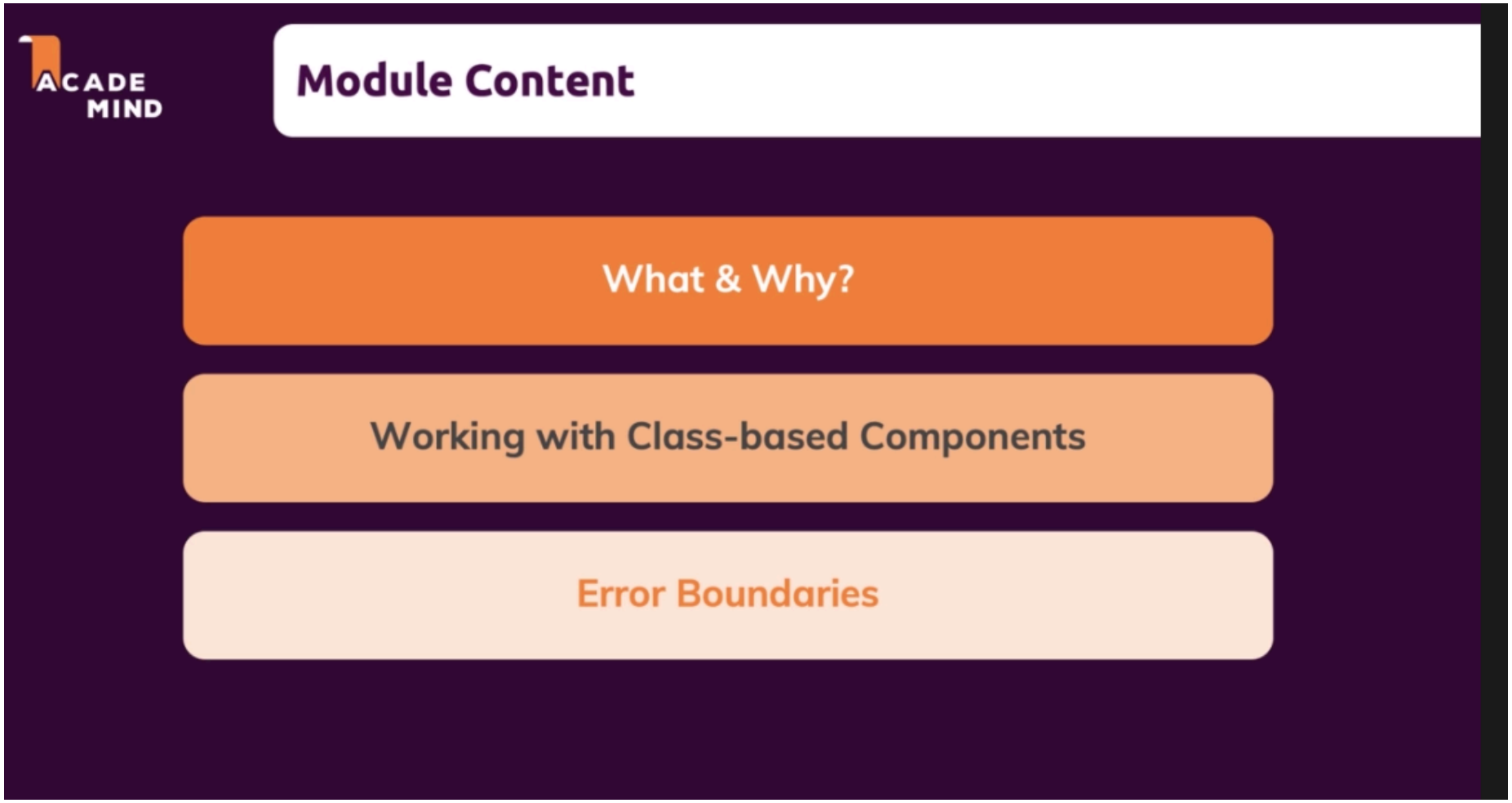
Front End

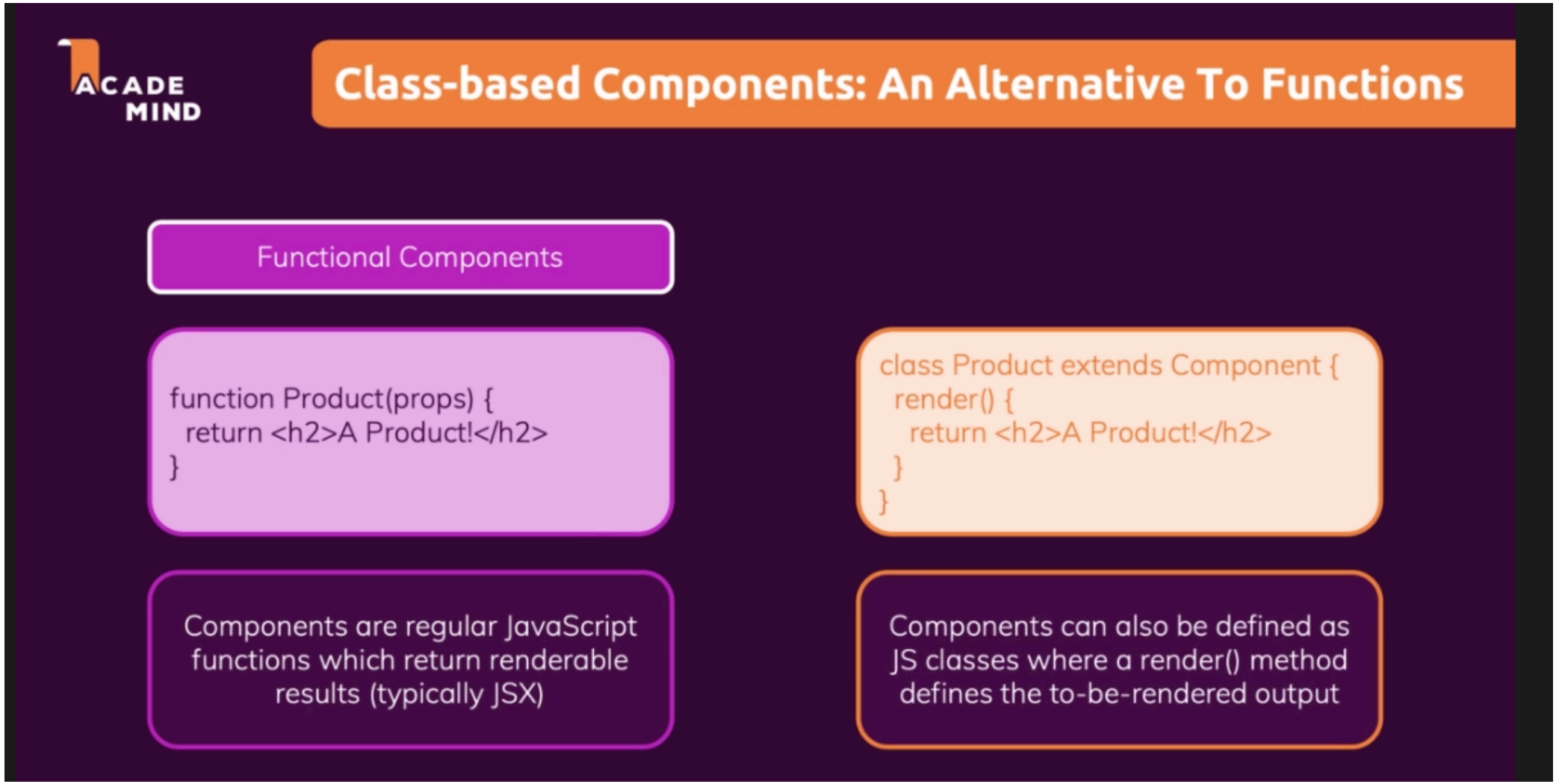
Class 12

September 22, 2021

Agenda

- Kahoot
- Review – Class-based components and TS
- Studio Review plus studio


ACADEMIND
Module Content
What & Why?
Working with Class-based Components
Error Boundaries



React < 16.8
16.8 introduced hooks
Class based can’t use react hooks

Class based Components

class keyword
constructor() for setup
render method. React calls the render method.
Equivalent to return in functional components.
Functional components get props
class based components – render method doesn’t receive props.
Need to import Component and extend it.
