**Please make sure you are able to answer the following questions:**

**Html:**

**What is doctype in html?**

* All HTML documents must start with a <!DOCTYPE> declaration. The declaration is not an HTML tag. It is an "information" to the browser about what document type to expect. The <!DOCTYPE> declaration is NOT case sensitive.
* (In older documents (HTML 4 or XHTML), the declaration is more complicated because the declaration must refer to a DTD (Document Type Definition).
* HTML 4.01:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

* XHTML 1.1:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">)

**What are new features in the html5?**

* New Semantic/Structural Elements: <article>, <aside>, <footer>, <header>, <nav>
* New Form Elements: <datalist>, <output>
* New Input Types: color, date, datetime, datetime-local, email, month, number, range, search, tel, time, url, week
* New Input Attributes: autocomplete, autofocus, form, formaction, formenctype, formmethod, formnovalidate, formtarget, height and width, list, min and max, multiple, pattern (regexp), placeholder, required, step
* HTML5 - New Attribute Syntax: Empty, Unquoted, Double-quoted, Single-quoted
* HTML5 Graphics: <canvas>, <picture>, <svg>
* New Media Elements: <audio>, <embed>, <picture>, <source>, <track>, <video>

**What is meta tag in the head tag?**

* Metadata is data (information) about data.
* <meta> tags always go inside the <head> element, and they provide metadata about the HTML document.
* <meta> tags are typically used to specify character set, page description, keywords, author of the document, and viewport settings.
* Metadata will not be displayed on the page, but is machine parsable.
* Metadata is used by browsers (how to display content or reload page), search engines (keywords), and other web services.
* There is a method to let web designers take control over the viewport (the user's visible area of a web page), through the <meta> tag (See "Setting The Viewport" example below).

**Css:**

**Explain me the position in the css?**

* The position property specifies the type of positioning method used for an element.
* There are five different position values: static, relative, fixed, absolute, sticky

**What is box model? Explain as more as u can**

* The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.

**What is specificity?**

* Specificity is a process of determining which CSS rule will be applied to an element. it actually determines which rules will take precedence.
* precedence(优先级为):   !important >inline style> #id > .class > tag > \* >继承 > 默认
* when the weight of selectors are same, it will apply the closet CSS rule. (就近原则)
* universal selector (\*) has no specificity.

**What are the differences between the pseudo elements and pseudo class?**

* pseudo class describe a special state. it allows to style element dynamically. The most popular one is :hover.
* pseudo-element matches virtual elements. it used to style specified parts of  an element.  we often use like " ::after ::before  ::first-line".

**How to achieve the RWD?**

* Responsive web design (RWD) makes web pages render well on all kinds of devices or screen sizes.
* 3rd party library: any design which is based on react application, styled component
* commonly used responsive design:  media query, flex box, grid layout

**How can we solve the cross-browser CSS issue?**

* used CSS reset style sheet to make sure every browser starts rendering with same basic set of rules.
* use third-party library to autocomplete "vendor specific css style", like -ms for IE, -moz for FireFox, -o for opera software, -webkit for Chrome.
* React supports most popular browsers, including Internet Explorer 9 and above.  but for the older browsers that don't support ES5 methods, we may find that some polyfills such as es5-shim and es5-sham.

**How can u center the element?**

* Center Align Elements: To horizontally center a block element (like <div>), use margin: auto;
* Center Align Text: To just center the text inside an element, use text-align: center;

## Center an Image: To center an image, set left and right margin to auto and make it into a block element;

**How can u hide the element?**

* Completely hiding elements can be done in 3 ways:
  + via the CSS property display, e.g. display: none;
  + via the CSS property visibility, e.g. visibility: hidden;
  + via the HTML5 attribute hidden, e.g. <span hidden>

**Js:**

**What will happen when we put the script tag inside the head tag in the html?**

* If one of our <scripts> fails to load, any content on our page physically lower than the failing <script> may not render or will error if we are trying to manipulate an element that doesn’t exist yet in the DOM. We could be left with a blank page with no HTML or styling! Also, if we are waiting for our <script> to download, the rest of the content on our page will have to wait to render until our <script> is loaded. It’s better practice to put our <script> elements before the closing </body> tag to make sure all our HTML content is read/rendered by the browser before trying to apply JavaScript.
* OR（Any script can insert its own HTML via document.write() or other DOM manipulations. This implies that the parser has to wait until the script has been downloaded & executed before it can safely parse the rest of the document. After all, the script could have inserted its own HTML in the document.
* However, most JavaScript developers no longer manipulate the DOM while the document is loading. Instead, they wait until the document has been loaded before modifying it. ）

