

# Frontend Quiz Preparation

Kristian Popov

12. Juli 2025

## Inhaltsverzeichnis

<b>1</b>	<b>CSS</b>	<b>2</b>
<b>2</b>	<b>JavaScript</b>	<b>8</b>
<b>3</b>	<b>Web Architecture</b>	<b>16</b>
<b>4</b>	<b>React</b>	<b>19</b>
<b>5</b>	<b>Answers</b>	<b>38</b>
5.1	CSS . . . . .	38
5.2	JavaScript . . . . .	40
5.3	Web Architecture . . . . .	49
5.4	React . . . . .	51

## 1 CSS

### Question 1

**What does “50vw” mean in CSS?**

- a. 50% of the content box
- b. 50% of the viewport’s width
- c. 50% of the parent element’s width
- d. 50 pixels

### Question 2

**What does `p.center` select?**

- a. All elements with both ID and class "`center`"
- b. All elements with the tag name "`center`"
- c. All `p` elements with class "`center`"
- d. All `p` elements with ID "`center`"

### Question 3

**What does the CSS selector `*` do?**

- a. It resets all browser styles
- b. It selects all elements with a class
- c. It selects all elements in the document
- d. It selects the first element

### Question 4

**What effect does increasing padding have on a box if `box-sizing: content-box` is used?**

- a. It has no visible effect
- b. Only the text gets smaller
- c. The total size of the element increases
- d. The border becomes thicker

**Question 5**

**What happens if an element is set to `display: none`;**

- a. It is hidden only on small screens
- b. It becomes invisible but still takes space
- c. It becomes transparent
- d. It is removed from the layout and not visible

**Question 6**

**What is a disadvantage of using inline styles?**

- a. They don't support media queries
- b. They reduce reusability and increase maintenance effort
- c. They always override external styles
- d. They are required for responsive design

**Question 7**

**What is the purpose of the `<link>` tag in the HTML `<head>`?**

- a. It links JavaScript functionality
- b. It applies internal styles
- c. It defines navigation links
- d. It connects an external CSS file to the HTML document

**Question 8**

**What makes inline-block different from inline?**

- a. It allows setting width and height
- b. It takes full width by default
- c. It is not affected by margin
- d. It always starts on a new line

**Question 9**

Where are internal CSS rules placed in an HTML document?

- a. Within an external CSS file
- b. In a `<script>` tag
- c. Inside a `<style>` block in the `<head>`
- d. Inside the `<body>` tag directly

**Question 10**

Which attribute in the `<link>` tag specifies the CSS file?

- a. href
- b. rel
- c. type
- d. src

**Question 11**

Which of the following is a correct CSS ID selector?

- a. `#main-header`
- b. `main-header`
- c. `.main-header`
- d. `id=main-header`

**Question 12**

Which rule would override all others (assuming all target the same element)?

- a. `#highlight { color: red; }`
- b. `.info { color: blue; }`
- c. `p { color: green; }`
- d. `div p { color: black; }`

**Question 13**

**Which selector has the highest specificity?**

- a. p
- b. .menu
- c. #main
- d. ul li a

**Question 14**

**Which statement is true about margin and padding?**

- a. Padding is inside the element's border, margin is outside
- b. They are visually identical
- c. Margin defines internal spacing, padding external
- d. Padding only works with inline elements

**Question 15**

**Which style declaration will override all others, assuming equal specificity?**

- a. External stylesheet
- b. Inline style
- c. User agent stylesheet
- d. Embedded style

**Question 16**

**Which of the following is NOT a flex container property?**

- a. flex-direction
- b. align-items
- c. justify-content
- d. flex-grow

**Question 17**

**What happens to a floated element?**

- a. It becomes inline-level and centered
- b. It is taken out of normal document flow
- c. It cannot contain any content

**Question 18**

**Which of the following CSS properties are inherited by default?**

- a. font-family
- b. border
- c. color
- d. margin

**Question 19**

**Which media query applies styles only on screens narrower than 800px?**

- a. @media (width: 800px)
- b. @media (min-width: 800px)
- c. @media (max-width: 800px)
- d. @media screen and (min-width: 801px)

**Question 20**

**Which value of position causes the element to scroll with the page?**

- a. absolute
- b. fixed
- c. static
- d. relative

**Question 21**

**What is the positioning context for an absolutely positioned element?**

- a. The body element regardless of positioning
- b. The nearest ancestor with a positioned value other than static
- c. The first ancestor with a class or ID
- d. The parent element if it has `display: block`

**Question 22**

**Which rule will be applied if both match the same element?**

- a. `.important { color: red !important; }`
- b. `div p { color: green; }`
- c. `#title { color: blue; }`

**Question 23**

**What makes `position: sticky` different from `fixed`?**

- a. Sticky elements scroll with the page until a threshold is reached
- b. Sticky elements ignore their container boundaries
- c. Sticky elements are always fixed to the viewport
- d. Sticky positioning is inherited by child elements

**Question 24**

**How do you fix a collapsing container that contains floated elements?**

**Question 25**

**What is the specificity of the selector `#main .highlight p`?**

## 2 JavaScript

### Question 1

**What is the difference between `async` and `defer` in loading external scripts?**

- a. `defer` cannot be used with external scripts.
- b. `async` scripts are guaranteed to execute in the order they appear in the document.
- c. Both `async` and `defer` delay the script download until after HTML parsing is complete.
- d. `async` scripts execute as soon as they are downloaded, potentially out of order.
- e. `defer` scripts wait for HTML parsing to finish before executing.

### Question 2

**Which statements about JavaScript execution are correct?**

- a. It is Just-In-Time compiled
- b. It is compiled ahead of time like C
- c. It is heavily optimized by modern browsers
- d. It is interpreted only with no compilation
- e. It must be compiled manually by the user

### Question 3

**Which ways can JavaScript be included in HTML?**

- a. Inline with `<script>`
- b. Inside `<style>` tags
- c. As a link to `.js`
- d. As an external file with `<script src="...">`
- e. In XML comments



**Question 4****What was the original purpose of JavaScript?**

- a. To build mobile apps
- b. To style HTML pages
- c. To manage databases
- d. To replace Java
- e. To add interactivity to web pages

**Question 5****Which statements about JavaScript functions are true?**

- a. Functions are always synchronous
- b. Functions are first-class objects
- c. Functions can be passed as arguments
- d. Functions must be declared before use
- e. Functions cannot be assigned to variables

**Question 6****What is a commonly used name for modern JavaScript versions starting from ES6?**

- a. DOMScript
- b. NextScript
- c. TypeScript
- d. JSX
- e. ES6+

**Question 7****Which model of object orientation does JavaScript use?**

- a. Trait-based
- b. Classical inheritance
- c. Aspect-oriented
- d. Functional-only
- e. Prototype-based

**Question 8****Who created JavaScript?**

- a. James Gosling
- b. Douglas Crockford
- c. Brendan Eich
- d. Tim Berners-Lee
- e. Bjarne Stroustrup

**Question 9****What was the original name of JavaScript?**

- a. Mocha
- b. LiveWire
- c. JavaLite
- d. ScriptEase
- e. CoffeeScript

**Question 10****What are Promises in JavaScript used for?**

- a. Defining variables
- b. Debugging
- c. Handling user input
- d. Error logging
- e. Asynchronous programming

**Question 11****Where can JavaScript code run?**

- a. In the browser
- b. In a Java Virtual Machine
- c. In a database engine
- d. Only in HTML files
- ✓ e. On the server using Node.js

**Question 12**

**JavaScript is standardized under which specification?**

- a. JavaSpec
- b. ECMAScript
- c. DOMSpec
- d. ISO-JavaScript
- e. WebScript

**Question 13**

**Which languages does JavaScript share syntactic similarities with?**

- a. Ruby
- b. Lisp
- c. Java
- d. Python
- e. C

**Question 14**

**Which statements about JavaScript's concurrency model are true?**

- a. JavaScript uses an event-driven model
- b. JavaScript uses blocking I/O
- c. JavaScript is single-threaded
- d. JavaScript supports native threads
- e. JavaScript uses multithreading

**Question 15**

**Which of the following best describe JavaScript's type system?**

- a. Type-safe
- b. Nominally typed
- c. Dynamically typed
- d. Strongly typed
- e. Statically typed

**Question 16**

**Which feature is unique to JavaScript compared to many other languages?**

- a. It allows prototype-based inheritance
- b. It compiles to machine code
- c. It requires semicolons
- d. It has static typing
- e. It uses manual memory management

**Question 17**

**Which of the following statements are true about the `defer` attribute in a `<script>` tag?**

- a. Scripts with `defer` maintain execution order.
- b. The script is executed after the HTML is parsed.
- c. The script blocks HTML parsing until it finishes loading.
- d. `defer` scripts are executed before `DOMContentLoaded` is fired.
- e. `defer` only works for inline scripts.

