| **Web page** | **Websites** | **Web Applications** |
| --- | --- | --- |
| Single document->HTML  Hyperlinks -> web pages  Access by Url | Group of well structured web page  Same domain  Static website and dynamic website . | Software program ->web browser -> web pages  Programming language + web app frameworks |
| Http request -> url ->server -> web page . | Static website->static web pages ->display as stored in the server.  Http request -> page  Html css javascript | Generates the web pages using the scripts and the programs |
| Document extensions | Dynamic website ->Dynamic web pages->scripting languages & database connection -> add delete restrict remove edit etcb  Html css javascript java    Url - > Http ->DNS -> server ->database .  Date | User engagement ->  End user content  Similar to desktop apps  Activity is done by calling functions |

**HTTP Protocol**

* World wide web communication
* Data exchange - videos,text etc.
* Client ->Http->Web server
* Request(Html page) and response(html file) cycle
* Connectionless protocol
* Stateless protocol - client servers know each other during the connection itself.
* Port no 80
* Request Response cycle

Req.( Http message)

Client(URL) —------> Web server

< —-----

Res.(Http message)

* Http message

Start line , Headers(information and rules) , Body - text or binary

Different for Request and Response

* Request Http message

| Method(what to do) Path/to/file.txt(URI) http/version |
| --- |
| Host : www.website.com  Accept-language:en-us  Accept: text/html (filetype/ext) |
| Some content |

* Response Http message

| http/version Status code(success or fail) 404 200:OK etc |
| --- |
| Host : www.website.com  Accept-language:en-us  Accept: text/html (filetype/ext) |
| File requested |

* Categories of headers :

Authentication ,caching, client hints , conditionals, connection management , controls , cookies , CORS ,

<https://www.geeksforgeeks.org/http-headers/>

* Types of headers

1. General header

1. Date and time

2. Connection of tcp open or close

1. Response header

1. Server : Apache server

2. Accept - ranges - bytes

1. Request header
2. Entity header (relates to the content message)

1. Content type - text/html

2. Content length - 200 bytes

2. Last modified date and time

* Request headers

Accept , Accept-charset,Accept-Encoding,Accept-Language,Authorization,

Cache control , connection , cookie , content length , content type , date host , if- modified-since , if-unmodified-since , i f-Match

if - none match

* Response headers -

Access-control-allow-origin , accept range, age,allow, cache control

,connection, connection , content- Disposition , content-encoding , content-language, content length , content type, date , etag , last modifies , expires , location , retry after , www-auhenticate

**HTTP Methods**

CRUD Operations -

GET - fetch the data , Read

POST - create

PUT - update

DELETE - delete

**Safe method :**

While fetching the date it is not changing on the server

Ex get head

**Usafe method**

Changing the data on the server

Delete put post

**Idempotent**

Multiple operations won't affect the server

Put delete

**GET**

We can attach header properties

If match

If unmodified

If None match

**POST**

Accepting entity in request body

Creates a new resource

**PUT**

Replace resource if exist or create new

**Delete**

Delete resource identified by the URI

**Head**

Requests a header without body , same as GET but without response body

**Options**

No operation

Request communication options

**PATCH**

Same as PUT but overrides

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Task 2**

**Javascript**

* Programming language and scripting language
* Client side and server side
* Loosely typed language
* Dynamic pages
* Interpreted programming language in browser
* Object oriented programming language
* Event responses
* Javascript run environments

1. Javascript engine - Browsers

Google engine v8

Firefox - spidermonkey

1. Node

C++ wrapper

Run javascript without browser anywhere

* Uses of javascript

Dynamic behavior

Game development

Web development

Web application

Web server

Mobile development

Animation

Client side validation

* **Importance**

Implement client side scripts

Write Server-Side Code

Frameworks

Responsive web design

* **Compilation and execution of javascript code**

Execution context of javascript

| **Memory / Variable environment** | **Thread of execution / code** |
| --- | --- |
| Key value pairs  A : 20 | Code is stored here |