**How does Js handle the sync and async operation?**

* JavaScript is always synchronous and single-threaded. If you're executing a JavaScript block of code on a page then no other JavaScript on that page will currently be executed.
* Async callbacks are functions that are specified as arguments when calling a function which will start executing code in the background. When the background code finishes running, it calls the callback function to let you know the work is done, or to let you know that something of interest has happened. Using callbacks is slightly old-fashioned now, but you'll still see them in use in a number of older-but-still-commonly-used APIs. Examples of an async callback is the second parameter of the addEventListener() method, and Promises

**pros and cons between using promise and callback?**

Promise Pros:

* Promise help developers avoid the ‘callback hell’. A piece of code that uses promises will be cleaner and  easy to understand.
* Promise provide catching mechanism (enhanced throw/catch mechanism.) which are not in callbacks.
* Promise also have good compatibility. it can support most of browser, unless IE 11 or old Android Browser.

Promise Cons:

* When in the Pending state, we can't know which stage is currently progressing. it's just started or it's about to complete (当处于Pending状态时，无法得知目前进展到哪一个阶段（刚刚开始还是即将完成）
* If you don't set the callback function, the error that is inside the Promise will not be reflected to the outside.(如果不设置回调函数，Promise内部抛出的错误，不会反应到外部)
* we can't cancel Promise, once it is created, it will be executed immediately and cannot be cancelled midway

Callback pros:  They are a well know pattern, they are easy to understand.

Callback cons:  when we have multiple nested levels. it will have callback hell.

**What will cause hoisting issue?**

* JavaScript only hoists declarations, not initializations.
* (Hoisting is JavaScript's default behavior of moving all declarations to the top of the current scope (to the top of the current script or the current function).
* Variables and constants declared with let or const are not hoisted!
* Hoisting is (to many developers) an unknown or overlooked behavior of JavaScript. If a developer doesn't understand hoisting, programs may contain bugs (errors). To avoid bugs, always declare all variables at the beginning of every scope. Since this is how JavaScript interprets the code, it is always a good rule. JavaScript in strict mode does not allow variables to be used if they are not declared.)
* <https://www.w3schools.com/js/js_hoisting.asp>

**Call, apply and bind?**

* You can use call()/apply() to invoke the function immediately. bind() returns a bound function that, when executed later, will have the correct context ("this") for calling the original function. So bind() can be used when the function needs to be called later in certain events when it's useful.
* call 需要把参数按顺序传递进去，而 apply 则是把参数放在数组里.( c for comma, a for array)

**Diff between Primitive data type and reference data type?**

* Primitive data types are: numbers, strings, booleans, undefined, and null, symbol, bigint
* That means the two reference types are objects
* Objects are mutable, meaning they can change (unless you freeze it). However, primitive types are immutable.

**1 + ‘3’ = ?, 3 - ‘2’ = ?, 33 + ‘5’ - ‘4’ + ’55’ - 0?**

* 1 + ‘3’ = ‘13’, string
* 3 - ‘2’ = 1, number
* 33 + ‘5’ - ‘4’ + ’55’ - 0=33155, number

**What is Event propagation?**

* Event propagation is a mechanism that defines how events propagate or travel through the DOM tree to arrive at its target and what happens to it afterward.
* (Event Bubling)

**How do u implement the shallow copy and deep copy?**

* Creating a new reference that points to the same memory location. This is also called a Shallow copy.
  + Shallow copy using …
  + Shallow copy using .slice()
  + Shallow copy using .assign()
  + Shallow copy arrays using Array.from()
* Creating a copy of object in a different memory location. This is called a Deep copy.
  + Deep copy with JSON.parse/stringify
  + Deep copy with lodash
  + Deep copy with Ramda
  + Deep copy with custom function
  + Really fast deep copy? Think rfdc

<https://medium.com/javascript-in-plain-english/how-to-deep-copy-objects-and-arrays-in-javascript-7c911359b089>

**What is Prototype in JS?**

* All JavaScript objects inherit properties and methods from a prototype.
  + Date objects inherit from Date.prototype
  + Array objects inherit from Array.prototype
  + Person objects inherit from Person.prototype
* The Object.prototype is on the top of the prototype inheritance chain:
* Date objects, Array objects, and Person objects inherit from Object.prototype.