**Question 18**

**Which statements about `async/await` are true?**

- a. `await` works outside `async` functions
- b. `async` code is executed synchronously
- c. `await` can only be used globally
- d. `await` pauses execution inside `async` functions
- e. `async` functions return Promises

**Question 19**

**Why is 'callback hell' considered a problem?**

- a. It makes code harder to read
- b. It leads to deeply nested functions
- c. It causes memory leaks
- d. It improves performance
- e. It is required for asynchronous programming

**Question 20****What is true about the 'class' syntax in JavaScript?**

- a. Classes use prototypes under the hood
- b. Classes are compiled like in Java
- c. Classes cannot have private fields
- d. It is syntactic sugar for function constructors
- e. Only one class per file is allowed

**Question 21****What happens when you forget new with a constructor function?**

- a. It creates a global object
- b. It binds `this` correctly
- c. It may return undefined or cause an error
- d. The prototype chain is still applied
- e. It automatically uses `new`

**Question 22****Which statements about the JavaScript event loop are correct?**

- a. It moves tasks from the queue when the call stack is empty
- b. Microtasks are executed after all macrotasks
- c. It uses multithreading
- d. Microtasks are executed before macrotasks
- e. Tasks are executed in parallel

**Question 23****Which statements about the fetch API are true?**

- a. It returns data directly
- b. It replaces XMLHttpRequest
- c. It cannot be used with `async/await`
- d. It blocks the event loop
- e. It returns a Promise

**Question 24****How does inheritance work in JavaScript?**

- a. Subclasses can override inherited methods
- b. Only static methods are inherited
- c. It uses the prototype chain
- d. Objects must use mixins for inheritance
- e. It uses classical class-based inheritance

**Question 25****Which ways to parse JSON are valid?**

- a. `response.json()`
- b. `string.toJSON()`
- c. `JSON.parse(string)`
- d. `eval(string)`
- e. `fetch.parse()`

**Question 26****Which statements about JavaScript modules are correct?**

- a. import must always use \*
- b. Modules use export/import
- c. Modules can only be used with Node.js
- d. Default exports must be unique
- e. All exports must be default

**Question 27****Which of the following statements about objects in JavaScript are true?**

- a. Keys must be numbers
- b. Keys can be strings or Symbols
- c. Objects must be declared with class
- d. Methods are not allowed in objects
- e. Objects are collections of key-value pairs

**Question 28**

**How can private fields be defined in modern JavaScript classes?**

- a. By using 'private' keyword
- b. They are public by default and cannot be private
- c. With underscore '\_'
- d. Through constructor scoping
- e. By using the '#' syntax

**Question 29**

**What happens when a Promise is fulfilled?**

- a. It calls the `.then()` handler
- b. It immediately returns the value
- c. It creates a new thread
- d. It queues the handler as a microtask
- e. It blocks the call stack

**Question 30**

**Which are valid states of a JavaScript Promise?**

- a. Resolved and Failed
- b. Rejected
- c. Pending
- d. Executing
- e. Fulfilled

**Question 31**

**Which of the following are true about prototypes in JavaScript?**

- a. Prototype methods are shared between instances
- b. Prototypes cannot be used with classes
- c. All objects have a prototype
- d. Prototypes are used only in modules
- e. Prototypes must be defined manually

**Question 32****What is true about Web Workers?**

- a. They are executed in the same thread
- b. They handle UI rendering
- c. They cannot access the DOM
- d. They replace Promises
- e. They run in background threads

**3 Web Architecture****Question 1****What is true about Client-Side Rendering (CSR)?**

- a. The DOM is updated dynamically in the browser
- b. CSR means rendering happens on the server
- c. The browser never interacts with APIs
- d. The HTML is fully generated on the server
- e. JavaScript generates the content based on data

**Question 2****Which frameworks typically use client-side rendering?**

- a. React
- b. Django
- c. Vue
- d. Laravel
- e. Angular



**Question 3****Which comparison between CSR and SSR is accurate?**

- a. CSR puts more load on the client, SSR on the server
- b. SSR cannot support interactive web apps
- c. SSR can show content faster for slow devices
- d. CSR does not require a build step
- e. CSR is generally better for SEO without extra effort

**Question 4****Which of the following statements are true for client-side rendering (CSR)?**

- a. The browser uses JavaScript to dynamically generate content
- b. The server generates complete HTML pages for each request
- c. The browser updates the DOM after loading a minimal HTML file
- d. No JavaScript is needed at all for CSR
- e. The browser is only used for layouting, not logic

**Question 5****Which techniques can be used to pass data between web pages in plain JavaScript apps?**

- a. CSS variables
- b. localStorage
- c. URL parameters
- d. React context
- e. sessionStorage

**Question 6****Which of the following are advantages of Server-Side Rendering (SSR)?**

- a. SSR requires localStorage
- b. No need for a server
- c. Better initial load time
- d. Full interactivity before HTML arrives
- e. Content is visible without JavaScript enabled

**Question 7**

**Which of the following statements about Single Page Applications (SPA) are correct?**

- a. Each screen is rendered on the server
- b. JavaScript dynamically updates the content without reloading
- c. SPAs require a different browser than MPAs
- d. The browser only loads one HTML file initially
- e. Every navigation requires a full page reload

**Question 8**

**Which statements about Server-Side Rendering (SSR) are correct?**

- a. SSR requires a JavaScript frontend to be rendered
- b. The browser creates the initial HTML based on templates
- c. SSR apps cannot include any client-side logic
- d. SSR pages are usually visible faster on slow devices
- e. The HTML is generated on the server and sent to the browser

**Question 9**

**Which technologies or frameworks are commonly used for server-side rendering?**

- a. Laravel
- b. Vue (without additional frameworks)
- c. React (without additional frameworks)
- d. Next.js
- e. Django

## 4 React

### Question 1

**What is the role of Babel in React development?**

- a. It compiles JSX into JavaScript
- b. It replaces Webpack
- c. It manages state
- d. It renders the DOM
- e. It styles components

### Question 2

**What must a React component return?**

- a. Multiple root elements
- b. A CSS block
- c. A Promise
- d. A single root element (usually JSX)
- e. An HTML string

### Question 3

**Which are valid ways to define React components?**

- a. Arrow functions
- b. `defineComponent()`
- c. Function declarations
- d. XML templates
- e. CSS functions

### Question 4

**Which files are part of a standard Create React App structure?**

- a. `build/src.js`
- b. `App.jsx.html`
- c. `src/App.js`
- d. `App.vue`
- e. `public/index.html`

**Question 5**

**Which commands are used to create and start a React project with Create React App?**

- a. `react-start`
- b. `npm start`
- c. `npm create react-app`
- d. `npx create-react-app my-app`
- e. `babel-run`

**Question 6**

**How is data inserted into JSX expressions?**

- a. Using curly braces `{}`
- b. Using double quotes
- c. Using `{{ }}`
- d. Using backticks
- e. With a special tag named `bind`

**Question 7**

**What is the purpose of `public/index.html` in a React project?**

- a. It is dynamically generated by ReactDOM
- b. It defines all components
- c. It includes all JavaScript code
- d. It contains the `<div id="root">` for React to mount
- e. It provides the root HTML structure for the app

