1. **Consider the following code from React Router. What do you call :id in the path prop? <Route path="/:id" />**

A. This is a route modal

https://react-location.tanstack.com/guides/routes

B. This is a route parameter

C. This is a route splitter

D. This is a route link

1. **If a function component should always render the same way given the same props, what is a simple performance optimization available for it?**

A. Wrap it in the React.memo higher-order component.

B. Implement the useReducer Hook.

C. Implement the useRouter Hook.

D. Implement the shouldComponentUpdate lifecycle method.

https://reactjs.org/docs/react-api.html

React.memo is a higher order component.

If your component renders the same result given the same props, you can wrap it in a call to React.memo for a performance boost in some cases by memoizing the result. This means that React will skip rendering the component, and reuse the last rendered result.

1. **What can you use to handle code splitting?**

A. React.memo B. React.split C. React.lazy D. React.falback

https://web.dev/code-splitting-suspense/

1. **What is styled-component styling in React?**

A. These styles are written as attributes and are passed to the element.

B. It is a JavaScript library for styling React applications. It removes the mapping between styles and components, and lets you write actual CSS augmented with JavaScript.

C. It is basically a .css file that is compiled. When compiled, it produces two outputs. One is CSS that is a modified version of input CSS with the renamed class names. The other is a JavaScript object that maps the original CSS name with the renamed name.

D. It offers a different approach in which no CSS needs to be written to style an application. Instead, It uses utility classes for each CSS property that you can use directly in your HTML or JSX.

https://www.section.io/engineering-education/working-with-styled-components-in-react/

Styled-components is a library built for React and React Native developers. It allows you to use component-level styles in your applications. Styled-components leverage a mixture of JavaScript and CSS using a technique called

1. **What is [e.target.id] called in the following code snippet?**

*handleChange(e) { this.setState({ [e.target.id]: e.target.value }) }*

A.a computed property name

B.a set value

C.a dynamic key

D.a JSX code string

1. **What is sent to an Array.map() function?**

A.a callback function that is called once for each element in the array

B.the name of another array to iterate over

C.the number of times you want to call the function

D.a string describing what the function should do

1. **What do you need to change about this code to get it to run?**

***class clock extends React.Component {***

***render() { return <h1>Look at the time: {this.props.time}</h1>; } }***

**https://reactjs.org/docs/components-and-props.html#rendering-a-component**

**Note: Always start component names with a capital letter.**

React treats components starting with lowercase letters as DOM tags. For example, <div /> represents an HTML div tag, but <Welcome /> represents a component and requires Welcome to be in scope.

A.Add quotes around the return value

B.Remove this

C.Remove the render method

D.Capitalize clock

1. **How do you invoke setDone only when component mounts, using hooks?**

*function MyComponent(props) {*

*const [done, setDone] = useState(false);*

*return <h1>Done: {done}</h1>;}*

A. useEffect(() => { setDone(true); });

B. useEffect(() => { setDone(true); }, []);

C. useEffect(() => { setDone(true); }, [setDone]);

D. useEffect(() => { setDone(true); }, [done, setDone]);

1. **You have created a new method in a class component called handleClick, but it is not working. Which code is missing?**

…class Button extends React.Component{

… constructor(props) { super(props);

…// Missing line

<https://reactjs.org/docs/faq-functions.html>

class Foo extends Component {

constructor(props) {

super(props);

this.handleClick = this.handleClick.bind(this); }

handleClick() {

console.log('Click happened'); }

render() {

return <button onClick={this.handleClick}>Click Me</button>; }}

}

handleClick() {...}

}

A. this.handleClick.bind(this);

B. props.bind(handleClick);

C. this.handleClick.bind();

D. this.handleClick = this.handleClick.bind(this);

1. **What property do you need to add to the Suspense component in order to display a spinner or loading state**?

function MyComponent() {

https://reactjs.org/community/support.html

The fallback prop accepts any React elements that you want to render while waiting for the component to load. You can place the Suspense component anywhere above the lazy component. You can even wrap multiple lazy components with a single Suspense component.

return (

<Suspense>

<div>

<Message />

</div>

</Suspense>

);

}

A. lazy B. loading C. fallback D. Spinner

**Interview Questions 15m**

1. **What is the difference betw Element and Component?**

**React Elements:**

1) A React Element is what gets returned from components. It’s an object that virtually describes the DOM nodes that a component represents.

2) With a function component, this element is the object that the function returns.

3) With a class component, the element is the object that the component’s render function returns. R

4) React elements are not what we see in the browser. They are just objects in memory and we can’t change anything about them.

5) React elements can have other type properties other than native HTML elements.

6) A react element describes what we want to see on the screen.

7) A React element is an object representation of a DOM node.

8) It’s important to make this distinction here because the element is not the actual thing we see on the screen, rather the object representation is what is rendered.

9) A react component render() function returns a DOM tree of react elements behind the scenes (This is the virtual DOM btw). There is some complex mapping and diff logic involved, but basically these React elements map to the DOM elements.

**React Components:**

1) A component is a function or a Class which optionally accepts input and returns a React element.

