**Web Fundamentals-Question**

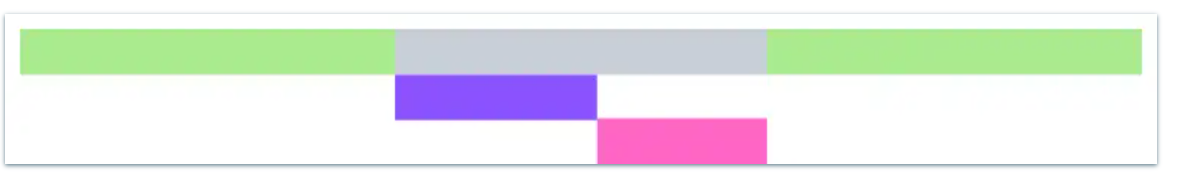
1. **What is the difference between the async and defer tag in the script?**

**🡪**Lets learn about the difference between the default ,async and defer tag:

* Before learning about them we need to understand that html is parsed line by line
* JS parsing is blocking in nature and it Blocks the html parsing and loading

Now letsunderstand :-

* **(Important)**When we write the script tag in html element, and soon as html meets the script tag it will start loading the scripting and stop parsing the html and if the script is heavy , blank page is shown on the screen , which makes loading slow
* **Default 🡪**When we just define the script tag without async or defer , the script is loaded as soon as it comes on html parsing and till the js parsing and execution is not finished then only html parsing will start .
  + - * **Some Very Very important because of the above explanation it is valid to define the script tag at the end of the htmlso the script tag only comes in html parsing when whole above html is parsed**



* + - **Async 🡪**When we just define the script tag with the **async tag** , the script will load along with the html parsing and as **soon as the script is loaded , then it will be executed while blocking the HTML parsing .**
      * **Async** tag is not viable to use as it can cause half domparsing
      * **But if we know the script tag has no dom interaction and we want fast script loading we can use async tag**

A colorful squares in a white background

Description automatically generated with medium confidence

* + - **Defer 🡪**Defer tag is best tag to load the javascriptfiles , as the defer tag makes javascipt to load with html parsing and **the javascript is executed always after the HTML is compelelty parsed**
      * **Using defer is same as loading the script at the end of the body tag.**

A green and purple rectangle

Description automatically generated

**Final Summary About Async , Defer**

1. If script is not depended on any other script then async is good. Because order of script execution will not affect anything.

2. If script depends upon another script then go with defer. Also if the script is depended on DOM.

3. If the script is small and is relied upon by an async script then use an inline script with no attributes placed above the async scripts.

2) **What is code splitting (Most Most Important 🡪 A Performance Optimization Techinque)**

**🡪** The code splitting is the mechanism where are sole purpose is to **load as less javascript on the load time** . The lesser the javascript on load time means **less parsing and less execution time .**

* **So how to implement the code splitting** 
  + - * **Lazy loading Techniques:-** We have to lazy load the component as much as possible for the components which are not required in the first render.

**How to Implement the lazy loading in React**

* + - * The React Provided built in function called **Lazy and using the lazy function we can import the component using the lazy function**
      * With lazy we need to wrap the lazy loaded component with the suspense actually what suspense did is we need to provide **fallback component will be rendered till the time lazy component is not loaded**

**Below is the syntax to how to load the lazy component.**

* constHome=React.lazy(()=>import("./components/Home"));
* constAbout=React.lazy(()=>import("./components/About"));
* const FAQ =React.lazy(()=>import("./components/FAQ"));
* const root =ReactDOM.createRoot(document.getElementById("root"));
* root.render(
* <React.StrictMode>
* <BrowserRouter>
* <Routes>
* <Route
* index
* element={
* <React.Suspense fallback={<>Loading</>}>
* <Home/>
* </React.Suspense>
* }
* />

1. **But Still how the code is spillted , what are the internal functioning of the code splitting?**

* So during the build time as we know babel transpiles the code and minification happens , which creates certain files .

Less understand what files are created :

During the build time of the app the separate , build folder is created , this build folder actually have the files which are being send to the browser to loading and execution.

