L1

01 March 2023 13:12

Execution context :
Memory and code
Single threaded<
For async : uses call stack, call by queue and event loop
Js creates global execution
Why use hoisting in JavaScript?
In JavaScript, hoisting allows you to use functions and variables before they're declared

Javascript makes global execution context even for empty program
Javascript is loosely binded language, where var a can store any data type .

In functional programming, data cannot be stored in objects, and it can only be transformed by creating functions. In object-oriented programming, data is stored in objects.

From <https://www.google.com/search?q=how+is+functional+programming+different+from+object+oriented+programming&rlz=1C1GCEA_enIN1046IN1047&oq=how+functional+programming+is+different+from+obprogramming&aqs=chrome.1.69i57j0i7i8i30j0i390l4.23202j0j7&sourceid=chrome&ie=UTF-8>

High Order Functions : - Function which takes a functions as a input or returns a function as a output.
Focus on Dry method (don't repeat yourself) try to break down the program into smaller logic.
And write a code into functions (javascript support functional programming).
Prototype : - it can make function available for all array (as a defined) and make function works similar to map.

Map , high order functions can be used for transformation of the values.
Filter ===> filters out the values by and returns values satisfying the filtering condition (E.G. maximum, minimum)

```
const arr = [5,3,2,16,1];
const output = arr.filter((x) => x<4);
//const output = arr.filter(isEven);
console.log(output);
```

```
const arr = [5,3,2,16,1];
function isEven(arr)
{
    for(let i=0;i<arr.length;i++)
     {   if(arr[i] % 2 === 0)
        console.log(arr[i]);
}
}
    isEven(arr);
```

Reduce ===> takes two argument function and a initial value for a function
returns a single object as a output(not a list like filter and map)

```
const arr = [5,3,2,16,1];
const output = arr.reduce(function(acc, curr){
    if(acc < curr)
        acc = curr;
    return acc;
}, 0);
console.log(output);
```

Example of map, filter, reduce ---

```
const users = [
    {firstname: "askahay", lastname: "saini", age:26},
    {firstname: "donald", lastname: "trump", age:75},
    {firstname: "elon", lastname: "musk", age:50},
    {firstname: "deepika", lastname: "padukone", age:26},
]


    //using reduce
       const output1 = users.reduce(function(acc, curr){
    if(curr.age < 30)
    acc.push(curr.firstname);
    return acc;
}, []);
        console.log(output1);


        //uisng filter and map
       const output = users.filter((x) => x.age < 30).map((x) => x.firstname);
       console.log(output);
```

L2

02 March 2023 11:22

Lexical Environment where child gets access to its own local env memory and its parent.

Let, const , var : use more in programing const	let	var

Let is only limited to the block scope and cannot be access before declaration or initialization.
Let value can be changed.
const cannot be redeclared or can be used again (name).

Let, const are declared in separate script
If let, var, const are declared in block i.e. { …} then they are declared in block memory ; if in function then they are declared in local memory of its function otherwise in global memory.
and that's why shadowing happens due to scope.
Block also follow lexical scope chain pattern.
Closure : function bundled with its lexical scope.
Var can be function
Entire function can be passed as an argument to a function, and return function(i.e. function name or function)(it returns entire function line by line).

While running setTimeOut in a loop using var it will refer to the same variable therefore use let as it has block scope it will refer to the new let value every time.

FUNCTIONS ACT AS A VALUE IN A JAVASCRIPT.

Functions statement and function expressions.
Anonymous functions
Named Functions

Functions

16 March 2023 11:04

Functions are the first class citizen i.e. first class functions
Functions can be used in many ways in javascript:
Using as a normal regular function syntax -

```
function a(){
}                              This is function statement aka function declaration.
a();
```

Using as a variable value --

```
Const b = function(){
}                                        This is function expression (function as a value to
b();                                     a variable.)
```

Function without name is called Anonymous function.
Anonymous function are mostly used when function are returned from or passed to functions as a arguments.
Anonymous function cannot used like normal function in program , it will give syntax error.

