* Define the closure concept in JavaScript and mention some of JavaScript’s features that are based on closure.

Closure means that the function has access to the variable environment of the of the execution context (outer function’s variable) in which it was created even after the execution context is gone.

Closure was created when we define function inside function.

Some JavaScript feature using closure:

1. We can use it to achieve encapsulation, we can create variables that can only accessed and modified by the inner function.
2. Also, we can use it in module pattern to create private method in function and return only the public one.
3. Can be used when we try to fetch data.

* Create an example using closure.

*function* incrementCounterToTen(*counter* = 0) {

  return {

    incrementInner: *function* () {

      if (counter < 10) {

        counter++;

      }

      return counter;

    },

  };

}

*let* increment = incrementCounterToTen(8);

console.log(increment.incrementInner());

console.log(increment.incrementInner());

console.log(increment.incrementInner());

her we apply the encapsulation by making the inner function only one that can modify the counter, as if we made it private and create setter and getter.

* Define the promise and what is the difference between fulfilling the promise using (Async/Await) and (Then) Make request using your function to consume the service of the following link <https://jsonplaceholder.typicode.com/users> using .then one time and using async/await

Promise is an object that represent the completion or failure of asynchronous task, it has three states.

1. Pending 🡪 initial state
2. Fulfilled 🡪 success.
3. Reject 🡪 failure.

We use .then method to specify the event should happened after the promise is succussed or failed

The difference between .then and async/await

1. . then 🡪

- when we use. then the execution of the function will not pause until the asynchronous task is done so the code after fetch function will complete execution normally

- handling errors using catch

*function* useThen() {

*let* example = fetch('https://jsonplaceholder.typicode.com/users').then(

*res* => {

      console.log(' inside the then() this will printed thired ');

    }

  );

  console.log('outside fetch  will printed first');

}

useThen();

console.log('outside function this will printed second');

1. async/await 🡪 this will pause the function until the asynchronous task is done.

so, the code in the function that include fetch will be executed after the fetch is done

* handling error using try/ catch.

*async* *function* usingAsyncAwait() {

*let* example = await fetch('https://jsonplaceholder.typicode.com/users');

  console.log('inside function will print second');

}

usingAsyncAwait();

console.log('outside function will print first');

fetch data using .then

*function* fetchUsersUsingThen() {

  fetch('https://jsonplaceholder.typicode.com/users')

    .then(*response* => {

      return response.json();

    })

    .then(*users* => {

      console.log(' using .then:', users);

    })

    .catch(*error* => {

      console.error('Error using .then:', error);

    });

}

fetchUsersUsingThen();

fetch data using async/await

*async* *function* fetchUsersUsingAsync() {

  try {

*const* response = await fetch('https://jsonplaceholder.typicode.com/users');

*const* users = await response.json();

    console.log('using async/await:', users);

  } catch (error) {

    console.error('Error using async/await:', error);

  }

}

fetchUsersUsingAsync();

* What is the difference between Object, Array, Function in JavaScript

Object 🡪 is a collection of key, value data

Ex :

Let obj = {

name : ‘sara’

}

Access nama 🡪 obj.name or obj[‘name’]

Array 🡪 is a collection of ordered element

Ex:

Let arr = [1,2,3]

Access element 🡪 arr[0]

Function 🡪 is a block of code to execute some instructions, can be reusable

Ex:

Function getCount() {

Return count;

}

Call function 🡪 getCount()

* Create a function that takes a name and returns age according to that name using switch case for one example and lookup object (lateral object) in another example

1. Using switch

*function* getAgeUsingSwitch(*name*) {

  switch (name) {

    case 'sara':

      return 27;

    case 'sedra':

      return 1;

    case 'daina':

      return 4;

    default:

      return 'Name not exist';

  }

}

console.log(getAgeUsingSwitch('sara'));

console.log(getAgeUsingSwitch('nada')); *// not exist*

1. Using object

*function* getAgeUsingObject(*name*) {

*let* ages = {

    sara: 27,

    sedra: 1,

    daina: 4,

  };

  return ages[name] ? ages[name] : 'Name not exist';

}

console.log(getAgeUsingObject('sedra'));

console.log(getAgeUsingObject('dina')); *//not exixt*

* What is the difference between reduce and map and filter in JavaScript arrays

Map 🡪 is used to transform the array to new array, we use map when we want to apply something to all array items and get the result in new array.

