FWP Semester 2, 2023

Week 02

More on components;

Interaction between components;

Incorporating data and

Styling in React

RMIT UNIVERSITY

Before we start

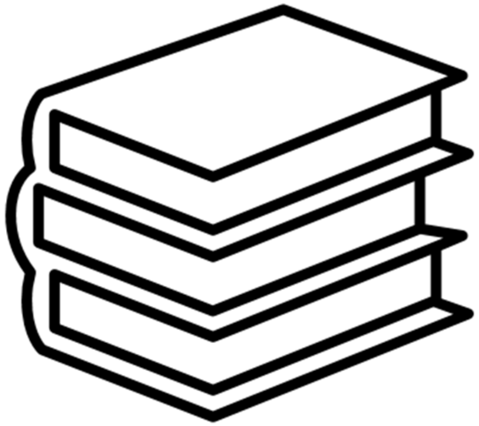
Lectorial

Tutorial/Lab

Assignments

Expectations

Introduction



This is week 2 and that means

- There was a week 1
- Which in turn means
  1. You need to read week 1 Lectorial
  2. Watch the associated recording
  3. Read the pre Lectorial recommendations for week(s) 1 and 2

- This is a fast-moving elective course, and you really need to keep up with the pace

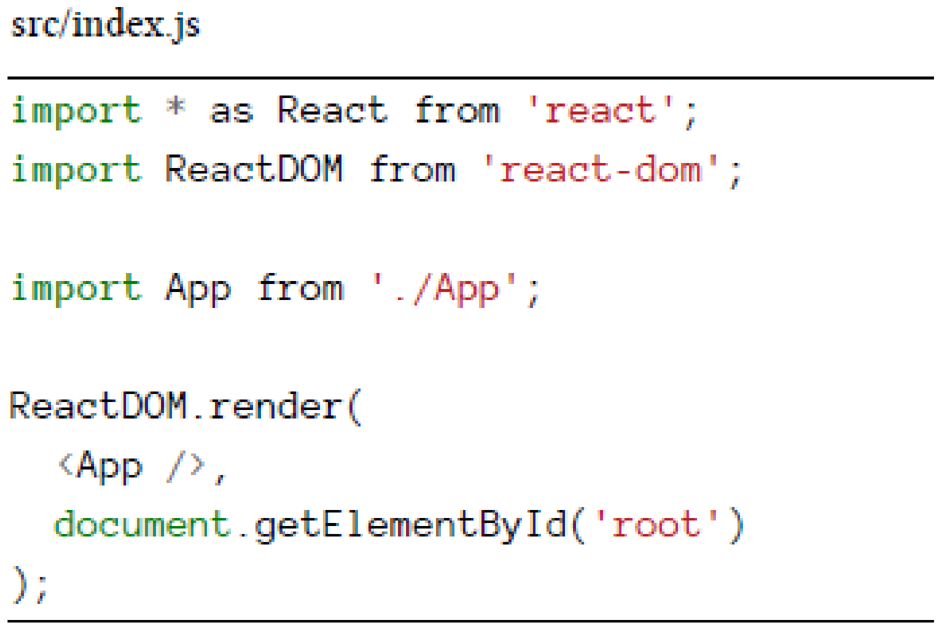
Segment 1

More on components

Interaction between components

React DOM

- **Think of all of the examples covered in week 1.**
- **Where do you think the- *App component instantiation is occurring?***
- **As discussed in week 1, it occurs in the src/index.js file**



