the Master Course

{CODENATION}

Working with State.

{CODENATION}

First things first

On a new CodePen (I'll send you one set up) create a main class component called Menu and a functional component called Fooditem.

Render the functional component 3 times, inside the class component, and pass down a type and cost prop.

{CODENATION}

Learning Objectives

To understand what state is and how we can work with it.

To be able to include methods in class components.

React.js

props

Getting information from outside our components.

{CODENATION}

React.js

State

Getting information from inside our components (the data is managed from inside the component).

State is used inside class components.

Remember, we should try to use functional components as often as possible – and use state with care.

Remember that javascript classes can have properties. State is a special property. Every time React detects the state has been changed, this triggers a re-render of components which have changed.

State is an object.

```
class App extends React.Component {
  state = {

  }


  render(){
    return(
        <div>
          <Person />
          <Person />
          <Person />
        </div>
    )
  }
}
```

React.js

Remember how we use the object structure. As state is an object we can store information as key-value pairs.

```
class App extends React.Component {
  state = {
    persons: [
      {name: "Dan", age: 33, pet: "polly"},
      {name: "Ben", age: 21},
      {name: "stuart", age: "30-something"}
    ]
  }


  render(){
    return(
        <div>
          <Person />
          <Person />
          <Person />
        </div>
    )
  }
}
```


In the state here I have a persons property which holds an array (made up of three objects).

If I wanted to pass my name as a prop to a Person component, what would I write?

```
class App extends React.Component {
  state = {
    persons: [
      {name: "Dan", age: 33, pet: "polly"},
      {name: "Ben", age: 21},
      {name: "stuart", age: "30-something"}
    ]
  }


  render(){
    return(
        <div>
          <Person name = {this.state.persons[0].name}/>
          <Person />
          <Person />
        </div>
    )
  }
```

Functional components are sometimes known as stateless components.

Class components are sometimes known as stateful components.

We can change the state of our components using methods. Let's have a look we might do that.

this.setState( )

Inside setState( ) we put the state property we wish to change. This change is merged with the current state, it does not replace the current state.

```
class App extends React.Component {
  state = {
    persons: [
      {name: "Dan", age: 33, pet: "polly"},
      {name: "Ben", age: 21},
      {name: "stuart", age: "30-something"}
    ]
  }

switchNameHandler = () => {
  this.setState({persons:  [
      {name: "Buzz Lightyear", age: 33, pet: "polly"},
      {name: "Ben", age: 21},
      {name: "stuart", age: "30-something"}
    ]})
};
  render(){
    return(
        <div>
          <Person
            name = {this.state.persons[0].name}/>
          <Person />
          <Person />
          <button onClick = {this.switchNameHandler}> I'm a button </button>
        </div>
    )
```

React.js

In our handler method we called the setState( ) method. Remember that this refers to the class.

```
class App extends React.Component {
  state = {
    persons: [
      {name: "Dan", age: 33, pet: "polly"},
      {name: "Ben", age: 21},
      {name: "stuart", age: "30-something"}
    ]
  }

switchNameHandler = () => {
  this.setState({persons:  [
      {name: "Buzz Lightyear", age: 33, pet: "polly"},
      {name: "Ben", age: 21},
      {name: "stuart", age: "30-something"}
    ]})
};
  render(){
    return(
        <div>
          <Person
            name = {this.state.persons[0].name}/>
          <Person />
          <Person />
          <button onClick = {this.switchNameHandler}> I'm a button </button>
        </div>
    )
```

React.js

We don't use brackets on this.switchNameHandler as this would call the function instantly when it is initially rendered.

{CN}®

However, sometimes we will want to pass an argument in our onClick, and this is a little bit different. Let's have a look at that.

```
class App extends React.Component {
  state = {
    persons: [
      {name: "Dan", age: 33, pet: "polly"},
      {name: "Ben", age: 21},
      {name: "stuart", age: "30-something"}
    ]
  }

switchNameHandler = (newName) => {
  this.setState({persons:  [
      {name: newName, age: 33, pet: "polly"},
      {name: "Ben", age: 21},
      {name: "stuart", age: "30-something"}
    ]})
};
  render(){
    return(
        <div>
          <Person
            name = {this.state.persons[0].name}/>
          <Person />
          <Person />
          <button onClick = {() => this.switchNameHandler("Briony")}> I'm a
            button </button>
        </div>
```

React.js

We use brackets here when passing an argument. We are passing an anonymous function which returns our method call.

