Page-04-Score Keeper App

What are React Components?

- they are individual chunks of data
- it keeps everthing in order
- we are going to be breaking down our app into 4 components
  - <TitleBar title ="Score Keep"/>
  - <Player />
  - <PlayerList />
  - <AddPlayer />
  - <App /> <----parent component that renders all components

ES6 Aside - Classes Part 1

- we capitalize the first letter when first defining the class
- head over to server main.js

```
import {Meteor} from 'meteor/meteor';
import {Players} from './../imports/api/players';

Meteor.startup(()=>{
 class Person{

 }
});
```

- now let's create a new insteance of Person

```
import {Meteor} from 'meteor/meteor';
import {Players} from './../imports/api/players';

Meteor.startup(()=>{
 class Person{

 }
 let me = new Person();
});
```

- we need a constructor to make an instance of Person

```
import {Meteor} from 'meteor/meteor';
import {Players} from './../imports/api/players';

Meteor.startup(()=>{
 class Person{
    constructor(name){
        this.name = name;
    }
 }
 let me = new Person('Andrew');
 console.log(me);
});
```

- we can also setup default values, if we don't pass an argument to the Person instance the default will appear

```
import {Meteor} from 'meteor/meteor';
import {Players} from './../imports/api/players';

Meteor.startup(()=>{
 class Person{
    constructor(name = 'Anonymous'){
        this.name = name;
    }
 }
 let me = new Person();
 console.log(me);
});
```

- Now we are going to create methods in our class Person
- this method is going to return a string, and we are going to make this happened by using ES6 template strings
- We are going to be using ` back ticks instead of quotes''

```
import {Meteor} from 'meteor/meteor';
import {Players} from './../imports/api/players';

Meteor.startup(()=>{
```

```
 class Person{
    constructor(name = 'Anonymous'){
        this.name = name;
    }
    getGreeting(){
        return `Hi I am ${this.name}.`;
    }
 }
 let me = new Person('Jose');
 console.log(me.getGreeting());
});
```

- as you can see we used ${} to inject js expressions
- And down below we just call it by doing me.getGreeting
- now we are going to be adding the age argument

ES6 Aside - Classes Part II

- we are going to creating sub classes by extending classes

```
import {Meteor} from 'meteor/meteor';
import {Players} from './../imports/api/players';

Meteor.startup(()=>{
 class Person{
    constructor(name = 'Anonymous', age = 0){
        this.name = name;
        this.age = age;
    }
    getGreeting(){
        return `Hi I am ${this.name}.`;
    }
    getPersonDescription(){
        return `${this.name} is ${this.age} year(s) old`
    }
 }

 class Employee extends Person{
```

```
}
```

- the above example demonstrates that Employee will now have everthing Person has
- our Employee class will take 3 arguments

```
class Employee extends Person{
    constructor(name, age, title){
        this.title = title;
    }
}
```

- We are not going to be defining name and title instead we are going to be calling our parents constructor function
- super() allows you to call the parent constructor

```
class Employee extends Person{
    constructor(name, age, title){
        super(name, age);
        this.title = title;
    }
    hasJob(){
        return !!this.title;
    }
}
let me = new Employee('Jose', 37, 'db admin');
console.log(me.getGreeting());
console.log(me.hasJob());

let person = new Employee('Andrew');
console.log(person.getPersonDescription());
console.log(person.hasJob());
```

- **the** hasJob() method will return false if an instance of Employee is created but no title argument is passed on and true if an argument is passes on
- we can also overide methods by just creating it again

```
import {Meteor} from 'meteor/meteor';
import {Players} from './../imports/api/players';
```

```
Meteor.startup(()=>{
 class Person{
    constructor(name = 'Anonymous', age = 0){
        this.name = name;
        this.age = age;
    }
    getGreeting(){
        return `Hi I am ${this.name}.`;
    }
    getPersonDescription(){
        return `${this.name} is ${this.age} year(s) old`
    }
 }

 class Employee extends Person{
    constructor(name, age, title){
        super(name, age);
        this.title = title;
    }
    getGreeting(){

    }
    hasJob(){
        return !!this.title;
    }
 }
 let me = new Employee('Jose', 37, 'db admin');
 console.log(me.getGreeting());
 console.log(me.hasJob());

 let person = new Employee('Andrew');
 console.log(person.getPersonDescription());
 console.log(person.hasJob());
});
```