**Question 8**

**What is the function of `src/index.js`?**

- a. It initializes the React app
- b. It renders the App component
- c. It contains all CSS styles
- d. It defines the Virtual DOM
- e. It loads JSX files directly

**Question 9**

**Which tool is required to transform JSX for the browser?**

- a. Babel
- b. ReactDOM
- c. npm
- d. Webpack
- e. TypeScript

**Question 10**

**Which statements about JSX are true?**

- a. JSX must be transformed before it can run
- b. JSX looks like HTML but compiles to JavaScript
- c. JSX can only be used in Node.js
- d. JSX is valid JavaScript
- e. JSX replaces JavaScript syntax

**Question 11**

**Which of the following statements about React are true?**

- a. It is used only for server-side rendering
- b. It replaces HTML
- c. It is a JavaScript library for building UIs
- d. It is a CSS framework
- e. It uses a Virtual DOM to improve performance

**Question 12**

**How can React render elements without JSX?**

- a. With string interpolation
- b. Using document.write
- c. Using React.createElement
- d. With HTML templates
- e. By using Babel directly

**Question 13****Who developed React and when?**

- a. Microsoft, 2014
- b. Google, 2012
- c. Facebook, 2013
- d. Twitter, 2011
- e. Mozilla, 2015

**Question 14****What is `React.StrictMode` used for?**

- a. It enables Babel
- b. It disables JSX
- c. It enforces runtime checks in production
- d. It applies automatic code formatting
- e. It helps identify potential problems in an application during development

**Question 15****Which statements about the Virtual DOM are true?**

- a. React applies only differences to the real DOM
- b. It is an in-memory representation of the DOM
- c. It is slower than direct DOM manipulation
- d. It renders changes directly to HTML
- e. It uses Shadow DOM internally

**Question 16****Which statements are true about class-based components?**

- a. They are preferred over function components
- b. They use a `render()` method to return JSX
- c. They do not use props
- d. They cannot hold state
- e. They are required for all components

**Question 17**

**Which of the following statements about React components are true?**

- a. Components modify the DOM directly
- b. Components cannot be nested
- c. Components are reusable units of UI
- d. Components can receive props and hold state
- e. Components must be class-based

**Question 18**

**Which is a valid way to define a React component?**

- a. As an event listener
- b. With inline CSS
- c. Using a function that returns JSX
- d. Using a JSON object
- e. Using a for loop

**Question 19**

**What are advantages of component composition?**

- a. Eliminates props
- b. Prevents dynamic rendering
- c. Improves code reuse
- d. Requires class components
- e. Encourages modular design

**Question 20**

**Which are valid ways to implement conditional rendering in JSX?**

- a. Using if-else outside the return block
- b. Using ternary operator
- c. Using HTML conditions
- d. Using JSX if-then statements
- e. Using switch directly in JSX

**Question 21****How do you pass a parameter to an event handler in JSX?**

- a. Bind in `render()`
- b. Add param after function name
- c. Use `apply()`
- d. Call it directly
- e. Wrap it in an arrow function

**Question 22****How do you prevent a function from being called immediately in JSX?**

- a. Use `bind()` always
- b. Add return before call
- c. Use function reference: `onClick={myFunction}`
- d. Use parentheses: `onClick={myFunction()}`
- e. Use `eval()`

**Question 23****What is true about explicit composition in React?**

- a. Renders children automatically
- b. Only works with arrays
- c. Requires external libraries
- d. Child elements are passed directly in JSX
- e. Accessed using `props.children`

**Question 24****What is true about helper functions in components?**

- a. They can compute derived values
- b. They replace hooks
- c. They have access to props and local state
- d. They must be declared globally
- e. They cannot return JSX



**Question 25****What describes implicit composition in React?**

- a. Only works with `props.children`
- b. Used for rendering dynamic collections
- c. Parent component generates children via `map()`
- d. Child components must control layout
- e. Requires `useEffect`

**Question 26****What must you do when rendering lists with `map()` in JSX?**

- a. Use a while loop
- b. Avoid using functions
- c. Provide a unique key for each element
- d. Use for-of loops
- e. Wrap each item in

**Question 27****What are props in React?**

- a. HTML attributes
- b. Component methods
- c. Global variables
- d. Input data passed from parent to child components
- e. React internal state

**Question 28****How can you access props using destructuring?**

- a. Access via `props.name()`
- b. Use `( { name, age } )` in the function parameters
- c. Use `const { name, age } = this`
- d. Bind props manually
- e. Use `props[name]`

**Question 29****What does the `useState()` hook return?**

- a. A Promise
- b. An array with current state and a function to update it
- c. An event object
- d. A reference to DOM node
- e. A single state value

**Question 30****Which is the correct way to update state in a click handler?**

- a. Using a global counter
- b. Direct assignment like `count++`
- c. `setState(count + 1)` inside a function
- d. Mutating the DOM
- e. Calling `setState()` outside render

**Question 31****How does a child component communicate with its parent?**

- a. By calling a function received via props
- b. Using Redux only
- c. Using global variables
- d. Via `useRef`
- e. By accessing the parent's state directly

**Question 32****What is true about event bubbling?**

- a. Child handler is triggered before parent by default
- b. Events are synchronous
- c. Event propagation starts at root
- d. Events propagate from target to ancestors
- e. Only one handler can be called

**Question 33**

**Which statements about event capturing are correct?**

- a. It processes events from root to target
- b. It only works with hooks
- c. It disables bubbling
- d. It ignores `stopPropagation`
- e. It requires a third argument set to `true`

**Question 34**

**Which statements about the `fetch` API in React are true?**

- a. It can be used inside `useEffect` to retrieve data
- b. It requires Redux
- c. It runs synchronously
- d. It directly updates the DOM
- e. It returns a `Promise`

**Question 35**

**How can a form input be reset after submission?**

- a. Using `innerHTML = ''`
- b. Reloading the page
- c. Calling `setTimeout()`
- d. Removing the input element
- e. By setting its state value to an empty string

**Question 36**

**How is form input typically handled in React?**

- a. Using global variables
- b. Using `useState` to bind input values
- c. Updating state on each `onChange` event
- d. Setting values via `innerHTML`
- e. Directly modifying DOM

**Question 37**

**Which is a correct structure for a controlled React form component?**

- a. Input without any event handling
- b. Input with value bound to state and `onChange` updating state
- c. `innerHTML` binding
- d. Form using `ref` as default
- e. Global event listener on `window`

**Question 38**

**Which of the following can be used for form validation?**

- a. HTML attributes like `required` and `minlength`
- b. CSS media queries
- c. JSX attributes only
- d. Custom validation logic in `handleSubmit`
- e. `React.StrictMode`

**Question 39**

**What is 'lifting state up' in React?**

- a. Moving state to a common parent component
- b. Binding refs
- c. Passing props downward
- d. Creating CSS modules
- e. Using session storage

**Question 40**

**What is the typical purpose of `onSubmit` in React forms?**

- a. To handle form submission and prevent default behavior
- b. To validate HTML structure
- c. To access Redux store
- d. To bypass event bubbling
- e. To reset the entire app

**Question 41**

**What happens when `event.stopPropagation()` is called?**

- a. The event handler is removed
- b. The event is canceled
- c. The DOM is re-rendered
- d. All child events are ignored
- e. Further propagation of the event is stopped

**Question 42**

**What are common use cases for the `useEffect` hook?**

- a. Rendering JSX directly
- b. Running before DOM updates
- c. Handling synchronous updates
- d. Performing side effects like data fetching
- e. Reacting to changes in props or state

**Question 43**

**What does the second argument of `useEffect` control?**

- a. When the effect function is re-executed
- b. What values are returned
- c. How JSX is rendered
- d. Which variables are global
- e. Whether props are required

**Question 44**

**What happens if you pass an empty array as the dependency list to `useEffect`?**

- a. It runs before render
- b. The effect runs only after the first render
- c. It never runs
- d. It causes an error
- e. It re-runs on every state change