Component class adds functionality and properties to your component.
props property gets added.
Class based and functional components can work together. Mix-match.
However, consistency matters for entire app.

Class based Components

Needed class-based components prior to 16.8 for managing state.

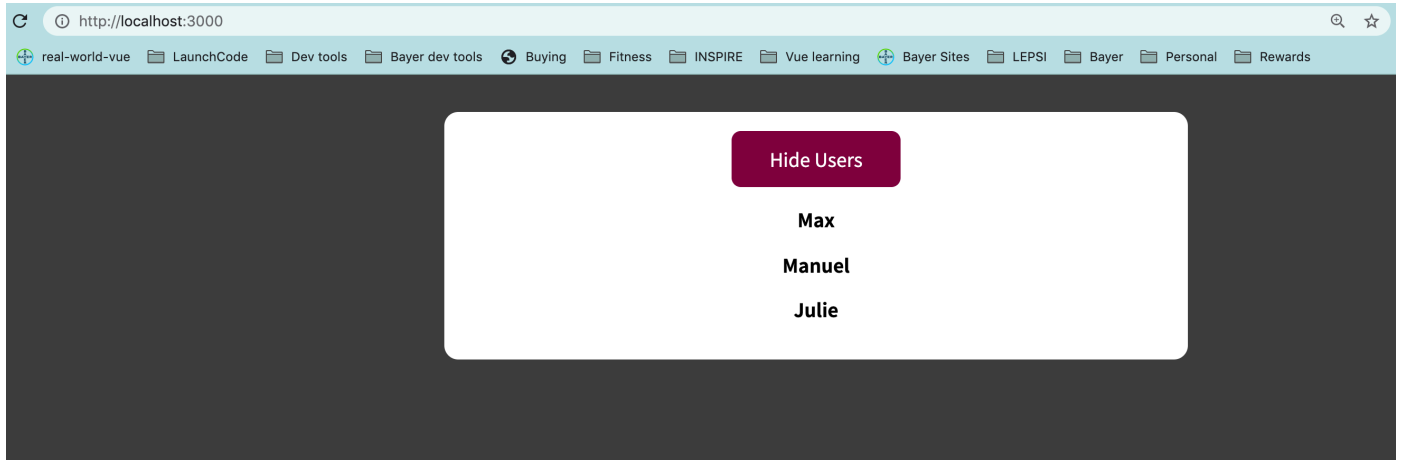
Constructor is called automatically for each component. Used to initialize state. State must be an object. The variable also needs to be named state. All state must be grouped together in the state object.

this.setState does not overwrite the state but merge the set values of properties within the state

In functional components it can be object or other data types.