- in the getGreeting() we are going to check if there is a greeting, if there is there will be a string, if not the getGreeting from the parent class will be called

```
  class Employee extends Person{
    constructor(name, age, title){
        super(name, age);
        this.title = title;
    }
    getGreeting(){
        if(this.title){
            return `Hi I am ${this.name}.  I work as a ${this.title}.`;
        }else{
            return super.getGreeting();
        }
    }
    hasJob(){
        return !!this.title;
    }
 }
 let me = new Employee('Jose', 37, 'db admin');
 console.log(me.getGreeting());

 let person = new Employee('Andrew');
 console.log(person.getGreeting());
});
```

- now lets create another class called Programmer that extends from Person, and it overwrites the getGreeting() method

```
class Programmer extends Person{
    constructor(name, age, preferredLanguage = 'assembly'){
        super(name, age);
        this.preferredLanguage = preferredLanguage;
    }
    getGreeting(){
        return `Hi I am ${this.name}.  I am a ${this.preferredLanguage} developer`;
    }
}
let userOne = new Programmer('Jose', 38, 'Java');
```

```
console.log(userOne.getGreeting())
});
```

Your First React Component

- first delete everything from server main.js
- head over to client main.js and delete a few things and make it look like this

```
Meteor.startup( () => {


  Tracker.autorun(() => {
    let players = Players.find().fetch();
    let title = 'Score Keep';
    let jsx = (
    <div>
      <h1>{title}</h1>
      {renderPlayers(players)}
      <form onSubmit = {handleSubmit}>
        <input type="text" name="playerName" placeholder="Player name"/>
        <button>Add Player</button>
      </form>
    </div>
    );
    ReactDOM.render(jsx, document.getElementById('app'))
  });
});
```

- we are going to start with the title component
- lets create a class in the client main.js

```
class TitleBar extends React.Component{
  render(){
    return(
      <div>
        <h1>My App Name</h1>
      </div>
    );
  };
```

```
}
```

- by default my react components only need to define one method render() that will return the jsx that the component should render to the screen. and above we have our very first component
- And we can add our TitleBar component down below

```
Meteor.startup( () => {


  Tracker.autorun(() => {
    let players = Players.find().fetch();
    let title = 'Score Keep';
    let jsx = (
    <div>
      <TitleBar />
      {renderPlayers(players)}
      <form onSubmit = {handleSubmit}>
        <input type="text" name="playerName" placeholder="Player name"/>
        <button>Add Player</button>
      </form>
    </div>
    );
    ReactDOM.render(jsx, document.getElementById('app'))
  });
});
```

- lets create a new folder in imports called ui
- and we are going to be creating a file for each component, lets start with TitleBar.js, lets take the class TitleBar from the client main.js and paste it on TitleBar.js
- we are going to be importing React to this file
- we also need to export our class so it can be used in client main.js
- and this will extend the React.Component

```
import React from 'react';


export default class TitleBar extends React.Component{
    render(){
      return(
        <div>
          <h1>My App Name</h1>
```

```
        </div>
      );
    };
  }
```

- and now let's go to client main.js and import this class we are exporting

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Meteor} from 'meteor/meteor';
import {Tracker} from 'meteor/tracker';

import {Players} from './../imports/api/players';
import TitleBar from './../imports/ui/TitleBar'
```

- now lets do another component called AddPlayer.js under ui

```
import React from 'react';

export default class AddPlayer extends React.Component{
    render(){
        return(
            <div>
                <p>AddPlayer will go here</p>
            </div>
        );
    }
}
```

- now we can use this component in our client main.js
- lets import it first

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Meteor} from 'meteor/meteor';
import {Tracker} from 'meteor/tracker';
```

```
import {Players} from './../imports/api/players';
import TitleBar from './../imports/ui/TitleBar'
import AddPlayer from './../imports/ui/AddPlayer';
```