For example, we want to multiply all element by three and store the result in new array

*const* arrNumber = [1, 2, 3, 4];

*const* newArr = arrNumber.map(*num* => num \* 3);

console.log(newArr);

filter 🡪 use to filter the array based on specific condition and store the result in new array.

*const* arr = [1, 2, 3, 4];

*const* filteredArray = arr.filter(*num* => num > 2);

console.log(filteredArray);

reduce 🡪 execute function on each element of the array and get a single value at the end.

*const* numbers = [1, 2, 3, 4];

*const* sumOFArray = numbers.reduce((*acc*, *num*) => acc + num, 0);

console.log(sumOFArray);

* What are the types of modules in JavaScript?

Modules used to organize code into small pieces and make code more reusable.

Types of modules in JS

1. CommonJS module
2. Asynchronous module definition (AMD)
3. Universal Module definition (UMD)
4. ES6 Module 🡪 recommend way in modern JavaScript.

Use ‘export’ and ‘import ‘

* List all loops syntax in javascript and explain the differences between them

1. For loop

for (*let* i = 0; i < 5; i++) {

  console.log(i);

}

1. While loop

*let* i = 0;

while (i < 5) {

  console.log(i);

  i++;

}

1. do .. while loop

*let* i = 0;

do {

  console.log(i);

  i++;

} while (i < 5);

1. for .. in loop

*const* person = { name: 'John', age: 30, city: 'New York' };

for (*let* key in person) {

  console.log(key + ': ' + person[key]);

}

1. for .. of loop

*const* array = ['a', 'b', 'c'];

for (*let* element of array) {

  console.log(element);

}

The difference between them 🡪

1. for loop 🡪 used when we know the number of iterations in advance
2. while 🡪 used when we loop based on condition and not know the number of iterations
3. do.. while 🡪 same as while but the condition is checked after the iteration
4. for .. in 🡪 used when we need to iterate in object properties
5. for .. of 🡪 used when er need to iterate over arrays, strings.

* What is the javascript rest parameter, spread operator and destructuring and swap a value of 2 variable using destructuring?

1. Rest parameter 🡪 allow function to take more args as array.

*function* getLength(...*numbers*) {

  return numbers.length;

}

console.log(getLength(1, 2, 3));

console.log(getLength(1, 2, 3, 4));

1. Spread operator 🡪 allows to expand iterabls like arrays, we can use it to combine two arrays into one.

*let* arr1 = [1, 2, 3];

*let* arr2 = [4, 5, 6];

*let* combinedArr = [...arr1, ...arr2];

console.log(combinedArr);

1. Destructuring 🡪 allows to get values from array to separated variables.

*let* [a, b, c] = [1, 2, 3];

console.log(a);

console.log(b);

console.log(c);

1. Swap two variable using destructuring

*let* x = 1;

*let* y = 2;

[x, y] = [y, x];

console.log(x);

console.log(y);

* Discuss the differences between synchronous and asynchronous JavaScript code execution.

In synchronous the code executed sequentially in order so if we have a long running operation it will block the other code until it finished

In Asynchronous, the code is not running sequentially, the code continue running to execute while the long running operation is running so the asynchronous not blocking the execution of the code

* What is the role of event-loop in JS

Event loop 🡪 is a part of runtime environment used to handle asynchronous tasks.

By continuously check the call stack where the code should execute if it is empty the eventloop take the first callback from callback queue and pass it to call stack to be executed

* What is the high order function? And give example

Heigh order function is function that take function or more as argument or return function.

Example of using it when we pass a callback to other function like map, filter, reduce.

*const* arr = [1, 2, 3, 4];

*const* filteredArray = arr.filter(*num* => num > 2);

console.log(filteredArray);

here is the filter function takes arrow function as a argument

* Explain the purpose of the bind, call, and apply methods in JavaScript with example.

Three methods are used to attach ‘this’ to function.

The difference between them

Call is used to invoke function with specific this and argument and arguments as a parameter.

