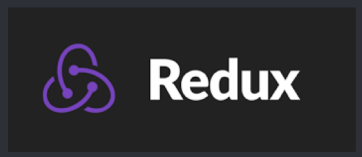
Exploring the CGSWeb Front-end Stack

Development Tools



Build | Develop | Test | Runtime

Build

- NodeJS
  - JavaScript runtime build on Chrome's V8 engine
  - Server-side JavaScript
- NPM – Node Package Manager
- Gulp
  - Task runner
- Webpack
  - Bundles all assets together (TS/JS, CSS, etc.)
  - Uses "loaders" to handle different file types

Develop

- Visual Studio Code
- Sass –Syntactically Awesome Style Sheets
  - CSS extension
  - Compiles down to CSS
- TypeScript
  - Superset of JavaScript
  - Static type checking
  - Superior refactoring and autocomplete

TypeScript - Type Inference

```ts
interface Person {
    firstName: string;
    middleInitial: string;
    lastName: string;
}

const printName = (person: Person) =>
    console.log(`${person.lastName}, ${person.firstName} ${person.middleInitial}.`);

const me = {
    firstName: 'Jeremy',
    lastName: 'Berglund'
};

printName(me);
```

[ts] Argument of type '{ firstName: string; lastName: string; }' is not assignable to parameter of type 'Person'. Property 'middleInitial' is missing

Testing

- Jest
  - Test runner and assertion library
  - Super easy to set up
- Selenium
  - Automated in-browser testing
  - Regression and integration testing

React

- What is React?
  - Declarative library for making UIs
  - Described as the “V” in “MVC”
  - Component architecture
    - Components take in props
    - Returns some elements to render on the screen
    - Components can manage their own state

Basic Component

```
interface Props {
    text: string;
    onClick: (e: React.MouseEvent<HTMLButtonElement>) => void;
}

function Button(props: Props) {
    return React.createElement('button', {
        text: props.text,
        onClick: props.onClick
    });
}
```

Component Using JSX Syntax

```tsx
interface Props {
    text: string;
    onClick: (e: React.MouseEvent<HTMLButtonElement>) => void;
}

function Button(props: Props) {
    return <button onClick={props.onClick}>{props.text}</button>;
}
```

```tsx
interface Props {

}
interface State {
    count: number;
}
class Counter extends React.Component<Props, State> {
    constructor(props: Props) {
        super(props);
        this.state = {
            count: 0
        }

        this.handleClick = this.handleClick.bind(this);
    }

    public render() {
        const { count } = this.state;
        return (
            <div className="counter">
                <Button text="Click me!" onClick={this.handleClick} />
                {`I've been clicked ${count} times`}
            </div>
        )
    }

    private handleClick(e: React.MouseEvent<HTMLButtonElement>) {
        this.setState({ count: this.state.count + 1 });
    }
}
```

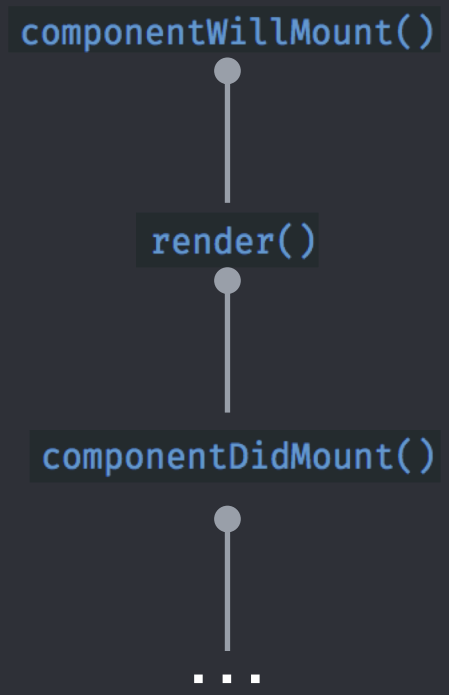
```
public render() {
    const { count } = this.state;
    return (
        <div className="counter">
            <Button text="Click me!" onClick={this.handleClick} />
            {`I've been clicked ${count} times`}
        </div>
    )
}
```