Class based Components

```
import { Component } from 'react';
import classes from './User.module.css';

class User extends Component {
  render() {
    return <li className={classes.user}>{this.props.name}</li>;
  }
}

// const User = (props) => {
//   return <li className={classes.user}>{props.name}</li>;
// };

export default User;
```



```jsx
import { Component } from 'react';
import User from './User';

import classes from './Users.module.css';

const DUMMY_USERS = [
  { id: 'u1', name: 'Max' },
  { id: 'u2', name: 'Manuel' },
  { id: 'u3', name: 'Julie' },
];

class Users extends Component {
  constructor() {
    super();
    this.state = {
      showUsers: true,
    };
  }
  toggleUsersHandler() {
    this.setState((curState) => {
      return { showUsers: !curState.showUsers };
    });
  }

  render() {
    const usersList = (
      <ul>
        {DUMMY_USERS.map((user) => (
          <User key={user.id} name={user.name} />
        ))}
      </ul>
    );
    return (
      <div className={classes.users}>
        <button onClick={this.toggleUsersHandler.bind(this)}>
          {this.state.showUsers ? 'Hide' : 'Show'} Users
        </button>
        {this.state.showUsers && usersList}
      </div>
    );
  }
}

export default Users;
```

this

https://academind.com/tutorials/this-keyword-function-references

This refers to the object that the property or function belongs to.

ACADEMIND

Class-based Component Lifecycle

Side-effects in Functional Components: **useEffect()**

Class-based Components can't use React Hooks!

componentDidMount()	Called once component mounted (was evaluated & rendered)	useEffect(..., [])
componentDidUpdate()	Called once component updated (was evaluated & rendered)	useEffect(..., [someValue])
componentWillUnmount()	Called right before component is unmounted (removed from DOM)	useEffect(() => { return () => {...}}, [])

57 people have written a note here

UserFinder Component – change to class-based Component.

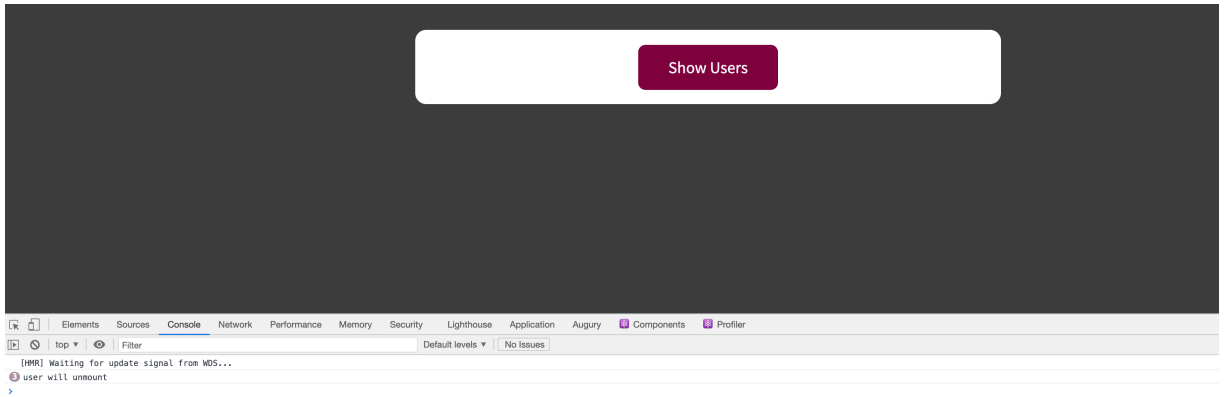
```jsx
import { Fragment, useState, useEffect } from 'react';

import Users from './Users';

const DUMMY_USERS = [
  { id: 'u1', name: 'Max' },
  { id: 'u2', name: 'Manuel' },
  { id: 'u3', name: 'Julie' },
];

const UserFinder = () => {
  const [filteredUsers, setFilteredUsers] = useState(DUMMY_USERS);
  const [searchTerm, setSearchTerm] = useState('');

  useEffect(() => {
    setFilteredUsers(
      DUMMY_USERS.filter((user) => user.name.includes(searchTerm))
    );
  }, [searchTerm]);

  const searchChangeHandler = (event) => {
    setSearchTerm(event.target.value);
  };

  return (
    <Fragment>
      <input type="search" onChange={searchChangeHandler} />
      <Users users={filteredUsers} />
    </Fragment>
  );
};

export default UserFinder;
```

```jsx
import { Fragment, Component } from 'react';

import Users from './Users';

const DUMMY_USERS = [
  { id: 'u1', name: 'Max' },
  { id: 'u2', name: 'Manuel' },
  { id: 'u3', name: 'Julie' },
];

class UserFinder extends Component {
  constructor() {
    super();
    this.state = {
      filteredUsers: DUMMY_USERS,
      searchTerm: '',
    };
  }
  componentDidUpdate(prevProps, prevState) {
    if (prevState.searchTerm !== this.state.searchTerm) {
      this.setState({
        filteredUsers: DUMMY_USERS.filter((user) =>
          user.name.includes(this.state.searchTerm)
        ),
      });
    }
  }
  searchChangeHandler(event) {
    this.setState({ searchTerm: event.target.value });
  }

  render() {
    return (
      <Fragment>
        <input type="search" onChange={this.searchChangeHandler.bind(this)} />
        <Users users={this.state.filteredUsers} />
      </Fragment>
    );
  }
}
```