**Question 45**

**What is useRef commonly used for in forms?**

- a. To create global variables
- b. To bind event listeners
- c. To trigger re-renders
- d. To access DOM elements directly
- e. To store component state

**Question 46**

**How do you correctly update an object state in Context API with useState?**

- a. By directly changing properties
- b. By using `Object.assign` in-place
- c. By using `push()` or `+=` operators
- d. By calling `setTimeout`
- e. By using the spread syntax to create a new object

**Question 47**

**Which state fields are typically used in Redux for async operations?**

- a. `error`
- b. `HTMLString`
- c. `loading`
- d. `responseTime`
- e. `retryCount`
- f. `data` or `user`

**Question 48**

**What does `dispatch()` return when used with a thunk?**

- a. The current component
- b. A `Promise` if the thunk is async
- c. Nothing
- d. An action object
- e. A DOM node

**Question 49**

**Why is direct mutation of state allowed in Redux reducers using Redux Toolkit?**

- a. Because Redux Toolkit uses `Immer` to track changes
- b. Because Redux ignores immutability
- c. Because reducers are only used for setup
- d. Because JavaScript allows it
- e. `Immer` converts mutations into immutable updates internally

**Question 50**

**What is the purpose of middleware in Redux?**

- a. To render UI
- b. To intercept actions and add behavior
- c. To handle side effects like logging or async calls
- d. To define routes
- e. To update CSS

**Question 51**

**Which of the following is a correct way to write a Redux reducer using Redux Toolkit?**

- a. Using `event.preventDefault()`
- b. Directly mutating the state (e.g., `state.count += 1`)
- c. Modifying props
- d. Returning a modified state manually
- e. Calling `setState`

**Question 52**

**What is a core difference between `useState` and Redux reducers?**

- a. `useState` must be global
- b. Reducers are only called once
- c. Redux never stores state
- d. `useState` updates require returning a new object
- e. Redux reducers with Immer allow mutation-style syntax

**Question 53**

**How would a 'Retry' button typically work in Redux?**

- a. It forces a React render
- b. It resets the component
- c. It reloads the entire page
- d. It dispatches the same async thunk again
- e. It clears all reducers

**Question 54**

**Where should API calls typically happen in a Redux app?**

- a. Inside reducers
- b. Directly in components
- c. Inside JSX expressions
- d. In the `index.js` file
- e. Inside thunks or middleware

**Question 55**

**How should you structure Redux state for a remote fetch?**

- a. Store HTML inside state
- b. Separate keys for loading, error, and data
- c. Only store raw response
- d. Use refs instead of state
- e. A single key with a long string



**Question 56**

**In what order are actions usually dispatched in a thunk for async requests?**

- a. Start → Success or Failure
- b. Only one action is dispatched
- c. Success → Start → Failure
- d. Failure → Retry → Start
- e. Start → End → Retry

**Question 57**

**What does a typical Redux Thunk do?**

- a. Modify the DOM manually
- b. Inject middleware automatically
- c. Dispatch multiple actions depending on async outcome
- d. Return HTML
- e. Render the React component directly

**Question 58**

**Which statements about Redux Thunks are true?**

- a. They allow dispatching asynchronous logic
- b. They return a function instead of an action object
- c. They mutate reducers
- d. They replace all reducers
- e. They require Context API

**Question 59**

**What are key benefits of using Redux Toolkit?**

- a. Built-in support for `createAsyncThunk`
- b. Simplified reducer logic with Immer
- c. Automatic UI testing
- d. Global CSS injection
- e. Less boilerplate code

**Question 60**

**What does `createAsyncThunk` help you do?**

- a. Generate async thunks with built-in action types
- b. Bind input fields
- c. Validate forms
- d. Replace `useEffect`
- e. Automatically create pending, fulfilled, and rejected actions

**Question 61**

**Which path is used to match all unknown routes?**

- a. `*`
- b. `/error`
- c. `**`
- d. `/notfound`

**Question 62**

**What component must wrap your entire React app to enable routing?**

- a. `RouteManager`
- b. `BrowserRouter`
- c. `RouteProvider`
- d. `AppRouter`

**Question 63**

**In a nested route, how is the default child defined?**

- a. `path=/element`
- b. `element={<Default />}`
- c. index element
- d. fallback element

**Question 64**

**Which of the following correctly defines a route to a LoginPage component?**

- a. `<Router path="/login" component={<LoginPage />}> />`
- b. `<Route path="/login">{<LoginPage />}</Route>`
- c. `<Route path="/login" element={<LoginPage />}> />`
- d. `<Route url="/login" render={<LoginPage />}> />`

**Question 65**

**How do you define a dynamic segment in a React Router path?**

- a. Using dollar sign, like \$id
- b. Using curly braces, like {id}
- c. Using a colon, like :id
- d. Using angle brackets, like <id>

**Question 66**

**Why might you use HashRouter instead of BrowserRouter?**

- a. It doesn't require server configuration for route handling
- b. It supports more modern features
- c. It allows usage of cookies
- d. It's the only router that works with JSX

**Question 67**

**What API does BrowserRouter rely on?**

- a. Window Navigation API
- b. React Context API
- c. HTML5 History API
- d. Session API

**Question 68**

**Which route element ensures that an invalid URL renders a fallback?**

- a. Route with fallback
- b. Route with no path
- c. ErrorBoundary
- d. Route with path=\*

**Question 69**

**What happens when you click a Link component in React Router?**

- a. An iframe opens the target page
- b. The URL changes without full page reload
- c. An anchor tag redirects to the route
- d. The server reloads the route from scratch

**Question 70**

**Which component provides navigation between routes in React Router?**

- a. Anchor
- b. Link
- c. NavLink
- d. Redirect

**Question 71**

**Which component is used as a placeholder for nested route content?**

- a. Placeholder
- b. Outlet
- c. Switch
- d. Content

**Question 72**

**What is the purpose of a `ProtectedRoute` component?**

- a. To handle 404 errors
- b. To define admin-only pages
- c. To cache route components
- d. To restrict access to certain routes based on authentication

**Question 73**

**What React Router hook is used to read URL parameters?**

- a. `useLocation`
- b. `useRoute`
- c. `useQuery`
- d. `useParams`

**Question 74**

**Which component is used to redirect the user to another path?**

- a. `RedirectTo`
- b. `RouterRedirect`
- c. `HistoryPush`
- d. `Navigate`

**Question 75**

**What type of data is returned by `useParams()`?**

- a. A query string
- b. An array of parameters
- c. A route object
- d. An object of strings

## 5 Answers

### 5.1 CSS

#### Question 1

What does **50vw** mean in CSS?

- ✓ a. 50% of the viewport's width
- ✗ b. 50% of the parent element's width
- ✗ c. 50% of the content box
- ✗ d. 50 pixels

#### Question 2

Correct answer: **c.** All **p** elements with class "**center**"

#### Question 3

Correct answer: **c.** It selects all elements in the document

#### Question 4

Correct answer: **c.** The total size of the element increases

#### Question 5

Correct answer: **d.** It is removed from the layout and not visible

#### Question 6

Correct answer: **b.** They reduce reusability and increase maintenance effort

#### Question 7

Correct answer: **d.** It connects an external CSS file to the HTML document

#### Question 8

What makes **inline-block** different from **inline**?

- ✗ a. It is not affected by margin
- ✗ b. It always starts on a new line
- ✗ c. It takes full width by default
- ✓ d. It allows setting width and height

**Question 9**

Correct answer: **c.** Inside a `<style>` block in the `<head>`

**Question 10**

Correct answer: **a.** href

**Question 11**

Correct answer: **a.** #main-header

**Question 12**

Correct answer: **a.** #highlight { color: red; }

**Question 13**

Correct answer: **c.** #main

**Question 14**

Correct answer: **a.** Padding is inside the element's border, margin is outside

**Question 15**

Correct answer: **b.** Inline style

**Question 16**

**Which of the following is NOT a flex container property?**

- ☒ a. justify-content
- ☒ b. flex-grow (It is a property applied to individual flex items)
- ☒ c. flex-direction
- ☒ d. align-items

**Question 17**

**What happens to a floated element?**

- ☒ a. It becomes inline-level and centered
- ☒ b. It is taken out of normal document flow
- ☒ c. It cannot contain any content

**Question 18**

Which of the following CSS properties are inherited by default?