src/index.js

```
import * as React from 'react';
import ReactDOM from 'react-dom';

import App from './App';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

React DOM

- Next to React which is imported from react, there is another imported library called react-dom,
  - in which a ReactDOM.render() function uses an HTML node to replace it with JSX.
- Essentially that’s everything needed to integrate React into any application which uses HTML.
- ReactDOM.render() expects two arguments; the first is to render the JSX
- The second argument specifies where the React application enters your HTML.
  - It expects an element with an id='root', found in the public/index.html file. This is a basic HTML file

React component definition

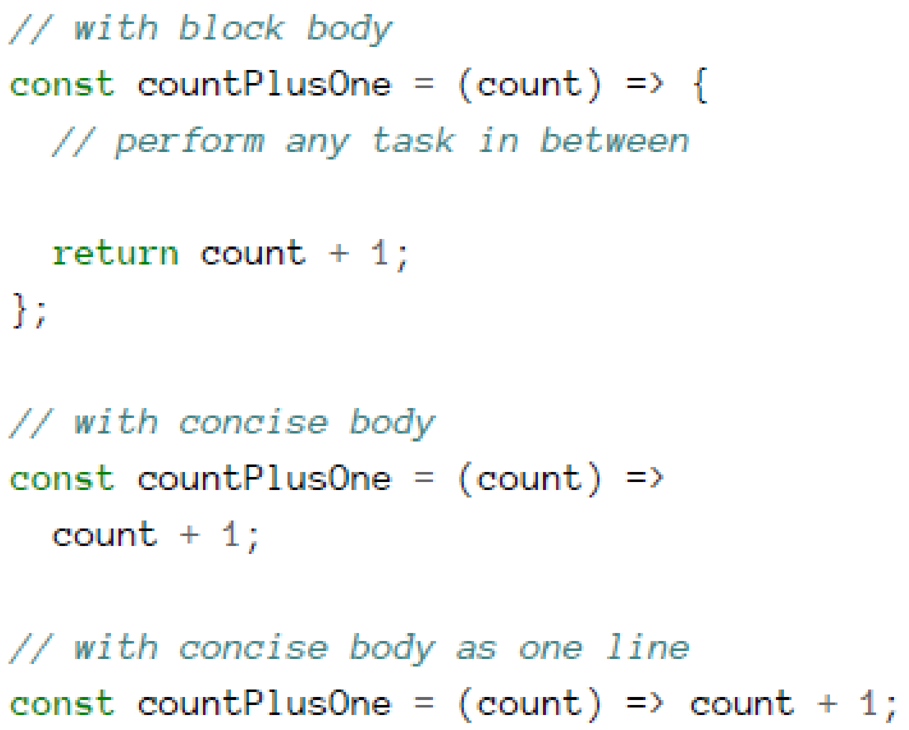
- There are multiple ways of declaring a component.
- So far, we have used the function statement, though arrow functions can be used more concisely

```
// function declaration
function () { ... }

// arrow function declaration
const () => { ... }
```

```
// allowed
const item => { ... }

// allowed (recommended)
const (item) => { ... }

// not allowed
const item, index => { ... }

// allowed (recommended)
const (item, index) => { ... }
```

React component definition

- If an arrow function’s only purpose is to return a value and it doesn’t have any business logic in between, you can remove the block body (curly braces) of the function.



```
// with block body
const countPlusOne = (count) => {
  // perform any task in between

  return count + 1;
};

// with concise body
const countPlusOne = (count) =>
  count + 1;

// with concise body as one line
const countPlusOne = (count) => count + 1;
```

REMEMBER WHAT YOU WERE TOLD
Can I use class components for this course?

- In this course we will only learn how to write new React i.e. *functional* components
- You are NOT allowed OR advised to write class components
  - You will get a zero in assessments for using class components
- Only legacy projects use class components
- So remember
  - Class components → ZERO

Lectorial Exercise

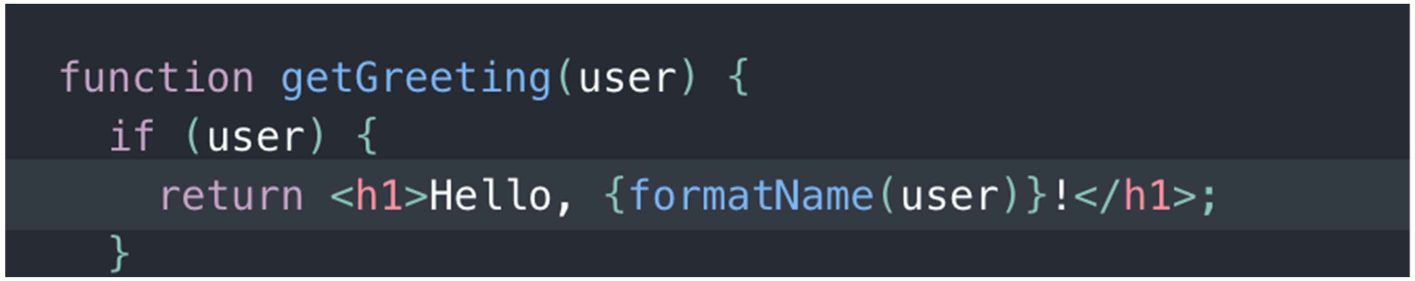
- What is *legacy code*?
- How to work effectively with a legacy code?

Going back to JSX

- It is called JSX, and it is a syntax extension to JavaScript.
- JSX produces React “elements”.
- You can embed expressions in JSX

  const name = ‘Jane Doe’;

  const element = <h1>Hello, {name}</h1>;
- After compilation, JSX expressions become regular JavaScript function calls and evaluate to JavaScript objects.



