<<<<<<<<<<<<<<<<<<<<<<<<< JAVASCRIPT >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

1.What is JavaScript?

=> Java script is a syncronouus single threaded language .

single threaded means => whole codes is executed one line at a time $ executed one comand at a time in a specefic order

2.Explain the difference between undefined and null.

=>1. Undefined means a variable has been declared but has yet not been assigned a value. Null is an assignment value.

It can be assigned to a variable as a representation of no value

2. undefined means the value hasnot been set, null means value has been set to be empty

ex=> 1. undefined => let a console.log(a)

2. null => let b = null

3. What are the different data types in JavaScript?

=> 1. number 2.string 3.booleon 4.undefined 5.null 6.bigint 7. symbol (premitives)

=> non premetives datatype is object

4.What is hoisting in JavaScript?

=> In JavaScript, Hoisting is the default behavior of moving all the declarations at the top of the scope before code execution. Basically, it gives us an advantage that no matter where functions and variables are declared, they are moved to the top of their scope regardless of whether their scope is global or local.

ex=>

5.Explain the concept of closures in JavaScript?

=>A closure is the combination of a function bundled together (enclosed) with references to its surrounding state (the lexical environment). In other words, a closure gives you access to an outer function's scope from an inner function.

6.What is the difference between let, const, and var in JavaScript?

=> 1.var => if we declere a variable from var, then we acan also declere it again with the same name,

and if we want to re-assign its value then we can do that too....

2.let => if we declare a variable with let, then we cannot declare it again with the same name,

but re-assign its value

3.const => and if we declare a variable with const , let we can neither declere it again,

nor re assign its value

7. What is the purpose of the this keyword in JavaScript?

=> 1.this keyword is a global object nothing but window object

2. in a method, this refers to owner object

8.What are the different ways to declare a function in JavaScript?

=>

variable diye jeta declere kor6i assign kor6i take function expression bole

* function decleration a function name thakkbe & hoisting support kore

9.Explain the concept of prototypal inheritance in JavaScript.

=>

10.What is the difference between == and === operators?

=> == is the abstract equality comparison operator and === is the strict equality comparison operator

11.How does event delegation work in JavaScript?

=>

12.Explain the concept of promises in JavaScript.

=>

below

13.What are arrow functions in JavaScript? How do they differ from regular functions?

=> In regular JavaScript functions, arguments keywords can be used to access the passed arguments when the function is invoked. But,

arrow functions do not have their own arguments and it uses the arguments from the outer function.

14. How can you handle errors in JavaScript?

=> cmplt

15.What are the different ways to loop over an array in JavaScript?

=> for each loop and its 3 arguments (value ,index and array)

=>

16.What is the purpose of the map() function in JavaScript?

=> map creates a new array by performing some operation on each array element .

17. What are some popular JavaScript libraries and frameworks?

=>

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Extra >>

A.

1. map => ekahne return korte hay and always new array creates kore

2. filter = > as a input callback function nay , return always booleeon values

3. reduce => The reduce() method executes a reducer function for array element. The reduce() method returns a single value: the function's accumulated result. The reduce() method does not execute the function for empty array elements. The reduce() method does not change the original array.

1. for each vs map = >>>>>>>>>>>>>>>

=> map can return new array by iterationg main array

=> for each can not return anything for each iterationg main array

2.Higher Order function =============== >

=> Higher order functions are functions that take other functions as arguments or return function as arguments

or returns functions as their results

3.closure =>>>>>>>>>>>>>

=> A closure is an inner function that has access to the outer function's variables and parameters.

It allows the inner function to access and manipulate the outer function's variables, even after the outer function has returned

2. A closure is an inner function that has access to the outer functions varuiable

ex:

4. javascript vs node js >>>>>>>>>>>>>>>>>>>>>>>>

javascript => Javascript is a programming language that is used for writing scripts on the website.It is basically used on the client-side.

node js => NodeJS is a Javascript runtime environment. It is mostly used on the server-side.

5. difference between filter and find method ?

=> filter method returns the matched values in an array from the collection

=> find method returns the first values that matches the value in findings, it will not check the remaining values in the array collection

6. higher order function?

=> A higher order function is a function that takes one or more functions as arguments, or returns a function as its result. There are several different types of higher order functions like map and reduce.

7. closure with example ==>>>>>>>>>>>>>>>>>>>>>

=> 1. a closure gives access to an outers functions scope from an inner function

2. Closure in JavaScript is a form of lexical scoping used to preserve variables from the outer scope of a function in the inner scope of a function.

8. waht is callback hell ?