- then down below lets include our component

```
Meteor.startup( () => {

  Tracker.autorun(() => {
    let players = Players.find().fetch();
    let title = 'Score Keep';
    let jsx = (
    <div>
      <TitleBar />
      {renderPlayers(players)}
      <AddPlayer />
      <form onSubmit = {handleSubmit}>
        <input type="text" name="playerName" placeholder="Player name"/>
        <button>Add Player</button>
      </form>
    </div>
    );
    ReactDOM.render(jsx, document.getElementById('app'))
  });
});
```

Props, Prop Types, and Prop Defaults

- we are going to be passing a prop to our TitleBar Component

```
Meteor.startup( () => {

  Tracker.autorun(() => {
    let players = Players.find().fetch();
    let title = 'Score Keep';
    let jsx = (
```

```
    <div>
      <TitleBar title={title}/>
      {renderPlayers(players)}
      <AddPlayer />
      <form onSubmit = {handleSubmit}>
        <input type="text" name="playerName" placeholder="Player name"/>
        <button>Add Player</button>
      </form>
    </div>
    );
    ReactDOM.render(jsx, document.getElementById('app'))
  });
});
```

- now head over to AddPlayer.js and change the code

```
import React from 'react';

export default class TitleBar extends React.Component{
    render(){
      return(
        <div>
          <h1>{this.props.title}</h1>
        </div>
      );
    };
  }
```

- we can also set default value to props and specify the type a prop should be, example like a string , function, number ets.. for more information about this go to google and search for **react type checking props**
- so right below our AddPlayer class lets define the type of the title

```
import React from 'react';

export default class TitleBar extends React.Component{
    render(){
      return(
```

```
        <div>
          <h1>{this.props.title}</h1>
        </div>
      );
    };
  }

  TitleBar.propTypes = {
      title: React.PropTypes.string.isRequired
  };
```

- this will throw a warning if the title is not a string
- we can also set a default prop, which means if a prop type is not provided it'll default when we add this

```
import React from 'react';

export default class TitleBar extends React.Component{
    render(){
      return(
        <div>
          <h1>{this.props.title}</h1>
        </div>
      );
    };
  }

  TitleBar.propTypes = {
      title: React.PropTypes.string
  };

  TitleBar.defaultProps = {
      title : 'Default Title'
  }
```

- and now if we remove the title from the client main.js

```
Meteor.startup( () => {
```

```
  Tracker.autorun(() => {
    let players = Players.find().fetch();
    let title = 'Score Keep';
    let jsx = (
    <div>
      <TitleBar/><--deleted title={title}
      {renderPlayers(players)}
      <AddPlayer />
      <form onSubmit = {handleSubmit}>
        <input type="text" name="playerName" placeholder="Player name"/>
        <button>Add Player</button>
      </form>
    </div>
    );
    ReactDOM.render(jsx, document.getElementById('app'))
  });
});
```

- but for now we are going to leave everthing how it is ...

```
import React from 'react';

export default class TitleBar extends React.Component{
    render(){
      return(
        <div>
          <h1>{this.props.title}</h1>
        </div>
      );
    };
  }

  TitleBar.propTypes = {
      title: React.PropTypes.string.isRequired
  };

  TitleBar.defaultProps = {
```

```
    //   title : 'Default Title'
  }
```

- and in client main.js

```
Meteor.startup( () => {


  Tracker.autorun(() => {
    let players = Players.find().fetch();
    let title = 'Score Keep';
    let jsx = (
    <div>
      <TitleBar title={title}/>
      {renderPlayers(players)}
      <AddPlayer />
      <form onSubmit = {handleSubmit}>
        <input type="text" name="playerName" placeholder="Player name"/>
        <button>Add Player</button>
      </form>
    </div>
    );
    ReactDOM.render(jsx, document.getElementById('app'))
  });
});
```

- lets create a subtitle
- lets head over to clent main.js

```
Meteor.startup( () => {


  Tracker.autorun(() => {
    let players = Players.find().fetch();
    let title = 'Score Keep';
    let subTitle = 'This is my SubTitle';
    let jsx = (
    <div>
      <TitleBar title={title} subtitle={subTitle}/>
      {renderPlayers(players)}
```

```
      <AddPlayer />
      <form onSubmit = {handleSubmit}>
        <input type="text" name="playerName" placeholder="Player name"/>
        <button>Add Player</button>
      </form>
    </div>
    );
    ReactDOM.render(jsx, document.getElementById('app'))
  });
});
```