Named Function Expression : function declared using name given to function.

First class function : Ability of function to be passed as arguments and return from function is first class function.

```
Eg: const  b = function(param1)
{
                  Return function xyz(){
                  }                 This will print --->
}                                   f xyz(){
Console.log(b());                   }
```

```
Var b = function xyz(){

}
xyz();
```

This will reference error because during hoisting it will declare the variable and functions in global space i.e. function a and const b are declared in global space but value assign to var b is local to b, hence cannot be accessed.

Let, Const and Var

16 March 2023 12:41

- Var :
1. is not block -scoped, but are function scoped.
2. can be redeclared inside the same scope.
3. Let can be initialized later after declaration.

- Let:
1. Are block scoped
2. They cannot be redeclared but updated.
   Eg: let fruit ="apple"
   let fruit = "mango" // error
   Fruit = "grapes" //updates the value of fruit to grapes.
3. Let can be initialized later after declaration.

- Const
4. Are block scoped
5. They can neither redeclared nor updated
6. Const should be initialized when declared.

While declaring

	Declaration after use	Initialization after use
var	No error i.e. are hoisted	No error (initialized with undefined) From <https://www.google.com/search?q=are+const+in+javascript+hoisted&rlz=1C1GCEA_enIN1046IN1047&oq=are+const+in+javascript+hoisted&aqs=chrome..69i57j0i22i30l3j0i390l3.12444j0j7&sourceid=chrome&ie=UTF-8>
let	No error i.e. are hoisted	Error (remain uninitialized)
const	No error i.e. are hoisted	Error(remain uninitialized)

Scope and Closures

16 March 2023 16:02

that the return statement does not execute the inner function — a function is executed only when followed by () — , but rather the return statement returns the entire body of the function.]

Understand the execution of this program to understand the closures.

```
function grandParent() {
    var house = 'GreenHouse';

    function parent() {
        var car = 'Tesla';
        house = 'YellowHouse';
        function child() {
            var scooter = 'Vespa';
            console.log('I have:', house, car, scooter);
        }
        child();
        return child;
    }
   parent();
    return parent;
}
var legacyGenX = grandParent();
console.log(typeof(legacyGenX)); //legacyGenX is of type function
var legacyGenY = legacyGenX();
console.log(typeof(legacyGenY)); //legacyGenY is of type function
legacyGenY(); // I have: YellowHouse Tesla Vespa
```

Since car and house is part of the outer function, car and house would only exist while the parent() function is in execution. Since the parent() function finished execution long before we invoked legacyGenX(), any variables within the scope of the parent() function cease to exist, and hence variable car and house no longer exists.

How does JavaScript handle this?

In other words, the inner function preserves the scope chain of the enclosing function at the time the enclosing function was executed, and thus can access the enclosing function's variables.

Website Structure

17 March 2023 11:36

Use lowercase and hypen for naming files and folder

- Javascript loads first
- Understand Developers tools, console.
- Javascript is case sensitive.
- Javascript code without ; may still work, but javascript requires ; at the backend, js interpreter add ; at the end of statement to distinguish between statements.3
- Javascript runs from top to bottom
- '+' operator can be used to append two string or integer with the string.
- Shorthand operator:
  - A += ;
  - A+ A =
- Used to check values in console and debugging
  - Document.write()
  - Console.log()
- Boolean (true/false) is enclosed in " ".
  - consta =100;constb ='100';console.log(a ==b)// true
  - From <https://www.freecodecamp.org/news/loose-vs-strict-equality-in-javascript/>
  - consta =100;constb ='100';console.log(a ===b);//false
  - From <https://www.freecodecamp.org/news/loose-vs-strict-equality-in-javascript/>
- !== and !=

```
if( '5'!== 5){
     return false}
else{
     return true;
}
Output: false

if( '5'!= 5){
     return false;
}else{
     return true;
}
Output: true
```

- Object is created with the var keyword.

```
Var myCar = new Object();
myCar.engine = "eve";
myCar.model = "eefe";
myCar.drive = function(){
 ▪ Console.log("it's a function");
}
Or
Var myCar = {
 ▪ engine: " eve";
 ▪ model:"eefe"
 ▪ drive:
 ▪  function(){
    □ Console.log("it's a function");
}
```

Method is set of instructions
 called on object. It is related to object.
Function is also set of instruction which is called by anyone, anywhere.