Other lifecycle methods.

```
componentDidMount() {
  //Will only run once, when the component is rendered the first time.
  //Send http request
  ///equivalent to using useEffect with no dependencies or with dependencies.
  this.setState({ filteredUsers: DUMMY_USERS });
}
```

```
class User extends Component {
  componentWillUnmount() {
    console.log('user will unmount');
  }
  render() {
    return <li className={classes.user}>{this.props.name}</li>;
  }
}
```



Class based Components and Context.

```jsx
import { Fragment, Component } from 'react';
import UsersContext from '../store/users-context';

import Users from './Users';

// const DUMMY_USERS = [
//   { id: 'u1', name: 'Max' },
//   { id: 'u2', name: 'Manuel' },
//   { id: 'u3', name: 'Julie' },
// ];

class UserFinder extends Component {
  static contextType = UsersContext;
  constructor() {
    super();
    this.state = {
      filteredUsers: [],
      searchTerm: '',
    };
  }
  componentDidMount() {
    //Will only run once, when the component is rendered the first time.
    //Send http request
    ///equivalent to using useEffect with no dependencies or with dependencies.
    this.setState({ filteredUsers: this.context.users });
  }
```

ACADEMIND

Class-based vs. Functional Components

Prefer functional components

Use class-based if...

- ...you prefer them
- ...you're working on an existing project or in a team where they're getting used
- ...you build an "Error Boundary"

Error Boundaries.

Error is a way to communicating via throwing an and embracing errors

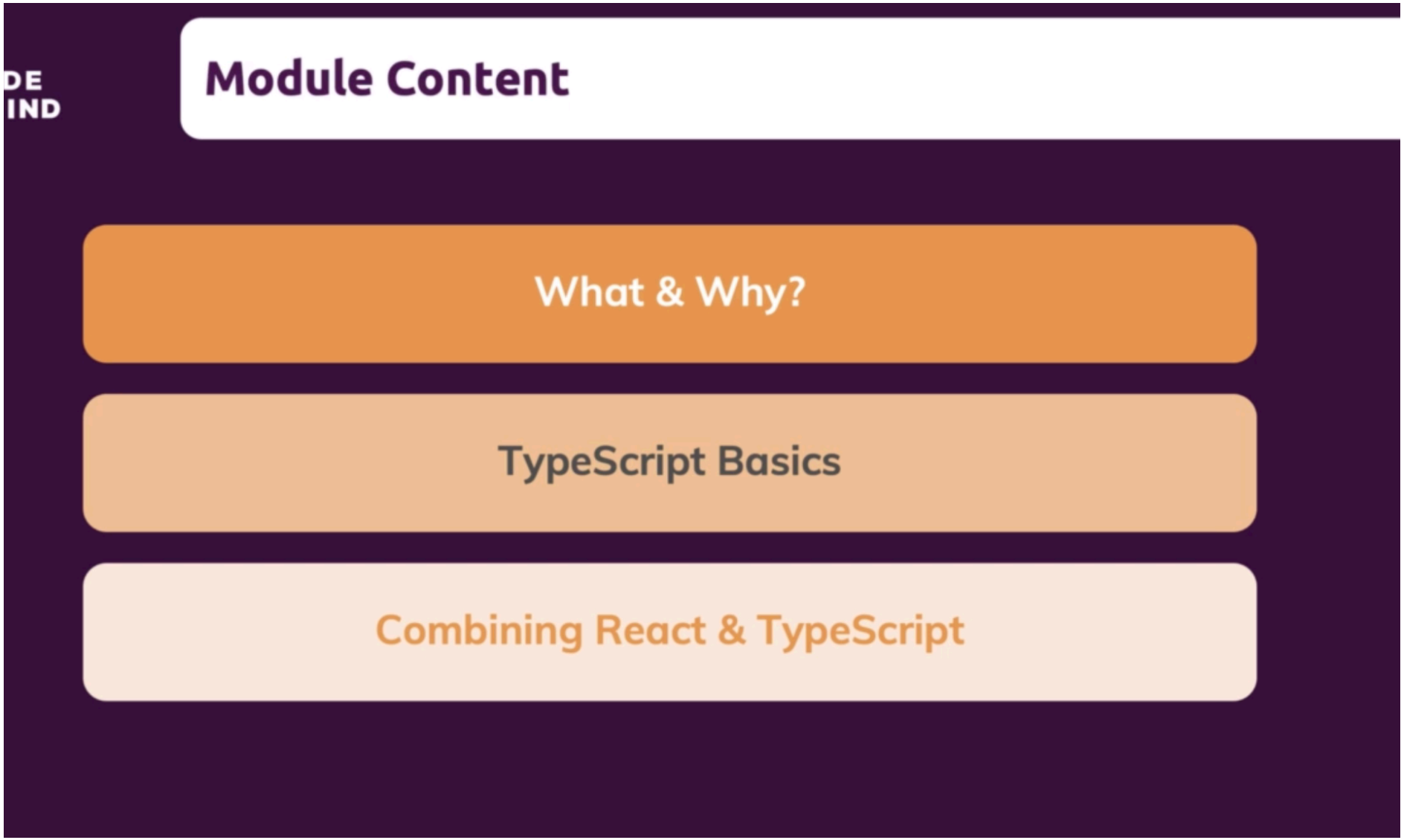
Try catch block is used for Error Handling in JS

