Page-05-Creating the Player Component

- create the Player.js under ui folder
- and import react , and Players
- then cut and paste what we have in the client main.js
- and replace everything with *this.props.player*

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

export default class Player extends React.Component{
    render(){
        return(
     <p key={this.props.player._id}>
      {this.props.player.name} has {this.props.player.score} point(s).
      <button onClick={()=>Players.update(this.props.player._id, {$inc:{score:1}})}>+1</button>
      <button onClick={()=>Players.update(this.props.player._id, {$inc:{score:-1}})}>-1</button>
      <button onClick={()=> Players.remove(this.props.player._id)}>X</button>
    </p>
        )
    }
}
```

- and now at client main.js include the component <Player />

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Meteor} from 'meteor/meteor';
import {Tracker} from 'meteor/tracker';
import Player from './../imports/ui/Player';
import {Players} from './../imports/api/players';
import TitleBar from './../imports/ui/TitleBar';
import AddPlayer from './../imports/ui/AddPlayer';

const renderPlayers = (playerList) => {
  return playerList.map((player) => {
    return (
      <Player key={player._id} player={player}/>
```

```
    );
  });
};
```

- the last thing we need to do is set up a prop type to an object, and we also need to import prop types

```
import React from 'react';
import {Players} from './../api/players';
import PropTypes from 'prop-types';

export default class Player extends React.Component{
    render(){
        return(
     <p key={this.props.player._id}>
      {this.props.player.name} has {this.props.player.score} point(s).
      <button onClick={()=>Players.update(this.props.player._id, {$inc:{score:1}})}>+1</button>
      <button onClick={()=>Players.update(this.props.player._id, {$inc:{score:-1}})}>-1</button>
      <button onClick={()=> Players.remove(this.props.player._id)}>X</button>
    </p>
        )
    }
}

Player.propTypes = {
    player: PropTypes.object.isRequired
}
```

List Based Component

- now lets make another component, under ui folder create a file called PlayerList.js

```
import React from 'react';

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

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

- now lets go to client main.js, and lets import this component so we can view it on the browser

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Meteor} from 'meteor/meteor';
import {Tracker} from 'meteor/tracker';
import Player from './../imports/ui/Player';
import {Players} from './../imports/api/players';
import TitleBar from './../imports/ui/TitleBar';
import AddPlayer from './../imports/ui/AddPlayer';
import PlayerList from './../imports/ui/PlayerList'

const renderPlayers = (playerList) => {
  return playerList.map((player) => {
    return (
      <Player key={player._id} player={player}/>
    );
  });
};




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)}
      <PlayerList />
      <AddPlayer />

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

- now lets pass the players prop to <PlayerList />
- delete {renderPlayers(players)} above it
- and cut the renderPlayers function to PlayerList.js file
- and this is how your main.js client should look like for now

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Meteor} from 'meteor/meteor';
import {Tracker} from 'meteor/tracker';
import Player from './../imports/ui/Player';
import {Players} from './../imports/api/players';
import TitleBar from './../imports/ui/TitleBar';
import AddPlayer from './../imports/ui/AddPlayer';
import PlayerList from './../imports/ui/PlayerList'

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}/>
      <PlayerList players={players}/>
      <AddPlayer />
```

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

- head on over to PlayerList.js
- import Player
- modify the function with this and inserted in our component

```
import React from 'react';
import Player from './Player';


export default class PlayerList extends  React.Component{
    renderPlayers(){
        return this.props.players.map((player) => {
            return (
              <Player key={player._id} player={player}/>
            );
          });
    }
    render(){
        return (
            <div>
               {this.renderPlayers()}
            </div>
        );
    }
};
```

- then we want to add new prop types definition

```
import React from 'react';
import Player from './Player';


export default class PlayerList extends  React.Component{
```

```
    renderPlayers(){
        return this.props.players.map((player) => {
            return (
              <Player key={player._id} player={player}/>
            );
          });
    }
    render(){
        return (
            <div>
               {this.renderPlayers()}
            </div>
        );
    }
};

PlayerList.propTypes = {
    players : React.PropTypes.array.isRequired
}
```

Conditional Rendering with JSX

- go to PlayerList.js we are going to be adding conditionals to our app
- we are going to check whether the PlayerList is empty or not
- if it's not than show it

```
import React from 'react';
import Player from './Player';



export default class PlayerList extends  React.Component{
    renderPlayers(){
        if(this.props.players.length === 0){

        }else{
            return this.props.players.map((player) => {
                return (
                  <Player key={player._id} player={player}/>
```

```
                );
              });
        }
    }
    render(){
        return (
            <div>
               {this.renderPlayers()}
            </div>
        );
    }
};

PlayerList.propTypes = {
    players : React.PropTypes.array.isRequired
}
```

- if it is empty

```
import React from 'react';
import Player from './Player';



export default class PlayerList extends  React.Component{
    renderPlayers(){
        if(this.props.players.length === 0){
            return <p>There are no players at the moment.  Please add your first player to get s
tarted!</p>
        }else{
            return this.props.players.map((player) => {
                return (
                  <Player key={player._id} player={player}/>
                );
              });
        }
    }
    render(){
```

```
        return (
            <div>
               {this.renderPlayers()}
            </div>
        );
    }
};

PlayerList.propTypes = {
    players : React.PropTypes.array.isRequired
}
```

