

# *Front End*

*Class 3*

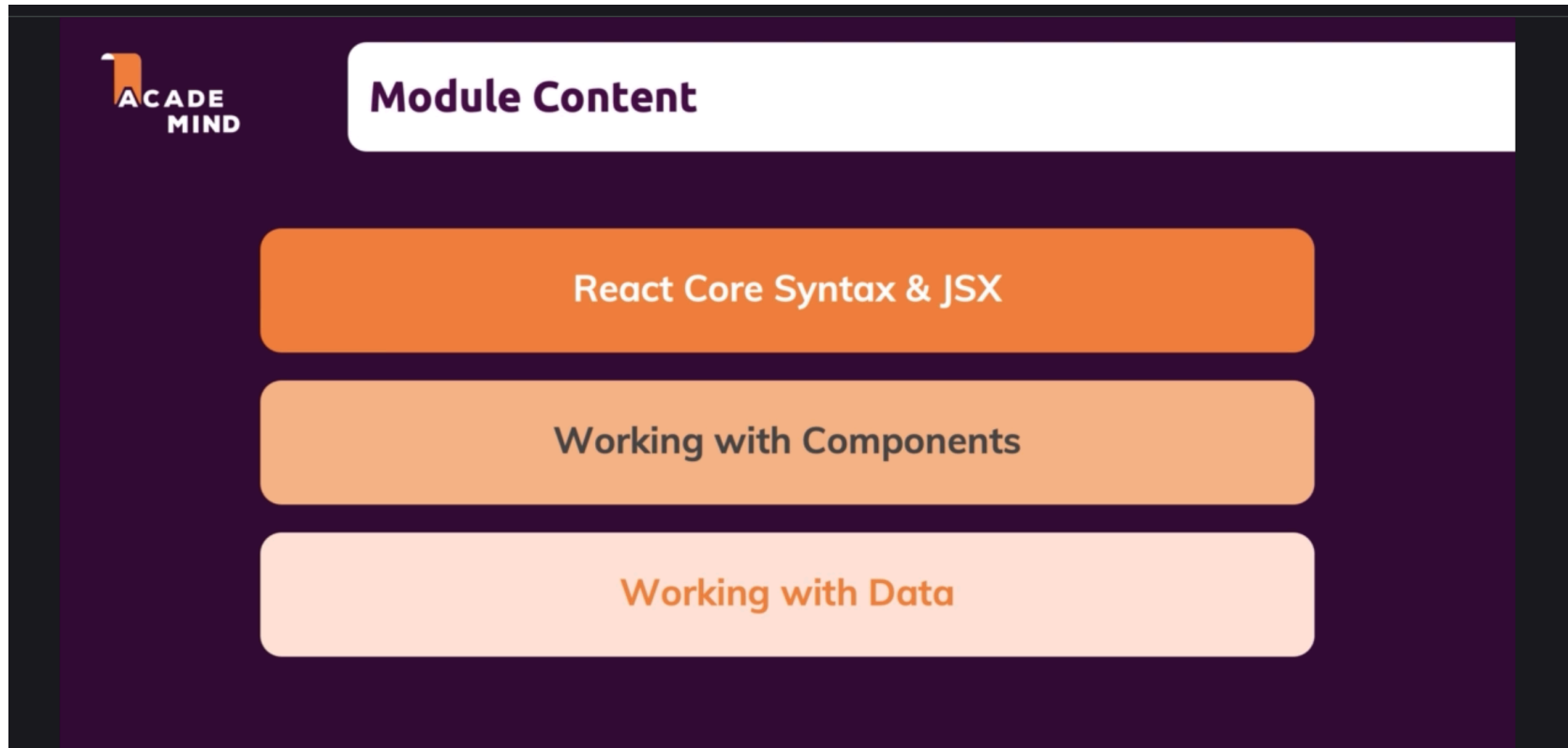
*July 28, 2021*

# Agenda

- Studio Review
- Kahoot
- Code walkthrough with slides
- Studio

# Summary

Composition is combining components



# Card Component

Composition is combining components.

Card is a container component.

Still a regular component, but it could return a div with a classname of card and goes with a card css.

Add some css in there to consolidate the styling.

