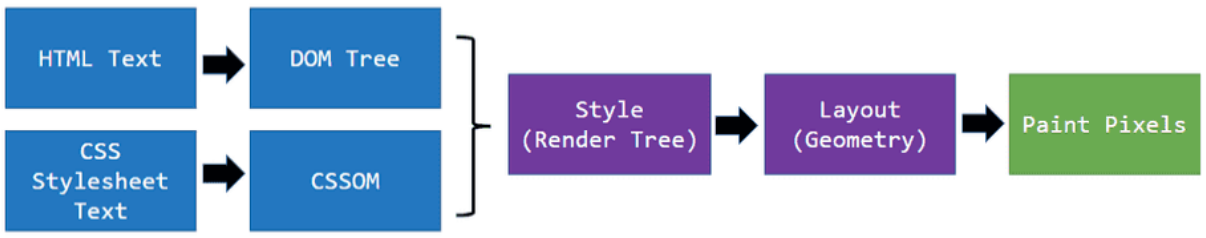
Assignment 1

1. **How does the browser parse and render HTML? Explain the rendering pipeline**



The browser's rendering pipeline involves several steps to convert HTML, CSS, and JavaScript into a visual display:

1. HTML Parsing to DOM Construction
   The browser reads the HTML document and convert it into Document Object Model (DOM) Tree
2. CSS Parsing to CSSOM Construction
   The CSS is parsed to create CSSOM (CSS Object Model) Tree
3. DOM Construction
   The DOM and CSSOM are combined to create a render tree, which represents the visible content on the page.
4. Layout
   The browser calculates the geometric layout of each element in the Render Tree, determining their size, position, and other visual properties.
5. Painting
   The browser paints the pixels on the screen, applying the styles and colors defined in the CSSOM to the elements in the Render Tree.

Source: https://webperf.tips/tip/browser-rendering-pipeline/

2. **What is the difference between position: relative and position: absolute in CSS?**

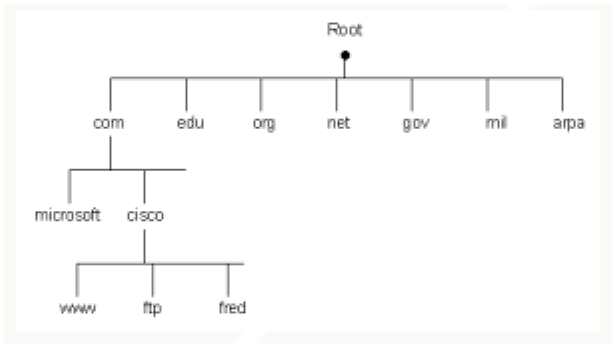
position: relative
- The element is positioned relative to its normal position
- It reserves space in the document flow.
- top, bottom, left, and right properties are used to offset the element from its original position.

Position: absolute
- The element is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- It is removed from the document flow and doesn't reserve space.

- top, bottom, left, and right properties are used to position the element relative to its containing element.

Source: https://www.w3schools.com/css/css_positioning.asp

3. **How does a browser handle DNS resolution when loading a webpage?**



Steps:
1. First user types a domain (example.com) into web browser and the query travels into the internet and is received by a DNS recursive resolver
2. The resolver then queries a DNS root nameserver (.), then the root server responds to the resolver with the address of Top Level Domain (TLD) DNS Server (.com, .net) which stores the information for its domain.
3. The TLD server responds with the IP address of the domain’s nameserver
4. Once the IP address is obtained, the browser can establish a TCP connection with the server

Source:
- DNS training at Endurance International Group (previous company)
- https://www.cloudflare.com/learning/dns/what-is-dns/