Rendering everything with an App Component

- first we are going to get rid of the subtitle in client main.js
- and then remove isRequired from the PropTypes
- we are then going to do another conditional statement to check if there is a subtitle or not
- go to TitleBar.js

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

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

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

- now lets make the App.js components
- go to ui folder and create it
- first we want to know if this static data will appear

```
import React from 'react';

export default class App extends React.Component{
    render(){
        return(
            <div>app component</div>
        );
    }
}
```

- then head over to client main.js and import it
- and we are going to be placing our App component on the ReactDom.render() method

```
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 App from './../imports/ui/App'
import TitleBar from './../imports/ui/TitleBar';
import AddPlayer from './../imports/ui/AddPlayer';
import PlayerList from './../imports/ui/PlayerList'


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}/>
      <PlayerList players={players}/>
      <AddPlayer />
    </div>
    );
    ReactDOM.render(<App/>, document.getElementById('app'))
  });
});
```

- now what we need to do is move our components to App.js, which means the imports TitleBar, AddPlayer and PlayerList and as well as our components <TitleBar title={title}/>, <PlayerList players={players}/>, <AddPlayer />
- so now our main.js should look like this

```
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 App from './../imports/ui/App'



Meteor.startup( () => {

  Tracker.autorun(() => {
    let players = Players.find().fetch();
    let title = 'Score Keep';
    let subTitle = 'This is my SubTitle';
    ReactDOM.render(<App title={title} players={players}/>, document.getElementById('app'))
  });
});
```

- and our App.js

```
import React from 'react';
import TitleBar from './TitleBar';
import AddPlayer from './AddPlayer';
import PlayerList from './PlayerList'

export default class App extends React.Component{
    render(){
        return(
            <div>
                 <TitleBar title={this.props.title}/>
                <PlayerList players={this.props.players}/>
                <AddPlayer />
            </div>
        );
    }
}
```

- and we also going to need a type for App

```
import React from 'react';
import TitleBar from './TitleBar';
import AddPlayer from './AddPlayer';
import PlayerList from './PlayerList'

export default class App extends React.Component{
    render(){
        return(
            <div>
                 <TitleBar title={this.props.title}/>
                <PlayerList players={this.props.players}/>
                <AddPlayer />
            </div>
        );
    }
}

App.propTypes = {
```

```
    title : React.PropTypes.string.isRequired,
    players: React.PropTypes.array.isRequired
}
```

Querying and sorting Player Document

- now we are going to sort the Player list by the score, the higher the score is the higher the player will on the list
- Mongo has this sorting feature
- first lets explore this in the terminal
- if mongo is not running, go ahead and open up a new tab and type

```
meteor mongo --release 1.4.2.1
```

- and then to see how many players we have

```
db.players.find()
```

- to sort inside of the console we call a cursor method .sort() and we want to pass in an object
- and inside the object we specify the key value pairs
- first lets try decensing sort (Z to A)

```
db.players.find().sort({name: -1})
```

- now if we switch it to 1 it will sort it (A to Z)

```
db.players.find().sort({name: 1})
```

- we can also sort score. If we want the highest score at the top we want a descending sort

```
db.players.find().sort({score: -1})
```

- and with the lowest score on top and the highest on the bottom

```
db.players.find().sort({score: 1})
```

- in our app is going to be different
- head over to client main.js
- the sort object gets passed on in our options object available on find()

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


  Tracker.autorun(() => {
    let players = Players.find().fetch();
    let title = 'Score Keep';
    let subTitle = 'This is my SubTitle';
    ReactDOM.render(<App title={title} players={players}/>, document.getElementById('app'))
  });
});
```

- find() takes an optional query
- for example we can query all players with all 0s

```
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 App from './../imports/ui/App'




Meteor.startup( () => {


  Tracker.autorun(() => {
    let players = Players.find({score: 0}).fetch();
    let title = 'Score Keep';
    let subTitle = 'This is my SubTitle';
    ReactDOM.render(<App title={title} players={players}/>, document.getElementById('app'))
  });
});
```

- and that will return all players with a score of 0
- we are going to leave the first object empty, and the second argument is the options argument

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

```
  Tracker.autorun(() => {
    let players = Players.find({}, {}).fetch();
    let title = 'Score Keep';
    let subTitle = 'This is my SubTitle';
    ReactDOM.render(<App title={title} players={players}/>, document.getElementById('app'))
  });
});
```

- in that argument sort gets set to an object

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

  Tracker.autorun(() => {
    let players = Players.find({}, {
      sort: {


      }
    }).fetch();
    let title = 'Score Keep';
    let subTitle = 'This is my SubTitle';
    ReactDOM.render(<App title={title} players={players}/>, document.getElementById('app'))
  });
});
```

- and now we do exactly what we did in the terminal

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

  Tracker.autorun(() => {
    let players = Players.find({}, {
      sort: {
        score: -1
      }
    }).fetch();
    let title = 'Score Keep';
    let subTitle = 'This is my SubTitle';
    ReactDOM.render(<App title={title} players={players}/>, document.getElementById('app'))
```

```
  });
});
```

-