```
function getGreeting(user) {
  if (user) {
    return <h1>Hello, {formatName(user)}!</h1>;
  }
```

Going back to JSX

- In JSX you can specify *attributes*
- JSX prevents *sql injection attacks*
  - By default, React DOM escapes any values embedded in JSX before rendering them.
- *JSX* represents object
- Read more at
  - [https://reactjs.org/docs/introducing-jsx.html]

Interaction between components

- **Components can refer to other components in their output.**
- **This lets us use the same component abstraction for any level of detail.**
- **Do not be afraid to split components into smaller components.**
- **Besides it is a <u>bad practice to write everything in ONE BIG component</u>**
- **When you separate components, they will often need to interreact with each other**

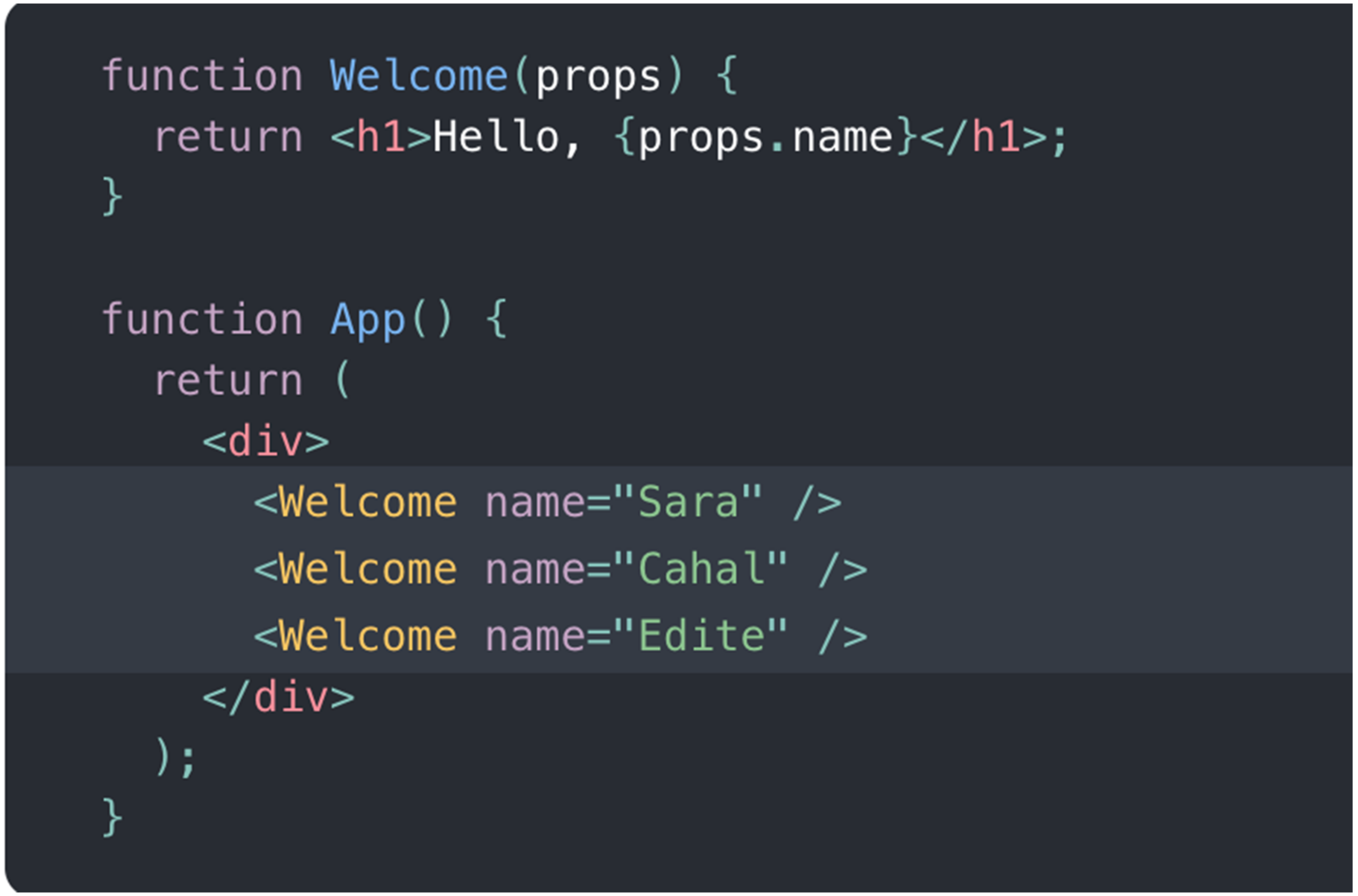
Interaction between components

- Extracting components might seem like grunt work at first, but having a palette of reusable components pays off in larger apps.
- A good *rule of thumb* is that if a part of your UI is used several times, or is complex enough on its own, it is a good candidate to be extracted to a separate component.
- Let us go through an example to demonstrate above
- Example 1
  - page layout using multiple components

React Props

- **Before we delve into an example, we need to understand another concept i.e.. *Props***
- **“Props” is a special keyword in React, which stands for properties and is used for passing data from one component to another.**
- **React Props are like function arguments in JavaScript and attributes in HTML.**
- **But (*there is always a but*):**
  - **data with props are being passed in a uni-directional flow (one way from parent to child)**
  - **props data is read-only, which means that data coming from the parent should not be changed by child components**

React Props