Handling State

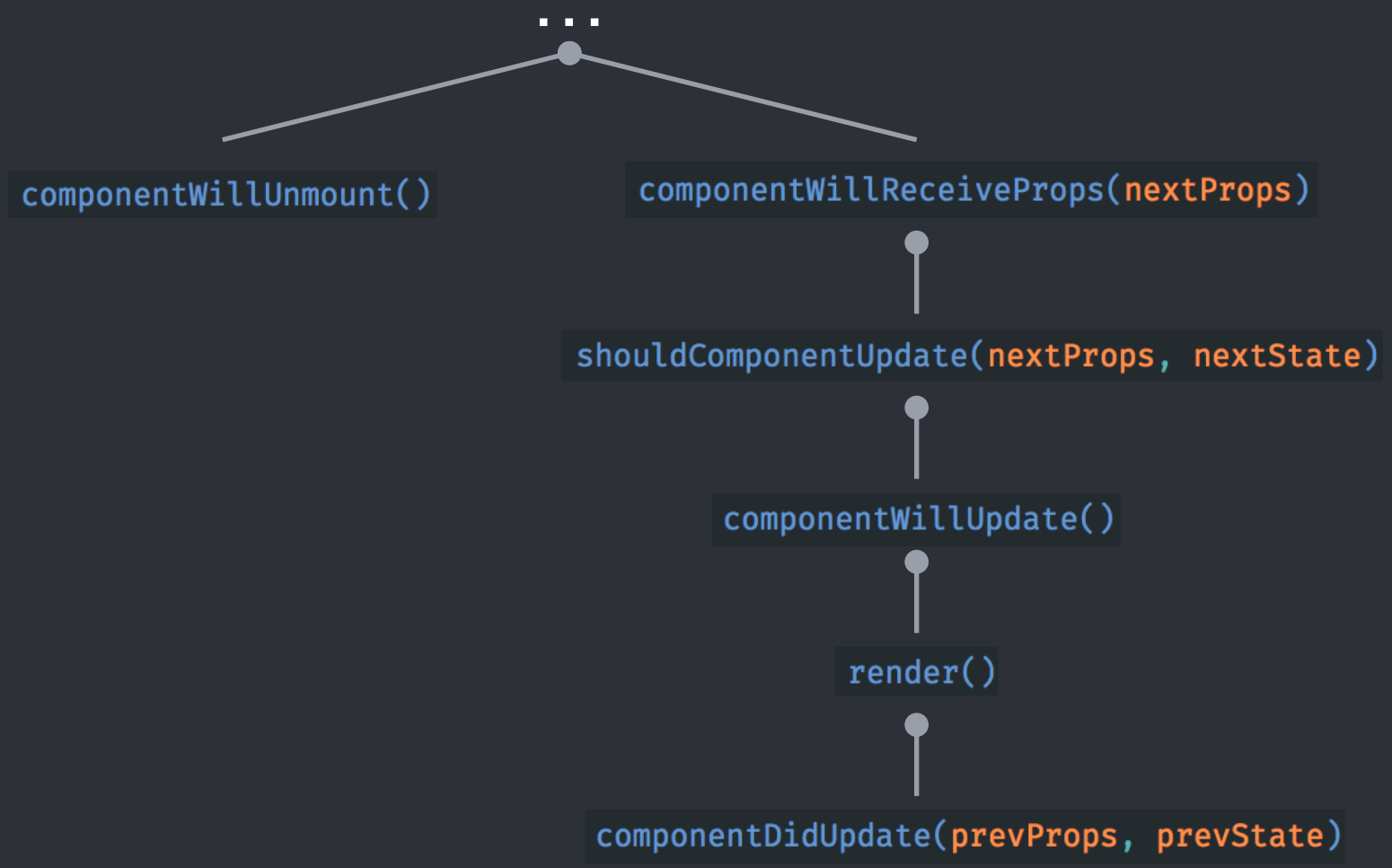
- Can’t modify directly, call setState instead
  - State is immutable
  - State changes are asynchronous
  - Calling setState will notify React it needs to re-render this component

```
private handleClick(e: React.MouseEvent<HTMLButtonElement>) {
    this.setState({ count: this.state.count + 1 });
}
```

React Component Lifecycle



React Component Lifecycle



Redux

- ”Predictable state container”
  - Single source of truth for application state (the Store)
  - Immutable state
  - Unidirectional data flow
  - Basically an event system for state changes
  - Actions, action creators, reducers
  - Doesn’t have to be used with React

Redux Actions

- Actions are objects
  - Has a type and a payload
  - Actions define what happened
  - Sends data from app to the store

```
const action = {
    type: 'ADD_MEMO',
    payload: {}
}
```

Action Creators

- Actions creators are simply functions that return actions

```
const addMemo = (title: string, body: string) => ({
    type: 'ADD_MEMO',
    payload: { title, body }
});
```

Reducers

- Respond to an action by creating a new state
- Actions are the “what”, reducers describe “how”

```
const reducer = (state, action) => {
    switch (action.type) {
        case 'ADD_MEMO':
            return {
                ...state,
                memos: [
                    ...state.memos,
                    {
                        title: action.payload.title,
                        body: action.payload.body
                    }
                ]
            };
        default:
            return state;
    }
};
```

Setting up the store

```
const reducers = combineReducers({ memo: memoReducer });

const store = createStore(reducers);
```

```
state = {
    memo: {
        memos: []
    }
}
```

Connecting to the Store

- “getState()”
  Returns entire application state object
- “dispatch(action)”
  Pass actions to trigger state changes
- “subscribe(listener)”
  Register a listener for state changes

Connecting to the Store

- “`getState()`”
  Returns entire application state object
- “`dispatch(action)`”
  Pass actions to trigger state changes
- “`subscribe(listener)`”
  Register a listener for state changes

```
store.dispatch(addMemo('My Memo','Lorem ipsum'))
```

Using Redux with React

- Redux by itself isn't very useful with React
  - Components could manually subscribe to state changes
  - Duplicated state
  - Lots of boilerplate
- React-Redux
  - Higher order components (HOC)
    - Takes in a Component and returns a new Component
    - Dependency injection
  - Application state is passed via props to your components

- The “**connect**“ HOC
  - Two arguments
    - mapStateToProps
    - mapDispatchToProps

```
const mapStateToProps = state => {
    return {
        memos: state.memos
    };
}

const mapDispatchToProps = { addMemo };

connect(mapStateToProps, mapDispatchToProps)(MemoList)
```

Example App

https://github.com/jtberglund/react-tech-talk-example

Resources

- Example app: https://github.com/jtberglund/react-tech-talk-example
- React: https://reactjs.org/
- Redux: https://redux.js.org/
- TypeScript: https://www.typescriptlang.org/