Milestone-7 All Necessary Notes

Module_37_How_JavaScript_and_Browser_Works

37-1 Module Introduction and how internet works

- Explained IP Address, Domain name, How does the internet work.

37-2 How Browser works, DOM tree, Render Tree

- When the browser receives an HTML file, it converts the HTML code into a DOM tree using HTML Parser. CSS Rules also does a similar task using CSS Parser. Then it combines render (Render Tree). Finally, it completes the painting and displays the user.

37-3 JavaScript Engine V8 Internal mechanism

- Browser (Google chrome) runs the JavaScript code using the **V8 engine**. V8 is Google's open-source high-performance JavaScript and WebAssembly engine, written in C++. It is used in Chrome and in Node.js, among others.
- V8 was first designed to increase the performance of JavaScript execution inside web browsers. In order to obtain speed, V8 translates JavaScript code into more efficient machine code instead of using an interpreter. It compiles JavaScript code into machine code at execution by implementing a **JIT (Just-In-Time) compiler** like a lot of modern JavaScript engines do such as SpiderMonkey or Rhino (Mozilla). The main difference here is that V8 doesn't produce bytecode or any intermediate code. [Source: How JS works]

37-4 setTimeout simple Asynchronous JS using

- Simply JavaScript code executes synchronously (line by line). But the code inside **setTimeout** function executes Asynchronously. It goes out of the normal flow, waits for all the other code to execute completely, then executes.
- **setTimeout** function takes multiple parameters. Normally it takes a call back function as the first parameter and a time period in milliseconds. The callback function executes after completing the time period.

37-5 Recognize fetch as an Asynchronous activity

- **fetch** is also works in an Asynchronous way like **setTimeout**.

37-6 setInterval and clearInterval with x++ and ++x

- **setInterval** also does its work asynchronously. It repeats the task infinitely by a given time interval (receive in milliseconds as the second parameter like setTimeout). It returns something that is used to stop the function. c**learInterval(returnVal)** is used used to stop the **setInterval** function. i.e.

```
// print from 1 to 16 by intervening 0.3 seconds and exit.
Let x = 0;
Let timeId = setInterval( () => {
        X++;
        console.log(x);
        If (x > 15) {
                clearInterval(timeId);
        }
}, 300);
```

37-7 (advanced) JavaScript event loop and concurrency

- JavaScript has a concurrency model based on an **event loop**, which is responsible for executing the code, collecting and processing events, and executing queued sub-tasks. This model is quite different from models in other languages like C and Java. [MDN]
- Must-watch What the heck is the event loop anyway? And read the Concurrency model and the event loop from MDN. This is a tricky and hard concept. But I have to know and understand it to become a better developer. So watch the video and read the article repeatedly, Explore from other sources if necessary in the future until the concepts have clear.

37-8 Integrate chrome devtool console with VS Code

- My VS Code interface is a little different due to switching on Ubuntu recently. Come back to this video a few days later after getting familiar with the new Operating System and its VS Code.

37-9 Module summary and HTTP layers

- JavaScript is a single-threaded synchronous Programming Language. But it has some functions which are work in an asynchronous way. setTimeout, setInterval, fetch, async-await are a few of them.

Module_38_Browser_API_and_Method

38-1 Module Overview, BOM vs DOM, Browser API

- All browsers have a set of built-in Web APIs to support complex operations and to help to access data. For example, the Geolocation API can return the coordinates of where the browser is located. [w3school]
- Read Introduction to Web API from MDN

38-2 Browser alert, confirm, prompt with examples

- **alert()** - Open a prompt from the top and show the passed string in it.
- **confirm()** - Open a prompt from the top, give two options Ok and Cancel, if the user clicks on Ok it returns **true**, and **false** for clicking Cancel.
- **prompt()** - Open a prompt from the top with an input field. Return the value that is entered by the user.

38-3 Location, URL parts, query string, href, hash, assign, reload

- URLs have various parts. A few of them are query string, href, hash, assign, reload, etc. To know about them read Location - Web APIs from MDN.
- We can access and set a new value for each and every part of the URL using the JavaScript location object.