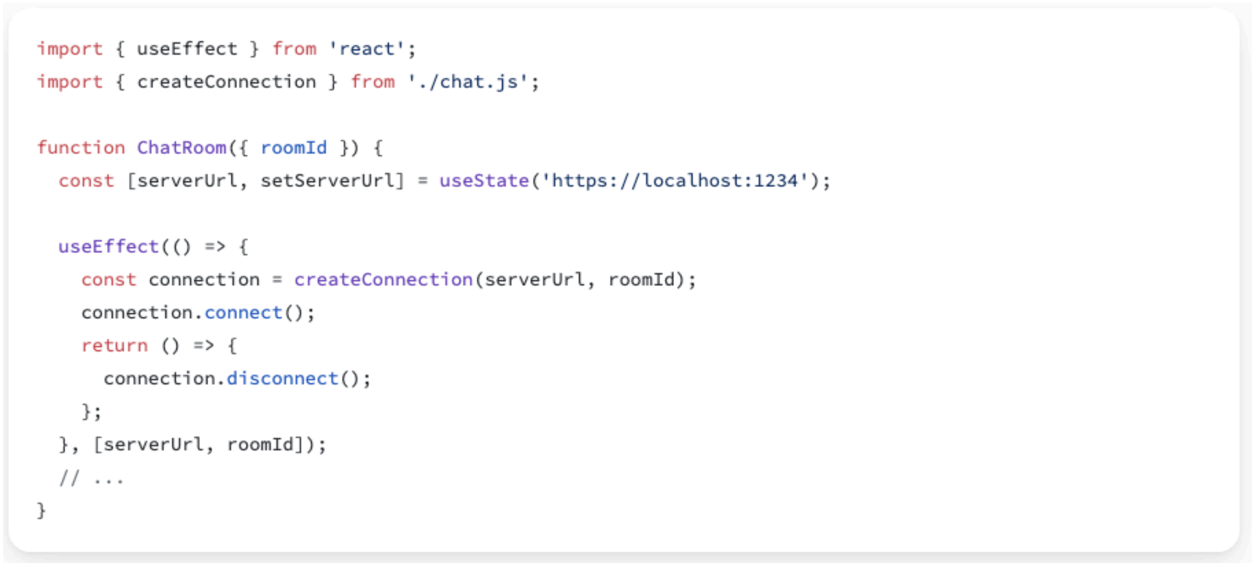
4. **What is CSRF (Cross-Site Request Forgery), and how do you defend against it?**
CSRF (Cross-Site Request Forgery) is an attack that forces end users to execute unwanted actions on web applications in which they’re currently authenticated.

To defend against it we can use:
1. Token Based Authentication, using unique tokens for each request and verifying them on the server
2. Enable same site cookie attribute, restricting cookie access to the originating site
3. HTTP Referer Header, checking the referer header to ensure request originate from the expected site

Source:
https://owasp.org/www-community/attacks/csrf

5. **Explain how React's useEffect hook works, and give an example of its usage.**
   The useEffect hook allows performing side effects in functional components. It uses two arguments, first effect callback and dependency array.
   1. Effect callback: A function containing the side effect to be executed
   2. Dependency array: Optional array of values that the effect depends on. If the values change, the effect is re-run.

   Example usage:



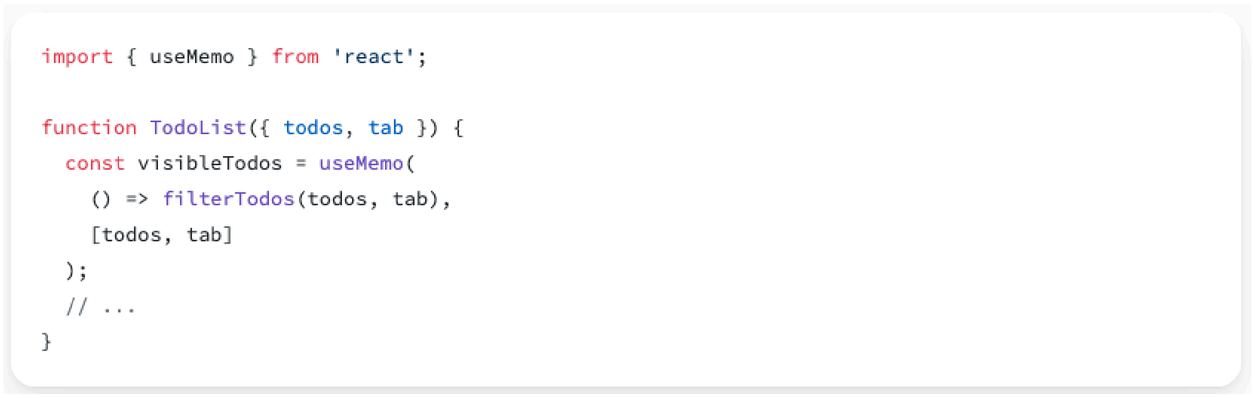
```
import { useEffect } from 'react';
import { createConnection } from './chat.js';

function ChatRoom({ roomId }) {
  const [serverUrl, setServerUrl] = useState('https://localhost:1234');

  useEffect(() => {
    const connection = createConnection(serverUrl, roomId);
    connection.connect();
    return () => {
      connection.disconnect();
    };
  }, [serverUrl, roomId]);
  // ...
}
```