Replace the built in div with the card component to get predefined styles.

```
function ExpenseItem(props) {  
  return (  
    <Card className='expense-item'>  
      <ExpenseDate date={props.date} />  
      <div className='expense-item__description'>  
        <h2>{props.title}</h2>  
        <div className='expense-item__price'>${props.amount}</div>  
      </div>  
    </Card>  
  );  
}
```

# Card Component

Children is reserved name that is available.

We also create a const for classes and add the css classes.

```
import "./Card.css";

const Card = (props) => {
  const classes = "card " + props.className;
  return <div className={classes}>{props.children}</div>;
};

export default Card;
```

# Organizing Component Files

Can create UI folder and Expenses Folder.

# Alternative Function Syntax

Can create Arrow functions.

Change all component functions to arrow functions.

```
const App = () => {  
}
```

# Learning Check

Question 1:

Which kind of code do you write when using React.js?

☐ Definitive JSX Code

☐ Imperative JavaScript Code

☒ Declarative JavaScript Code



# Learning Check

Question 2:

What is "JSX"?

- ☐ It's a standard JavaScript syntax
- ☒ It's a special, non-standard syntax which is enabled in React projects
- ☐ It's a special string which you can pass to React functions

# Learning Check

Question 3:

Why is React all about "Components"?

- ☒ Every UI in the end up is made up of multiple building blocks (= components), hence it makes sense to think about user interfaces as "combinations of components"
- ☐ React projects are configured to only work with components, hence you have to use them when writing React code.
- ☐ Components offer better performance than "standard user interfaces" that don't use components.

# Learning Check

Question 4:

What does "declarative" mean?

- ☐ "Declarative" is the same as "imperative"
- ☐ You define the individual steps that need to be taken to achieve a desired outcome (e.g. a target UI).
- ☒ You define the desired outcome (e.g. a target UI) and let the library (React) figure out the steps.

# Learning Check

Question 5:

What is a "React Component"?

- ☒ It's a JavaScript function which typically returns HTML (JSX) code that should be displayed.
- ☐ It's a replacement for standard HTML which is supported by modern browsers.
- ☐ It's a JavaScript function that must not return anything.

# Learning Check

Question 6:

How many custom React components must a React app have?

☐ At least 2

☐ At most 99

☒ That's totally up to you

# Learning Check

Question 7:

Which statement is correct?

- ☐ With React, you build multiple sibling component trees that are then mounted into the same DOM node.
- ☒ With React, you build a component tree with one root component that's mounted into a DOM node.
- ☐ With React, you always mount every component into it's own DOM node.

# Learning Check

Question 8:

What does "component tree" mean?

- ☒ It means that you have a root node which then has more components nested beneath it.
- ☐ It means that you must always return more than one component or HTML element per component function.
- ☐ It means that you can build multiple components.

# Learning Check

Question 9:

How do you pass data between components?

- ☐ Via global JavaScript variables that are accessible in all files
- ☒ Via "custom HTML attributes" (better known as "props")
- ☐ Via standard HTML attributes which you can use in non-React apps as well



# Learning Check

Question 10:

How can you output dynamic data in React components (i.e. in the returned JSX code)?

- ☒ You can use single curly braces (opening & closing) with any JS expression between them.
- ☐ React has a special syntax that allows you to output variable values (i.e. values stored in variables) and nothing else: Opening & closing curly braces
- ☐ You can't

# Listener

Want to have applications that are interactive – reactive application with state changes.  
Event handlers.

<https://developer.mozilla.org/en-US/docs/Web/API/HTMLButtonElement>

<https://developer.mozilla.org/en-US/docs/Web/API/Element#events>

React exposes default events as props which start with on. onClick takes a function.

```
<button  
  onClick={() => {  
    console.log("clicked");  
  }}  
>
```

Change Title

```
</button>
```

But don't want to get accrued away.

# Listener

```
const clickHandler = () => {  
  console.log("Clicked!");  
};
```

```
<button onClick={clickHandler}>Change Title</button>
```

Naming conventional is to end with handler when a function that is attached to an event listener and is invoked when an event occurs.

We now want to change what is showing up on the screen.

# React State

```
import './ExpenseItem.css';
import ExpenseDate from './ExpenseDate';
import Card from '../UI/Card';
const ExpenseItem = (props) => {
  const clickHandler = () => {
    title = "updated";
  };

  let title = props.title;

  return (
    <Card className="expense-item">
      <ExpenseDate date={props.date} />
      <div className="expense-item__description">
        <h2>{title}</h2>
        <div className="expense-item__price">${props.amount}</div>
      </div>
      <button onClick={clickHandler}>Change Title</button>
    </Card>
  );
}
```

# React State

Title does not update.

