**How JavaScript works**

It will be done by Execution Context.

Execution Context: The environment which executes the javascript code.

**Two phases:**

Memory Phase: Memory is allocated before the execution of code

Code Phase: Code will be executed line by line. Javascript is a synchronous single threaded language

Console.log(a) //undefined

Let a=10;

Console.log(a) //10

* node -v
* npm -v
* npm init – For initiating the package json file which contains the properties of javascript file

package name: (js-basics)

version: (1.0.0)

description:

entry point: (index.js)

test command:

git repository:

keywords:

author: suraj

license: (ISC)

keywords:

author: suraj

license: (ISC)

author: suraj

license: (ISC)

license: (ISC)

About to write to D:\JS-Basics\package.json:

{

{

"name": "js-basics",

"version": "1.0.0",

{

"name": "js-basics",

"version": "1.0.0",

"description": "",

"name": "js-basics",

"version": "1.0.0",

"description": "",

"main": "index.js",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "suraj",

"license": "ISC"

}

* npm i – npm install
* node index.js – To interact with node js ie instead of browser and run the program

**JavaScript Basics**

**Primitive/Value Types**

String

Number

Boolean

Undefined

Null

-----------------------------------------------------------------------------------------------------------------

**Reference Types**

Object

Array

Function

Let and var are blockscoped. It means if they are declared within a flower braces, then they cannot be accessed out of that.

**//---------------------------Pre-Requisites of React----------------------**

export const DoSomething=()=>

{

}

**//Components**

const MyComponent =()=>

{

    return <div> </div>

}

<button onClick={()=>

    {

        console.log("Hello World");

    }

}></button>;

**//Conditional Statements**

let age2=10;

let name2="Pedro";

if(age2>10)

{

    name2="Pedro";

}else{

    name2="Jack";

}

console.log(name2);

------------------------------------------------------------

const Component = () => {

  const age = 15; // Define age here

  return age > 10 ? <div>Pedro</div> : <div>Jack</div>;

};

export default Component;

**Functions in React**

Functions is the set of code that can be reused in React

To create React app

npm create vite@latest

give app name

cd my-react-app

npm install

npm run dev

Card Project

**App.jsx**

import Card from './Card.jsx'

function App() {

  return(

    <>

    <Card/>

    {/\* Two cards will be displayed \*/}

    <Card/>

    </>

);

}

export default App

**Card.jsx**

import neokred\_logo from './assets/neokred\_logo.jpg'

//In Component we can use the html, the className should be same used in the Index.css file. This card component is called in the App Component which is root component

function Card()

{

    return (

        <div className="card">

        <img className="card-image" src={neokred\_logo} alt="profile picture"></img>

        <h2 className="card-title">Neokred</h2>

        <p className="card-text">Fintech Company</p>

        </div>

    );

}

export default Card

**Index.css**

.card{

  border: 1px solid black;

  border-radius:10px;

  box-shadow:5px 5px 5px hsla(0, 0%, 0%, 0.1);

  padding:40px;

  margin: 10px;

  text-align:center;

  max-width:250px;

  display:inline-block;

}

/\* Access the card-image class in the Card image   \*/

.card .card-image{

  max-width: 60%;

  height:auto;

  border-radius:50%;

  margin-bottom:10px;

}

.card .card-title{

  font-family:Arial,sans-serif;

  margin:0;

  color:hsl(0,0%,20%);

}

.card .card-text{

  font-family:Arial,sans-serif;

  color:hsl(0,0%,20%);

}

}

**Props in React**

In React, "props" (short for properties) are used to pass data from one component to another, usually from a parent component to a child component. Props are read-only and allow you to make components dynamic and reusable by changing their values from outside the component.

How Props Work

1. Passing Props: Props are passed from the parent component to the child component as attributes in the JSX.
2. Accessing Props: In the child component, props are accessed using props (an object) or by destructuring the properties.
3. Immutable: Props are immutable, meaning they cannot be changed within the child component.

Example

**Parent Component (Passing Props)**

import React from 'react';

import Greeting from './Greeting';

function App() {

return (

<div>

<Greeting name="John" age={30} />

</div>

);

}

export default App;

**Child Component (Receiving and Using Props)**

import React from 'react';

function Greeting(props) {

return (

<div>

<h1>Hello, {props.name}!</h1>

<p>Age: {props.age}</p>

</div>

);

}

export default Greeting;

In React, "props" (short for properties) are used to pass data from one component to another, usually from a parent component to a child component. Props are read-only and allow you to make components dynamic and reusable by changing their values from outside the component.