Synchronous single threaded language

One command at a time in a sequence

Two phases

1. Memory execution case
2. Code execution phase (runs line by line )

Example program

*Var n= 2 ;*

*Function square(num)*

*{*

*Var ans = num \*num ;*

*Return ans;*

*}*

*Var square2 = square(n);*

*Var square4 = square (4);*

| Memory | Thread of execution |
| --- | --- |
| n : undefined(P1) , 2(P2)  Square : *Var ans = num \*num ;*  *Return ans; —---->*  Square 2 : undefined (P1)  Square4 : undefined(P1) | Function     | Memory | Thread of execution | | --- | --- | | num :  undefined(P1)  2(P2)  num : ans(p1)  4(P2) | <—-num \*num  —--> return ans |   Return transfers the control and the instance is deleted . |

**Call stack**

* Execution context are placed inside the call stack
* Bottom is of Global execution stack
* Functions or new EU are placed over the global execution stack

**JavaScript Engine**

* Javascript can run anywhere
* It just needs an JS run time environment or container
* JRE has Javascript engine , API, Event loop , call back queue , microtask queue
* API’s may have different implementations inside browser and Node.JS

**JS engine architecture**

* Code -> parsing -> compilation -> execution
* Parsing

Code is broken down into tokens

Convert it into Abstract syntax tree (json)

Astexplorer.net

* Compilation and execution phase

JIT(Just in time compilation)

Turbo fan Compiler -> efficiency

Ignition Interpreter -> faster execution

AST -> Interpreter ->high level code -> byte code

Compiler optimizes(inlining,copy elision, inline caching) the code and feeds it to the interpreter.

Byte code -> execution phase

* Call stack , memory heap , Garbage collector(mark and sweep algorithm)

**Hoisting and Scope chaining**

**Hoisting**

* Accessing of variables and functions before the definition of the function or value of the variable
* Functions can be accessed before the declaration
* Arrow functions act as the variables hence they throw error
* Assignments are not hoisted

**Scope chaining**

* Whenever the GEC is created the lexical environment is created
* Lexical environment is local memory + lexical environment of its parent
* Hierarchy
* Gec points to NULL

