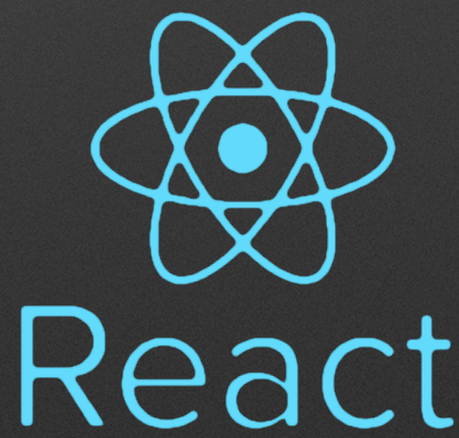
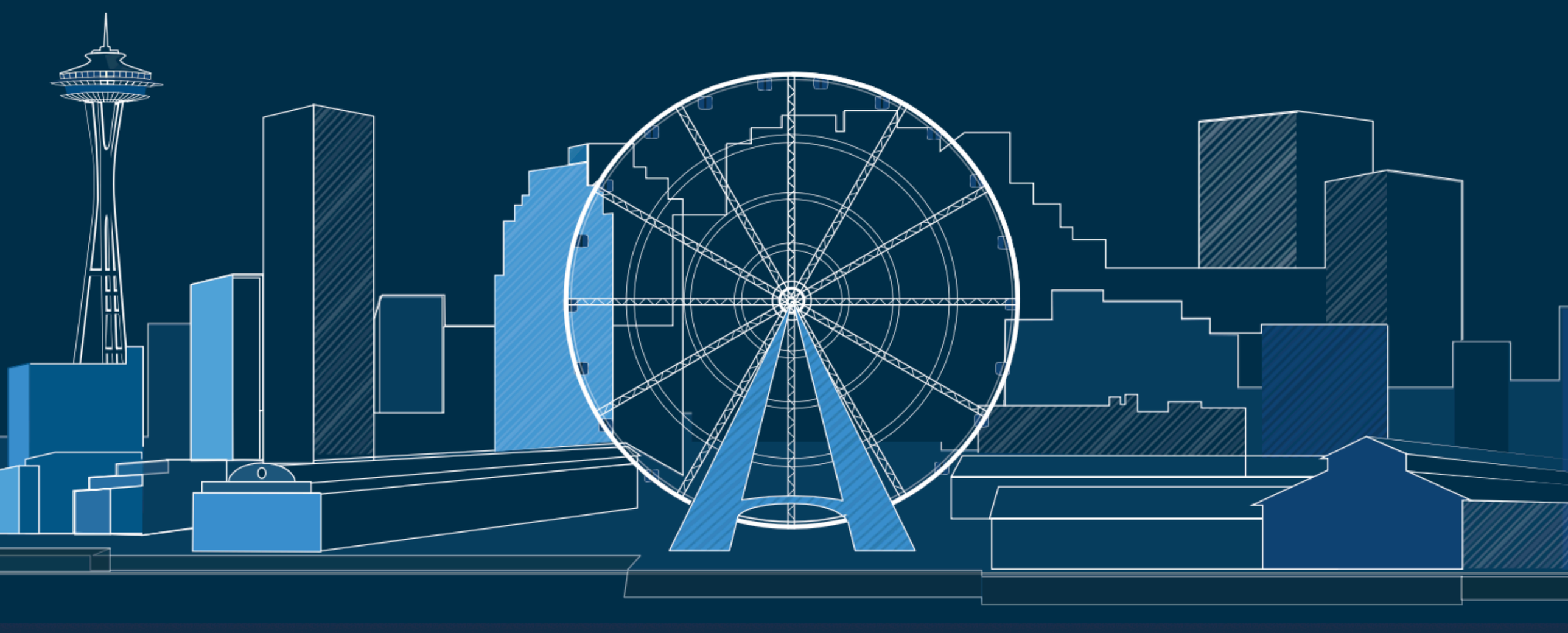
# Real World Apps with React and TypeScript

Kurt Wiersma
Tech Lead - OneOme
@kwiersma

# Agenda

- Brief intro to TypeScript
- How to setup a React app with TypeScript
- Building React components with TypeScript
- Building services to fetching data with TypeScript
- Managing state
- Testing React apps with TypeScript and React Testing Library



# TypeScript

## JavaScript that scales.

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.

Any browser. Any host. Any OS. Open source.