38-4 History api to navigate back, forward, go, history length

- We can navigate the history of the current tab using the browser's **history** API. History API (object) has a few properties and methods. i.e **back, forward, go, length,** etc. Explore more about History API from MDN - History API

38-5 (advanced) Cookies, dev tool application tab, get cookie value

- We can also access the cookie of a website using the JavaScript **document.cookie;** It is a long string. It has contained name-value pair separated by a semicolon.
- Explore more about cookies: What is a Cookie from Kaspersky, and Document.cookie from MDN.
- To check a website's cooke from the browser using Application Tab -> Storage -> Cookies

38-6 Local storage, session storage, storage info in browser storage

- We can also access session storage and local storage using Chrom DevTool. Open Dev Tool, Go Application -> Storage -> Local Storage | Session Storage
- We can also access it using JavaScript browser API **sessionStorage,** and **localStorage.**
- Must check Introduction to Local Storage and Web Storage API from MDN to know about how to change, set, and get local storage and session key-value.
- Difference between local storage and session storage and Cookies- Interview Important

38-7 Dynamically set and read from local, session storage

- **localStorage.*getItem(key)*** - To get an item from current local storage
- **localStorage.*setItem(key, value)*** - To add an item to current local storage.

38-8 (advanced) retrieve local storage value and display them

- **localStorage.*removeItem(itemKey)*** - To remove a item from Local Storage.
- There is no operational difference between Local Storage and Session Storage. The main difference is Local Storage is saved on the browser for future uses but Session Storage deletes as soon as the tab is closed.

38-9 Module Summary, ContentEditable, chrome restart

- chrome://restart - Restart Chrome without losing any opened tab.
- chrome://newtab - Open a new Tab
- Ctrl + Shift + T - Reopen the last closed tab
- Shift + Esc - Task Manager of Google Chrome
- Ctrl + L - Select current tab URL

Module_39_JavaScript_debug_web_debug_dev_tool_mastering

39-1 Module overview, Dev tool overview, settings, shortcut keys

- Every developer needs to have these 3 skills, Google Search, Understand other developer's code, and Debugging.
- Explore Settings (Preference) from Chrome DevTools and check this Keyboard Shortcut List

39-2 Element tab, HTML event listener, accessibility list

- To check Event Listener of a specific element -> Right Click on the element and Inspect -> Go to Event Listener under Element Tab (Right side).
- Open Element Tab, Right-click on an element, and Explore the options.
- **Accessibility** is the practice of making websites usable by as many people as possible. We traditionally think of this as being about people with disabilities, but the practice of making sites accessible also benefits other groups such as those using mobile devices, or those with slow network connections. [Source: MDN - What is accessibility]

39-3 Console tab, preserve log, log levels, console api

- The shortcut way to select any element from the console is **$('css-selector')**;
- The basic log levels are ERROR, WARNING, INFO & Regular Console. Search for software engineering log level or take a look at Logging Levels
- Explore some other console properties and methods like console.error, info, count, etc.
- Take a look at the Console API reference for more detail about Console API.

39-4 Sources tab, break point, call stack, event break

- Explore Source Tab -> Debugger, breakpoint, call stack, XHR (fetch), snippet, event break, etc.
- Ctrl+G - To go any particular line of code.
- Ctrl+Shif+O -> To find only the functions.

39-5 Debug fancy slider and fix image display, console table

- Clone fancy-slider, Open the index.html, search for an image. The search result is not showing, try to fix this using Console, Source, Breakpoints, Debugger, etc.

39-6 More debug to fix slider to slide by getting duration

- After getting the search results and selecting multiple images, when the user clicked on create slides, it is not working. Also, try to fix it using dev tools.

39-7 Fix negative duration and how to read others code

- Try to read and understand all the code of this repo (fancy-slider), rewrite all the code after getting understand if possible.
- If the user enters any negative value or text into the Duration field, the slider image moving unexpectedly faster. Try to fix it and give an alert if the user enters anything other than a number or number less than 1000.

39-8 Performance tab, memory tab, Lighthouse tab

- The **performance** tab is used to check the overall performance of the website. It shows tons of information that is really hard to understand for Junior developers. Generally, these Performance-related tasks are handled by a senior developer.
- The **memory** tab indicates the overall memory uses and related analysis. It is also handled by senior developers.
- The **Lighthouse** tab is used to generate an overall lab report of a website including performance, accessibility, best practices, SEO, etc. This report helps developers to improve every area of a website. It is also handled by senior developers.
- Come back to this Console API Reference again and again throughout the journey to be a web developer. Check, read, and explore it to be a better developer.

39-9 Module Summary, Try out Create React APP

- Create React app into project folder for testing using the following command - or Visit [Create a New React App]
    - npx create-react-app my-app
    - cd my-app
    - npm start

Module_40_More_Debug_and_Regular_Expression

40-1 Module Introduction and Dev tool Element tab

- Clone salary-app, Explore DevTools one more time.