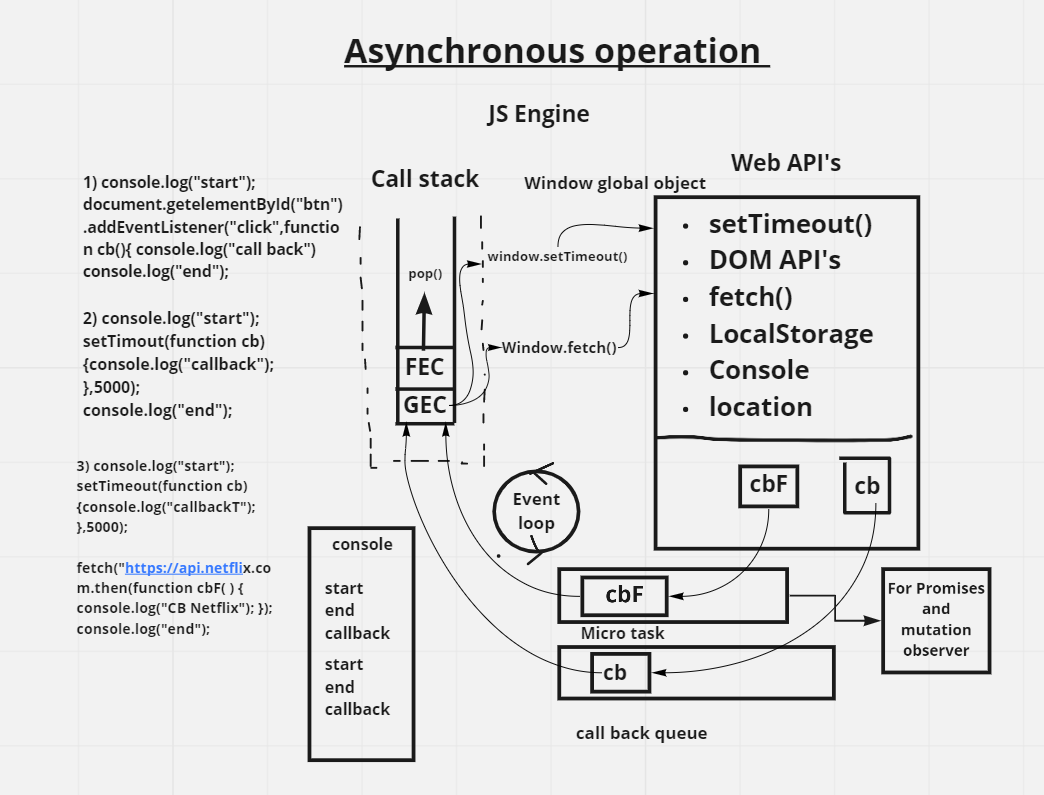
**Let vs var vs const**

Var -> redeclare , value change , If users use the var variable before the declaration, it initializes with the undefined value

Let -> cannot redeclare , modified , block scoped , if used before declaration it gives error

Const -> same as let but cannot modify

**Asynchronous operation**

****

**Block scope and shadowing**

* Let and const are block scoped
* They cannot be accessed outside the block scope
* Using Var Same variable which is declared outside the block is shadowed by the variable in the block scope because both points to the same memory location
* Also applicable for functions
* Illegal shadowing by let -> var
* Var -> let valid shadowing
* Block scope follows lexical scope

**Closures**

* Function bundled together with its lexical environment is called closure

*function x()*

*{*

*var a=7;*

*function y(){*

*console.log(a);}*

*return y;*

*}*

*var z= x();*

*console.log(z);*

*z();*

***Op****-*

*function y(){*

*console.log(a);}*

*return y;*

*}*

*7*

* The returned function (y) contains all the lexical scope along with the closure .
* Function bundled with its lexical environment is known as a closure. Whenever function is returned, even if its vanished in execution context but still it remembers the reference it was pointing to. Its not just that function alone it returns but the entire closure
* Module design pattern
* Currying
* Functions like once
* Memoize
* Maintaining state in async world
* setTimeous
* Iterators

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures>

**setTimeouts**

* Accepts function reference
* No function()
* Returns id into a variable
* cleartimeout() is to stop the setTimeout execution
* Nested
* Trust issues

**prompt alert confirm**

**Alert** - pop up

Return undefined

Modal window

**Prompt -** accepts a value and returns the value

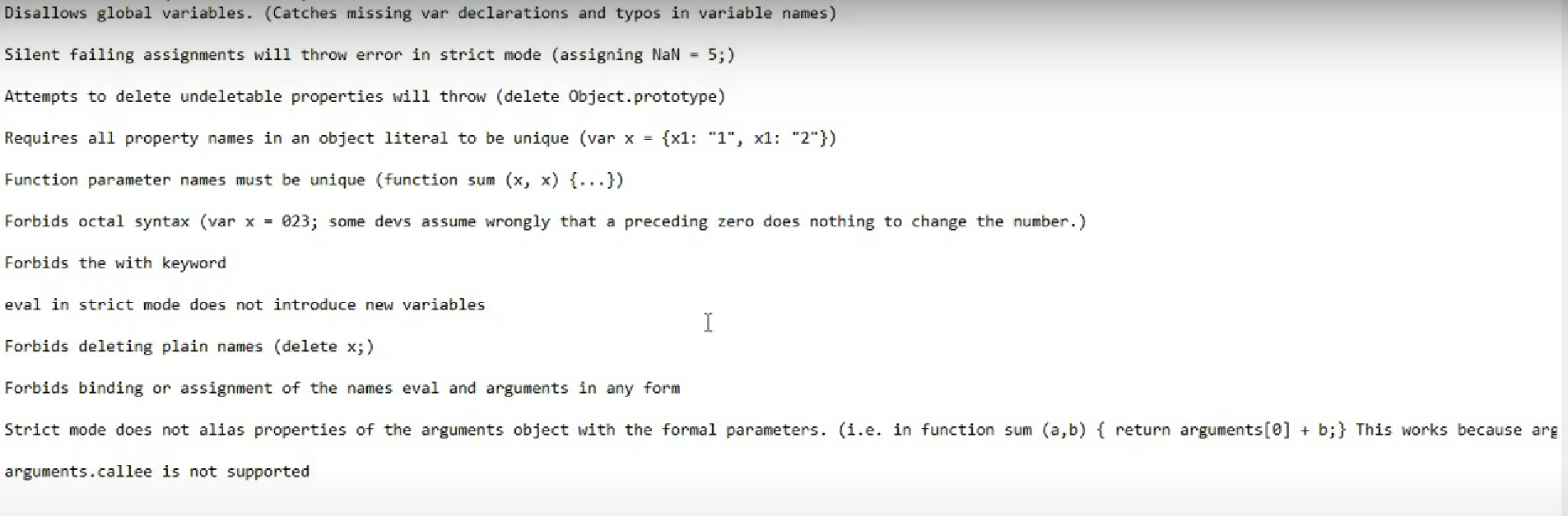
prompt( string , placeholder)