React does not work like this.

Component is a function. Only special thing is that it returns JSX. Who calls the function?

Under the hood, React is aware of components (functions). React evaluated the JSX code and invokes the component functions. Invokes all dependent functions.

Flow is started from index.js

React does not repeat the rendering.

To update what was visible on screen, we need to tell React that something has changed, and it needs to re-render that element.

Here is where state comes in the picture.

# React State

Need to import something from React library

```
import React, { useState } from "react";
```

Must be called inside function directly.

```
const ExpenseItem = (props) => {  
  useState();  
}
```

It takes in a variable and returns a function. Can use Array destructuring to always return two values.

```
const [title, setTitle] = useState(props.title);
```

Title is the current state value(variable), and second parameter is the name of the function to invoke to change it's value.

Name Convention is to use setVariable name.

# React State

By calling the function we are telling react that we updating a value.  
If data change is possible, u need state.

React will only update the component in which state was registered.

Separate state is created for every time the component is created.

State is on a per component instance.

We need to keep the value of each item for example, separate.

Using const because we are not assigning a new value. Value is updated elsewhere by React. We don't see the variable anywhere.

React keeps track of when the useState is called.


With state we can react to userInput and events.

# Form Inputs

Title

Amount

Date



Add expense

August 14 2020	Toilet Paper	\$94.12	Change Title
March 12 2021	New TV	\$799.49	Change Title
March 28 2021	Car Insurance	\$294.67	Change Title



# Listening to user Inputs

```
You, seconds ago | 1 author (You)
import "../ExpenseForm.css";
import React from "react";

const ExpenseForm = () => {
  const titleChangeHandler = (event) => {
    console.log(event.target.value);
  };
  return (
    <form>
      <div className="new-expense__controls">
        <div className="new-expense__control">
          <label>Title</label>
          <input type="text" onChange={titleChangeHandler} />
        </div>
        <div className="new-expense__control">
          <label>Amount</label>
          <input type="number" min="0.01" step="0.01" />
        </div>
        <div className="new-expense__control">
```

# Listening to user Inputs

```
import './ExpenseForm.css';
import React, { useState } from 'react';

const ExpenseForm = () => {
  const [enteredTitle, setEnteredTitle] = useState('');

  const titleChangeHandler = (event) => {
    setEnteredTitle(event.target.value);
  };

  const [enteredAmount, setEnteredAmount] = useState('');

  const amountChangeHandler = (event) => {
    setEnteredAmount(event.target.value);
  };

  const [enteredDate, setEnteredDate] = useState('');

  const dateChangeHandler = (event) => {
    setEnteredDate(event.target.value);
  };
}
```

# Using one state instead

```
import './ExpenseForm.css';
import React, { useState } from 'react';

const ExpenseForm = () => {
  // const [enteredTitle, setEnteredTitle] = useState('');
  const [userInput, setUserInput] = useState({
    enteredTitle: '',
    enteredAmount: '',
    enteredDate: '',
  });
  const titleChangeHandler = (event) => {
    // setEnteredTitle(event.target.value);
    setUserInput({
      ...userInput,
      enteredTitle: event.target.value,
    });
  };

  // const [enteredAmount, setEnteredAmount] = useState('');
```

# Updating state that depends on previous state

Not preferred to depend on previous state snapshot

Because React schedules state updates

By using `prevState` as parameter to function then it's more guaranteed to have updated values.

# Updating state that depends on previous state

```
import './ExpenseForm.css';
import React, { useState } from 'react';

const ExpenseForm = () => {
  // const [enteredTitle, setEnteredTitle] = useState("");
  const [userInput, setUserInput] = useState({
    enteredTitle: "",
    enteredAmount: "",
    enteredDate: "",
  });
  const titleChangeHandler = (event) => {
    // setEnteredTitle(event.target.value);
    setUserInput((prevState) => {
      return { ...prevState, enteredTitle: event.target.value };
    });
  };

  // const [enteredAmount, setEnteredAmount] = useState("");
```

# Handling Form Submission

Button type submit then form will emit submit event.