1. **main.[hash].js:-**This the file which will be loaded on the first load , heavier this file will more javascript will be loaded and hence the performance issue will be their(make this file as less as possible) , all the libraries + components will be loaded in this file.
2. **main.[hash].map:-** This is the file which will create the map which contains the file chunk which are attached to main file + the chunk of the file required by those files also in cascading way.
3. **[hash].**js:- this are the chunks of the js file which are required to be loaded **not in the first load but on the time it will be required . So we need to focus to create this chunk files .That why its name is code splitting.**

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

With Lazy Loading

Without Lazy Loading

**Question 3) What are the difference performance optimization techniques?**

* Performance optimization is one of the most crucial thing to noted in development
  + **Code Splitting**
  + **Lazy Loading**
  + **Tree shaking / Removing the Dead Pattern**
  + **Image Optimization (Use of CDN / webp and avif images)**
  + **Skeleton UI for loading Optimization**
  + **Use of performance optimization hooks like useMemo ,memo and callback to handle unnecessary and heavy rendering**
  + **Image and font optimization**
  + **Use of server side rendering and client side rendering pattern**

**Question 4) What is Serverside rendering and client side rendering?(Most Important Part of the next js and optimization patterns)**

* Currently Just a flavour , but complete explanation will be afterwards

**A diagram of a server side rendering

Description automatically generated**

**A diagram of a server

Description automatically generated**

**A diagram of a software company

Description automatically generated**

**Question 6) What are Cross Site Scripting XSS and how to take measure to control cross site scripting?**

🡪The Cross site scripting attack is the frontend application attacks where the some hacker can induce some script through web loopholes and that script can get the browser store cookies , tokens from localstorage which can be used to do things from the users data like making unauthorized api call, even getting bank details

The XSS only occurs due to using poor web practices.

**For Example Setting the inner HTML (Never Ever Set the Inner HTML)**

*//Poor practices of the Cross site scripting*

conststr="<script>//some malicious script</script>"

constparagraph=document.querySelector('.paragraph')

paragraph.innerHTML=str

What are the Type of XSS attacks:

* **Stored XSS**
* **Reflected XSS**
* **DOM XSS**

**Lets Learn About the each of the following :**

1. **Stored XSS :**This is the process where the hacker any how stores the malicious script in the database and whenever that script is being fetched from the db to the browser the malicious script will run and infect which ever user browser it will ran.

**Example of this scenario:** A Chat application if not preserved the hacker can send the malicious code to the more privileged user which Is support person and can get their credentials

**Another Example :** The Product Review , if the textbox is not properly sanitized then the reviewer can submit the review with the malicious script tag and if it stores in the db , then every user which opens that pdp page the script will run and get the data

1. **Reflected XSS:** This is the process where the script needs not to stored into the database but based on the page rendering either client side or server side ,

**If the some person make the url with**

http[semicolon]//www[dot]somesite.com?q=nike+<script>MaliciousCode</script>/

Then the whoso ever will click the link will can run the script based on the rendering pattern this script can run on the server side also. And client side also

In the case of [server-side rendering](https://learnersbucket.com/examples/web/single-page-application-csr-vs-multi-page-application-ssr), the query is passed to the server and then it is returned and displayed on the screen, thus it is known as a reflected XSS attack.

1. **Dom Based XSS :**This XSS can be stored XSS or reflected XSS but it uses the DOM APIs to inject the script into the dom .

**Question 7) Ways to prevent the Cross Site Scripting (XSS)?**

* All the input sections should be input box , text area should be sanitized so that all the tags should be parsed as string only
* Use **DomPurify**library to sanitize html before injection
* **Use Safe Sink always use add new node if we want to add new child in dom using the vanilla js , never use** the innerHTML .
* **For the places where we have to show string as it is , encode the string to special characters escape them like & to "&amp;", < to &lt;, > to &gt;, " to &quot;, ' to &#x27;.**

**Question 8) How HTML is rendered into the browser ?**

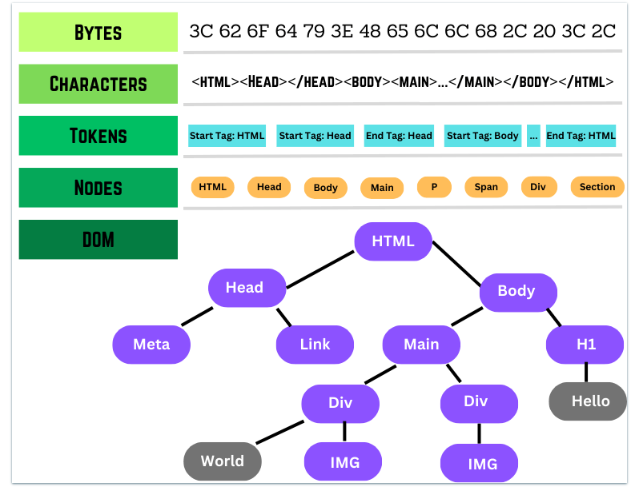
* To better understand about how web works we need to understand how the HTML is rendered onto the browser:-

The HTML rendering on the browser takes in five steps below are the following steps :-

* + - **DOM Formation**
    - **CSSOM Formation**
    - **Creation of Render Tree**
    - **Layouting**
    - **Browser Paint**

**Lets Learn about them Step by step :-**

* **DOM Formation:** This is the step 1 of the Process when the HTML is being send from the browser :
  + - * **HTML from the Server is bytes format**
      * **The Bytes are converted to the characters**
      * **Characters are converted to ElementTags**
      * **Element Tags are converted to nodes**
      * **Based on nodes the DOM tree is formed .**

****

During the HTML parsing the CSS and Script if present will be parsed and executed which are blocking in nature and due to html parsing maybe blocked .