40-2 Dev Tool Console Source Tab Review

- Explore Console API reference and Console Utilities API one more time.

40-3 Explore Salary App and try to add a Record

- Add new Salary Record. But it will not add properly. Try to find out the problem.

40-4 Logical Not, double Not, fix add to Record to chart

- Logical Not or Bang (!) convert true to false and false to true. Double bang (!!) convert twitch.
- Fix add to Record to chart.

40-5 Lost with errors and trying to find ways to fix it

- It shows 14 records after clicking the Record Count button. But we can see 11 records on the chart. Try to fix this error by debugging.

40-6 Fix Duplicate names on the chart and display chart

- Add **!== undefined** in the uniquifyNames function > map> if condition. To fix the issue. But I couldn't understand it properly. Hope to come back here again.

40-7 Explains Unique name generation for duplicate data

- It quite understands. Spend a few more hours for feeling comfortable.

40-8 Simple Introduction to Regular expression

- Regular expression means write the JavaScript expression using some special character. It allows us to do more things by writing less code. It is specially used to check password requirements or the presence of a particular character in a string. We can fill our requirement of Regular expression by google search. To get an overview read This Article

40-9 Module Summary

- [x] ~~When clicking on show last, a modal will open and display the lastly added record.~~
- [x] ~~Some error message is showing when clicking on Show the Last button. Fix it.~~
- [x] ~~Clear the name and salary field after clicking on Add Record button and add them to the record.~~

Module_41_Getting_Started_with_TypeScript

41-1 Module Introduction and Setup Typescript

- TypeScript is a programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript and adds optional static typing to the language. TypeScript is designed for the development of large applications and transcompiles to JavaScript.[Wikipedia]
- Visit TypeScript's official website and install it on the project folder via npm.

41-2 Create output and connect html file for typescript

- We have to need to compile .ts file to convert it into a .js file. Initialize tsc using **tsc --init**
- We can edit the tsconfig.json file and change **"target": "es6"**, and **"outDir": "./js"** to compile it in es6 and compiled the file into the **/js** folder. Use **tsc** to compile all ts files.

41-3 Basic typescript type number string boolean

- We have to specify the type of a variable when declaring it in TS and can not assign the new type of value to it after declaration but same type.. i.e. **const fullName: string = 'Mahtab Uddin'; let job: boolean = true; const roll: number = 3232;**

- [ ] Check out the Basic Documentation of Typescript.

41-4 Set function parameter type and function return type

- We have to specify the function argument and return type in TS. i.e. **function add(num1: number, num2: number) : number { return num1 + num2 }**

- Use **void** if the function would not return anything and single or-operator (‘|”) to make a multi-purpose function.

41-5 Apply types for array and get error for type mismatch

- We can declare the type of array as well functions.. i.e. **const friends: string[] = [‘Rakib’, ‘Sakib’]**
  const salary: number[] = [7400, 12000, 14500];

41-6 Object types, type and arrow function type

- Type of object and arrow function is declared following.
- Object: **const student { name: string, age: number, job: boolean } = { name: ‘Sadar Uddin’, age: 27, job: true }**
  The left side can be declared before object declaration and used that: **type Person = { name: string, age: number, job: boolean }**
  const student: Person = { name: ‘Sadar Uddin’, age: 27, job: true }
- Arrow Function: **cont add = (num1: number, num2: number): number => num1 + num2;**

41-7 (advanced) Interface and extend interfaces

- There are a lot of use cases of **interface** instead of **type. Extends** one interface for another, and question mark **(?)** Sets a property optional. **i.e**
  interface Person { name: string, age: number },
  interface Player extends Person {club: ‘string’, wife?: boolean}
  const messy: Player { club: ‘Barselona’, name: ‘L Messy’, age: 35 }

41-8 (advanced) class and explore methods private fields

- When we use the constructor function in a class, we have to declare the type of the constructor function’s parameter before the constructor function declaration inside the class. Though it is not necessary for ES6.
- We can also privately declare a property using the **private** keyword in class.

41-9 Module Summary, How much typescript you should know

- Google Search and read 2-3 articles on each topic
  - ☐ What is TypeScript?
  - ☐ TypeScript vs JavaScript or Difference between TypeScript and JavaScript or Pros and cons of TypeScript

    - [ ] Read Documentation: The Basics and Everyday Types
- Being an expert in TypeScript is not necessary for React developers. Maybe 20% of cases will need TypeScript. It is very important (99%) for Angular people. But keeping a basic idea and a mindset of learning it when needed is very very important.