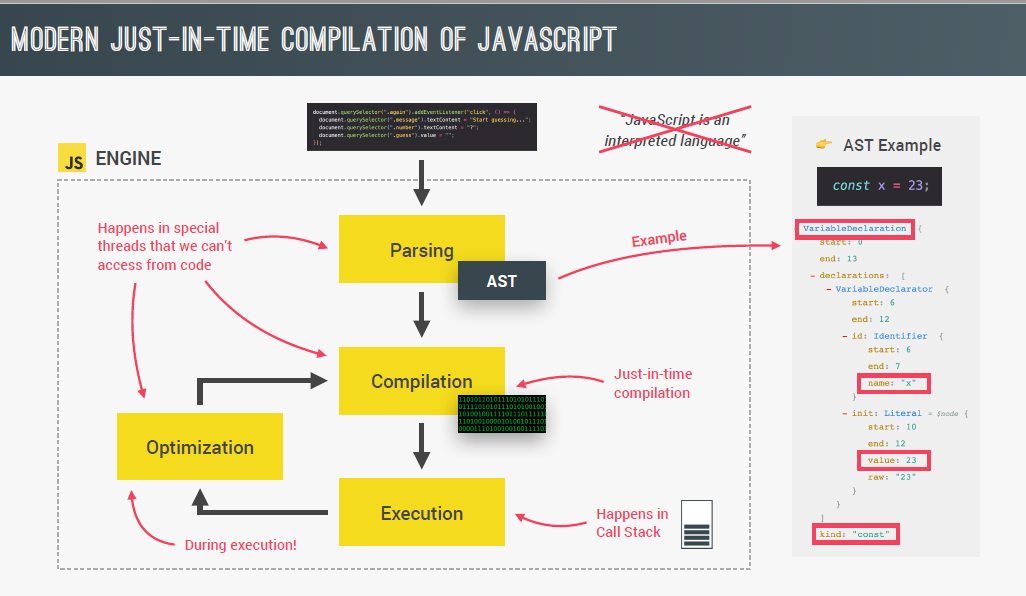
2. in this parsing phase any syntax error is also checked. And this AST is later used to generate machine code.

3. this AST tree has no relation to DOM.

4. then this AST is compiled into machine code. After that code is executed. Execution happens inside call stack.

5. just to make code running fast code the JIT process generates machine code unoptimized without applying too much logic. And once the code is already in running state in call stack, a separate thread will revisit the syntax tree to see for further optimization and then the optimized version of machine code is replaced with un-optimized code.

6. this special thread we cannot access from with-in our code. And it is not part of main thread.



## 4. JavaScript Runtime

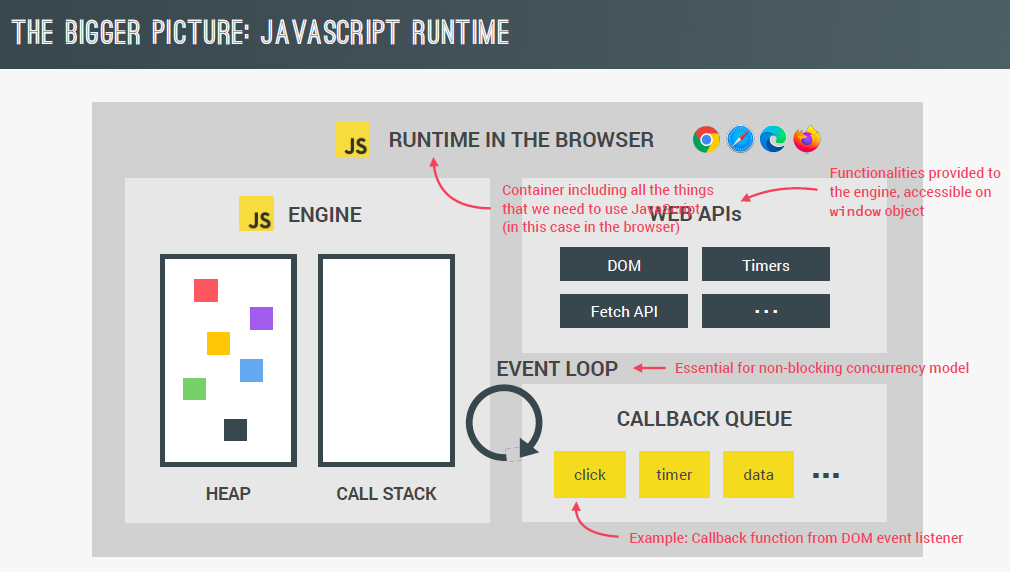
1. Runtime is a container which has everything which is needed by javascript. In most of the cases it is browser. It can also be node.js.