*function* printPersonName(*greeting*, *emotion*) {

  console.log(greeting + ', ' + *this*.name + emotion);

}

*const* personObj = { name: 'sara' };

printPersonName.call(personObj, 'Hello', ':)');

Apply Call is used to invoke function with specific this and argument and arguments as an array.

*function* printPersonName(*greeting*, *emotion*) {

  console.log(greeting + ', ' + *this*.name + emotion);

}

*const* personObj = { name: 'sara' };

printPersonName.apply(personObj, ['Hello', ':)']);

Bind is used only bind ‘this’ to function but not call it, and the arguments are optional in this step, and we can add them when calling the function

*function* printPersonName(*greeting*, *emotion*) {

  console.log(greeting + ', ' + *this*.name + emotion);

}

*const* personObj = { name: 'sara' };

*let* printName = printPersonName.bind(personObj, 'Hello');

printName(':)');

* Discuss the concept of memoization and how it can improve performance in JavaScript.

Memorization is a technique used to optimize performance by caching the function results, and when it called again with the same arguments it checks if the result already stored in cache it return the result without computing it

This improves the performance by reducing the number of calculations, and save the time of getting the results, it also useful when getting data from API, er can get it one time and stored in cache.

* Define what polyfills is in JavaScript and provide an example of its usage.

Polyfill is a code used to support older browsers by provide the modern functionality to them to ensure that web apps can run on the same ways across different environment.

Example:

Promise was not existed before ES6 so the old browsers may not support it but we can use polyfill to add this functionality

* Define the difference between backward and forward compatibility and JS is backward or forward compatible .

Backward compatibility 🡪 means that we can run or use old versions of programs in newer versions.

forward compatibility 🡪 means that we can run or use newer versions of programs in old versions.

javaScript is backward compatibility this means we can not use modern JavaScript in older versions of browsers for example, but we can handle this by using polyfills, and the older version can be used in newer environment.

* Describe the differences between the splice, slice, and concat methods in JavaScript arrays.

Splice 🡪 used to adding or deleting elements to the original array, so it modified the original array

*let* numbersArray = [1,2,3,4];

*// Removing*

numbersArray.splice(1, 2);

console.log(numbersArray);

*// Adding*

numbersArray.splice(1, 0,5, 6);

console.log(numbersArray);

slice 🡪 return a partial copy from the original array and not modify it.

*let* numbersArray = [1, 2, 3, 4];

console.log(numbersArray.slice(0, 2));

*//orignal*

console.log(numbersArray);

concat 🡪 used to marge two arrays into new array and not modify the original ones.

*let* numbersArray = [1, 2, 3, 4];

*let* array2 = [5, 6]

console.log(numbersArray.concat(array2));

* Define what Transpilers is in JavaScript and provide an example of its usage.

Used to convert the code written in modern JavaScript code into old version to ensure the compatibility with older browsers.

We can use babel to achieve this.

* What are the scopes and what is the scope chain in JS.

Scopes means the accessibility of variables in the code.

There are three scops:

1. Global scope 🡪 variables declare outside any functions can be accessed from anywhere.
2. Function scope 🡪 variables declared in the function, accessed in the function only.
3. Block scope 🡪 variable declared within {} like inside ‘if’ but they must be let or const.

Scope chain 🡪 the order in which scope are searcher for the variables starts from inner to outer.

* What is the difference between var, let, const and their scopes.

var 🡪 used to declare variables that can be modified letter, var is not block scope it only function scope, var also hoisted and take initial value of ‘undefined.’

let 🡪 used to declare variables that can be modified letter, let is block scope and function scope, var also hoisted but not initialized so it will give us reference error, it still in TDZ.

const 🡪 used to declare variables that cannot be modified letter, const is block scope and function scope, var also hoisted but not initialized so it will give us reference error, it still in TDZ.

* What is the type of functions on JS and which of these hoisted.

1. Function declarations

*function* helloWorld() {

  console.log("Hello Worled!");

}

1. Function expression

*let* helloFun = *function*() {

  console.log("Hello Worled!");

}

1. Arrow Function

*let* helloFun = () => console.log('Hello Worled!');

1. Immediately invoked function expressions (LIFE)

(*function* () {

  console.log('Hello Worled!');

})();

Function declaration the only one can be hoisted and used before its declaration.