```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

function App() {
  return (
    <div>
      <Welcome name="Sara" />
      <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>
  );
}
```

Interaction between components

- How to use props?
- Firstly, define an attribute and its value(data)
- Then pass it within component(s) by using *props*
- Finally, render the props Data
- Time to now tie all this up with the help of a neat example
- Example 2

The rule of the props

- React has one weird strict rule:
  - ***All React components must act like pure functions with respect to their props.***
  - **[https://reactjs.org/docs/components-and-props.html]**
- ***WHAT DOES THAT MEAN!***
- **So we need to learn something that will help us make those changes**
- **This is where concept of "state" will come to rescue**
  - **State allows React components to change their output over time in response to user actions, network responses, and anything else, without violating this rule.**

Handling events

- Event handling makes it possible for your users to interact with the React app
- As per React documentation- handling events with React elements is similar to handling events on DOM elements except
  - React events are named using camelCase, rather than lowercase.
  - With JSX you pass a function as the event handler, rather than a string
- If you’re familiar with how events work in standard HTML and JavaScript, it should be easy for you to learn how to handle events in React.

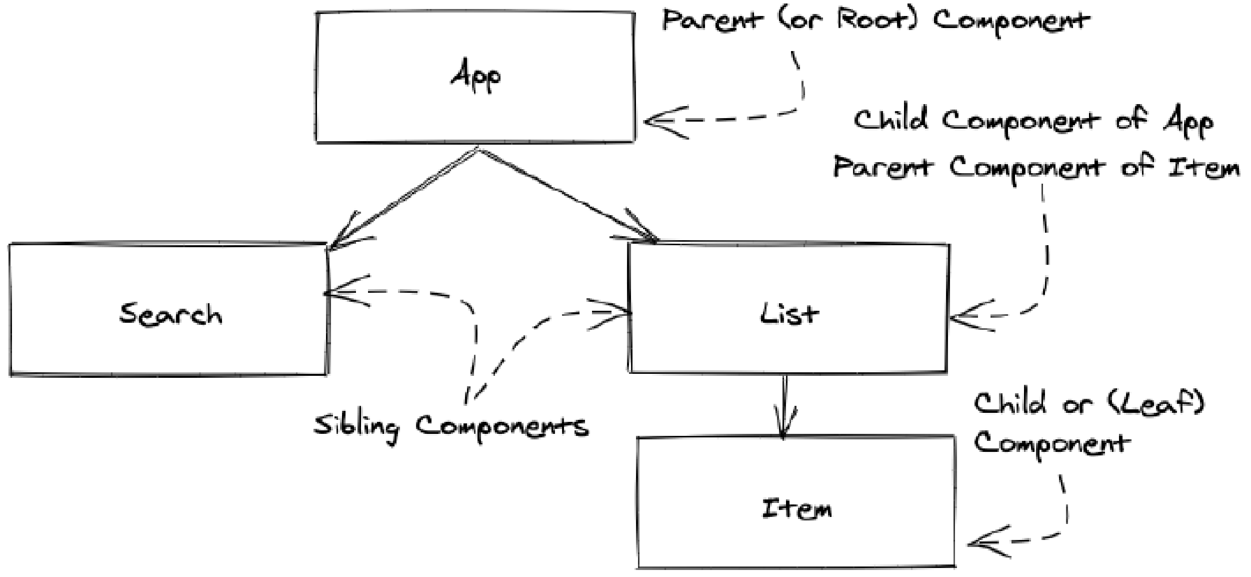
Handling events: functional component

- We looked at one example in week 1
- Here is another one
- Example 3

- We will revisit events in week 03
