JavaScript

Question: Explain arrays in JavaScript

Answer: An array in JavaScript is a special type of object used to store multiple values in a single variable. It allows us to hold a collection of elements, such as numbers, strings, or objects, which can be accessed using an index. JavaScript arrays are dynamic, meaning they can grow or shrink in size, and they support various built-in methods like push(), pop(), map(), filter(), and forEach() for efficient manipulation of data. Example

let fruits = ["Apple", "Banana", "Cherry"];

console.log(fruits[0]); // Output: Apple

fruits.push("Mango"); // Adds "Mango" to the array

console.log(fruits.length); // Output: 4

TypeScript

Question

Answer

HTML5

Question: How to call base class constructor from a child class?

Answer: In JavaScript, you can call a base class constructor from a child class using the super keyword inside the child class constructor.

class Parent {

constructor(name) {

this.name = name;

console.log(`Parent constructor called with name: ${name}`);

}

}

class Child extends Parent {

constructor(name, age) {

super(name); // Calls the constructor of Parent

this.age = age;

console.log(`Child constructor called with age: ${age}`);

}

}

const childObj = new Child("John", 25);

CSS

Question: What is an iframe and how does it work?

Answer: An **<iframe> (Inline Frame)** is an HTML element that embeds another document (webpage, video, map, etc.) within the current HTML page. It acts as a "window" to display external content seamlessly inside your webpage.

Example: <iframe src="URL" width="600" height="400" title="Example"></iframe>

React

Question: How does react works??

Answer: React is a JavaScript library for building **interactive user interfaces**. It follows a **component-based architecture** and efficiently updates and renders only the necessary parts of a web page using a **virtual DOM**

React-hooks

Question: How to call loading function in react useEffect only once?

Answer: To call a **loading function only once** in React's useEffect, pass an **empty dependency array ([])** as the second argument. This ensures the effect runs only after the **first render** and never again.

Example:

import { useEffect } from "react";

function App() {

useEffect(() => {

console.log("Component Mounted - This runs only once!");

loadData(); // Call your function here

}, []); // Empty dependency array ensures it runs only once

const loadData = () => {

console.log("Fetching data...");

// Simulate data fetching

};

return <h1>React useEffect Example</h1>;

}

export default App;

OOP

Question: What is the difference between a class and a structure?

Answer: A **class** is a reference type, meaning its objects are stored in the heap, managed by garbage collection, and passed by reference, making them suitable for complex objects that require dynamic memory allocation. On the other hand, a **structure** is a value type, stored in the stack, and passed by value, making it more efficient for lightweight data models. Classes support features like **inheritance, polymorphism, and encapsulation**, allowing for more flexible and reusable code, whereas structures do not support inheritance and are primarily used for simple data storage. Additionally, in a class, members are **private** by default, ensuring better encapsulation, whereas structure members are **public** by default. Due to these differences, classes are typically used for objects that represent entities with behaviors, while structures are used for small, immutable data objects that do not require complex relationships

Redux

Question: What is a flux?

Answer: **Flux** is an **application architecture** used in **React** for managing unidirectional data flow. It was introduced by **Facebook** to address the complexity of state management in large-scale applications. Flux follows a strict pattern consisting of **four main components**:

1. **Actions** – Objects that contain information (such as user interactions) sent to a central dispatcher.
2. **Dispatcher** – A central hub that receives actions and forwards them to relevant stores.
3. **Stores** – Containers for application state and logic, which update in response to dispatched actions.
4. **View (React Components)** – UI components that listen for changes in stores and re-render accordingly.

Unlike **MVC (Model-View-Controller)**, Flux enforces a **single-direction data flow**, making state changes predictable and easier to debug. It serves as the foundation for **Redux**, a popular state management library that extends Flux principles with a centralized store and strict immutability rules

Git

Question: What is git?

Answer: Git is a distributed version control system that allows developers to track and manage changes to code over time. It enables multiple developers to work on the same project by creating branches for independent work, and then merging changes back into the main codebase. Git stores a complete history of changes in a repository, allowing developers to revert to previous versions or collaborate more effectively. Its distributed nature means each developer has their own local copy of the project, making it faster and more reliable, with popular platforms like GitHub and GitLab providing online hosting for collaboration.

Web Security

Question: What is an SQL injection??

Answer: SQL injection is a type of cyberattack where malicious SQL code is inserted into a web application's input fields, such as forms or URLs, with the aim of manipulating the database. This happens when user input is not properly sanitized and is directly used in SQL queries. The attacker can then exploit this vulnerability to gain unauthorized access to the database, retrieve, modify, or delete sensitive data, or even execute administrative operations. SQL injection can be prevented by using parameterized queries, prepared statements, and proper input validation to ensure user input is safe and cannot alter the structure or logic of SQL queries

Unit Testing

Question: Should unit test be done for getters and setters?

Answer: Unit tests for getters and setters are generally not necessary because these methods typically have simple, predictable behavior: getters return values, and setters assign values to fields. Since they usually don’t contain complex logic, testing them separately is often redundant. However, if a getter or setter includes additional logic, such as validation or transformations, it should be tested. Instead of testing these methods in isolation, it is more effective to focus on testing the overall behavior of the class where they are used, ensuring that the system works as expected in real-world scenarios.

Agile and Scrum

Question: What is sprint in scrum??

Answer: In Scrum, a **sprint** is a fixed-length iteration during which a team works to complete a set of predefined tasks or user stories. Typically lasting between one and four weeks, a sprint aims to deliver a potentially shippable product increment. At the start of each sprint, the team selects items from the product backlog (prioritized by the product owner) and commits to completing them by the end of the sprint. During the sprint, the team holds daily stand-up meetings (also known as **daily scrums**) to discuss progress, challenges, and plans. At the end of the sprint, the team reviews the work completed and holds a retrospective to discuss improvements for the next sprint. Sprints are a core element of Scrum’s framework, allowing teams to deliver value incrementally and iteratively.