=> Callback Hell: Callback Hell is essentially nested callbacks stacked below one another forming a pyramid structure. Every callback depends/waits for the previous callback, thereby making a pyramid structure that affects the readability and maintainability of the code.

9. what is promise?

=> a promise is a returned object to which you attach callbacks, instead of passing callbacks into a function.

=> A Promise is a JavaScript object that links producing code and consuming code

A JavaScript Promise object can be:

Pending => result undefined

Fulfilled => result is a value

Rejected => the result is an error object.

ex=>

```
let myPromise = new Promise(function(myResolve, myReject) {
// "Producing Code" (May take some time)

  myResolve(); // when successful
  myReject();  // when error
});

// "Consuming Code" (Must wait for a fulfilled Promise)
myPromise.then(
  function(value) { /* code if successful */ },
  function(error) { /* code if some error */ }
);
```

10. what is async & await ?

=> async makes a function return a Promise. await makes a function wait for a Promise.

=> The async and await keywords enable asynchronous, promise-based behavior to be written in a cleaner style,

11. what is then() in javascript ?

=> The then() method in JavaScript has been defined in the Promise API and is used to deal with asynchronous tasks such as an API call.

1. Callback Function =>

a callback function is a function passed into another function as an arguments ex:

```
function func1(){
    console.log("hiw i am anupam ");
}


function func2(callback){
    console.log("my age is 22");
    callback()
}


func2(func1)
```

2. all Types of functions ====>

1. function statement ==> // hosting support // also known as Function decleration

```
  function a (){
    console.log("hello");
}
a()
```

2. function expression // not support hoisting

=>

```
let x = function(){
    console.log("anupam");
}


x()
```

3.function Declaration

4.anonomous function

=> function without a name doesnot have identity return syntax error

uses=> we can use as an value

5. named functtion expression // create inside not global // create local variable

```
=> let x = function xyz(){
   console.log(xyz);
}
x()
```

6. difference between parameter and arguments?

=>

7. first class function ?

=> same as callback

8. Higher order function ?

=> Higher function are functions that take other functions as arguments or return functions as their results.

ex=>

9. call, apply & bind ?

=> Call invokes the function and allows you to pass in arguments one by one.

Apply invokes the function and allows you to pass in arguments as an array. Bind returns a new function,

allowing you to pass in a this array and any number of arguments.

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< Array methods >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

1. filter => Creates a new array with every element in an array that pass a test

2. find => Returns the value of the first element in an array that pass a test

3. forEach => Calls a function for each array element

4. map => Creates a new array with the result of calling a function for each array element

5. pop => Removes the last element of an array, and returns that element

6.push => Adds new elements to the end of an array, and returns the new length

7.Reduce => Reduce the values of an array to a single value (going left-to-right)

8.reverse => Reverses the order of the elements in an array

9. shift => Removes the first element of an array, and returns that element

10. slice => Selects a part of an array, and returns the new array

11. sort => Sorts the elements of an array

12.splice => Adds/Removes elements from an array

13.unshift => Adds new elements to the beginning of an array, and returns the new length

14 join => Joins all elements of an array into a string

10. closure example=>

```
function hello(name) {

  var message = "hello " + name;

  return function hello() {

    console.log(message);

  };

}

//generate closure

var helloWorld = hello("World");

//use closure
```

```
helloWorld();
```

11. explain import and exports =?

=>

12. this keyword in javascript ?

=> The Keyword ‘this’ in JavaScript is used to call the current object as a constructor to assign values to object properties.

13. What is the difference between Call and Apply? (explain in detail with examples)

Call

Call uses arguments separately.

Example:

```
function sayHello()

{

  return "Hello " + this.name;

}

var obj = {name: "Sandy"};

sayHello.call(obj);

// Returns "Hello Sandy"
```

Apply

Apply uses an argument as an array.

Example:

```
function saySomething(message)

{

  return this.name + " is " + message;

}

var person4 = {name:  "John"};

saySomething.apply(person4, ["awesome"]);
```

14. Explain Hoisting in javascript. (with examples)

Hoisting in javascript is the default process behavior of moving declaration of all the variables and functions on top of the scope where scope can be either local or global.

Example 1:

```
hoistedFunction();  // " Hi There! " is an output that is declared as function even after it is called

function hoistedFunction(){

  console.log(" Hi There! ");

}
```

Example 2:

```
hoistedVariable = 5;

console.log(hoistedVariable); // outputs 5 though the variable is declared after it is initialized

var hoistedVariable;
```

15. currying in JavaScript (with examples)

In JavaScript, when a function of an argument is transformed into functions of one or more arguments is called Currying.

Example:

```
function add (a) {

  return function(b){

    return a + b;

  }

}

add(3)(4)
```