**Important Points**

**Script Async, defer:-**

Timeline

Description automatically generated **async:- here also html parsing not stop. js file download it execute immediately.**

**Defer:- no need to stop html parsing. Loaded complete in browser html then js file will execute.**

**stopPropagation**:- prevents further propagation of the current event in the capturing and bubbling phases.

**preventDefault**:- prevents the default action the browser makes on that event.

**Fetch :-**

fetch('/article/fetch/post/user', {

method: 'POST',

headers: {

'Content-Type': 'application/json;charset=utf-8'

},

body: JSON.stringify(user)

}).then(response => response.json())

.then(result => alert(JSON.stringify(result, null, 2)))

1. What is Ecma script(ES6)  
   A) let and const key words., Arrow functions, multi-line string, Default Parameters, Template Literals, Destructing, Promises, classes, Modules
2. Data types
3. Data type methods.
4. Array map, foreach, .  
    a. map:- will create new array of values, and also perform actions
   1. forEach:- it will operate function it not retun any array.
   2. Filter:- based on conduction check and result as new array.
   3. Find:- it checks array if conduction is true are not and return value. if it has duplicate values it ignore remaining it return first match only.
   4. Findindex:- it retuns value index, not find it return -1. it ignore remaining it return first match only
5. Diff b/w normal and arrow functions.  
    arrow function are best for callbacks or methods.
   1. Don't work with new keyword and cannot be used when creating prototype
   2. Fixed this bound to scope at initialisation

https://www.geeksforgeeks.org/when-should-one-use-arrow-functions-in-es6/  
<https://www.geeksforgeeks.org/difference-between-regular-functions-and-arrow-functions/?ref=rp>

Regular function created using function declartion or expressions are constructible and callable.

since regular functions are constructible they can call by using new key word.

however the arrow function are only callable and not constructible, arrow function can neaver be as costructor function.

**this** Binding: Arrow functions do not bind their own **this** value. Instead, they inherit **this** from the enclosing context (lexical scoping).

function Person(name) {

this.name = name;

}

const john = new Person('John');

const printArguments = () => {

console.log(arguments); // Error: arguments is not defined

};

printArguments(1, 2, 3);

1. Rest and spread operator.  
     
   The rest operator (…) allows us to call a function with any number of arguments and then access those excess arguments as an array. The rest operator also allows us in destructuring array or objects.

The spread operator (…) allows us to expand an iterable like array into its individual elements.

console.log([...[1,2,3],...[4,5]]);

1. Event bubling.

* **stopPropagation**:- prevents further propagation of the current event in the capturing and bubbling phases.
* **preventDefault**:- prevents the default action the browser makes on that event.

1. Call backs call back hell.
2. Promises

var a = new Promise((res,rej)=>{

return res(0);

})

a.then((res)=>{

console.log('$$$',res);

})

1. Call arguments, apply[] , bind mehod

In JavaScript, the **call**, **apply**, and **bind** methods are used to manipulate the **this** value and pass arguments to functions. They are commonly used for function invocation and controlling the context in which a function is executed. Let's explore each method:

Call:- we need to send number of arguments.

Apply:- we need to send array of arguments.  
Bind: which will returns method.

1. Diff b/w Json and object (in JSON all keys must be quoted, while in object literals this is not necessary)  
    json is name/value pair.

data = {version:1.0,stage:'test1'};

JSON.stringify(data);

// {'version':1.0,'stage':'test1'}

1. Hosting

at time of hosting memory creation. it declare varable with 'a' at top of the function (not take the assignment it will take only declartion)

Var is functional scope.

Const and Let are block scope.

JavaScript hoisting occurs during the creation phase of the execution context that moves the variable and function declarations to the top of the script.

The JavaScript engine hoists the variables declared using the let keyword, but it doesn’t initialize them as the variables declared with the var keyword.

The JavaScript engine doesn’t hoist the function expressions and arrow functions.

1. Shallow copy and deap copy and spreed operater is deep copy or shallow copy.  
     
   A deep copy means that all of the values of the new variable are copied and disconnected from the original variable.;  
    Ex:- JSON.parse(JSON.stringify(employee)) ;
2. A shallow copy means that certain (sub-)values are still connected to the original variable.  
    Ex:- spreed and normal assignment value. (Object.assign and spread operaer it does only shallow copy.)
3. Object freaze and seal

**Object.seal()** allows changes to the existing properties of an object whereas

**Object.freeze()** does not allow so.

**Object.freeze()** makes an object immune to everything even little changes cannot be made.

**Object.seal()** prevents from deletion of existing properties but cannot prevent them from external changes.

1. How to prevent hosting,

Which throws error . if we use var it not throw error in function variable it show ‘undefined’.  
let animal = "lion";

var favoriteAnimal = function () {  
 console.log("Original favourite animal: " + animal);  
 let animal = "giraffe";  
 console.log("New favourite animal: " + animal);  
};

favoriteAnimal()

In the creation phase, the execution context exhibits a behavior called hoisting. Hoisting is the behavior of moving all the declarations at the top of the scope before code execution which happens in the execution phase. The javascript engine goes through the code and looks for all “var”s and “function”s and declares them, initializing to undefined, before actually starting to execute the statements. In other words, it assigns memory for all vars and functions in the code before executing the code statements.