Can build and utilize an Error Boundary for JSX code.

```jsx
import { Component } from 'react';

class ErrorBoundary extends Component {
  constructor() {
    super();
    this.state = { hasError: false };
  }

  //lifecycle method is trigerred when a child component throws an error and that handles errors
  componentDidCatch(error) {
    console.log(error);
    this.setState = { hasError: false };
  }

  render() {
    if (this.state.hasError) {
      return <p>Something went wrong!</p>;
    }
    return this.props.children;
  }
}

export default ErrorBoundary;
```

```jsx
render() {
  return (
    <Fragment>
      <input type="search" onChange={this.searchChangeHandler.bind(this)} />
      <ErrorBoundary>
        <Users users={this.state.filteredUsers} />
      </ErrorBoundary>
    </Fragment>
  );
}
```

React and TS


DE
IND
Module Content
What & Why?
TypeScript Basics
Combining React & TypeScript

React and TS

TS is static typed

Advantages – avoid passing wrong types in code

Download and use typescript

Compiled into JS to run in browser

Types – number, string, boolean, array, object, null, undefined

Type inference

Union string | number

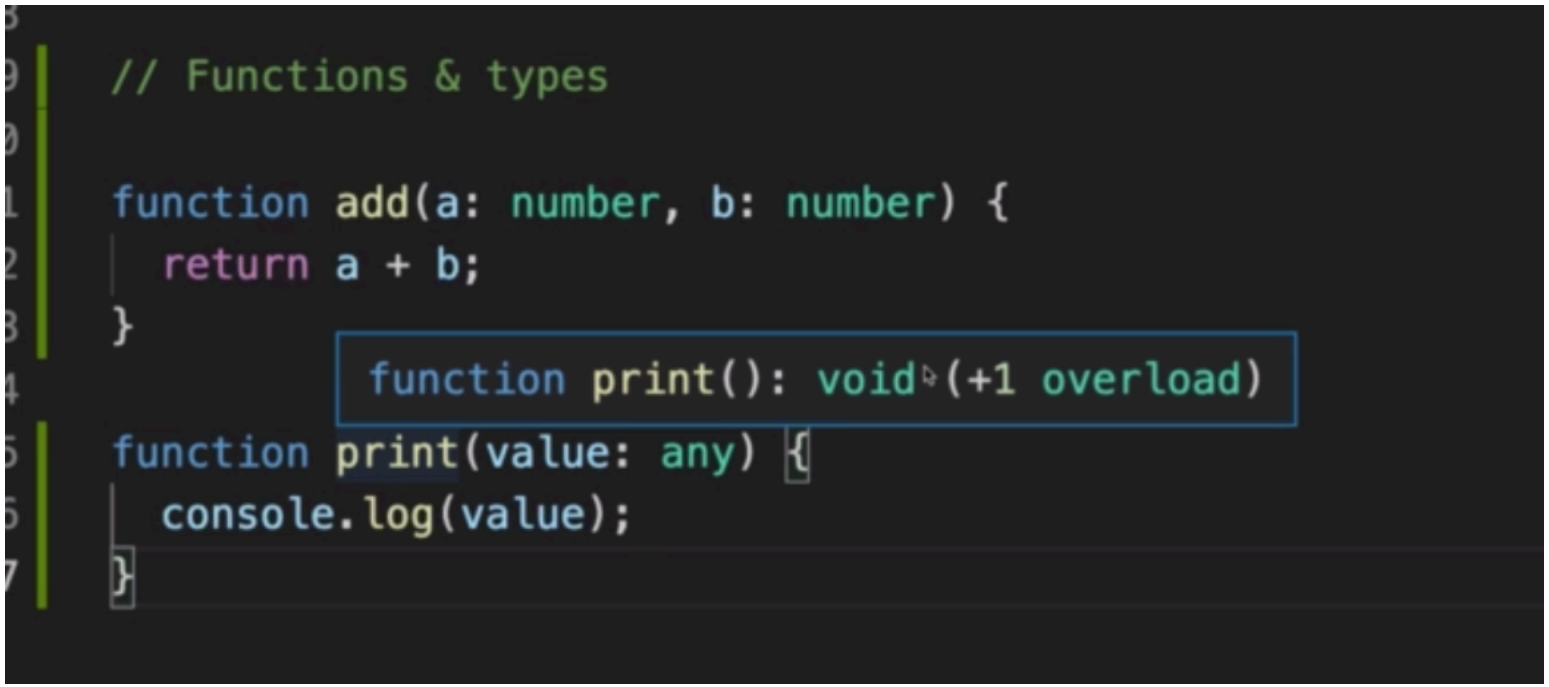
React and TS

type alias

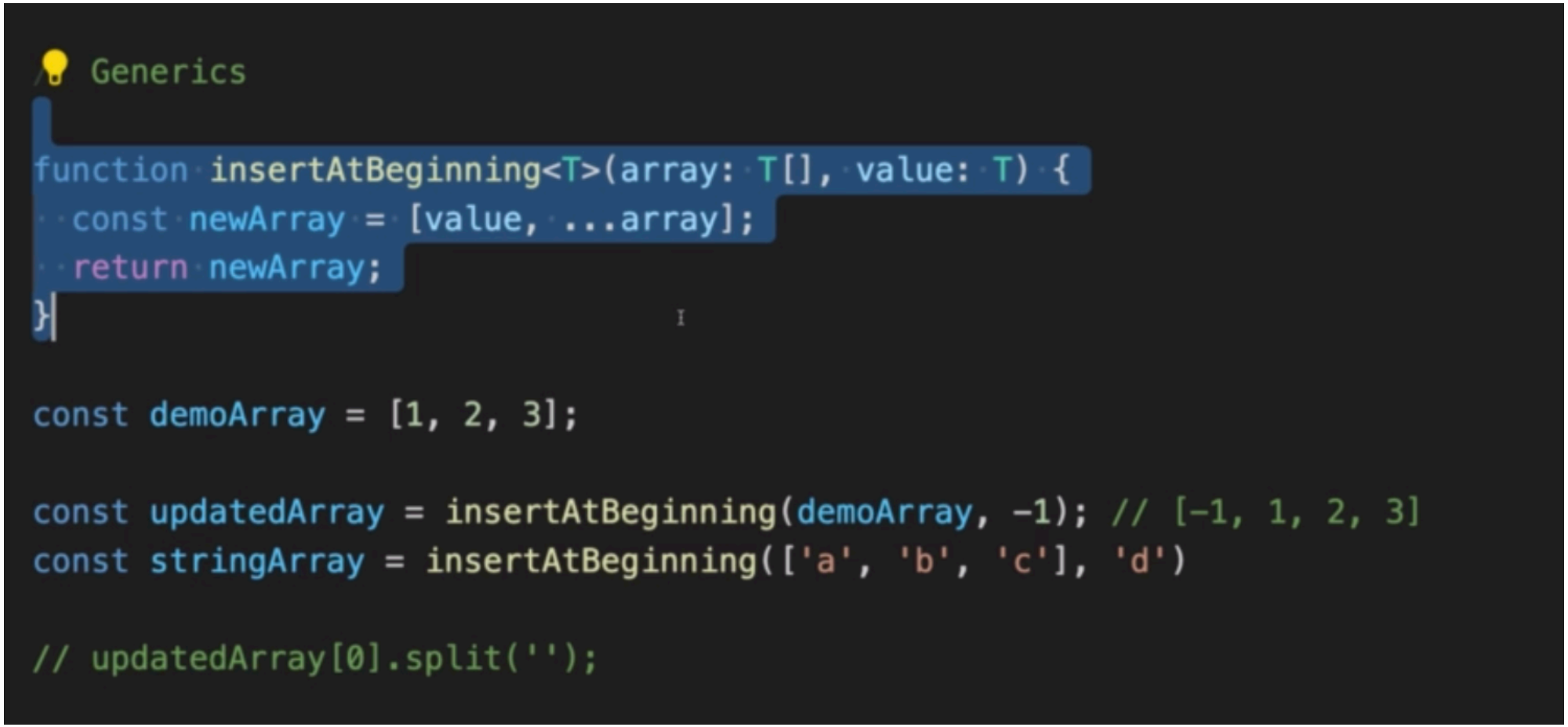
Example
type Person = {
name: string,
age: number
}

person: person = {
name: “Arianna”
age: 25
}

Functions and Types



Diving into Generics



Creating a React and TypeScript Project

https://create-react-app.dev/docs/adding-typescript/

Can create a new project using TS or add to an existing project

npx create-react-app my-app --template typescript

Extension is tsx

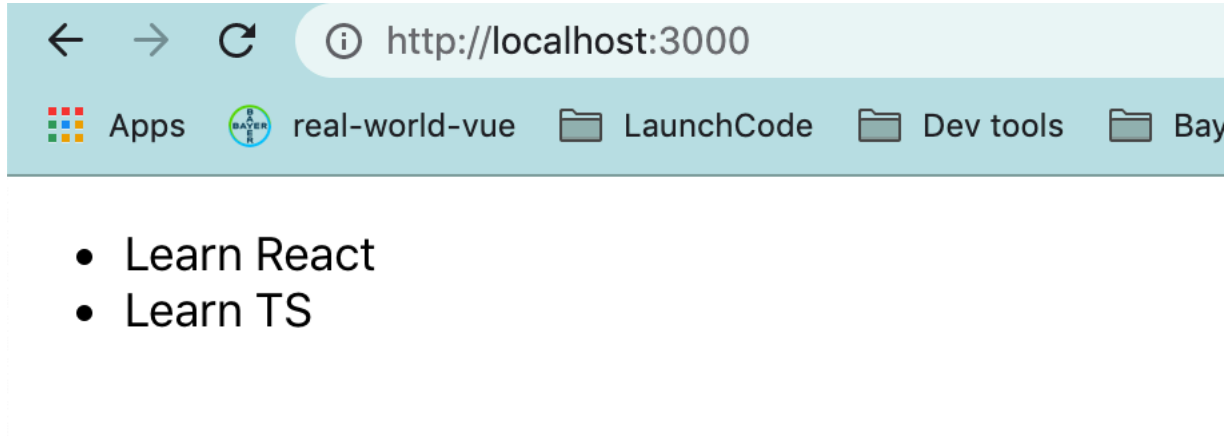
Compiled into JS code.