- We need to look at
  - Events and state
  - Synthetic events
  - Custom events

Lectorial Exercise

- **Complete the event handlers for example 03 from Lectorial 1.**



Segment 2

Incorporating data using data structure

Styling in React

Hooks again &

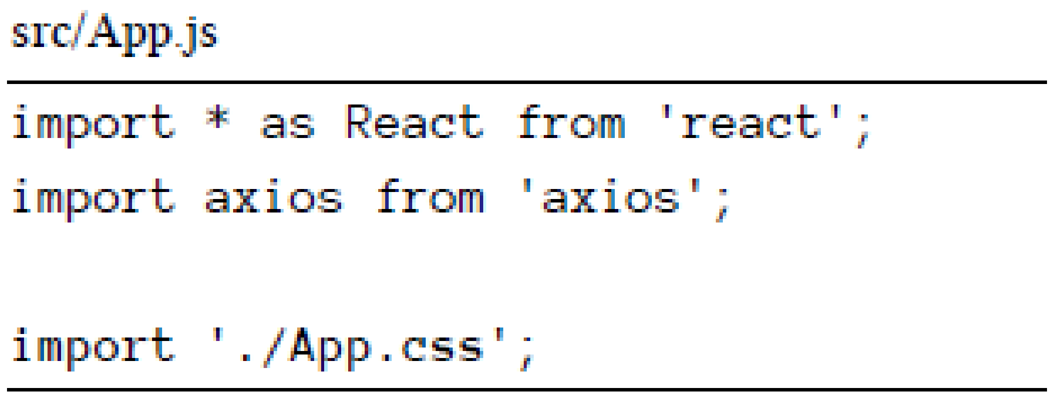
Assignment 1 discussion

Styling in React

- Assumption
  - You remember basic CSS syntax and concepts
  - If not, revise these

CSS in React

- Common CSS in React is similar to the standard CSS you may have already learned on your own.
- Each web application gives HTML elements a class (in React it's className) attribute that is styled via a CSS file:



src/App.js

```
import * as React from 'react';
import axios from 'axios';

import './App.css';
```

- The CSS file is present as src/App.css

Styling in React

- You can apply styling as
  - Inline CSS properties in React code
  - Using external CSS file
  - Using CSS modules
- While you can dabble with CSS yourself, there are other ways to style a react app
  - Bootstrap
  - Material-UI: *left as self-exercise*

Styling with Bootstrap

- First we need Bootstrap library, you can do either:
  1. Use CDN OR
  2. Import Bootstrap in React as a dependency
  3. Install a React Bootstrap package (such as *bootstrap-react* or *reactstrap*)
- First approach is the easiest one- add reference to CDN in index.html
- Bootstrap CDN

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<!-- jQuery library -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<!-- Popper JS -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<!-- Latest compiled JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
```