1. How legacy js function retain scope.
2. How to find duplicates in array, how to remove duplacte values in array.  
   remove duplicate form array.

["a","a","b","c","c"].filter((val,i,arr)=>{

return arr.indexOf(val) == i;

})  
  
to get duplicate array  
["a","a","b","c","c"].filter((val,i,arr)=>{

return arr.indexOf(val) !== i;

})

1. What is event loop and is js is async and sync how to make async.  
     
   <https://www.digitalocean.com/community/tutorials/understanding-the-event-loop-callbacks-promises-and-async-await-in-javascript>  
     
   Promise.resolve schedules a microtask and the setTimeout schedules a macrotask. And the microtasks are executed before running the next macrotask.
2. Propetys of inhertence.   
     
   interface ClockInterface {

currentTime: Date;

setTime(d: Date): void;

}

class Clock implements ClockInterface {

currentTime: Date = new Date();

setTime(d: Date) {

this.currentTime = d;

}

constructor(h: number, m: number) {}

}

interface Shape {

color: string;

}

interface Square extends Shape {

sideLength: number;

}

let square = {} as Square;

square.color = "blue";

square.sideLength = 10;

1. Es6 feature in .bind method.
2. Java script function.
3. Async and await.  
     
   https://www.digitalocean.com/community/tutorials/understanding-the-event-loop-callbacks-promises-and-async-await-in-javascript
4. Type of in type script.
5. Api call states.
6. Navigate.platform

Tells browser details.

1. Slice and splice.  
     
   var array=[1,2,3,4,5];

console.log(array.splice(2));

This will return [3,4,5]. The **original array is affected** resulting in array being [1,2].

var array=[1,2,3,4,5]

console.log(array.slice(2));

This will return [3,4,5]. The **original array is NOT affected** with resulting in array being [1,2,3,4,5].

1. Typeof and instanceof operator.

Typeof:- retuns a string of what type he operand it.

Instanceof:- does not work with primitive data types. Works on object retuns what type of object.  
var p = new Person("Jon");

p instanceof Person

That is p instanceof Person is true since p inherits from Person.prototype.

28)Localstorage and cookies.

**Capacity:**

* **Local Storage:** 10MB
* **Cookies:** 4kb

**Browser Support:**

* **Local Storage:** HTML5
* **Cookies:** HTML4, HTML5

**Storage Location:**

* **Local Storage:** Browser Only
* **Cookies:** Browser & Server

**Send With Request:**

* **Local Storage:** Yes
* **Cookies:** No

**Accessed From:**

* **Local Storage:** Any Window
* **Cookies:** Any Window.

**Expiry Date:**

* **Local Storage:** Never Expire, until done by javascript.
* **Cookies:** Yes, Have expiry date.

1. Promise states

* **Pending** - Initial state before being resolved or rejected
* **Fulfilled** - Successful operation, promise has resolved
* **Rejected** - Failed operation, promise has rejected.

1. **What is a constructor in java script  
     
   simple function which is used to create object. Any function with new   
     
   let Author = function(name,age,book){  
    this.name = name;  
    this.age = age;  
    this.book = book;  
   }  
     
   var obj = new Author(“ABC”,20,”BOOK1”);**
2. **Inheritance is possible in java script?  
     
   var num = 10;**

num.\_\_proto\_\_

object inherits another object  
  
let animal = {

eats: true

};

let rabbit = {

jumps: true

};

rabbit.\_\_proto\_\_ = animal; // (\*)

// we can find both properties in rabbit now:

alert( rabbit.eats ); // true (\*\*)

alert( rabbit.jumps ); // true  
  
String.prototype.show = function() {

alert(this);

};

"BOOM!".show();

const ABC = function(){  
 this.a = 30;  
 }  
  
 const child = function(){  
  
 }  
  
 child.prototype = new ABC();  
 new child();  
  
 function Person(firstName, lastName) {

this.FirstName = firstName || "unknown";

this.LastName = lastName || "unknown";

}

Person.prototype.getFullName = function () {

return this.FirstName + " " + this.LastName;

}

function Student(firstName, lastName, schoolName, grade)

{

Person.call(this, firstName, lastName);

this.SchoolName = schoolName || "unknown";

this.Grade = grade || 0;

}

//Student.prototype = Person.prototype;

Student.prototype = new Person();

in typescript we need to use :- extends for class  
 implements use for interface (we can use ‘n’ number of interface’s).  
  