{CN}®

```
class App extends React.Component {
  state = {
    persons: [
      {name: "Dan", age: 33, pet: "polly"},
      {name: "Ben", age: 21},
      {name: "stuart", age: "30-something"}
    ]
  }

switchNameHandler = (newName) => {
  this.setState({persons:  [
      {name: newName, age: 33, pet: "polly"},
      {name: "Ben", age: 21},
      {name: "stuart", age: "30-something"}
    ]})
};
  render(){
    return(
        <div>
          <Person
            name = {this.state.persons[0].name}/>
          <Person />
          <Person />
          <button onClick = {() => this.switchNameHandler("Briony")}> I'm a
            button </button>
        </div>
```

React.js

This is passed to our method, which we can then use as a variable anywhere in our function, including setState( ).

{CN}®

We can put a lot of different information in our **state object**.

We are now going to look at rendering dynamic content. Basically rendering a component, when a certain condition is met.

In our state, we could add the showPersons property and set it equal to false.

```
class App extends React.Component {
  state = {
    persons: [
      {name: "Dan", age: 33, pet: "polly"},
      {name: "Ben", age: 21},
      {name: "stuart", age: "30-something"}
    ],
    showPersons: false
  }
```

We can render content conditionally in a few different ways. I'll show you two. Let's have a look!

So we created a togglePersonsHandler method which set the state to the opposite of what was currently stored there. So if the current state was false, it would change it to true and vice versa.

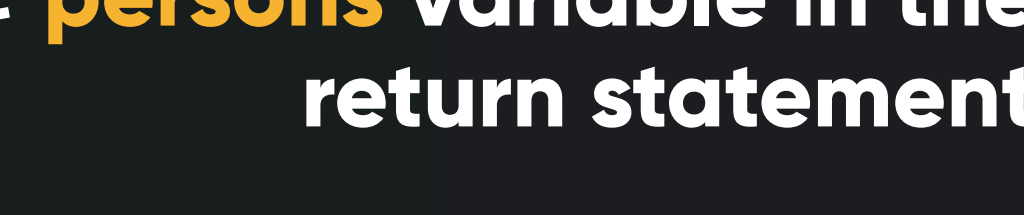
We can put javascript logic inside our render( ) method. This will be fired every time a re-render is triggered. And remember, a re-render happens every time the state or props change.

```
togglePersonsHandler = () => {
  const show = this.state.showPersons;
  this.setState({showPersons: !show})
}
```

This method stores the current state of showPersons in a variable. Then when the function runs, it sets the state to the opposite.

