Day 2

2d) Answer the following questions (if you skip this part, you have wasted your time)

- What is a functional component?
  - ➢ It's a component which accepts a single "props" object with a data and returns a react element.it is called functional because it's a JavaScript function.
- What is a Class Component?
  - ➢ It's a component that accepts multiple object with a data and return a react element through a render () method.
- What is the idea with props
  - ➢ Unchangeable parameters used to customise most components when they are being created. (props are READ ONLY)
- Provide a simple example in how you write a Component that accepts props
  - ➢
```
function Welcome(props) {
 return<h1>Hello, {props.name} </h1>;
```
- Provide a simple example (could be a line from the exercise above) that demonstrates how you pass props into a Component
  - ➢
```
function App2(props){
  return (
   <div className="App2">
    <header >
     <h1 className="App2-title">Welcome to React</h1>
    </header>
    <p className="App2-intro">
     To get started, edit <code>src/App.js</code> and save to reload.
    </p>
   </div>
  );
}
```

3f) Answer the following questions (if you skip this part, you have wasted your time)

- Would it be possible to rewrite the Clock component into a functional component (provide arguments for your answer)?
  - ➢
- How do you set new values for state: In the constructor, and all other places?
  - ➢ By using this.setState function which will accept an object that will be merged into components current state.
- How is it possible to "tell" React that you want the UI to be updated (re-rendered)?
  - ➢ By calling this.forceUpdate() method or this.setState() method
- What is the difference(s) between state and props?
  - ➢ State can change inside a component while props cannot.
  - ➢ Props can change in parent and child component while state cannot change.
- How do you pass in prop values to a Component?
  - ➢ Binding it in the constructor.
- What is the purpose of React Components Life Cycle Methods?
  - ➢ Through lifecycle methods, we can then control what happens when each tiny section of your UI renders, updates, thinks about re-rendering, and then disappears entirely.

5f) Answer the following questions (if you skip this part, you have wasted your time)

- What is the purpose of this line in the constructor: `this.handleClick = this.handleClick.bind(this);`
  - ➢ It is binding which enables “this” to work in the callback.
- How can we disable the default behavior of an event handler (for example prevent a submit handler to submit?)
  - ➢ By writing it as an arrow function

```
handleClick=()=> {
        this.setState(prevState => ({
          isToggleOn: !prevState.isToggleOn
        }));
    }
```

  - ➢ Instead of

```
handleClick() {
      this.setState(prevState => ({
        isToggleOn: !prevState.isToggleOn
      }));
    }
```

- What is the benefit(s) you get from using arrow-functions in a ES6 class?
  - ➢ It automatically bind “this”

Day 3

2g) Answer the following questions, before you continue (questions we will ask during the examination)

- What is the purpose of the key value, which must be given to individual rows in a react-list
  - ➢ Because the key is always unique to identify individuals rows.
- It's recommended to use a unique value from you data if available (like an ID). How do you get a unique value in a map callback, for data without a unique id?
  - ➢ By using a filter()method it traverses the array from left to right invoking a callback function on each element.
- What is the difference(s) between state and props?
  - ➢ State can change inside a component while props cannot.
  - ➢ Props can change in parent and child component while state cannot change.
- For which scenarios would you use props, and for which would you use state?
  - ➢
- Where is the only place you can set state directly as in: `this.state = {name: "Peter"};`
  - ➢ In a functional component
- How must you set state all other places?
  - ➢ By using this.setState

4d) Answer the following questions, before you continue (questions we will ask during the examination)

- In a Controlled Component React state is made the "Single source of truth", so how:
  1. Do we ensure that input controls like `text`, `textarea` or `select` always presents the value found in the components state?
     - ➢ By lifting state up to their common ancestor
  2. Do we ensure that a controls state, always matches the value found in an input control?
- What is the purpose of doing `event.preventDefault()` in an event handler?
  - ➢ If this method is called, the default action of the event will not be triggered.
- What would be the effect of NOT doing event.preventDefault in a submit handler?
  - ➢
- Why don't we want to submit the traditional way, in a single page application?
  - ➢
- What are the two different ways we can use to make this works as expected for our event handlers?
  - ➢
- Explain in words what it takes to implement the "Controlled Component" pattern for a form
  - ➢

5d) Answer the following questions, before you continue (questions we will ask during the examination)

- What is meant by the react term “Lifting State Up”?
  - ➢ To share a state between two components, the most common operation is to move it up to their closest common ancestor.
- Where do you lift it up to?
  - ➢ Usually, the state is first added to the component that needs it for rendering. Then, if other components also need it, you can lift it up to their closest common ancestor.
- Which way does React recommend data to flow: From sibling to sibling, from bottom to top or from top to bottom?
  - ➢ Top to bottom
- Lifting state up, involves a great deal of boilerplate code, what is the benefit we get from this strategy

  - ➢ it takes less work to find and isolate bugs.