Styling with Bootstrap

- In order to use Bootstrap, you need to be familiar with basic bootstrap syntax
- Revise it from
  - [https://www.w3schools.com/bootstrap4/default.asp]
- Let us look at two examples
  - Example 4: using CDN
  - Example 5: using react-bootstrap

Data storage

- In any web application, you will need to store data one way or the other
- Examples- store user details, product preferences, items bought, etc.
- The data storage may be temporary or persistent
- We will learn the data storage via the databases in the latter half of this course where we will use MySQL database *(more for assignment 2*).
- But before that, there are other ways of storing data
  - Data structures in JS
  - HTML5's localStorage object
  - *You will need these for assignment 1*

Data storage with data structures

- You can use any data structure available in JavaScript
  - Binary search tree
  - Stack
  - Queue
  - Linked List
  - Hash Table
  - Maps
  - Sets
  - JSON data
- The disadvantage being- it will be lost when you shut down the browser.

Lectorial Exercise

- What is localStorage?
- What do you know about it?

Do you remember useState() *hook?*

1. **Call useState() hook to enable state in a functional component.**

2. **The first argument of the useState(initialValue) is the state's initial value.**

3. **[state, setState] = useState(initialValue) returns an array of 2 items: the state value and a state updater function.**

Do you remember useState() *hook?*

4. **Invoking the state updater function setState(newState) with the new value updates the state.**
   - **Alternatively, you can invoke the state updater with a callback setState(prev => next), which returns the new state based on previous.**

5. **After the state updater is called, React makes sure to re-render the component so that the new state becomes actual.**

Bring localStorage into the equation

- Time to look at an example – here we will use a data structure and localStorage to store the data.

Example 6

- A login-logout system
- It will also be covered in the forthcoming lab

Example 6- *crib notes*

- Uses some concepts that will cover ahead
  - Forms in React
  - Conditional rendering of components
  - React Router dom:
    - `install as npm install react-router-dom`
  - React router has been used to control which component is shown based on the current page, with location and navigation support
  - The router, switch and route paths can be found in App.js. The links using these paths can be found in Navbar.js.

Example 6- *crib notes*

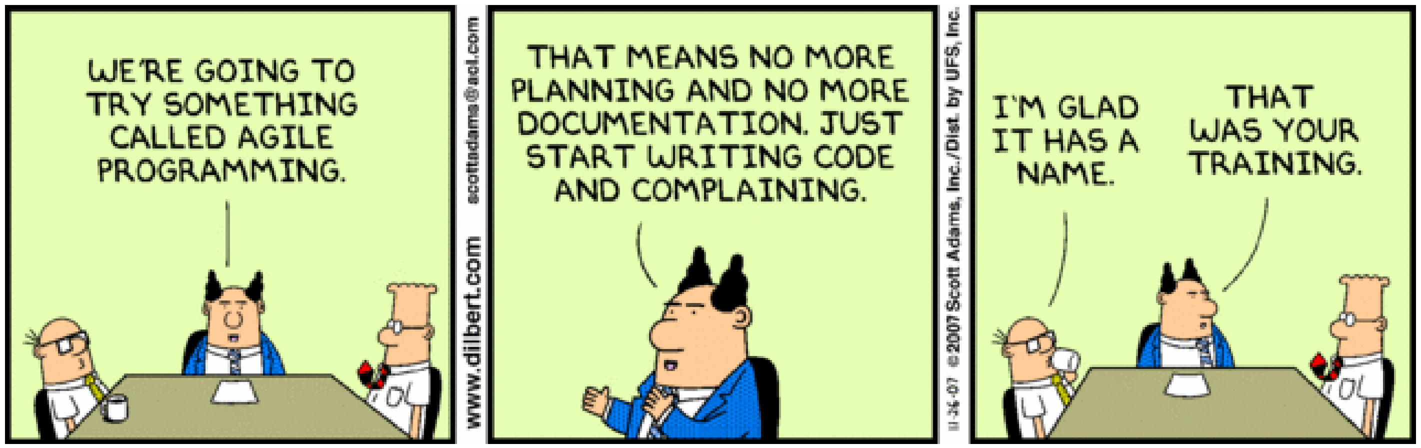
- The logged in username state is stored in the App parent component and is passed down to components that reference this state as props / properties.
- To modify the state functions implemented within the App component called loginUser and logoutUser are implemented. These functions are passed as props / properties to the Navbar and Login components; the Navbar component uses logoutUser and Login component uses loginUser.
- Lastly there is some conditional rendering is included in the Home and Navbar components to render different output if the user is logged in or not.
- The user data can be found in src/data/repository.js