 32) NaN === NaN and NaN == NaN // false

1. What is constructor in java script?  
     
   java script constructor is an function which created object  
     
   var fun1 = function(){  
    this.a = 30  
   }  
     
   funobj = new fun1();  
   in class using extend key word
2. Higher-order function.  
     
   is a function that receives a function as an argument or retuns the function as output  
     
   Array.prototype.map  
   ‘’ ‘’ .reduce  
   ‘’ ‘’ .filter
3. Java script curring  
     
   let fun6 = function(a){

return function(b){

if(b){

console.log('111',a,b);

return fun6(a+b);

}else{

console.log('222',a,b);

return a;

}

}

}

console.log(fun6(4)(6)(7)());

1. y.getA().getB()  
     
   let y = {

a:1,

b:2,

getA(){

console.log(this.a);

return this;

},

getB(){

console.log(this.b)

}

}

1. [1,2].print();  
     
   Array.prototype.print = function (value,i){

this.forEach((val)=>{

console.log(val);

})

}

1. Const newB = new b(‘x’,’y’);  
     
   const a = function(x){

this.x = x;

this.getA = function(){

console.log(this.x);

}

}

const b = function(x,y){

this.y = y;

a.call(this,x);

this.getB = function(){

console.log(this.y);

}

}

let newB = new b('x','y');

newB.getB();

newB.getA();

1. Deap Clone  
     
   const obj = {

a:{

b:{

c:1

}

}

};

const cloneObj = JSON.parse(JSON.stringify(obj));

cloneObj.a.b.c = 2;

console.log(obj.a.b.c);

1. Sort Array   
     
   const arr1 = [1,2,3,7,9];

const arr2 = [2,5,7,12,100];

const arr3 = [...arr1,...arr2];

const sort = arr3.sort((a,b)=>{

return a - b;

})

console.log(sort);

1. obj1.getX() ?  
   const obj1 = {

x:1,

getX(){

var self = this;

const inner = function(){

console.log(self.x); // if you use this.x get undefined. Or use arrow function it will work.

}

inner();

}

}

obj1.getX();

1. Flatten array   
     
   const merge1 = [[1],[2],[3]];

const merge2 = [[4],[5],[6]];

const merge4 = [...merge1,...merge2];

var merge3 = [].concat.apply([],merge4);

console.log('merge3',merge3); or we can use arrays.flat(1); with depth.

## What is the Temporal Dead Zone in ES6

## I've heard that accessing let and const values before they are initialized can cause a ReferenceError because of something called the **temporal dead zone**.

## NaN == NaN? NaN typeOf is number

## What is use of type script

## What use of let.

## Destructing in java script.

## Async and await ? what is await.

## For in and for of. The for...of loop is used to iterate through the values of an iterable.

## The for...in loop is used to iterate through the keys of an object

## Type script data types and protoTypes. string,Boolean,any,voide,number, tuple

## Promise all and race. any -> it check which one is success .

## ,race -> which promise execute first if is success or fail

## all -> it will resolve all promises, if any promise fail it throws error

## allSettled -> I will waite to all promises if it is rejected

## The "use strict" Directive The purpose of "use strict" is to indicate that the code should be executed in "strict mode".

## "use strict";

## x = 3.14; // This will cause an error because x is not declared

## Please note that null and undefined behave differently here: null becomes zero while undefined becomes NaN.

## Number(undefined) – null

## Number(null) – zero.

## alert( null === undefined ); // false

## alert( null == undefined ); // true

alert( null > 0 ); // (1) false

alert( null == 0 ); // (2) false

## alert( null >= 0 ); // (3) true

## null convert to zero. Where ‘undefined’ convert to ‘null’

## let num = 255;

## alert( num.toString(16) ); // ff

## alert( num.toString(2) ); // 11111111

## function abc (a,b,a){

## console.log(a,b,a);

## } abc(1,2,3) // 3,2,3 // duplicate parameters.

## [1,2] + [3,4] / / '1,23,4';

## function abc (){

## abc.greet = "hello";

## console.log(abc.greet); // hello

## }

1. how to create object

<https://www.freecodecamp.org/news/a-complete-guide-to-creating-objects-in-javascript-b0e2450655e8/>

1. [Gulp](http://gulpjs.com/) is a build system and JavaScript task runner which can automate common tasks of any website like minification, checking js errors, bundling of various js and css files, compile SASS, optimize images, create sprites, concatenate files and many more.
2. diff b/w http and https