- and now let's head over to TitleBar.js and add it to our class

```
import React from 'react';

export default class TitleBar extends React.Component{
    render(){
      return(
        <div>
          <h1>{this.props.title}</h1>
          <h2>{this.props.subtitle}</h2>
        </div>
      );
    };
  }

  TitleBar.propTypes = {
      title: React.PropTypes.string.isRequired,
      subtitle: React.PropTypes.string.isRequired
  };
```

Prop Type Update

- we are going to grab a new package from npm
- go to the terminal and in a new tab

```
meteor npm install prop-types@15 --save
```

- head over to TitleBar.js
- import it
- and down below instead of doing React.PropTypes, we can remove React and leave PropTypes

```
import React from 'react';
import PropTypes from 'prop-types';

export default class TitleBar extends React.Component{
    render(){
      return(
        <div>
          <h1>{this.props.title}</h1>
          <h2>{this.props.subtitle}</h2>
        </div>
      );
    };
  }

  TitleBar.propTypes = {
      title: PropTypes.string.isRequired,
      subtitle: PropTypes.string.isRequired
  };
```

Custom Component Methods

- head over to client main.js and cut and paste the form to AddPlayer.js and for now let's just add an empty arrow function to the onSubmit event

```
import React from 'react';

export default class AddPlayer extends React.Component{
    render(){
        return(
            <div>
                <form onSubmit = {() => {}}>
                    <input type="text" name="playerName" placeholder="Player name"/>
                    <button>Add Player</button>
                </form>
```

```
            </div>
        );
    }
}
```

- head back over to client main.js and cut out the handleSubmit function and paste above the AddPlayer class
- we are not leaving there but for now we are going to use it as a reference

```
import React from 'react';


const handleSubmit = (e) => {
    let playerName = e.target.playerName.value;
    e.preventDefault();
    if(playerName){
      e.target.playerName.value = '';
      Players.insert({
        name: playerName,
        score: 0
      });
    }
  }


export default class AddPlayer extends React.Component{
    render(){
        return(
            <div>
                <form onSubmit = {() => {}}>
                    <input type="text" name="playerName" placeholder="Player name"/>
                    <button>Add Player</button>
                </form>
            </div>
        );
    }
}
```

- let's also import our name export Players from a local file

```
import React from 'react';
import {Players} from './../api/players';

const handleSubmit = (e) => {
    let playerName = e.target.playerName.value;
    e.preventDefault();
    if(playerName){
      e.target.playerName.value = '';
      Players.insert({
        name: playerName,
        score: 0
      });
    }
  }

export default class AddPlayer extends React.Component{
    render(){
        return(
            <div>
                <form onSubmit = {() => {}}>
                    <input type="text" name="playerName" placeholder="Player name"/>
                    <button>Add Player</button>
                </form>
            </div>
        );
    }
}
```

- we are now going to create a custom method in our component

```
import React from 'react';
import {Players} from './../api/players';

const handleSubmit = (e) => {
    let playerName = e.target.playerName.value;
    e.preventDefault();
    if(playerName){
```

```
      e.target.playerName.value = '';
      Players.insert({
        name: playerName,
        score: 0
      });
    }
  }

export default class AddPlayer extends React.Component{
    handleSubmit(e){

    }
    render(){
        return(
            <div>
                <form onSubmit = {this.handleSubmit}>
                    <input type="text" name="playerName" placeholder="Player name"/>
                    <button>Add Player</button>
                </form>
            </div>
        );
    }
}
```

- cut and paste what we have in the handleSubmit outside the class inside our new handleSubmit() and remove the old function

```
import React from 'react';
import {Players} from './../api/players';



export default class AddPlayer extends React.Component{
    handleSubmit(e){
        let playerName = e.target.playerName.value;
        e.preventDefault();
        if(playerName){
          e.target.playerName.value = '';
```

```
            Players.insert({
              name: playerName,
              score: 0
            });
        }
    }
    render(){
        return(
            <div>
                <form onSubmit = {this.handleSubmit}>
                    <input type="text" name="playerName" placeholder="Player name"/>
                    <button>Add Player</button>
                </form>
            </div>
        );
    }
}
```