   Source:
   https://react.dev/reference/react/useEffect

6. **UseMemo and useCallback are optimization hooks in React. In what situations should each be used, and what precautions should be taken? Please explain with specific examples.**
   - useMemo
     Memoize the result of a calculation or expensive function

     Situation should be used for expensive calculations so the page didn’t need to re-render (avoid unnecessary re-render)

     Example usage:



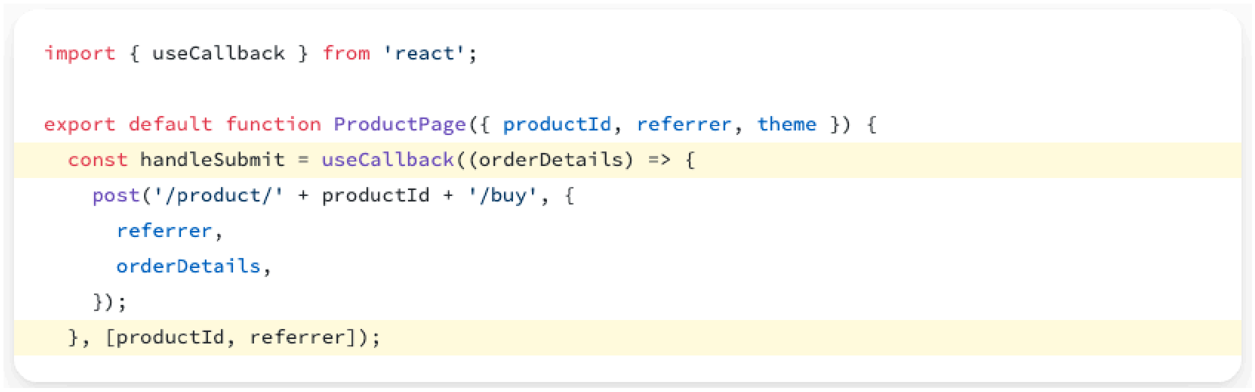
```
import { useMemo } from 'react';

function TodoList({ todos, tab }) {
  const visibleTodos = useMemo(
    () => filterTodos(todos, tab),
    [todos, tab]
  );
  // ...
}
```

Source: https://react.dev/reference/react/useMemo

- useCallback
  Memoize a function

  Situation should be used for unnecessary re-render when the function is triggered.

  Example usage:



```
import { useCallback } from 'react';

export default function ProductPage({ productId, referrer, theme }) {
  const handleSubmit = useCallback((orderDetails) => {
    post('/product/' + productId + '/buy', {
      referrer,
      orderDetails,
    });
  }, [productId, referrer]);
```

Source: https://react.dev/reference/react/useCallback

7. **Server-side rendering (SSR) and client-side rendering (CSR) in Next.js each have their own advantages and disadvantages. Explain the differences between them and discuss which type should be used for different applications, providing specific examples.**
   - SSR
     - Renders the page on the server and sends HTML to the client
     - Pros: Improved SEO, faster initial page load
     - Cons: Higher server load, more complex setup
     - Use cases: content-heavy website, marketing landing pages, e commerce site

   - CSR
     - Renders the initial HTML on the client
     - Pros: Less server load

    - Cons: Slower initial load, SEO challenges
    - Use cases: Single page applications(SPAs), interactive dashboard

    Source:https://www.geeksforgeeks.org/how-does-ssrserver-side-rendering-differ-from-csrclient-side-rendering/

8. **Explain Single responsibility principle, Open/closed principle.**
   Single responsibility principle: A class should have only one reason to change.

   Open/closed principle: Software entities (classes, modules, functions, etc.) should be open for extension but closed for modification.

   Source:
   https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

9. **Explain Scalability, Maintainability, Readability.**
   - Scalability: Ability to handle increased load (example: add users or features)
   - Maintainability: Ease of updating and fix the codebase
   - Readability: Ease which a system's code can be understood

10. **What is important when developing in a team? provide your own opinion with your concrete episode.**
    - Communication: Effective communication and regular team meetings are essential
    - Code Review: Peer reviews help maintain code quality and consistency
    - Version control: Version control system (Git) is vital for tracking changes and collaborating efficiently

    In a previous project I implemented git flow to ensure consistent commit messages across the team, improving collaboration and traceability.