1. `<link href="style.css" />` **vs** `@import("something.css")`

Both methods serve the same purpose: linking CSS files to an HTML document. However, `<link>` is preferred as it is faster and more efficient. It allows the browser to load CSS files concurrently with HTML. On the other hand, `@import` is slower as it delays CSS loading and blocks rendering of the page. It's generally better to use `<link>` in modern development.

2. **Spread and Rest**

Both spread and rest operators in JavaScript are denoted by `...`. The spread operator is used to unpack elements from an array or object, typically to pass them as arguments. The rest operator collects multiple elements into a single array. They look identical but serve opposite purposes in handling data.

3. **ES6 vs ES5 Features**

ES6, or ECMAScript 2015, introduced many modern features such as arrow functions, promises, template literals, and `let`/`const` for variable declaration. It also added modules, classes, and destructuring. ES5, the previous version, lacked these features and used `var` for variable declarations, along with callback-based handling for asynchronous code.

4. **How React Components React to Each Other**

In React, components can communicate via props and state. Parent components pass data to child components through props, and child components can send data back using callback functions. React's virtual DOM ensures efficient updates and renders only the parts of the UI that change.

5. **Communication Between Two Browser Tabs**

Tabs can communicate via the `localStorage` or `sessionStorage` events. One tab can store data in these storage mechanisms, and the other tab can listen for changes using the `storage` event. This allows two tabs to exchange information in real-time.

6. **rem vs em**

Both `rem` and `em` are relative units used for font sizes and layout styling in CSS. `rem` stands for "root em" and is based on the font size of the root element (usually `<html>`), making it consistent throughout the page. `em` is relative to the font size of its parent element, so it can change depending on its context.

7. **Closure**

A closure is a function that retains access to its lexical scope, even after the outer function has finished executing. This allows inner functions to "remember" the environment in which they were created, making them useful for creating private variables or functions.