TypeScript | Select... | Share | Run | JavaScript

```
1 function greeter(person) {
2     return "Hello, " + person;
3 }
4
5 var user = "Jane User";
6
7 document.body.innerHTML = greeter(user);
8
9
```

```
1 function greeter(person) {
2     return "Hello, " + person;
3 }
4
5 var user = "Jane User";
6
7 document.body.innerHTML = greeter(user);
8
```

# JavaScript is valid TypeScript

Syn

```tsx
type MyProps = {
  // using `interface` instead 'type' is also ok
  message: string;
};

type MyState = {
  count: number;
  label: string;
};

class Counter extends React.Component<MyProps, MyState> {
  state: MyState = {
    // optional second annotation for better type inference
    count: 0,
    label: "",
  };

  constructor(props: MyProps) {
    super(props);
    this.state.label = props.message + ": ";
  }

  increment = (amt: number): void => {
    this.setState((state) => ({
      count: state.count + amt,
    }));
  };

  render() {
    return (
      <p>
        {this.state.label} {this.state.count}
      </p>
    );
  }
```

# Why would you want types?

- Structure for large code bases and/or teams
- Catch errors early
- Provide a better structured, documented API
- Tooling can provide better code completion & refactoring

Ghetto League | Players | Teams

**Round 1**

4. ChumpsnSalsa (Steph)
5. Taco Charlton (Pingle)
6. Suburbia (Marc)
7. Woodstock Nation (Russ)
8. Daddy Shark (Todd)
9. The Bear Den (Will)
10. Funicular Goats (Kurt)

**Round 2**

11. Funicular Goats (Kurt)
12. The Bear Den (Will)
13. Daddy Shark (Todd)
14. Woodstock Nation (Russ)

Taco Charlton (Pingle)

| Player | Pos | Bye |
|---|---|---|

**Last name:** ( ) Any ( ) Available ( ) Drafted **Position:** Any

| Player | Rank | Pts | Bye | ADP | R, P | Fantasy Team |
|---|---|---|---|---|---|---|
| Barkley, Saquon<br>NYG - RB | 1 | 256 | 11 | 1.3 | R 1 P 1 | The Blue Sky Bullets (Swig) |
| Kamara, Alvin<br>NO - RB | 2 | 237 | 9 | 2.6 | R 1 P 3 | Techicolor Unicorns (Rebecca) |
| McCaffrey, Christian<br>Car - RB | 3 | 237 | 7 | 3.5 | R 1 P 2 | Don Sowo (Kristina) |
| Chubb, Nick<br>Cle - RB | 4 | 161 | 7 | 10.8 | R P | |
| Elliott, Ezekiel<br>Dal - RB | 5 | 220 | 8 | 4.2 | R P | |
| Johnson, David<br>Ari - RB | 6 | 168 | 12 | 6.5 | R P | |
| Hopkins, DeAndre<br>Hou - WR | 7 | 160 | 10 | 7 | R P | |
| Adams, Davante<br>GB - WR | 8 | 164 | 11 | 9.5 | R P | |
| Conner, James<br>Pit - RB | 9 | 197 | 7 | 11 | R P | |
| Jones, Julio<br>Atl - WR | 10 | 150 | 9 | 12.1 | R P | |

10

1 2 3 4 5 > >>

On the clock: (Steph) (R 1 P 4)
On deck: (Pingle)
Last Pick: Kamara, Alvin by Techicolor Unicorns (Rebecca)
Before Last Pick: McCaffrey, Christian by Don Sowo (Kristina)
10028:46
On the clock !

# Real world app demo

# Project setup

```
$ npx create-react-app my-app --template typescript
$ cd my-app
$ npm start
```

```
interface TeamsState {
    teams: FantasyTeam[];
}

export class Teams extends Component<{}, TeamsState> {

   constructor(props: {}) {
        super(props);
        this.state = {
            teams: [],
        };
    }

    componentDidMount() {
        draftAPI.getFantasyTeams().then((teams: FantasyTeam[]) => {
            this.setState({ teams });
        });
    }

    render() {
        let {teams} = this.state;

        const teamRows = teams.map((team, idx) => {
            return (
                <tr key={team.id}>
                    <td>{team.draftorder}</td>
                    <td>{team.name}</td>
                    <td>{team.owner}</td>
                </tr>
            );
        });

        return (
```

```tsx
interface IState {
    playerFilter: PlayersFilter;
}

interface IPlayerFilterProps {
    onChange: (playerFilter: PlayersFilter) => void;
}

export class PlayersFilter {
    public lastname: string = "";
}

export class PlayerFilter extends Component<IPlayerFilterProps, IState> {

  handleLastNameChange = (e: FormEvent<HTMLInputElement>) => {
    const newPlayerFilter = this.state.playerFilter;
    newPlayerFilter.lastname = e.currentTarget.value;
    this.setState({
        playerFilter: newPlayerFilter
    });
    this.props.onChange(newPlayerFilter);
  };

  render() {
    return (
        <label className="control-label" htmlFor="lastname">Last name: </label>
        <input type="text" id="lastname"
               className="form-control"
               value={this.state.playerFilter.lastname}
               onChange={this.handleLastNameChange}/>
    )}
}
```

```
1  import {FantasyTeam} from "../model";
2
3  const baseUrl = "";
4  let teams: FantasyTeam[];
5  let players: Player[];
6
7  const getFantasyTeams = (): Promise<FantasyTeam[]> => {
8      const teamsUrl = baseUrl + "/api2/teams";
9      if (teams) {
10         return new Promise<FantasyTeam[]>((resolve) => {
11             return resolve(teams)
12         });
13     } else {
14         return fetch(teamsUrl, {credentials: "include"})
15             .then((response) => {
16                 if (response.status === 404) {
17                     return null;
18                 }
19                 return response.json();
20             })
21             .then((response: any): FantasyTeam[] => {
22                 teams = response as FantasyTeam[];
23                 return teams;
24             });
25     }
26 };
27
28 export const draftAPI = {
29     getFantasyTeams,
```