- ☒ a. margin
- ☒ b. border
- ☒ c. color
- ☒ c. font-family

**Question 19**

Correct answer: **c.** @media (max-width: 800px)

**Question 20**

Correct answer: **d.** relative

**Question 21**

Correct answer: **b.** The nearest ancestor with a positioned value other than static

**Question 22**

Correct answer: **a.** .important { color: red !important; }

**Question 23**

Correct answer: **a.** Sticky elements scroll with the page until a threshold is reached

**Question 24**

Correct answer: **clearfix**

**Question 25**

What is the specificity of the selector #main .highlight p?

Correct answer: **1,1,1**

## 5.2 JavaScript

**Question 1**

What is the difference between async and defer in loading external scripts?

- ☒ a. defer cannot be used with external scripts.



- ☒ b. **async** scripts are guaranteed to execute in the order they appear in the document.
- ☒ c. Both **async** and **defer** delay the script download until after HTML parsing is complete.
- ☒ d. **async** scripts execute as soon as they are downloaded, potentially out of order.
- ☒ e. **defer** scripts wait for HTML parsing to finish before executing.

**Question 2**

**Which statements about JavaScript execution are correct?**

- ☒ a. It is Just-In-Time compiled
- ☒ b. It is compiled ahead of time like C
- ☒ c. It is heavily optimized by modern browsers
- ☒ d. It is interpreted only with no compilation
- ☒ e. It must be compiled manually by the user

**Question 3**

**Which ways can JavaScript be included in HTML?**

- ☒ a. Inline with `<script>`
- ☒ b. Inside `<style>` tags
- ☒ c. As a link to `.js`
- ☒ d. As an external file with `<script src="...">`
- ☒ e. In XML comments

**Question 4**

**What was the original purpose of JavaScript?**

- ☒ a. To build mobile apps
- ☒ b. To style HTML pages
- ☒ c. To manage databases
- ☒ d. To replace Java
- ☒ e. To add interactivity to web pages

**Question 5****Which statements about JavaScript functions are true?**

- ☒ a. Functions are always synchronous
- ☒ b. Functions are first-class objects
- ☒ c. Functions can be passed as arguments
- ☒ d. Functions must be declared before use
- ☒ e. Functions cannot be assigned to variables

**Question 6****What is a commonly used name for modern JavaScript versions starting from ES6?**

- ☒ a. DOMScript
- ☒ b. NextScript
- ☒ c. TypeScript
- ☒ d. JSX
- ☒ e. ES6+

**Question 7****Which model of object orientation does JavaScript use?**

- ☒ a. Trait-based
- ☒ b. Classical inheritance
- ☒ c. Aspect-oriented
- ☒ d. Functional-only
- ☒ e. Prototype-based

**Question 8****Who created JavaScript?**

- ☒ a. James Gosling
- ☒ b. Douglas Crockford
- ☒ c. Brendan Eich
- ☒ d. Tim Berners-Lee
- ☒ e. Bjarne Stroustrup

**Question 9****What was the original name of JavaScript?**

- ✓ a. Mocha
- ✗ b. LiveWire
- ✗ c. JavaLite
- ✗ d. ScriptEase
- ✗ e. CoffeeScript

**Question 10****What are Promises in JavaScript used for?**

- ✗ a. Defining variables
- ✗ b. Debugging
- ✗ c. Handling user input
- ✗ d. Error logging
- ✓ e. Asynchronous programming

**Question 11****Where can JavaScript code run?**

- ✓ a. In the browser
- ✗ b. In a Java Virtual Machine
- ✗ c. In a database engine
- ✗ d. Only in HTML files
- ✓ e. On the server using Node.js

**Question 12****JavaScript is standardized under which specification?**

- ✗ a. JavaSpec
- ✓ b. ECMAScript
- ✗ c. DOMSpec
- ✗ d. ISO-JavaScript
- ✗ e. WebScript

**Question 13**

**Which languages does JavaScript share syntactic similarities with?**

- ☐ a. Ruby
- ☐ b. Lisp
- ☒ c. Java
- ☐ d. Python
- ☒ e. C

**Question 14**

**Which statements about JavaScript's concurrency model are true?**

- ☒ a. JavaScript uses an event-driven model
- ☐ b. JavaScript uses blocking I/O
- ☒ c. JavaScript is single-threaded
- ☐ d. JavaScript supports native threads
- ☐ e. JavaScript uses multithreading

**Question 15**

**Which of the following best describe JavaScript's type system?**

- ☐ a. Type-safe
- ☐ b. Nominally typed
- ☒ c. Dynamically typed
- ☐ d. Strongly typed
- ☐ e. Statically typed

**Question 16**

**Which feature is unique to JavaScript compared to many other languages?**

- ☒ a. It allows prototype-based inheritance
- ☐ b. It compiles to machine code
- ☐ c. It requires semicolons
- ☐ d. It has static typing
- ☐ e. It uses manual memory management

**Question 17**

**Which of the following statements are true about the defer attribute in a <script> tag?**

- ✓ a. Scripts with `defer` maintain execution order.
- ✓ b. The script is executed after the HTML is parsed.
- ✗ c. The script blocks HTML parsing until it finishes loading.
- ✓ d. `defer` scripts are executed before `DOMContentLoaded` is fired.
- ✗ e. `defer` only works for inline scripts.

**Question 18**

**Which statements about `async/await` are true?**

- ✗ a. `await` works outside `async` functions
- ✗ b. `async` code is executed synchronously
- ✗ c. `await` can only be used globally
- ✓ d. `await` pauses execution inside `async` functions
- ✓ e. `async` functions return Promises

**Question 19**

**Why is 'callback hell' considered a problem?**

- ✓ a. It makes code harder to read
- ✓ b. It leads to deeply nested functions
- ✗ c. It causes memory leaks
- ✗ d. It improves performance
- ✗ e. It is required for asynchronous programming

**Question 20**

**What is true about the 'class' syntax in JavaScript?**

- ✓ a. Classes use prototypes under the hood
- ✗ b. Classes are compiled like in Java
- ✗ c. Classes cannot have private fields
- ✓ d. It is syntactic sugar for function constructors
- ✗ e. Only one class per file is allowed

**Question 21****What happens when you forget `new` with a constructor function?**

- ☒ a. It creates a global object
- ☒ b. It binds `this` correctly
- ☒ c. It may return undefined or cause an error
- ☒ d. The prototype chain is still applied
- ☒ e. It automatically uses `new`

**Question 22****Which statements about the JavaScript event loop are correct?**

- ☒ a. It moves tasks from the queue when the call stack is empty
- ☒ b. Microtasks are executed after all macrotasks
- ☒ c. It uses multithreading
- ☒ d. Microtasks are executed before macrotasks
- ☒ e. Tasks are executed in parallel

**Question 23****Which statements about the fetch API are true?**

- ☒ a. It returns data directly
- ☒ b. It replaces XMLHttpRequest
- ☒ c. It cannot be used with `async/await`
- ☒ d. It blocks the event loop
- ☒ e. It returns a Promise

**Question 24****How does inheritance work in JavaScript?**

- ☒ a. Subclasses can override inherited methods
- ☒ b. Only static methods are inherited
- ☒ c. It uses the prototype chain
- ☒ d. Objects must use mixins for inheritance
- ☒ e. It uses classical class-based inheritance

**Question 25****Which ways to parse JSON are valid?**

- ✓ a. `response.json()`
- ✗ b. `string.toJSON()`
- ✓ c. `JSON.parse(string)`
- ✗ d. `eval(string)`
- ✗ e. `fetch.parse()`