2. Runtime has three components- JS Engine, Web API’s and CallBack Queue.

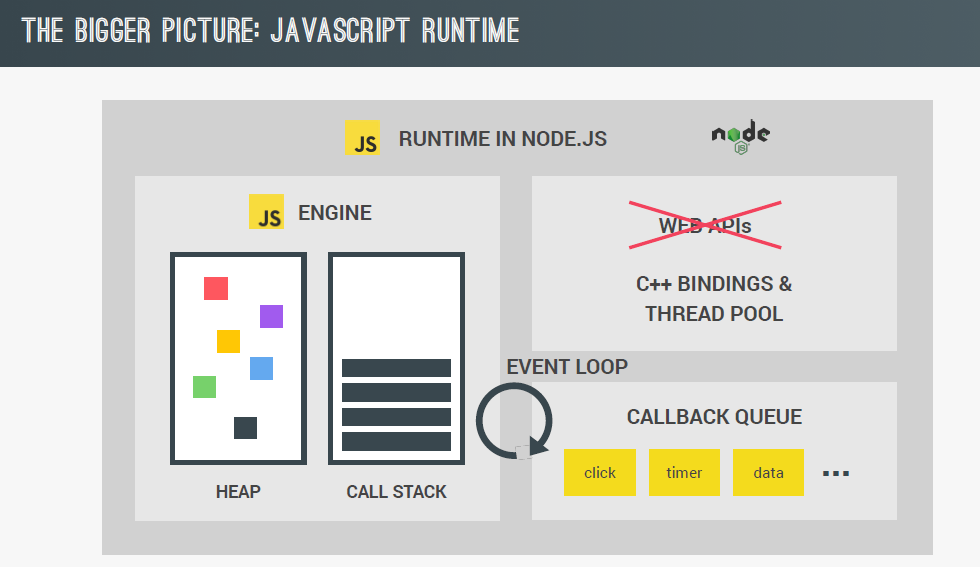
3. Web api are the functionality provided to the engine which can be used inside our javascript code like DOM, Timers, console interaction via window object.

4. all the callbacks are added into callback queue and on execution it is added into call stack by event loop. For example function of click event of button is callback function which is added to callback queue on registration of click event. This function is moved into callstack once button is clicked.

5. with the help of event loop this click event function call is non blocking.



============================JavaScript Runtime in Node.js ================================



In Node.js environment WEB api does not exists as it is provide by browsers. Here instead C++ bindings and Thread Pool is used.

## 5. Execution context and call stack.

1. once the code is compiled into machine code. Execution phase starts.

2. In it first global execution context object is created. Which is responsible for executing code which is outside the function.

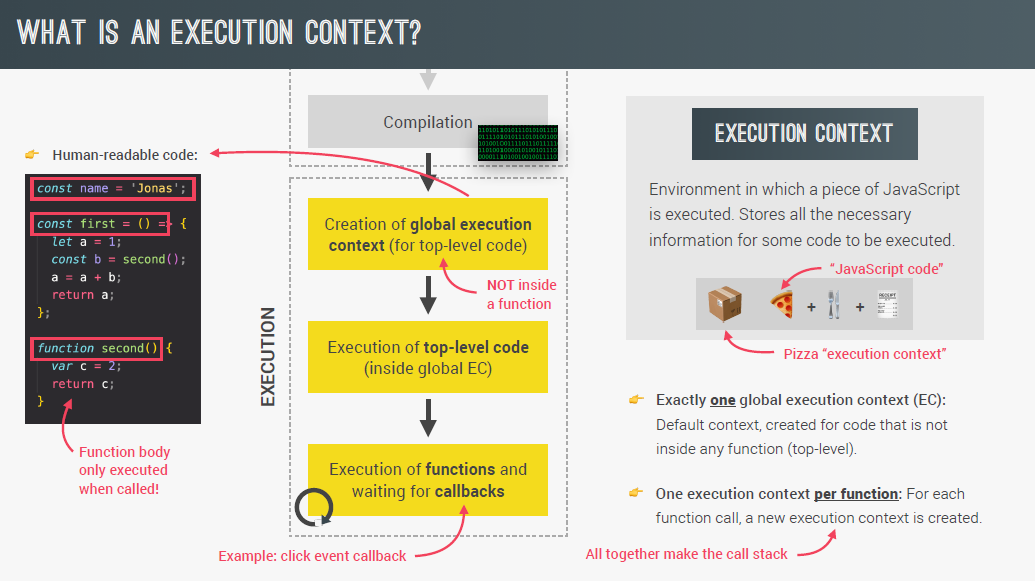
3.Execution Context – is an environment in which a peace of javascript code can be executed.it stores all the necessary information for that peace of code to be executed.

