TW-011 TEAM LEAD VERSION (Sprint-7 Week-1)

TEAMWORK

CLARUSWAY
WAY TO REINVENT YOURSELF

Meeting Agenda

▶ Icebreaking

▶ Questions

▶ Interview Questions

▶ Coding Challenge

▶ Video of the week

▶ Retro meeting

▶ Case study / project

Teamwork Schedule

Ice-breaking 5m

- Personal Questions (Study Environment, Kids etc.)
- Any challenges (Classes, Coding, studying, etc.)
- Ask how they're studying, give personal advice.
- Remind that practice makes perfect.

Team work 5m

- Ask what exactly each student does for the team, if they know each other, if they care for each other, if they follow and talk with each other etc.

Ask Questions 15m

1. The maximum number of global symbols a module can define is _____ .

A. 2
B. 3
C. 4
D. 1

Answer: D
Explanation: Generally, the various modules are allowed to run in the pristine (or near pristine) environment that it expects. The modules should minimize the number of global symbols they define – ideally, no module should define more than one.

2. Modules that have more than one item in their API can _____ .

A. Assign itself to a global variable
B. Invoke another module of the same kind
C. Return a namespace object
D. Invoke another module of the same kind

Answer: C
Explanation: Namespace is a container for a set of identifiers, functions, methods and all that. It gives a level of direction to its contents so that it will be well distinguished and organized. Modules that have more than one item in their API can return a namespace object.

3. The properties() method is a _____ .

A. Enumerable method
B. Non-enumerable method
C. Operational method
D. Calling method

Answer:B
Explanation: The `properties()` *is a method to get information of the properties of a particular object.* `properties()` *method is a non-enumerable method.*

4. Which of the following languages support functional programming paradigm?

A. Javascript
B. R
C. Closure
D. All of them

Answer: D

5. One of the side effects of functional programming if not implemented can be _____ .

A. More memory utilization because it relies on immutable data
B. More CPU cycles may be needed to crunch and analyze data
C. None of these
D. Both A and B

Answer: D

6. What are the various programming paradigms?

A. Functional Programming
B. Object Oriented Programming
C. compImperative Programming
D. All of them

Answer: A

7. Functional Programming heavily relies on the use and re-use of _____ .

A. Objects
B. Functions
C. Variables
D. All of them

Answer: B

8. 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
B. This is a route parameter
C. This is a route splitter
D. This is a route link

Answer: B

9. 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.

Answer: A

10. What can you use to handle code splitting?

A. `React.memo`
B. `React.split`
C. `React.lazy`
D. `React.fallback`

Answer: C

11. When might you use `React.PureComponent`?

A. when you do not want your component to have props
B. when you have sibling components that need to be compared
C. when you want a default implementation of `shouldComponentUpdate()
D. when you do not want your component to have state

Answer: C

12. You have written the following code but nothing is rendering. How do you fix this problem?

```
const Heading = () => {
  <h1>Hello!</h1>;
};
```

A. Add a render function.
B. Change the curly braces to parentheses or add a return statement before the `h1` tag.
C. Move the `h1` to another component.
D. Surround the `h1` in a `div`.

Answer: B

13. Which option is correct for State vs Props

A. Props is something that the parent doesn't need and decide to throw around among other parents; State is the parent's favorite child and something the component wants to nurture.
B. Props get passed to the component using naming conventions, like a function parameter; State is managed within the component and holds some information that may change over the lifetime of the component.
C. Props is a copy of real DOM; State is the definition of the real DOM.
D. Prop is managed within the component and holds some information that may change over the lifetime of the component; State gets passed to the component, like a function parameter

Answer: B

Explanation: We can update state within the component using setState() , then we can pass the state to its children. Its children would use this.props to take whatever state from its parents.

14. 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.

Answer: B

Interview Questions 15m

1. What is Functional Programming?

Answer: Functional programming is a programming paradigm that dictates the way we build apps. Functional programming treats functions in programs like mathematical functions. This means that we avoid changing state and creating mutable data. It also means that functions are pure, that is, a function that returns the same thing given the same input. Functions also shouldn't have side effects in functional programming because it makes functions impure.

2. What makes JavaScript a Functional Language?

Answer: JavaScript functions are also first-class functions. This means that we can have functions that have functions as arguments or functions that return functions. It also supports closures, where we return functions that can use some values inside the parent function. For example, JavaScript arrays have the map, filter, and reduce methods, which take callback functions that are called as they process the items in the array. map, filter, and reduce are higher-order functions since they accept functions as arguments. Functions that return functions are also higher-order functions.

