1.

Synchronous and asynchronous:

Synchronous is a blocking architecture ,so the execution of each operation depends on completing the one before it.

Asynchronous is a non blocking architecture ,so the execution of one task isn’t dependent on another tasks can run simultaneously.

2.

Callback:

Callbacks are a main aspect of Javascript ,as they allow you to run code after an asynchronous operation has been completed.

Callback is a function that is passed as an argument to another function,and is called after the main function has finished its execution.

3.

Fetch API:

Fetch API comes with the fetch() method,which is used to fetch data from different sources.The fetch API is a feature that allows to make HTTP requests such as GET,POST<PUT<DELETE to a web server.

4.

Syntax for Fetch API:

fetch(url)

  .then(response ){

  };

5.

Handling errors in fetch API:

\*try or catch is used to get errors in fetch API when the promise gets rejected.

And also there are different types of errors also there :

\*Reference Error.

\*Syntax Error.

\*Type Error.

The different types of errors are handling in different ways.

6.

Fetch API in JSON data:

Fetch API starts a GET request, and returns a response object when the request completes .

And from the server u can extract the response to JSON file and store it. This is the handling of fetch API in JSON data. Fetch API mainly uses GET request for the handling of JSON data .Because the it returns the fast response to the object when the request completes successfully.

7.

Function Argument Vs Parameter:

The values that declared within a function when the function is called are known as an argument.

The variables that are defined when the function is declared are known as Parameter.

8.

Arrow functions:

An Arrow function expression is a compact alternative to a traditional function expression, but it is very limited,and we should not use it in all situations.

The arrow function offers shorter syntax comparing to other functions in javascript and it have lexical scoping of “this”, and can’t be used as a constructor.

9.

Function and Method:

Function is a set of instructions that perform a task.

Method is a set of instructions that are associated with an object.

10.

This:

In Javascript the this keyword refers to an object.

The this keyword refers to the current object in a method or constructor.And also it refers the current text or scope.

11.

Limitations of arrow function:

The main limitation of arrow function is that it cannot be used as a constructors.

And the other limitations are;

\*The arrow functions are only anonymous.

\* The arrow functions have no prototype property.

12.

Destructring:

Destructring is a way to unpack arrays and objects and assigning to a distinct variable.

There are different ways of destructring is there;

\*Destructring in arrays .

\*Deststructring during iteration.

\*Destructring Object.

13.

\*\*Destructring in arrays.\*\*

const numbers = [1, 2, 3]

  let [numOne, numTwo, numThree] = numbers

  console.log(numOne, numTwo, numThree)

14.

\*\*Destructring object.\*\*

const rectangle = {

  width: 20,

  height: 10,

  area: 200

}

let { width, height, area, perimeter } = rectangle

console.log(width, height, area, perimeter)

15.

Javascript allows to destructre arrays, objects,and function parameters.

let employee\_details = ["Rohit", 1211, " Engineer"];

let getDetails = ([name\_of\_an\_employee, employee\_id, employee\_designation]) => {

  console.log(

    `Employee's name : ${name\_of\_an\_employee} , Employee's ID: ${employee\_id}, Employee's Designation: ${employee\_designation}`

  );

};

getDetails(employee\_details);

16.

Throw custom errors:

The throw statement allows us to create a custom error.It can be a string, number,Boolean, or an object.use the throw statement to throw an exception.

\*throw’Error2’

\*throw 21

\*throw true

\*throw new Error(‘Required’)

const throwErrorExampleFun = () => {

  let message

  let x = prompt('Enter a number: ')

  try {

    if (x == '') throw 'empty'

    if (isNaN(x)) throw 'not a number'

    x = Number(x)

    if (x < 5) throw 'too low'

    if (x > 10) throw 'too high'

  } catch (err) {

    console.log(err)

  }

}

throwErrorExampleFun()

17.

Try, Catch, Block;

\*Try:

The try statement defines a code block to run(to try).

\*Catch:

The catch defines a code block to handle any error.

18.

try {

  tryStatements

} catch (exceptionVar) {

  catchStatements

}

19.

try {

    // try\_statements

}

catch(error) {

    // catch\_statements

}

finally() {

    // codes that gets executed anyway

}

20.

Ternary operator:

Ternary operator is a concise way to write a conditional(if else) statement.Ternary operator takes three operands … Condition, True Value and False value.