- over at client main.js add a prop to the component <AddPlayer>

```
Meteor.startup( () => {

  Tracker.autorun(() => {
    let players = Players.find().fetch();
    let title = 'Score Keep';
    let subTitle = 'This is my SubTitle';
    let jsx = (
    <div>
      <TitleBar title={title} subtitle={subTitle}/>
      {renderPlayers(players)}
      <AddPlayer score={10}/>

    </div>
    );
    ReactDOM.render(jsx, document.getElementById('app'))
  });
});
```

- and now over to AddPlayer.js change the score key value

```
import React from 'react';
import {Players} from './../api/players';



export default class AddPlayer extends React.Component{
    handleSubmit(e){
        let playerName = e.target.playerName.value;
        e.preventDefault();
        if(playerName){
          e.target.playerName.value = '';
          Players.insert({
            name: playerName,
            score: this.props.score
          });
        }
    }
```

- the above is just to illustrate that this will cause an error
- in this example we are losing the this. binding , the this binding is no longer refering to the instance of the AddPlayer component, it refers to the global 'window' object
- we are going to do an example first and then implemented in our program
- head over to server main.js

```
import {Meteor} from 'meteor/meteor';
import {Players} from './../imports/api/players';


Meteor.startup(()=>{
    let obj =  {
        name: 'Andrew',
        printName(){
            console.log(`Name: ${this.name}`);
        }
    }
    setTimeout(obj.printName, 1000)
});
```

- the above will produce an error undefined, to fix this we need the bind method

```
import {Meteor} from 'meteor/meteor';
import {Players} from './../imports/api/players';

Meteor.startup(()=>{
    let obj =  {
        name: 'Andrew',
        printName(){
            console.log(`Name: ${this.name}`);
        }
    }
    setTimeout(obj.printName.bind(obj), 1000)
});
```

- delete the above example
- head over to AddPlayer.js

```
import React from 'react';
import {Players} from './../api/players';



export default class AddPlayer extends React.Component{
    handleSubmit(e){
        let playerName = e.target.playerName.value;
        e.preventDefault();
        if(playerName){
          e.target.playerName.value = '';
          Players.insert({
            name: playerName,
            score: this.props.score
          });
        }
    }
    render(){
        return(
            <div>
                <form onSubmit = {this.handleSubmit.bind(this)}> <--(this)refers to the render()
                    <input type="text" name="playerName" placeholder="Player name"/>
```

```
                    <button>Add Player</button>
                </form>
            </div>
        );
    }
}
```

- in the screen you should a default of 10 on score when adding a new player
- we are going to put it back to 0

```
import React from 'react';
import {Players} from './../api/players';



export default class AddPlayer extends React.Component{
    handleSubmit(e){
        let playerName = e.target.playerName.value;
        e.preventDefault();
        if(playerName){
          e.target.playerName.value = '';
          Players.insert({
            name: playerName,
            score: 0
          });
        }
    }
    render(){
        return(
            <div>
                <form onSubmit = {this.handleSubmit.bind(this)}>
                    <input type="text" name="playerName" placeholder="Player name"/>
                    <button>Add Player</button>
                </form>
            </div>
        );
    }
}
```

- and over at client main.js remove the prop from <AddPlayer/>

```
Meteor.startup( () => {

  Tracker.autorun(() => {
    let players = Players.find().fetch();
    let title = 'Score Keep';
    let subTitle = 'This is my SubTitle';
    let jsx = (
    <div>
      <TitleBar title={title} subtitle={subTitle}/>
      {renderPlayers(players)}
      <AddPlayer />

    </div>
    );
    ReactDOM.render(jsx, document.getElementById('app'))
  });
});
Meteor.startup( () => {

  Tracker.autorun(() => {
    let players = Players.find().fetch();
    let title = 'Score Keep';
    let subTitle = 'This is my SubTitle';
    let jsx = (
    <div>
      <TitleBar title={title} subtitle={subTitle}/>
      {renderPlayers(players)}
      <AddPlayer />

    </div>
    );
    ReactDOM.render(jsx, document.getElementById('app'))
  });
});
```