**Question 26****Which statements about JavaScript modules are correct?**

- ✗ a. import must always use `*`
- ✓ b. Modules use `export/import`
- ✗ c. Modules can only be used with Node.js
- ✓ d. Default exports must be unique
- ✗ e. All exports must be default

**Question 27****Which of the following statements about objects in JavaScript are true?**

- ✗ a. Keys must be numbers
- ✓ b. Keys can be strings or Symbols
- ✗ c. Objects must be declared with `class`
- ✗ d. Methods are not allowed in objects
- ✓ e. Objects are collections of key-value pairs

**Question 28****How can private fields be defined in modern JavaScript classes?**

- ✗ a. By using `'private'` keyword
- ✗ b. They are public by default and cannot be private
- ✗ c. With underscore `'_'`
- ✗ d. Through constructor scoping
- ✓ e. By using the `'#'` syntax

**Question 29****What happens when a Promise is fulfilled?**

- ✓ a. It calls the `.then()` handler
- ✗ b. It immediately returns the value
- ✗ c. It creates a new thread
- ✓ d. It queues the handler as a microtask
- ✗ e. It blocks the call stack

**Question 30****Which are valid states of a JavaScript Promise?**

- ✗ a. Resolved and Failed
- ✓ b. Rejected
- ✓ c. Pending
- ✗ d. Executing
- ✓ e. Fulfilled

**Question 31****Which of the following are true about prototypes in JavaScript?**

- ✓ a. Prototype methods are shared between instances
- ✗ b. Prototypes cannot be used with classes
- ✓ c. All objects have a prototype
- ✗ d. Prototypes are used only in modules
- ✗ e. Prototypes must be defined manually

**Question 32****What is true about Web Workers?**

- ✗ a. They are executed in the same thread
- ✗ b. They handle UI rendering
- ✓ c. They cannot access the DOM
- ✗ d. They replace Promises
- ✓ e. They run in background threads



### 5.3 Web Architecture

#### Question 1

**What is true about Client-Side Rendering (CSR)?**

- ✓ a. The DOM is updated dynamically in the browser
- ✗ b. CSR means rendering happens on the server
- ✗ c. The browser never interacts with APIs
- ✗ d. The HTML is fully generated on the server
- ✓ e. JavaScript generates the content based on data

#### Question 2

**Which frameworks typically use client-side rendering?**

- ✓ a. React
- ✗ b. Django
- ✓ c. Vue
- ✗ d. Laravel
- ✓ e. Angular

#### Question 3

**Which comparison between CSR and SSR is accurate?**

- ✓ a. CSR puts more load on the client, SSR on the server
- ✗ b. SSR cannot support interactive web apps
- ✓ c. SSR can show content faster for slow devices
- ✗ d. CSR does not require a build step
- ✗ e. CSR is generally better for SEO without extra effort

#### Question 4

**Which of the following statements are true for client-side rendering (CSR)?**

- ✓ a. The browser uses JavaScript to dynamically generate content
- ✗ b. The server generates complete HTML pages for each request
- ✓ c. The browser updates the DOM after loading a minimal HTML file
- ✗ d. No JavaScript is needed at all for CSR
- ✗ e. The browser is only used for layouting, not logic

**Question 5**

**Which techniques can be used to pass data between web pages in plain JavaScript apps?**

- ☐ a. CSS variables
- ☒ b. localStorage
- ☒ c. URL parameters
- ☐ d. React context
- ☒ e. sessionStorage

**Question 6**

**Which of the following are advantages of Server-Side Rendering (SSR)?**

- ☐ a. SSR requires localStorage
- ☐ b. No need for a server
- ☒ c. Better initial load time
- ☐ d. Full interactivity before HTML arrives
- ☒ e. Content is visible without JavaScript enabled

**Question 7**

**Which of the following statements about Single Page Applications (SPA) are correct?**

- ☐ a. Each screen is rendered on the server
- ☒ b. JavaScript dynamically updates the content without reloading
- ☐ c. SPAs require a different browser than MPAs
- ☒ d. The browser only loads one HTML file initially
- ☐ e. Every navigation requires a full page reload

**Question 8**

**Which statements about Server-Side Rendering (SSR) are correct?**

- ☐ a. SSR requires a JavaScript frontend to be rendered
- ☐ b. The browser creates the initial HTML based on templates
- ☐ c. SSR apps cannot include any client-side logic
- ☒ d. SSR pages are usually visible faster on slow devices
- ☒ e. The HTML is generated on the server and sent to the browser

**Question 9**

**Which technologies or frameworks are commonly used for server-side rendering?**

- ✓ a. Laravel
- ✗ b. Vue (without additional frameworks)
- ✗ c. React (without additional frameworks)
- ✓ d. Next.js
- ✓ e. Django

**5.4 React****Question 1**

**What is the role of Babel in React development?**

- ✓ a. It compiles JSX into JavaScript
- ✗ b. It replaces Webpack
- ✗ c. It manages state
- ✗ d. It renders the DOM
- ✗ e. It styles components

**Question 2**

**What must a React component return?**

- ✗ a. Multiple root elements
- ✗ b. A CSS block
- ✗ c. A Promise
- ✓ d. A single root element (usually JSX)
- ✗ e. An HTML string

**Question 3**

**Which are valid ways to define React components?**

- ✓ a. Arrow functions
- ✗ b. `defineComponent()`
- ✓ c. Function declarations
- ✗ d. XML templates
- ✗ e. CSS functions

**Question 4**

**Which files are part of a standard Create React App structure?**

- ☐ a. build/src.js
- ☐ b. App.jsx.html
- ☒ c. src/App.js
- ☐ d. App.vue
- ☒ e. public/index.html

**Question 5**

**Which commands are used to create and start a React project with Create React App?**

- ☐ a. react-start
- ☒ b. npm start
- ☐ c. npm create react-app
- ☒ d. npx create-react-app my-app
- ☐ e. babel-run

**Question 6**

**How is data inserted into JSX expressions?**

- ☒ a. Using curly braces {}
- ☐ b. Using double quotes
- ☐ c. Using {{ }}
- ☐ d. Using backticks
- ☐ e. With a special tag named bind

**Question 7**

**What is the purpose of public/index.html in a React project?**

- ☐ a. It is dynamically generated by ReactDOM
- ☐ b. It defines all components
- ☐ c. It includes all JavaScript code
- ☒ d. It contains the <div id="root"> for React to mount
- ☒ e. It provides the root HTML structure for the app

**Question 8****What is the function of `src/index.js`?**

- ✓ a. It initializes the React app
- ✓ b. It renders the App component
- ✗ c. It contains all CSS styles
- ✗ d. It defines the Virtual DOM
- ✗ e. It loads JSX files directly

**Question 9****Which tool is required to transform JSX for the browser?**

- ✓ a. Babel
- ✗ b. ReactDOM
- ✗ c. npm
- ✗ d. Webpack
- ✗ e. TypeScript

**Question 10****Which statements about JSX are true?**

- ✓ a. JSX must be transformed before it can run
- ✓ b. JSX looks like HTML but compiles to JavaScript
- ✗ c. JSX can only be used in Node.js
- ✗ d. JSX is valid JavaScript
- ✗ e. JSX replaces JavaScript syntax

**Question 11****Which of the following statements about React are true?**

- ✗ a. It is used only for server-side rendering
- ✓ b. It replaces HTML
- ✗ c. It is a JavaScript library for building UIs
- ✗ d. It is a CSS framework
- ✓ e. It uses a Virtual DOM to improve performance

**Question 12****How can React render elements without JSX?**

- ☐ a. With string interpolation
- ☐ b. Using document.write
- ☒ c. Using React.createElement
- ☐ d. With HTML templates
- ☐ e. By using Babel directly

**Question 13****Who developed React and when?**

- ☐ a. Microsoft, 2014
- ☐ b. Google, 2012
- ☒ c. Facebook, 2013
- ☐ d. Twitter, 2011
- ☐ e. Mozilla, 2015