Arrow functions are always unnamed. If the arrow function needs to call itself, use a named function expression instead. You can also assign the arrow function to a variable so it has a name.

```
// Traditional Function
functionbob(a)
{returna +100;}
// Arrow Function
constbob2=(a)=>a +100;
```

From <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions>

- Date object is very useful it gives time :
  Tue Jun 06 2023 11:42:12 GMT+0530 (India Standard Time)
  And has method like

Method	**Description**
getFullYear()	Get **year** as a four digit number (yyyy)
getMonth()	Get **month** as a number (0-11)
getDate()	Get **day** as a number (1-31)
getDay()	Get **weekday** as a number (0-6)
getHours()	Get **hour** (0-23)
getMinutes()	Get **minute** (0-59)
getSeconds()	Get **second** (0-59)
getMilliseconds()	Get **millisecond** (0-999)
getTime()	Get **time** (milliseconds since January 1, 1970)

From <https://www.w3schools.com/js/js_date_methods.asp>

- DOM
- HTML elements
  - Document is the primart object
  - Each element in the HTML is the object
  - It can be accessed using document.getElements......
  - Different methods are used with the elements to alter , update, create, remove
  - setAttribute, getAttribute, innerHTML, createElement, appendChild

- HTML events
  - onClick , onMo
  - use such events occur when each activity is done on webpage

- Windows.onLoad: in index.html the webpage content loads according to the place in code, if script is present before body then the changes are not applied hence windows.onLoad is used to avoid it.

- Forms

- Forms Validation using javascript onsubmit function.

- Windows is the object in the javascript it is global object of browser properties. Enables access any browser property,
  - SetTimeout allows the event to happen after some specified time using window
  - SetTimeInterval sets the execution of function continuously after set time interval.
  - CloseTimeInterval stops the execution of function in setTimeInterval.
  - This all are properties of windows
  - Browser: room and window : is a tool/object using which we can do change in room and access properties of room.

- Async Javascript
- ###
- # due to callback junction we can do async operations with javascript #
- ###
- Async code: code that start something now but finish later
- Async Js :governs how we perform such tasks which take some time to complete.
- Like fetching data from database, API.
- Event : anything that's happen in webpage, e.g. click the button, move cursor over element
- Event handler when events occur then what should be done that can be specified by event handler.

PROMISES
There are two outcomes of the promise either result is successfully executed and error if not.
i.e. we say we resolve the promise if they the output or reject if any error occur.

There is access to two parameter in promise that are (resolve, reject) parameters .
These two are the functions built in promise API.
Promises has executor function in constructor. And handler: .then(), .catch(), .finally()
Function has two function as a parameter resolve and reject which these handler function handles.

Handler is executed on promise object. which has executor function in which resolve or reject is specified.
Apply .then() on function which returns a promise object or on a direct name of promise object , both are correct.

.then() always returns a promise with pending state and undefined value. It can create a chain on it. . For this case example refer: https://www.freecodecamp.org/news/javascript-promise-tutorial-how-to-resolve-or-reject-promises-in-js/
Or write a function in then() that return a new promise again can apply .then() on it but this time it will resolve on the method that new promise contains. For this case example refer: [Asynchronous JavaScript Tutorial #8 - Chaining Promises]()

typescript

26 June 2023 12:26

Use any than union

Union | is used to define type as or

Eg. const username: string | number | boolean

Or

To restrict the values that can be assigned to the variable

E.g. const role: "user" | "admin" | "editor"

Function declaration:

createUser : () => <return_type>

createUser(): <return_type> ------> more strict telling that createUser will be always function.

Variable declared **private** are only accessible inside the class.

Setter function in typescript does not return any value/type

Variables declared **protected** are accessible within the class and classes which inherit the parent class