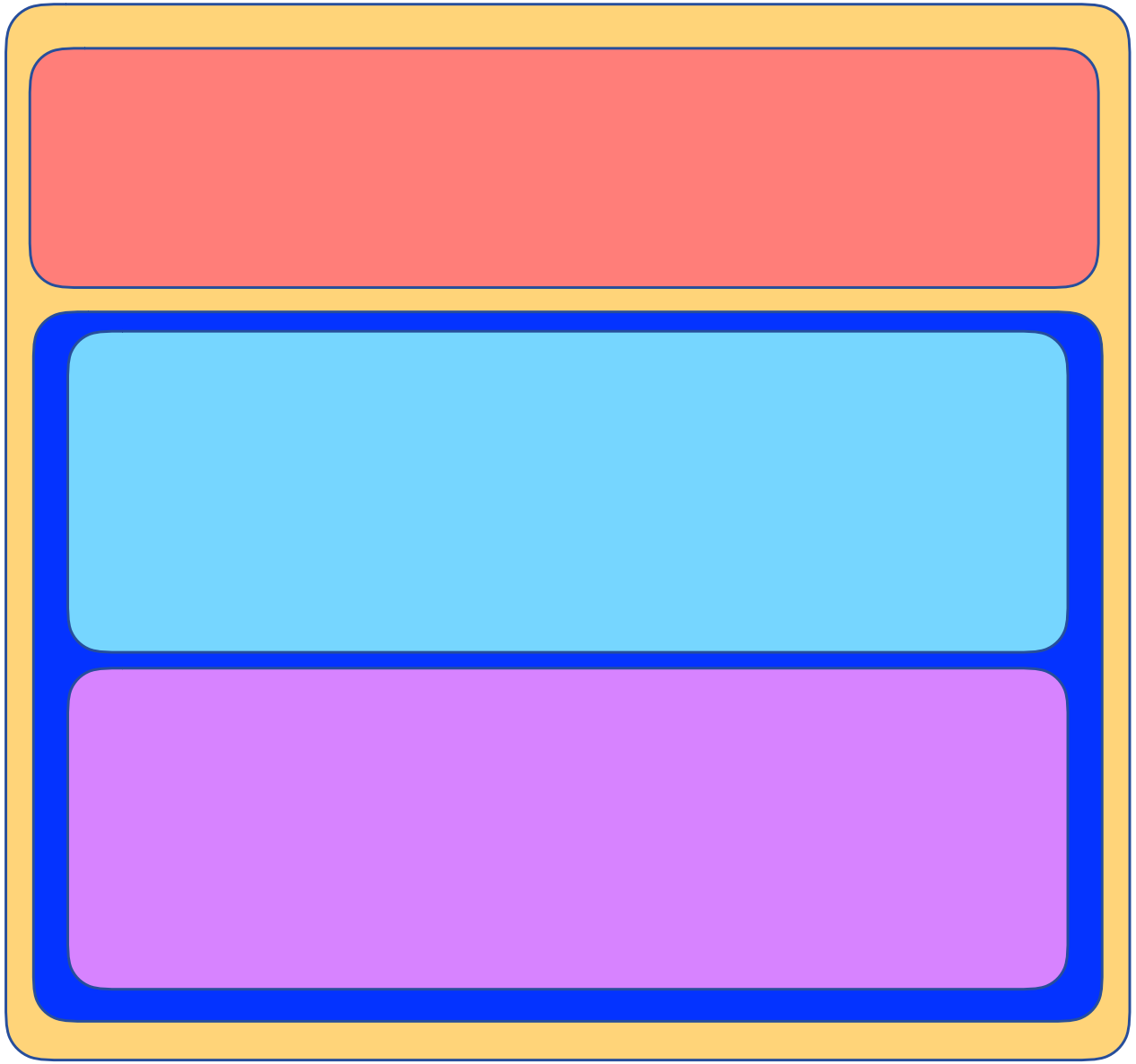
```
render(){

  let persons = null;

  if (this.state.showPersons == true){
      persons = (<div>
                   <Person
                   name = {this.state.persons[0].name}/>
                 <Person />
                 <Person />
                  </div>)
      }

  return(
      <div>
        {persons}
        <button onClick = {this.togglePersonsHandler}> toggle persons
        </button>
        <button onClick = {() => this.switchNameHandler("Briony")}> I'm a
          button </button>
      </div>
  )
```

We can put javascript functionality inside the render method. In this case, notice that we include the persons variable in the return statement



{CODENATION}

Class Components



Properties and methods go here

Render( ) method

JS logic can go here.

return statement

We can also include more JS code in our return statement. To get the same functionality for our toggle method, we could have used a ternary operator.

If we use JS in our return statement, we need to **wrap it in curly brackets**. I'll show your a ternary statement in action.

React.js

Task one:

In VSCode create a button which adds 1 to (increments) a counter every time you click it.

Task Two:

Add a button which decrements the counter.

{CN}®

React.js

Task Three:

Let's create a simple desktop calculator using React.

{CN}®

Revisiting Learning Objectives

To understand what state is and how we can work with it.

To be able to include methods in class components.