* **Step 2 Formation of CSSOM:-** CSSOM called css object model , as the DOM is tree containing all the content . The CSS object model contains all the CSS properties node wise .

The formation of the CSSOM is blocking in nature ,as the CSSOM works in cascading and overriding manner .

* **Step 3 Formation of Render Tree**

Once the DOM and CSSOM is being Created, the **Render Tree is formed , the Render Tree is formed , The Render is the combination of Both DOM and CSSOM ,** where the DOM merges with CSSOM , the Render Tree contains **only those elements which will shown into the Browser .** If the node has **display:none** property , **that element + its decendants child will not be present in the render tree.**

* **Step 4 Layouting :-**

Once the Render tree is created the layout of the page is being formed , The layout the page is formed based on the device viewport . **If their no tag <meta device-width=”full-wdith”/> , then default size 960 px based layout is formed .**

**If the tag is present then the layout will always capture the device width .**

* **Step 5 Paint** 
  + - * After the layout is being formed then the page is painted with the content which is loaded like static images and content.

**Frontend System Design**

**Question 1) What is the monolithic Architecture?**

* Monolithic architecture is the pattern of designing the application where whole code lies in the simple repo or single folder structure .

Where all the code of frontend and backend lies in the single repository . This is the architecture which is used to follow in older applications or the applications which are used to design low to medium scale applications.

**Lets see the Pros and Cons of the monolithic Architecture and where we can use it**

**Pros of the Monolithic Applications:-**

* **Easy Deployment and Synchornization–** As all the code lies in the same place the deployment of the monolithic applications is relatively easy as whole code acts as the single source of the truth.
* **Small to Medium Scale Systems –** This is usefull to create small to medium scale applications .
* **Simplified Testing**

**Cons of the Monolithic Architecture:-**

* **Headless Development Not Possible : -** The developer cant do headless development where they use separation of concerns . In the monolithic architecture is not possible
* **Tight Coupling and New Framework Adoption issues:-** The tight coupling issue between the code and dependencies so new framework adoption is really hard and tedious
* **Scaling Issues :** As the whole code is single unit and couldn’t be broken down , they we can see some scaling issues. Might be due to infrastructure and growing code issues

**Question 2) What are the usecases and anti usecases for the monolithic applications?**

* **Usecases**
  + - * Building small scale applications for MVP or POC
      * Applications which has smaller development team for rapid prototyping
      * Applications where scaling doesn’t matter much
    - **AntiUsecases**
      * Building Large scale web applications with larger user database and larger code contributors
      * Applications where headless development is required , and separation of concern.
      * Applications where the different system need to be scaled differently and different scaling strategy is required.

**Question 3 ) What Microfrontends and how they solve the problem of the monolithic Architecture?**

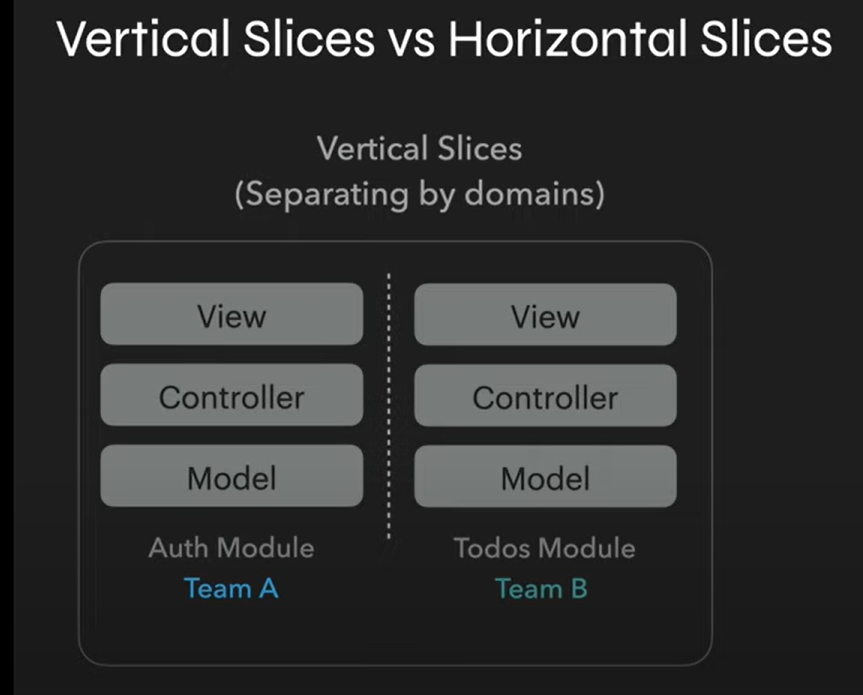
* The main issue with monolithic applications are is the separation of the concerns and scalability.