# Fetching Remote Data

- Add proxy config to package.json
- Run the dev server for the backend in a separate process
- In production run both the backend and front end on the same domain
- If you fetch() a polyfill is need for IE 11

```
1 {
2   "name": "my-app",
3   "version": "1.0.0",
4   "proxy": "http://localhost:8000",
5 }
```

# Managing State

- Keep as much state local to each component as you can
- Shared state options:
  - React Hooks with Context
  - Mobx (Observables)
  - Redux

# Redux Toolkit - Before

```
1   import { Item } from "../models/item";
2   import { createStore } from "redux";
3   import { initialItemState } from "./initialData";
4   import { ItemState, UpdateOwnerAction } from "./interfaces";
5
6   const UPDATE_OWNER = "UPDATE_OWNER";
7   export const updateOwner = (item: Item, user: string): UpdateOwnerAction => ({
8     type: UPDATE_OWNER,
9     payload: { item, user }
10  });
11
12  const itemReducer = (
13    state: ItemState = initialItemState,
14    { type, payload }: UpdateOwnerAction
15  ) => {
16    switch (type) {
17      case UPDATE_OWNER:
18        return {
19          ...state,
20          [payload.item.id]: { ...state[payload.item.id], owner: payload.user }
21        };
22      default:
23        return state;
24    }
25  };
26
27  export const store = createStore(itemReducer);
28
```

# Redux Toolkit - After

```
1   import { createStore } from "redux";
2   import { createSlice, createEntityAdapter } from "@reduxjs/toolkit";
3   import { initialItemState } from "./initialData";
4
5   const itemsAdapter = createEntityAdapter();
6
7   const { actions, reducer } = createSlice({
8     name: "items",
9     initialState: initialItemState,
10    reducers: {
11      updateOwner: itemsAdapter.updateOne
12    }
13  });
14
15  export const updateOwner = actions.updateOwner;
16
17  export const store = createStore(reducer);
18
```

# Testing

- React-Testing-Library promotes testing React components like a user would use them
- Runs on top of Jest which runs tests in node and not in a browser instance so it only has a “mini-dom”
- You should avoid testing internal component state

```
$ npm install @types/jest @testing-library/react \
      @testing-library/jest-dom ts-jest
```

# Example Test

```tsx
import * as React from "react";
import App from "./App";
import { render } from "@testing-library/react";

test("should render app", async () => {
  const { getByText } = render(<App />);

  const navbar = await getByText("Ghetto League");
  expect(navbar !== undefined);
});

```

App.test.tsx

```tsx
import * as React from "react";
import { PlayerFilter, PlayersFilter } from "./PlayerFilter";
import { render, fireEvent } from "@testing-library/react";

test("should render last name field", async () => {
    const changeHandler = jest.fn();
    const { getByLabelText } = render(<PlayerFilter onChange={changeHandler} />);

    const input = getByLabelText("Last name:") as HTMLInputElement;
    expect(input !== undefined);

    fireEvent.change(input, { target: { value: 'a' } });

    expect(input.value).toBe('a');
    expect(changeHandler).toHaveBeenCalledTimes(1);
    let playerFilter = new PlayersFilter();
    playerFilter.lastname = "a";
    expect(changeHandler).toHaveBeenCalledWith(playerFilter);
});

```

# Resources

- Sample app
  - https://github.com/kwiersma/react-league
- React + TypeScript Cheatsheets
  - https://github.com/typescript-cheatsheets/react-typescript-cheatsheet
- Create React App Redux TypeScript:
  - https://github.com/reduxjs/cra-template-redux-typescript
  - https://redux-starter-kit.js.org/usage/usage-with-typescript
- Testing with React + TypeScript + React Testing Lib (Kamran Ayub)
  - https://www.pluralsight.com/guides/how-to-test-react-components-in-typescript

# Questions & Answers (hopefully)

Fire away!