**Confirm** returns boolean values

Warning

**Use strict**

Must declare variable



**Destructuring**

Const details = {age:20,name:keshav}

Const { age,name} = details

Age and name can be accessed using object name itself

`my age is ${age }`

**datatype**

Primitive string , numbers , boolean , undefined , null , undefined, symbol

Reference - objects literals , arrays-object , dates , functions

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Functions**

* Anonymous function - a function without a name they can be used as values

*Var b = function (){*

*console.log(“anonymous”);*

*}*

* Function statement is normally which we define function
* Function can be assigned to a variable called is function expression

*Var b = function y(){*

*console.log(“anonymous”);*

*}*

*y() → Invalid as y is not in the outer scope its a local expression*

**First class functions:-**

The ability to use functions as values , passing the function as argument and return a function in a function is first class functions ,

*A = function() {*

*Return function()*

*{*

*console.log(“hello world”);*

*}*

*}*

*A();*

**Call back functions**

Functions which are awaited to be executed

Never block the main thread use async operations

*Function attach(){*

*Let count =0;*

*document.getElementById(“click me”)*

*.addEventlistener(“click”,function(){ console.log(“clicked”,++count);}*

*attach();*

***Arrow functions***

Shorter syntax

*Let Greet = () => {*

*console.log(“hello”)*

*}*

*setTimeout(()=>{ console.log(“this is call back”},4000);*

*Let sum = (a,b) => {return a+b}; //flower braces must for return*

*sum(2,4)*

*Let half = a => a/2;*

*Lexical this -> arrow function uses this to refer the parent objects not its own unlike the normal functions*

*Greeting : "good"; X*

*let obj1 = {*

*Greeting : "good" , X,*

*names : ["harry","rohan","ramwsj ","google"],*

*speak(){*

*this.names.forEach((student)=>{*

*Greeting : "good"; X*

*console.log(this.Greeting+"kk"+student);*

*});*

*}*

*}*

*obj1.speak();*

***Data types***

*Primitive - string , numbers , boolean , null, undefined , symbol*

*Reference - arrays , object literals ,functions ,dates*

*typeof*

*Object wrappers - str.toUpperCase()*

***Type conversion and coercion***

*Myvar = String(34); // converted to string*

*myvar.toString()*

*Myvar = parseInt(“22”);*

*Myvar = parseFloat(“22.45”)*

*Myvar = (“22.45”)*

*Myvar = Number(“22dd”);//NaN not a number*

*Myvar = 223.33445454545*

*Myvar.toFixed(3)// no of decimal places*

*The Boolean value of undefined is false. The value of Not only undefined but also null, false, NaN, empty string is also false*

*Arrays properties and methods*

*Length*

*constructor*

*Find , filter , map , reduce , filter , join arrays properties , utility functions, object properties and methods*