Micro-frontends is the architecture principle where **the whole application is being broken into the smaller sensible modules which creates a whole application combined.**

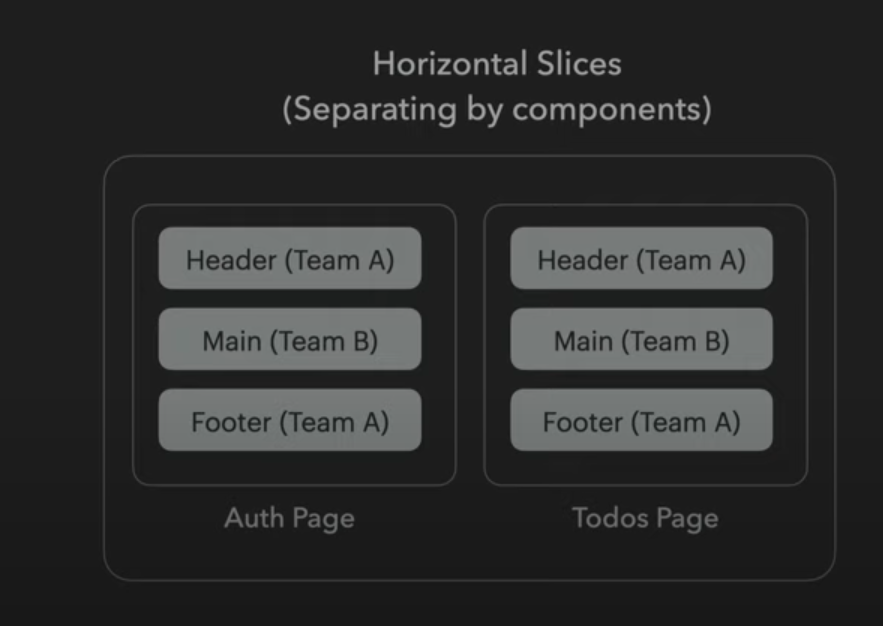
Each Module is sensible code module where developers or contributors can work independently with no to very less development dependency on another modules , the functional dependency may be their **.**

**Types of how to breakdown the applications:-**

1. **Vertical Slicing (Module Wise Distribution):-** The Vertical slicing means in this we break the application on the module basis.

****

1. **Horizontal Slicing(Component wise distribution)**

****

**Question 4) What are the usecases for the horizontal and vertical slicing ?**

* **Horizontal Slicing -**  This type of component differentiation is used for the applications where the application has many pages but with similar components with minor tweaks , like the ecommerce applications **(which has many same type components with minor tweaks)**
* **Vertical Slicing :** This type of component differentiation where the slicing is done based on module functionality like **AuthModule , UserModule and Main Module just for the example.**

It will be used generally for the complex applications which has more separate features which can be developed separately

As team can take any of the ways or a hybrid ways also which as both types of separations.

**Question 5) What are some of the pros and cons for the microfrontend architecture?**

* **The Pros of the Microfrontend Architecture:-**
  + **Easy and Fast Feature Development -**  As the modules are separated , each modules can develop the features with their own speed and fast feature development could be their
  + **Easy and Fast Deployment –** As modules are separated we can deploy the focus module separately which will be fast in nature rather than deploying whole application
  + **Multiple Scaling Strategies Can be used –** As the modules are now separated we can scale the each module or pod separately . if the scaling is required.

**The cons of the microfrontend Architecture:-**

* **Large Number of Tools and Pods handling –** As the services and modules increases the number of dependencies will also increase and we have to handle them each one of them , and it may become the straining issue.
* **Synchornization Issue –** Now as the services are being deployed independently their will be cases of synchronization issues , where one service the forward deployment ,and due to this the functional dependent services may have some issues
* **UX in compatibility –** As the several modules are being developed by different people and if the system is not strong and adherable their will always will be chances where their will be feature mismatch and adherence to bad practices between the modules