**Question 14****What is React.StrictMode used for?**

- ☐ a. It enables Babel
- ☐ b. It disables JSX
- ☐ c. It enforces runtime checks in production
- ☐ d. It applies automatic code formatting
- ☒ e. It helps identify potential problems in an application during development

**Question 15****Which statements about the Virtual DOM are true?**

- ☒ a. React applies only differences to the real DOM
- ☒ b. It is an in-memory representation of the DOM
- ☐ c. It is slower than direct DOM manipulation
- ☐ d. It renders changes directly to HTML
- ☐ e. It uses Shadow DOM internally

**Question 16****Which statements are true about class-based components?**

- ☒ a. They are preferred over function components
- ☒ b. They use a `render()` method to return JSX
- ☒ c. They do not use props
- ☒ d. They cannot hold state
- ☒ e. They are required for all components

**Question 17****Which of the following statements about React components are true?**

- ☒ a. Components modify the DOM directly
- ☒ b. Components cannot be nested
- ☒ c. Components are reusable units of UI
- ☒ d. Components can receive props and hold state
- ☒ e. Components must be class-based

**Question 18****Which is a valid way to define a React component?**

- ☒ a. As an event listener
- ☒ b. With inline CSS
- ☒ c. Using a function that returns JSX
- ☒ d. Using a JSON object
- ☒ e. Using a for loop

**Question 19****What are advantages of component composition?**

- ☒ a. Eliminates props
- ☒ b. Prevents dynamic rendering
- ☒ c. Improves code reuse
- ☒ d. Requires class components
- ☒ e. Encourages modular design

**Question 20****Which are valid ways to implement conditional rendering in JSX?**

- ✓ a. Using if-else outside the return block
- ✓ b. Using ternary operator
- ✗ c. Using HTML conditions
- ✗ d. Using JSX if-then statements
- ✗ e. Using switch directly in JSX

**Question 21****How do you pass a parameter to an event handler in JSX?**

- ✓ a. Bind in render()
- ✗ b. Add param after function name
- ✗ c. Use apply()
- ✗ d. Call it directly
- ✓ e. Wrap it in an arrow function

**Question 22****How do you prevent a function from being called immediately in JSX?**

- ✗ a. Use bind() always
- ✗ b. Add return before call
- ✓ c. Use function reference: `onClick={myFunction}`
- ✗ d. Use parentheses: `onClick={myFunction()}`
- ✗ e. Use eval()

**Question 23****What is true about explicit composition in React?**

- ✗ a. Renders children automatically
- ✗ b. Only works with arrays
- ✗ c. Requires external libraries
- ✓ d. Child elements are passed directly in JSX
- ✓ e. Accessed using `props.children`



**Question 24****What is true about helper functions in components?**

- ✓ a. They can compute derived values
- ✗ b. They replace hooks
- ✓ c. They have access to props and local state
- ✗ d. They must be declared globally
- ✗ e. They cannot return JSX

**Question 25****What describes implicit composition in React?**

- ✗ a. Only works with `props.children`
- ✓ b. Used for rendering dynamic collections
- ✓ c. Parent component generates children via `map()`
- ✗ d. Child components must control layout
- ✗ e. Requires `useEffect`

**Question 26****What must you do when rendering lists with `map()` in JSX?**

- ✗ a. Use a while loop
- ✗ b. Avoid using functions
- ✓ c. Provide a unique key for each element
- ✗ d. Use for-of loops
- ✗ e. Wrap each item in

**Question 27****What are props in React?**

- ✗ a. HTML attributes
- ✗ b. Component methods
- ✗ c. Global variables
- ✓ d. Input data passed from parent to child components
- ✗ e. React internal state

**Question 28****How can you access props using destructuring?**

- ☒ a. Access via `props.name()`
- ☒ b. Use `({ name, age })` in the function parameters
- ☒ c. Use `const { name, age } = this`
- ☒ d. Bind props manually
- ☒ e. Use `props[name]`

**Question 29****What does the `useState()` hook return?**

- ☒ a. A Promise
- ☒ b. An array with current state and a function to update it
- ☒ c. An event object
- ☒ d. A reference to DOM node
- ☒ e. A single state value

**Question 30****Which is the correct way to update state in a click handler?**

- ☒ a. Using a global counter
- ☒ b. Direct assignment like `count++`
- ☒ c. `setState(count + 1)` inside a function
- ☒ d. Mutating the DOM
- ☒ e. Calling `setState()` outside render

**Question 31****How does a child component communicate with its parent?**

- ☒ a. By calling a function received via props
- ☒ b. Using Redux only
- ☒ c. Using global variables
- ☒ d. Via `useRef`
- ☒ e. By accessing the parent's state directly

**Question 32****What is true about event bubbling?**

- ✓ a. Child handler is triggered before parent by default
- ✗ b. Events are synchronous
- ✗ c. Event propagation starts at root
- ✓ d. Events propagate from target to ancestors
- ✗ e. Only one handler can be called

**Question 33****Which statements about event capturing are correct?**

- ✓ a. It processes events from root to target
- ✗ b. It only works with hooks
- ✗ c. It disables bubbling
- ✗ d. It ignores `stopPropagation`
- ✓ e. It requires a third argument set to `true`

**Question 34****Which statements about the `fetch` API in React are true?**

- ✓ a. It can be used inside `useEffect` to retrieve data
- ✗ b. It requires Redux
- ✗ c. It runs synchronously
- ✗ d. It directly updates the DOM
- ✓ e. It returns a `Promise`

**Question 35****How can a form input be reset after submission?**

- ✗ a. Using `innerHTML = ''`
- ✗ b. Reloading the page
- ✗ c. Calling `setTimeout()`
- ✗ d. Removing the input element
- ✓ e. By setting its state value to an empty string

**Question 36****How is form input typically handled in React?**

- ☐ a. Using global variables
- ☒ b. Using `useState` to bind input values
- ☒ c. Updating state on each `onChange` event
- ☐ d. Setting values via `innerHTML`
- ☐ e. Directly modifying DOM

**Question 37****Which is a correct structure for a controlled React form component?**

- ☐ a. Input without any event handling
- ☒ b. Input with value bound to state and `onChange` updating state
- ☐ c. `innerHTML` binding
- ☐ d. Form using `ref` as default
- ☐ e. Global event listener on `window`

**Question 38****Which of the following can be used for form validation?**

- ☒ a. HTML attributes like `required` and `minlength`
- ☐ b. CSS media queries
- ☐ c. JSX attributes only
- ☒ d. Custom validation logic in `handleSubmit`
- ☐ e. `React.StrictMode`

**Question 39****What is 'lifting state up' in React?**

- ☒ a. Moving state to a common parent component
- ☐ b. Binding refs
- ☐ c. Passing props downward
- ☐ d. Creating CSS modules
- ☐ e. Using session storage

**Question 40****What is the typical purpose of `onSubmit` in React forms?**

- ✓ a. To handle form submission and prevent default behavior
- ✗ b. To validate HTML structure
- ✗ c. To access Redux store
- ✗ d. To bypass event bubbling
- ✗ e. To reset the entire app

**Question 41****What happens when `event.stopPropagation()` is called?**

- ✗ a. The event handler is removed
- ✗ b. The event is canceled
- ✗ c. The DOM is re-rendered
- ✗ d. All child events are ignored
- ✓ e. Further propagation of the event is stopped

**Question 42****What are common use cases for the `useEffect` hook?**

- ✗ a. Rendering JSX directly
- ✗ b. Running before DOM updates
- ✗ c. Handling synchronous updates
- ✓ d. Performing side effects like data fetching
- ✓ e. Reacting to changes in props or state

**Question 43****What does the second argument of `useEffect` control?**

- ✓ a. When the effect function is re-executed
- ✗ b. What values are returned
- ✗ c. How JSX is rendered
- ✗ d. Which variables are global
- ✗ e. Whether props are required

**Question 44**

**What happens if you pass an empty array as the dependency list to `useEffect`?**

- ☐ a. It runs before render
- ☒ b. The effect runs only after the first render
- ☐ c. It never runs
- ☐ d. It causes an error
- ☐ e. It re-runs on every state change