2) A React Component is a template. A blueprint. A global definition. This can be either a function or a class (with a render function).

3) If react sees a class or a function as the first argument, it will check to see what element it renders, given the corresponding props and will continue to do this until there are no more createElement invocations which have a class or a function as their first argument.

4) When React sees an element with a function or class type, it will consult with that component to know which element it should return, given the corresponding props.

5) At the end of this processes, React will have a full object representation of the DOM tree. This whole process is called reconciliation in React and is triggered each time setState or ReactDOM.render is called.

6) React Element does not have any methods and nothing on the prototype. This also makes them fast.

7) A ReactElement is a light, stateless, immutable, virtual representation of a DOM Element

8) You can also create a Element directly React.createElement(arg) where arg can be a html tag name, or a React Component class.

**Class-Based Components**:

1) Class syntax is one of the most common ways to define a React component. While more verbose than the functional syntax, it offers more control in the form of lifecycle hooks.

2) We can render many instances of the same component.

3) The instance is the “this” keyword that is used inside the class-based component.

4) Is not created manually and is somewhere inside React’s memory.

**Function-Based Components:**

1) Do not have instances.

2) Can be rendered multiple times but React does not associate a local instance with each render.

3) React uses the invocation of the function to determine what DOM element to render for the function.

**React Öğeleri:**

1) React Element, bileşenlerden döndürülen şeydir. Bir bileşenin temsil ettiği DOM düğümlerini sanal olarak tanımlayan bir nesnedir.

2) Bir işlev bileşeni ile bu öğe, işlevin döndürdüğü nesnedir.

3) Bir sınıf bileşeniyle, öğe, bileşenin oluşturma işlevinin döndürdüğü nesnedir. r

4) React elementleri tarayıcıda gördüğümüz şeyler değil. Onlar sadece hafızadaki nesnelerdir ve onlar hakkında hiçbir şeyi değiştiremeyiz.

5) React öğeleri, yerel HTML öğelerinden başka tür özelliklerine sahip olabilir.

6) Bir tepki öğesi ekranda görmek istediğimiz şeyi açıklar.

7) Bir React öğesi, bir DOM düğümünün nesne temsilidir.

8) Burada bu ayrımı yapmak önemlidir, çünkü eleman ekranda gördüğümüz gerçek şey değil, nesne temsili işlenen şeydir.

9) Bir tepki bileşeni render() işlevi, sahne arkasındaki tepki öğelerinin bir DOM ağacını döndürür (Bu, sanal DOM btw'dir). Bazı karmaşık haritalama ve fark mantığı vardır, ancak temel olarak bu React öğeleri, DOM öğeleriyle eşlenir.

**React Bileşenleri:**

1) Bir bileşen, isteğe bağlı olarak girişi kabul eden ve bir React öğesi döndüren bir işlev veya Sınıftır.

2) Bir React Bileşeni bir şablondur. Bir plan. Küresel bir tanım. Bu, bir işlev veya bir sınıf olabilir (bir oluşturma işleviyle).

3) Reaksiyon ilk argüman olarak bir sınıf veya fonksiyon görürse, karşılık gelen donanımlar göz önüne alındığında hangi elemanı oluşturduğunu kontrol edecek ve şu şekilde bir sınıfa veya fonksiyona sahip olan createElement çağrıları kalmayıncaya kadar bunu yapmaya devam edecektir. onların ilk argümanı.

4) React, bir işleve veya sınıf tipine sahip bir öğe gördüğünde, karşılık gelen aksesuarlar verildiğinde hangi öğeyi döndürmesi gerektiğini öğrenmek için o bileşene danışacaktır.

5) Bu işlemlerin sonunda React, DOM ağacının tam bir nesne temsiline sahip olacaktır. Tüm bu sürece React'te mutabakat denir ve setState veya ReactDOM.render her çağrıldığında tetiklenir.

6) React Element'in prototipinde herhangi bir metod ve hiçbir şey yoktur. Bu da onları hızlı yapar.

7) Bir ReactElement, bir DOM Öğesinin hafif, durumsuz, değişmez, sanal bir temsilidir

8) Arg öğesinin bir html etiket adı veya bir React Component sınıfı olabileceği doğrudan React.createElement(arg) öğesini de oluşturabilirsiniz.

Class-Based Bileşenler:

1) Sınıf sözdizimi, bir React bileşenini tanımlamanın en yaygın yollarından biridir. İşlevsel söz diziminden daha ayrıntılı olsa da, yaşam döngüsü kancaları biçiminde daha fazla kontrol sunar.

2) Aynı bileşenin birçok örneğini oluşturabiliriz.

3) Örnek, sınıf tabanlı bileşen içinde kullanılan "this" anahtar sözcüğüdür.

4) Manuel olarak oluşturulmaz ve React'in hafızasında bir yerdedir.

Function-Based Bileşenler:

1) Örnekleri yok.

2) Birden çok kez oluşturulabilir ancak React, yerel bir örneği her bir oluşturma ile ilişkilendirmez.

3) React, işlev için hangi DOM öğesinin oluşturulacağını belirlemek için işlevin çağrılmasını kullanır.