**How do u delete the element from the array?**

* pop - Removes from the End of an Array
* shift - Removes from the beginning of an Array
* splice - removes from a specific Array index
* filter - allows you to programatically remove elements from an Array

**What is data storage?**

* Data storage is a term for how information is kept in a digital format that may be retrieved at a later time. Computers, laptops, tablets, smartphones, and other devices all store data. The methods and technologies used vary greatly, but the basic concept is always the same: information is kept so that it can be accessed again later.
* Web storage
  + LocalStorageis local storage for your browsers, it can save up to10MB.
  + SessionStorageis session based and will be deleted after closing your browser, also can save less than LocalStorage, like up to5MB
  + Cookiesare very tiny data storing in your browser, that can save up4KBand can be accessed through server or browser both.

**What is diff between null and undefined? console.log(typeof null)? console.log(typeof undefined)?**

* Null: has a value null.
* Undefined: empty value
* console.log(typeof null) //object
* console.log(typeof undefined) //undefined

**New features in the ES6?**

* Promise
* Let, const
* Class
* Arrow function
* Literal template: ``
* Destructuring: this.state… this,props…
* Set and Map: Set remove Doppelganger
* Generator

**React:**

**What are the main features of the react?**

* The Virtual DOM
* One-way Data Binding
* Components
* JSX
* Conditional Statements
* Lifecycle Methods
* Reusable
* <https://acowebs.com/react-js-features/#Features_of_ReactJS>

**How virtual Dom update the real Dom?**

* When new elements are added to the UI, a virtual DOM, which is represented as a tree is created. Each element is a node on this tree. If the state of any of these elements changes, a new virtual DOM tree is created. This tree is then compared or “diffed” with the previous virtual DOM tree. Once this is done, the virtual DOM calculates the best possible method to make these changes to the real DOM. This ensures that there are minimal operations on the real DOM. Hence, reducing the performance cost of updating the real DOM.
* <https://programmingwithmosh.com/react/react-virtual-dom-explained/>

**Walk me through all the knowledge of jsx u know?**

* JSX stands for JavaScript XML.
* JSX allows us to write HTML in React.
* JSX makes it easier to write and add HTML in React.
* JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods. JSX converts HTML tags into react elements.
* Expressions in JSX
  + With JSX you can write expressions inside curly braces { }.
  + The expression can be a React variable, or property, or any other valid JavaScript expression. JSX will execute the expression and return the result
* Inserting a Large Block of HTML
  + To write HTML on multiple lines, put the HTML inside parentheses
* One Top Level Element
  + The HTML code must be wrapped in ONE top level element.
  + So if you like to write two headers, you must put them inside a parent element, like a div element
* Elements Must be Closed
  + JSX follows XML rules, and therefore HTML elements must be properly closed.
* [**https://www.w3schools.com/react/react\_jsx.asp**](https://www.w3schools.com/react/react_jsx.asp)

**List some of the major advantages of react?**

* It facilitates the overall process of writing components
* It boosts productivity and facilitates further maintenance
* It ensures faster rendering
* <https://da-14.com/blog/its-high-time-reactjs-ten-reasons-give-it-try>
* React JS makes the process of writing components smoother

### React JS increases efficiency and makes maintenance easier

### React JS provides an ideal solution to high-load application

### React JS comes with useful developer tools

* <https://codecondo.com/most-powerful-features-of-react-js/>

**Why can’t browsers read jsx?**

* Browsers can only read JavaScript objects but JSX is not a regular JavaScript object. Thus to enable a browser to read JSX, first, we need to transform JSX file into a JavaScript object using JSX transformers like Babel and then pass it to the browser.

**Which lifecycle is mandatory?**

* render can return your JSX, render is like a body of my component.this is the only method that is mandatory to have. we can not use "setState" here otherwise you go in an infinite loop.

**Tell me all the lifecycle u familiar with?**

**A screenshot of a cell phone

Description automatically generated**

**Give me a example of how can u embed two or more components into one?**

* Import

**What are diffs between the state and props?**

* In a React component, props are variables passed to it by its parent component. State on the other hand is still variables, but directly initialized and managed by the component.

**Why should not update the state directly?**

* Basically, if you modify this.state directly, you create a situation where those modifications might get overwritten. Optimized components might not re-render if you do, and the rendering bugs will be tricky to track down.
* everything is object, only change the value, not the reference(address) which won’t cause re-render

**How do u understand refs in react?**

* React supports a special attribute that you can attach to any component. The ref attribute can be an object created by React.createRef() function or a callback function, or a string (in legacy API). When the ref attribute is a callback function, the function receives the underlying DOM element or class instance (depending on the type of element) as its argument. This allows you to have direct access to the DOM element or component instance.