**How Props Work**

1. **Passing Props:** Props are passed from the parent component to the child component as attributes in the JSX.
2. **Accessing Props:** In the child component, props are accessed using props (an object) or by destructuring the properties.
3. **Immutable:** Props are immutable, meaning they cannot be changed within the child component.

**Example**

**Parent Component (Passing Props)**

jsx

Copy code

import React from 'react';

import Greeting from './Greeting';

function App() {

return (

<div>

<Greeting name="John" age={30} />

</div>

);

}

export default App;

**Child Component (Receiving and Using Props)**

jsx

Copy code

import React from 'react';

function Greeting(props) {

return (

<div>

<h1>Hello, {props.name}!</h1>

<p>Age: {props.age}</p>

</div>

);

}

export default Greeting;

**You can also use destructuring for a cleaner syntax:**

function Greeting({ name, age }) {

return (

<div>

<h1>Hello, {name}!</h1>

<p>Age: {age}</p>

</div>

);

}

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**DOM – Document Object Model**

The Html page will be displayed on the browser. Browser engine will get convert into bytes.

A black text with black lines

Description automatically generated with medium confidence

How to access HTML element in the javascript?

Id value

Class value

Tag name

name attribute

**Event listener and Event Handler**

event ----------actions

event listener- It is used to catch the event occurred on the element. It will start with “on” keyword

event handler - It is function or JS function call

<button onClick =fn()”>Click me</button>

onClick is the eventHandler

**Spread Operator**

Spread Operator is used to copy the properties of one object to another object. The three dots(…) is used as Spread Operator

**Destructuring of Array or Object**

It is technique in JavaScript to unpack properties or elements from the object or Array.

**Closures in JavaScript**

It is a technique to access scope of outer function inside the inner function

Or It is a technique to define a function inside another function

**ES Module Pattern**

We have to create a .mjs file instead of writing the

**This keyword**

“this” is a special keyword in JS which will point to window object

**Call, Apply, Bind**

They are all predefined functions which is used to change the reference of the this keyword.

Call - There is no restriction, how much arguments it will take

Apply- It will take only two arguments in the array format

Bind – Same like call but it will return function

**Scope in Javascript**

Scope refers to accessibility

There are 3 scopes in javascript

1.Global Scope

2. Function Scope

3.Module Scope

**Scope Statement**

**var:**

* **Scope**: Function scope (i.e., it's scoped to the function in which it's declared or globally if declared outside any function).
* **Declaration and Initialization**: Both declaration and initialization are possible. var is hoisted and initialized with undefined.
* **Reassigning**: Reassigning is possible.
* **Shadowing**: Shadowing is possible. However, var can be shadowed by let and const within a block scope.

**let:**

* **Scope**: Block scope (i.e., it's confined to the block {} in which it's declared).
* **Declaration and Initialization**: Both declaration and initialization are possible. let is hoisted but not initialized, leading to a ReferenceError if accessed before initialization (due to the temporal dead zone).
* **Reassigning**: Reassigning is possible.
* **Shadowing**: Shadowing is possible. let can shadow variables declared with var or let in an outer scope, including the global scope.

**const:**

* **Scope**: Block scope (i.e., it's confined to the block {} in which it's declared).
* **Declaration and Initialization**: Declaration is possible, but initialization is mandatory (i.e., you must initialize const when you declare it).
* **Reassigning**: Reassigning is not possible. Once assigned, the value of a const variable cannot be changed.
* **Shadowing**: Shadowing is possible. Like let, const can shadow variables declared in an outer scope.

**Constructor**

Constructor is a special method in the class. Constructor purpose is to initialize the properties of the class.

While defining the constructor name should be constructor only and while calling we should use constructor name as class name.

**Promises in the JavaScript**

Promise is a special object in the JavaScript which contains success information or Error information(failure information)

It is created using special constructor called Promise()

There is a special way to add data inside the Promise Object

Promise object has 3 states

Pending state – Empty Promise Object

Fulfilled state – Data stored in the promise

Rejected State – Data stored in the promise using reject function

Consume or Access the Promise Object

1)then() and catch()

2)async await