**2. What is the difference between state and props?**

| **State** | **Props** |
| --- | --- |
| Should have an initial value. The initial value can be got from a parent component | Passed from a parent component. May be empty. We can set the default value for props if it is empty |
| Read and write | Read-only |
| Only the component that owns state can change it. State is private | The component can’t change incoming props. Only a parent is allowed to change props of the child component. |
| Make the component interactive for users | Make non-interactive components |
| Is also responsible for fetching remote data | Just display the incoming data |
| We can pass state as props to child components within the render method of the parent component | Pass information from component to component |
| Requires in higher-order components | Can provide the same functionality as higher-order components without using state and with fewer lines of code |

State

Bir başlangıç değerine sahip olmalıdır. Başlangıç değeri bir ana bileşenden alınabilir.

Oku ve yaz

Yalnızca duruma sahip olan bileşen onu değiştirebilir. Devlet özeldir

Bileşeni kullanıcılar için etkileşimli hale getirin

Uzak verilerin alınmasından da sorumludur

Ana bileşenin render yönteminde durumu alt bileşenlere props olarak iletebiliriz.

Daha yüksek dereceli bileşenlerde gerektirir

Porps

Bir üst bileşenden geçirildi. Boş olabilir. Eğer boşsa, props için varsayılan değeri ayarlayabiliriz.

Sadece oku

Bileşen, gelen aksesuarları değiştiremez. Yalnızca bir ebeveynin alt bileşenin aksesuarlarını değiştirmesine izin verilir.

Etkileşimli olmayan bileşenler yapın

Sadece gelen verileri göster

Bileşenden bileşene bilgi aktarın

Durum kullanmadan ve daha az kod satırıyla üst düzey bileşenlerle aynı işlevselliği sağlayabilir

1. **What is the difference between Shadow DOM and Virtual DOM?**

**Virtual DOM** is any kind of representation of a real DOM. Virtual DOM is about avoiding unnecessary changes to the DOM, which are expensive performance-wise, because changes to the DOM usually cause re-rendering of the page. It allows to collect several changes to be applied at once, so not every single change causes a re-render, but instead re-rendering only happens once after a set of changes was applied to the DOM.

Sanal DOM, gerçek bir DOM'nin her türlü temsilidir. Sanal DOM, DOM'de yapılan gereksiz değişikliklerden kaçınmakla ilgilidir; bu, performans açısından pahalıdır, çünkü DOM'de yapılan değişiklikler genellikle sayfanın yeniden oluşturulmasına neden olur. Aynı anda uygulanacak birkaç değişikliğin toplanmasına izin verir, bu nedenle her bir değişiklik yeniden oluşturmaya neden olmaz, bunun yerine yeniden oluşturma yalnızca DOM'ye bir dizi değişiklik uygulandıktan sonra gerçekleşir.

**Shadow DOM** is mostly about encapsulation of the implementation. A single custom element can implement more-or-less complex logic combined with more-or-less complex DOM. Shadow DOM refers to the ability of the browser to include a subtree of DOM elements into the rendering of a document, but not into the main document DOM tree.

Shadow DOM, çoğunlukla uygulamanın kapsüllenmesiyle ilgilidir. Tek bir özel öğe, az çok karmaşık DOM ile birlikte az çok karmaşık mantığı uygulayabilir. Gölge DOM, tarayıcının DOM öğelerinin bir alt ağacını bir belgenin oluşturulmasına dahil etme yeteneğini ifade eder, ancak ana belge DOM ağacına eklemez.

1. **What are the lifecycle methods of React?**

The three phases are: **Mounting, Updating, and Unmounting**.

**Mounting:** Mounting means putting elements into the DOM.

React has **four built-in methods**:

***constructor(),***

***getDerivedStateFromProps(),***

***render(),***

***componentDidMount()***

The render() method is required and will always be called, the others are optional and will be called if you define them.

**Updating:**

The next phase in the lifecycle is when a component is updated.

A component is updated whenever there is a change in the component's state or props.

React has five built-in methods that gets called, in this order, when a component is updated:

**getDerivedStateFromProps()**

**shouldComponentUpdate()**

**render()**

**getSnapshotBeforeUpdate()**

**componentDidUpdate()**

The render() method is required and will always be called, the others are optional and will be called if you define themconstructor()

**Unmounting**

The next phase in the lifecycle is when a component is removed from the DOM, or unmounting as React likes to call it.

React has **only one built-in method** that gets called when a component is unmounted:

componentWillUnmount(): The componentWillUnmount method is called when the component is about to be removed from the DOM.

**5. What is React Router?** React Router is a selection of navigation components, and is mainly used for handling page routing in the C#Bot client-side. It allows us to build a single-page web application with navigation, that doesn’t refresh each time the user navigates.

React Router, navigasyon bileşenlerinin bir seçimidir ve esas olarak C#Bot istemci tarafında sayfa yönlendirmeyi işlemek için kullanılır. Kullanıcı her gezindiğinde yenilenmeyen, navigasyonlu tek sayfalık bir web uygulaması oluşturmamıza olanak tanır.