**Question 45**

**What is `useRef` commonly used for in forms?**

- ☐ a. To create global variables
- ☐ b. To bind event listeners
- ☐ c. To trigger re-renders
- ☒ d. To access DOM elements directly
- ☐ e. To store component state

**Question 46**

**How do you correctly update an object state in Context API with `useState`?**

- ☐ a. By directly changing properties
- ☐ b. By using `Object.assign` in-place
- ☐ c. By using `push()` or `+=` operators
- ☐ d. By calling `setTimeout`
- ☒ e. By using the spread syntax to create a new object

**Question 47**

**Which state fields are typically used in Redux for async operations?**

- ✓ a. `error`
- ✗ b. `HTMLString`
- ✓ c. `loading`
- ✗ d. `responseTime`
- ✗ e. `retryCount`
- ✓ f. `data` or `user`

**Question 48**

**What does `dispatch()` return when used with a thunk?**

- ✗ a. The current component
- ✓ b. A `Promise` if the thunk is async
- ✗ c. Nothing
- ✗ d. An action object
- ✗ e. A DOM node

**Question 49**

**Why is direct mutation of state allowed in Redux reducers using Redux Toolkit?**

- ✓ a. Because Redux Toolkit uses `Immer` to track changes
- ✗ b. Because Redux ignores immutability
- ✗ c. Because reducers are only used for setup
- ✗ d. Because JavaScript allows it
- ✓ e. `Immer` converts mutations into immutable updates internally

**Question 50**

**What is the purpose of middleware in Redux?**

- ✗ a. To render UI
- ✓ b. To intercept actions and add behavior
- ✗ c. To handle side effects like logging or async calls
- ✗ d. To define routes
- ✗ e. To update CSS

**Question 51**

**Which of the following is a correct way to write a Redux reducer using Redux Toolkit?**

- ☒ a. Using `event.preventDefault()`
- ☒ b. Directly mutating the state (e.g., `state.count += 1`)
- ☒ c. Modifying props
- ☒ d. Returning a modified state manually
- ☒ e. Calling `setState`

**Question 52**

**What is a core difference between `useState` and Redux reducers?**

- ☒ a. `useState` must be global
- ☒ b. Reducers are only called once
- ☒ c. Redux never stores state
- ☒ d. `useState` updates require returning a new object
- ☒ e. Redux reducers with Immer allow mutation-style syntax

**Question 53**

**How would a 'Retry' button typically work in Redux?**

- ☒ a. It forces a React render
- ☒ b. It resets the component
- ☒ c. It reloads the entire page
- ☒ d. It dispatches the same async thunk again
- ☒ e. It clears all reducers

**Question 54**

**Where should API calls typically happen in a Redux app?**

- ☒ a. Inside reducers
- ☒ b. Directly in components
- ☒ c. Inside JSX expressions
- ☒ d. In the `index.js` file
- ☒ e. Inside thunks or middleware



**Question 55****How should you structure Redux state for a remote fetch?**

- ☐ a. Store HTML inside state
- ☒ b. Separate keys for loading, error, and data
- ☐ c. Only store raw response
- ☐ d. Use refs instead of state
- ☐ e. A single key with a long string

**Question 56****In what order are actions usually dispatched in a thunk for async requests?**

- ☒ a. Start → Success or Failure
- ☐ b. Only one action is dispatched
- ☐ c. Success → Start → Failure
- ☐ d. Failure → Retry → Start
- ☐ e. Start → End → Retry

**Question 57****What does a typical Redux Thunk do?**

- ☐ a. Modify the DOM manually
- ☐ b. Inject middleware automatically
- ☒ c. Dispatch multiple actions depending on async outcome
- ☐ d. Return HTML
- ☐ e. Render the React component directly

**Question 58****Which statements about Redux Thunks are true?**

- ☒ a. They allow dispatching asynchronous logic
- ☒ b. They return a function instead of an action object
- ☐ c. They mutate reducers
- ☐ d. They replace all reducers
- ☐ e. They require Context API

**Question 59****What are key benefits of using Redux Toolkit?**

- ✓ a. Built-in support for `createAsyncThunk`
- ✓ b. Simplified reducer logic with Immer
- ✗ c. Automatic UI testing
- ✗ d. Global CSS injection
- ✓ e. Less boilerplate code

**Question 60****What does `createAsyncThunk` help you do?**

- ✓ a. Generate async thunks with built-in action types
- ✗ b. Bind input fields
- ✗ c. Validate forms
- ✗ d. Replace `useEffect`
- ✓ e. Automatically create pending, fulfilled, and rejected actions

**Question 61****Which path is used to match all unknown routes?**

- ✓ a. `*`
- ✗ b. `/error`
- ✗ c. `**`
- ✗ d. `/notfound`

**Question 62****What component must wrap your entire React app to enable routing?**

- ✗ a. `RouteManager`
- ✓ b. `BrowserRouter`
- ✗ c. `RouteProvider`
- ✗ d. `AppRouter`

**Question 63**

**In a nested route, how is the default child defined?**

- ☒ a. `path=/element`
- ☒ b. `element={<Default />}`
- ☒ c. index element
- ☒ d. fallback element

**Question 64**

**Which of the following correctly defines a route to a LoginPage component?**

- ☒ a. `<Router path="/login" component={<LoginPage />}> />`
- ☒ b. `<Route path="/login">{<LoginPage />}</Route>`
- ☒ c. `<Route path="/login" element={<LoginPage />}> />`
- ☒ d. `<Route url="/login" render={<LoginPage />}> />`

**Question 65**

**How do you define a dynamic segment in a React Router path?**

- ☒ a. Using dollar sign, like `$id`
- ☒ b. Using curly braces, like `{id}`
- ☒ c. Using a colon, like `:id`
- ☒ d. Using angle brackets, like `<id>`

**Question 66**

**Why might you use HashRouter instead of BrowserRouter?**

- ☒ a. It doesn't require server configuration for route handling
- ☒ b. It supports more modern features
- ☒ c. It allows usage of cookies
- ☒ d. It's the only router that works with JSX

**Question 67**

**What API does BrowserRouter rely on?**

- ☐ a. Window Navigation API
- ☐ b. React Context API
- ☒ c. HTML5 History API
- ☐ d. Session API

**Question 68**

**Which route element ensures that an invalid URL renders a fallback?**

- ☐ a. Route with fallback
- ☐ b. Route with no path
- ☐ c. ErrorBoundary
- ☒ d. Route with path="\*"

**Question 69**

**What happens when you click a Link component in React Router?**

- ☐ a. An iframe opens the target page
- ☒ b. The URL changes without full page reload
- ☐ c. An anchor tag redirects to the route
- ☐ d. The server reloads the route from scratch

**Question 70**

**Which component provides navigation between routes in React Router?**

- ☐ a. Anchor
- ☒ b. Link
- ☒ c. NavLink (Note: I also think it is correct but the Prof not)
- ☐ d. Redirect

**Question 71**

**Which component is used as a placeholder for nested route content?**

- ☐ a. Placeholder
- ☒ b. Outlet
- ☐ c. Switch
- ☐ d. Content

**Question 72**

**What is the purpose of a ProtectedRoute component?**

- ☐ a. To handle 404 errors
- ☐ b. To define admin-only pages
- ☐ c. To cache route components
- ☒ d. To restrict access to certain routes based on authentication

**Question 73**

**What React Router hook is used to read URL parameters?**

- ☐ a. useLocation
- ☐ b. useRoute
- ☐ c. useQuery
- ☒ d. useParams





**Question 74**

**Which component is used to redirect the user to another path?**

- ☐ a. RedirectTo
- ☐ b. RouterRedirect
- ☐ c. HistoryPush
- ☒ d. Navigate

**Question 75**

**What type of data is returned by `useParams()`?**

-  a. A query string
-  b. An array of parameters
-  c. A route object
-  d. An object of strings