4.for each function a new execution context is created to execute code inside that function. They all together make the call stack.

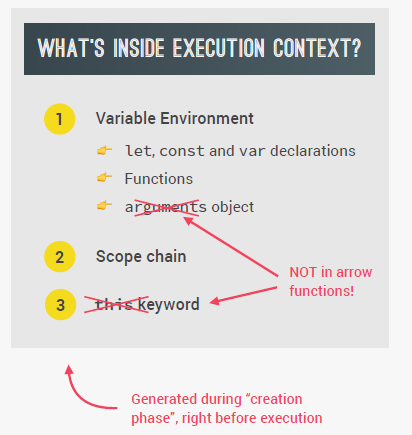
5.Only one global execution context is created. And it will execute top level code.

6. only when the function is called then only it’s specific execution context is created and that code is executed.

7. In Execution CPU basically execute the machine code.



8. Inside execution context below data is present –

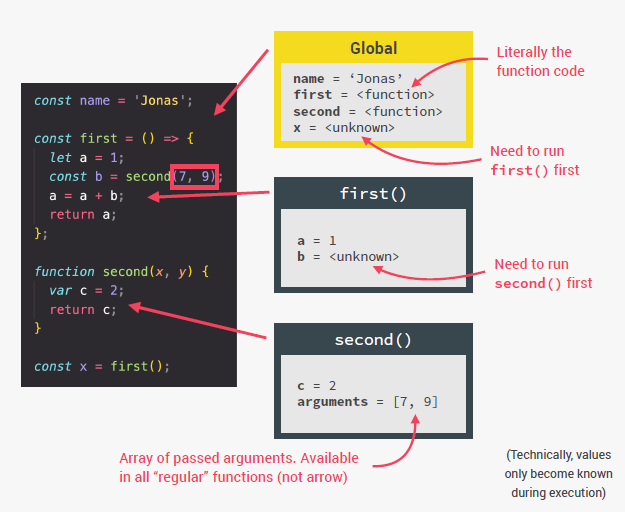


* Variable environment contains all the variable declarations, functions code and argument object contains the list of objects need to be passed into function call.
* Scope chain-> by it reference is made between variables of different scopes to be accessed.
* This keyword.

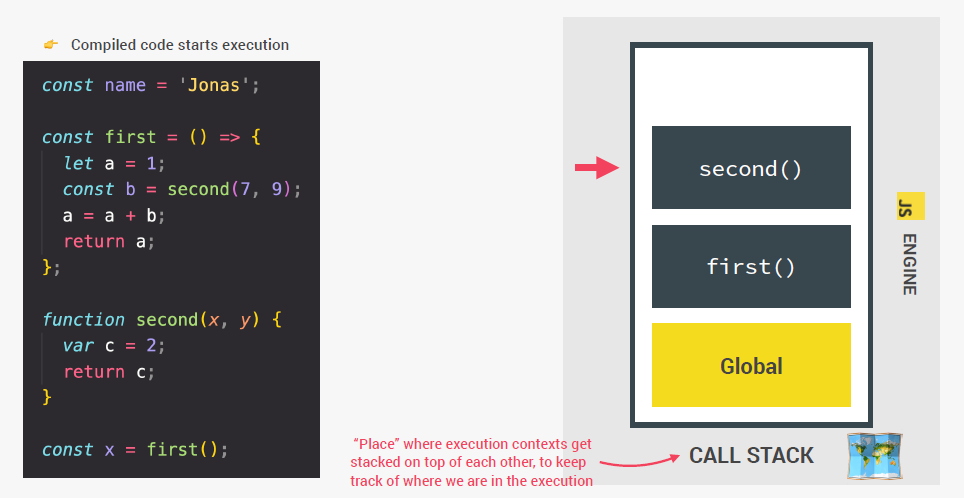
9. In arrow function this and arguments object is not available.

10. All the above information data is prepared and available just before starting execution.

Example 🡪 shows the initial phase of execution context.



11. Flow explanation ->



1. When code starts first global execution context is created and pushed into call stack.

2. Inside it call to first method is invoked. Now first execution context is created and pushed into stack. Now code inside first method executing.

3. Now call to second method is made. Its execution is created and pushed into stack. when second method finishes it is pooped from call stack. And first method remaining lines of code start executing.

4. now when first method finishes. It is popped from call stack.

5. when last line executed. Even after that global EC remains in stack. When the browser is closed then only global EC removed from call stack.