Types libraries – translations between TS and JS (annotations)

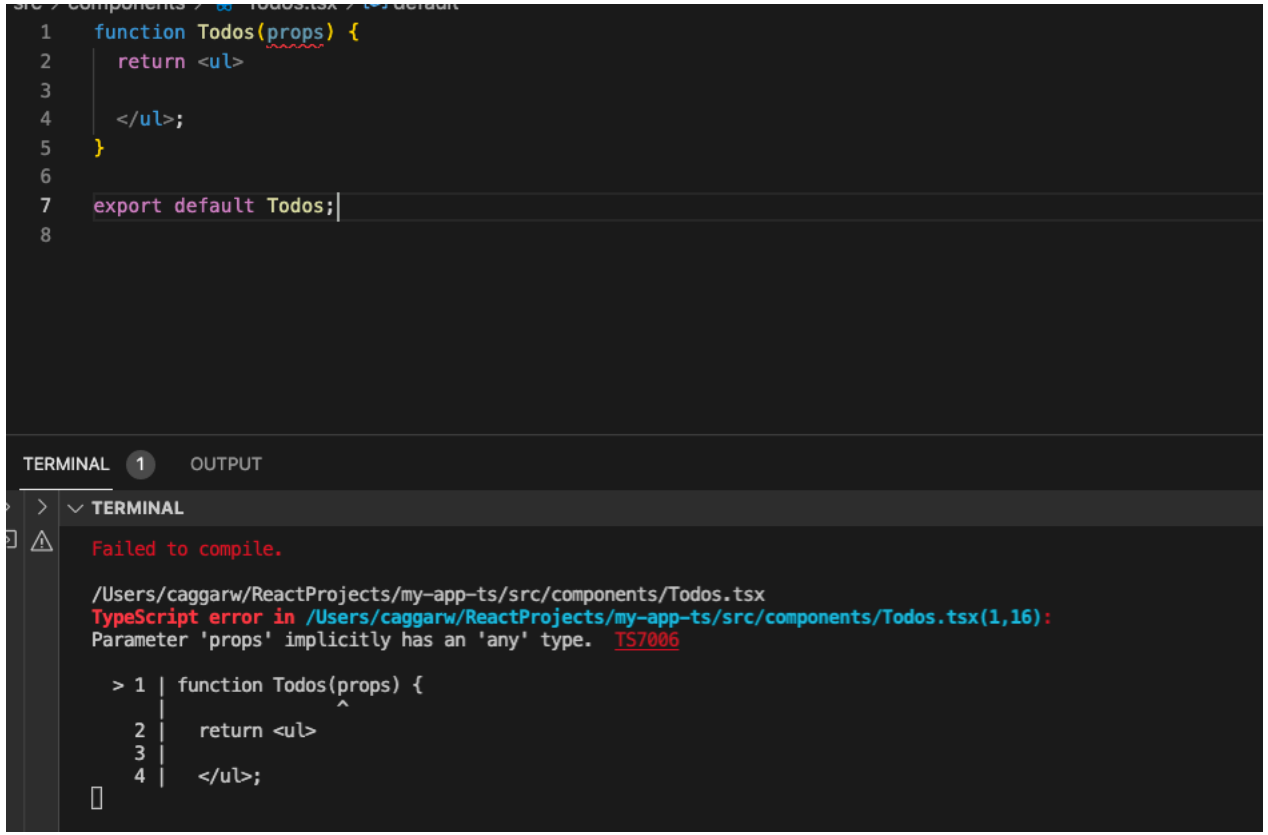
Working with Components and TS

TODO app to add and remove TODO’s

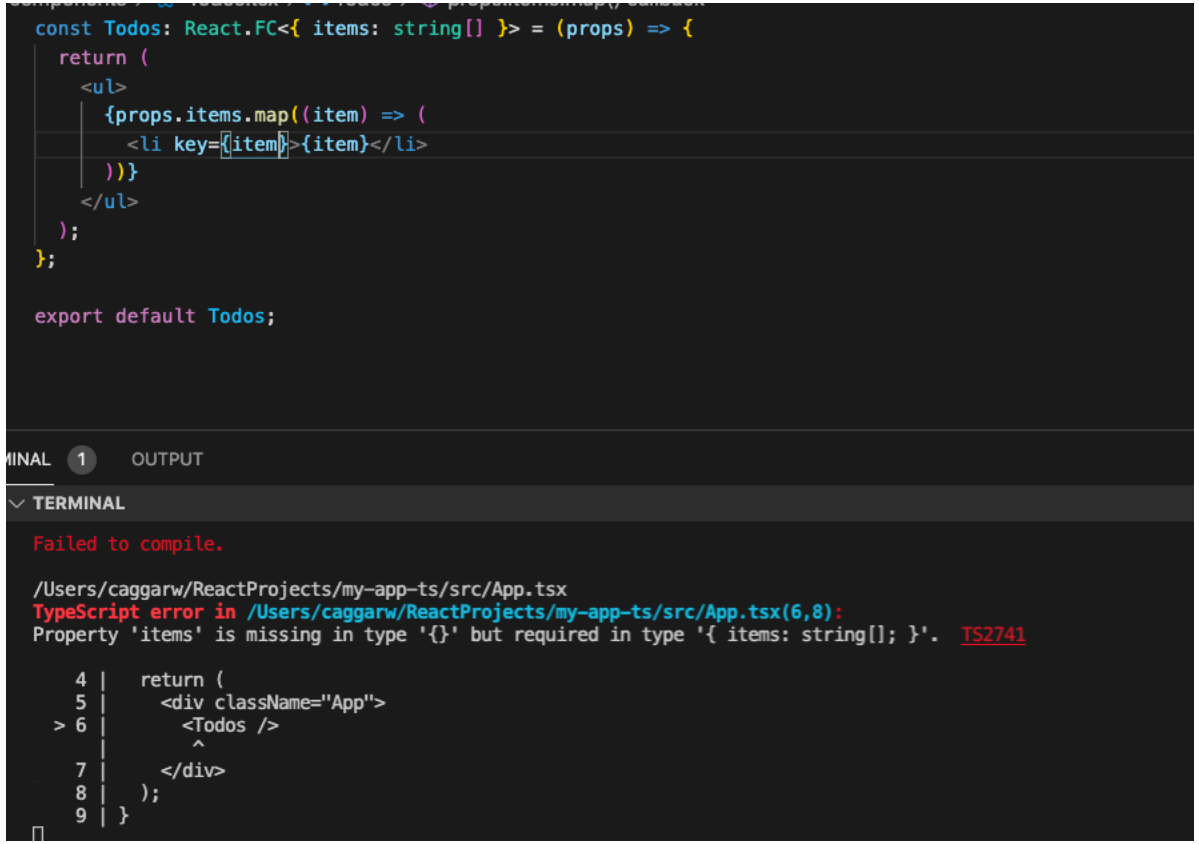
```tsx
function Todos() {
  return (
    <ul>
      <li>Learn React</li>
      <li>Learn TS</li>
    </ul>
  );
}

export default Todos;
```



Working with Components and TS



Working with Components and TS

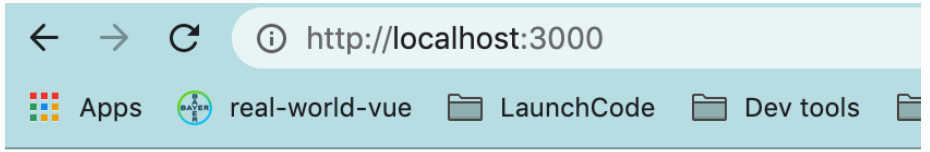


```tsx
const Todos: React.FC<{ items: string[] }> = (props) => {
  return (
    <ul>
      {props.items.map((item) => (
        <li key={item}>{item}</li>
      ))}
    </ul>
  );
};

export default Todos;
```

```
INAL 1  OUTPUT

TERMINAL

Failed to compile.

/Users/caggarw/ReactProjects/my-app-ts/src/App.tsx
TypeScript error in /Users/caggarw/ReactProjects/my-app-ts/src/App.tsx(6,8):
Property 'items' is missing in type '{}' but required in type '{ items: string[]; }'.  TS2741

    4 |   return (
    5 |     <div className="App">
  > 6 |       <Todos />
      |        ^
    7 |     </div>
    8 |   );
    9 | }
```

Working with Components and TS