3. What are Higher Order Functions?

Answer: Higher-order functions are functions that take functions as arguments or return functions. For example, if we have:

```
const hoc = (fn)=> fn();
hoc(()=>{
 console.log('foo');
});
```

The code above has the hoc function which takes a function fn and runs it. Then when we make the following call:

```
hoc(()=>{
 console.log('foo');
});
```

Then we get foo logged since the function we passed in is run in the hoc function.

4. What is styled-component styling in React?

Answer: With styled-components, we can write actual CSS in our JavaScript file. This means you can use all the features of CSS — like media queries, pseudo-selectors, nesting, etc. — in JavaScript.

styled-components uses ES6's tagged template literals to style components. With it, the mapping between components and styles is removed. This means that when you're defining your styles, you're actually creating a normal React component that has your styles attached to it.

Using styled-components, we can create reusable components with styles. It is quite exciting to create and use. Explaining with an example here will do us a lot of good.

First, we need to install it, so run the following in your React app's directory:

```
$ npm install --save styled-components
```

Let's go back to our to-do app and make our component use styled components. First, we'll import the styled-components package:

```
import styled from 'styled-components';
```

We can start using it right away. We'll first create a styled component, then see how we'll use it:

```
const TodoComponent = styled.div`
  background-color: #44014C;
  width: 300px;
  min-height: 200px;
  margin: 30px auto;
  box-sizing: border-box;
`;
```

Above we created a component that can be used the same as any React component. However, notice that we are using pure CSS in a JavaScript file. Next, let's put this component to use:

```
function ToDoApp() {
 //...

 return (
   <TodoComponent>
     <h2>ToDo</h2>
     <div>
       <Input onChange={this.handleChange} />
       <p>{this.state.error}</p>
       <ToDoList value={this.state.display} />
     </div>
   </TodoComponent>
 );
}
```

In the code above, we used the styled component we created — TodoComponent — on line 5 like we'll use any other HTML element. The only difference is that it comes with its own predefined styles.

We can do the same for other parts of the component:

```
function ToDoApp() {
 //...

 return (
```

```
    <TagComponent>
      <Header>ToDo</Header>
      <div>
        <Input onChange={this.handleChange} />
        <ErrorMessage>{this.state.error}</ErrorMessage>
        <ToDoList value={this.state.display} />
      </div>
    </TagComponent>
  );
}
```

5. What is React Router?

Answer: A tool that allows you to handle routes in a web app, using dynamic routing. Dynamic routing takes place as the app is rendering on your machine, unlike the old routing architecture where the routing is handled in a configuration outside of a running app. React router implements a component-based approach to routing. It provides different routing components according to the needs of the application and platform.

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.

Most of the applications built with React.js are SPA (single page application), but it doesn't mean your app will have only one view. It means your app doesn't need to reload to another view, but how can we change views and go into the next page? We can use a react router for that! React router is the official and standard routing package that we use in React.js to change views, and move between pages.

With the React router, we can specify the whole routing for our modules that will decide what view should be visible when we enter the specified URL but not only. React router gives us the possibility to create protected views like, for example, the view that we need to be logged in or have any special requirements to visit.

One more useful feature of the React Router is the routing history, that can keep all of the history of our views, and come back to the specified step when needed.

The id of element, to render the route that can show specified elements, and give you access to that param. We can use routing navigation in a few ways. The most popular is to type URL, visit URL by a link inside the menu, or add data to the routing history.

On the example below, you can simple routing:

```
<Switch>
  <Route path="/about">
    <About />
  </Route>
  <Route path="/contact/:id">
    <Contact />
  </Route>
  <Route path="/contact">
    <AllContacts />
```

```
    </Route>
    <Route path="/">
      <Home />
    </Route>
  </Switch>
```

Coding Challenge 20m

- Coding Challenge: JS-CC-008: Sliding Window Maximum

☕

Coffee Break 10m

☕

Video of the Week 5m

- What to do in an interview
- React Router Setup in 5 minutes

Retro Meeting on a personal and team level 5m

Ask the questions below:

- What went well?
- What went wrong?
- What is the improvement areas?

Case study/Project 15m

Case study should be explained to the students during the weekly meeting and has to be completed in one week by the students. Students should work in small teams to complete the case study.

- RC-005 Clarusway Website Page with Router

Closing 5m

-Next week's plan

-QA Session