**What are HOC? What can u do with HOC?**

* A high-order component is an techquie in React for reusing component logic.
* A high-order component is a function that takes a component and returns a new component with the behavior we want.
* The purpose of HOC is to enhance a component with extra functionality.

**Explain me what lifting state up in react?**

* When several components need to share the same changing data, then it is recommended to lift the shared state up to their closest common ancestor.
* That means if two child components share the same data from its parent, then move the state to parent instead of maintaining local state in both of the child components.

**Why do we prefer using id as keys than the index?**

* Key is unique
* Let’s say there are index [0, 1, 2, 3], if the values in index[2] is deleted, then the index 2 is still there, only the values of it has change to index[3], the original index[3] is deleted instead of index 2

**Do I need to keep all my state into redux?**

* There is no “right” answer for this. Some users prefer to keep every single piece of data in Redux, to maintain a fully serializable and controlled version of their application at all times. Others prefer to keep non-critical or UI state, such as “is this dropdown currently open”, inside a component's internal state.

**What are differents between controlled component and uncontrolled components**

* controlled component:
  + An input form element whose value is controlled by React is called a controlled component.
  + In a controlled component, form data is handled by a React component. (same, but in another words)
* uncontrolled components:
  + The alternative is uncontrolled components, where form data is handled by the DOM itself.
  + An uncontrolled component works like form elements do outside of React.(same, but in another words)

**Explain me the redux data flow?**

* Redux architecture revolves around a strict unidirectional data flow.
* This means that all data in an application follows the same lifecycle pattern, making the logic of your app more predictable and easier to understand. It also encourages data normalization, so that you don't end up with multiple, independent copies of the same data that are unaware of one another.

**Walk me through all the knowledge u know about actions and reducer?**

* Pure function, copy state.
* Reducers: As we already know, actions only tell what to do, but they don't tell how to do, so reducers are the pure functions that take the current state and action and return the new state and tell the store how to do.

**What is the middle ware? Why we need it?**

* Middleware allows for side effects to be run without blocking state updates.
* We can run side effects (like API requests) in response to a specific action, or in response to every action that is dispatched (like logging). There can be numerous middleware that an action runs through before ending in a reducer.

**What is react router? Why we want to use switch keyword? ﻿**

* React Router, and dynamic, client-side routing, allows us to build a single-page web application with navigation without the page refreshing as the user navigates. React Router uses component structure to call components, which display the appropriate information.
* The <Switch /> component will only render the first route that matches/includes the path. Once it finds the first route that matches the path, it will not look for any other matches. Not only that, it allows for nested routes to work properly, which is something that <Router /> will not be able to handle

**What kind of tool u used to do the test? What is coverage of test?**

* Jest is a JavaScript test runner that lets you access the DOM via jsdom. While jsdom is only an approximation of how the browser works, it is often good enough for testing React components. Jest provides a great iteration speed combined with powerful features like mocking modules and timers so you can have more control over how the code executes.
* LightHouse: a chrome extension. coverage: Performance, Accessibility, Best Practices, SEO, Progressive Web App

**Accessibility:**

**What is accessibility?**

* Web accessibility (also referred to as a11y) is the design and creation of websites that can be used by everyone.
* Accessibility support is necessary to allow assistive technology to interpret web pages.

**How many kind of values of tabindex?**

* 负值，0， 正值

**What is the aria attribute? List me some examples?**

* ARIA(Accessible Rich Internet Applications) is a set of attributes that define ways to make web content and web applications more accessible to people with disabilities.
* aria-label, aria-labelledby

**Explain me what the role is?**

* Role is used to define a type of user interface element. Once a role is set for an element, it does not change.

**Performance:**

**How can u increase the performance in react?**

* If performance is related in terms of speed, we need to profile the app and then run the app to see if which areas of the app is behaving badly, use shouldComponentUpdate to avoid unnecessary re-render.
* Use webpack dynamically to import lazy loading.
* Because react’s lazing loading does not support server side rendering.
* Use arrow function
* Debounce and throttling to prevent a function from being called multiple times.
* Use the Production build to minimize the size of the app
* setState is async function, if you update state multiple times, react will wait for event handling to finish before re-rendering, setState can reduce the times of virtual dom comparison