```tsx
import Todos from './components/Todos';

function App() {
  return (
    <div className="App">
      <Todos items={['learn react', ' learn TS']} />
    </div>
  );
}

export default App;
```



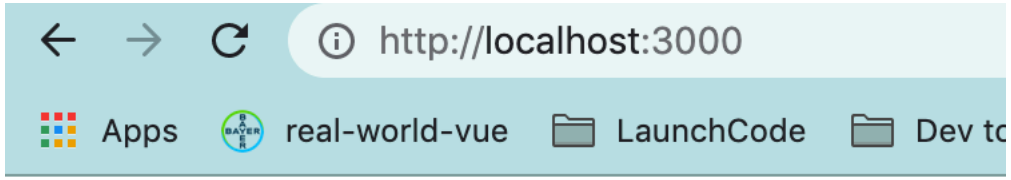
- learn react
- learn TS

Adding a Data Model

```ts
class Todo {
  id: string;
  text: string;

  constructor(todoText: string) {
    this.text = todoText;
    this.id = new Date().toISOString();
  }
}

export default Todo;
```

```ts
import Todo from '../models/todo';
const Todos: React.FC<{ items: Todo[] }> = (props) => {
  return (
    <ul>
      {props.items.map((item) => (
        <li key={item.id}>{item.text}</li>
      ))}
    </ul>
  );
};

export default Todos;
```

```ts
You, seconds ago | 1 author (You)
import Todos from './components/Todos';
import Todo from './models/todo';

function App() {
  const todos = [new Todo('learn react'), new Todo('learn TS')];
  return (
    <div className="App">
      <Todos items={todos} />
    </div>
  );
}

export default App;
```



- learn react
- learn TS

Exploring tsconfig.json

Studio Review + Studio