Self-exercise

- Those of you who are interested should search and learn about- how to store offline data using Dexie.js
- It uses something known as IndexedDB to store offline data
- Think about
  - Why to use this option?
  - What advantages does it offer?
  - How to set it up?
  - How to use it?
- *Note: this is for enthusiastic learners- it is optional!*

Code Elegance

Writing good code



WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.
www.dilbert.com scottadams@aol.com
©2007 Scott Adams, Inc./Dist. by UFS, Inc.

Coding convention: React

- Some of these exist- slightly different for each organisation
- These can become tedious at times to follow- *but if you end up working at a place that mandates a standard, you won't have much choice in the matter!*
- Do not think of these are *hard and fast rules*
- Adhering to a standard way of writing code leads to a professional codebase (*though it is debatable*)
- Do not pay attention to
    - Bad practices prevalent in industry
    - Cowboy coders
    - Frustrated and negative people who criticise everything
    - Arrogant, burnt out and jaded professionals

Coding convention: React

- Learn from positive, experienced people
- Often experts are invisible to amateurs-
  - someone with none or little experience (a project or 1-2 years) has a lot to learn
- Clean code writing is often hard and a regimen that arrogant developers abhor. Traits of an arrogant developer -
  - Assume that they're the smartest person in the room.
  - Refuse to explain something because the other person "wouldn't understand".
  - Talk down to others / condescend.
  - Pretend to be smarter than they really are.
  - Assume they can't learn new things from other people.

Code commenting

- **One of the arguments often made by developers- *code commenting is a waste of time***
- **While many developers do not use it in the right spirit- it is important to have sensible code commenting**
  - **When code becomes complex, sometimes the developer might not even realise how complex their code has gotten. A facile argument often overheard is- *but my code is not complex!-….* 'complex' is a relative term**
  - **Another really good use case for code commenting is when an anomaly occurs, say for example because of browser differences you have to do something a bit differently or have a bit of extra unusual looking code in there.**

Code commenting

  - Also if you have a bug and find a solution on Stack Overflow in Github issues or something, it’s best to leave a link to that page in your code.
  - Clarification comments are intended for anyone (including your future self) who may need to maintain, refactor, or extend your code.
- So always add comments and add them sensibly
- You are going to lose marks in assessments in absence of code commenting

Plugins for cleaner code – VS Code

- You can use plugins in VS Code
- *Prettier*-an opinionated code formatter. It obviously supports JavaScript but also many other languages like JSX, CSS, JSON or Vue.
  - It is easy to install and use
- *EsLint*- It's an open-source project initially created by Nicholas C. Zakas, which provides a pluggable linting utility for JavaScript.
- Here is a good reading:
  - [https://thomaslombart.com/setup-eslint-prettier-react]

References

- Reference: ***The road to react (2021 edition), by Robin Weiruch; Leanpub***
- **The above will be the prescribed reference textbook for the first few week(s) for this course.**

Assignment 1 *first discussion*

☹

- **It is online**
- **Deadline: *Check Canvas***
- **Worth 25%**
- **To be completed group of 2**
- **Client-side React website prototype**
- **Based on week(s) 1-5**
- **Discussion..**

Next week

- More on React components
- Components interacting - part 2
- incorporating data
- State
- Forms
- props
- Styling in React