React.js

17. september 2018

Nicolai Hagen & Kristofer Selbekk

Who am I?

- Master's thesis on DHIS2 at UiO

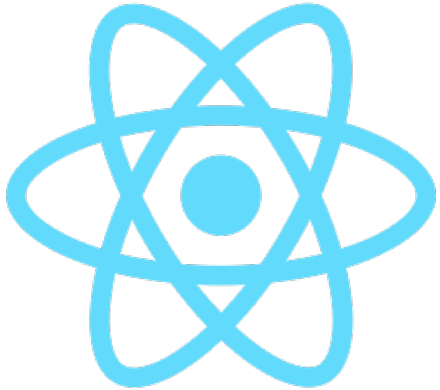
- Consultant at Bekk

- Started with React ~two years ago

A programmer in its unnatural habitat

Agenda

1. Frontend development
2. What is React?
3. Why React?
4. Core technical concepts
5. Reusability and design systems
6. The road ahead
7. Live coding of an Todo-app 🎉
8. This weeks assignments

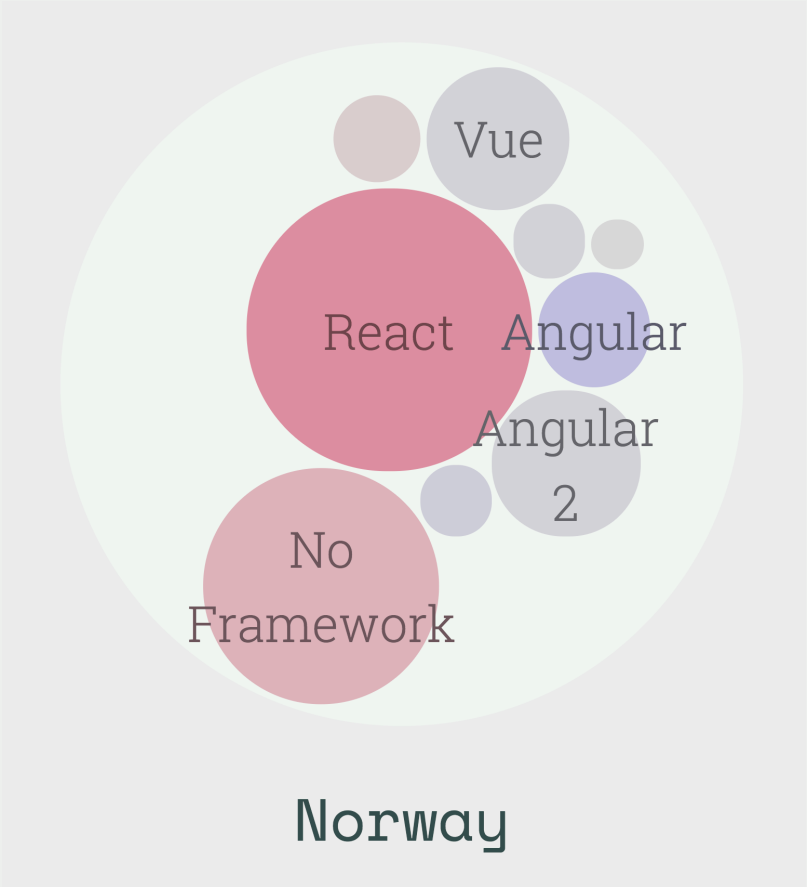


Frontend development

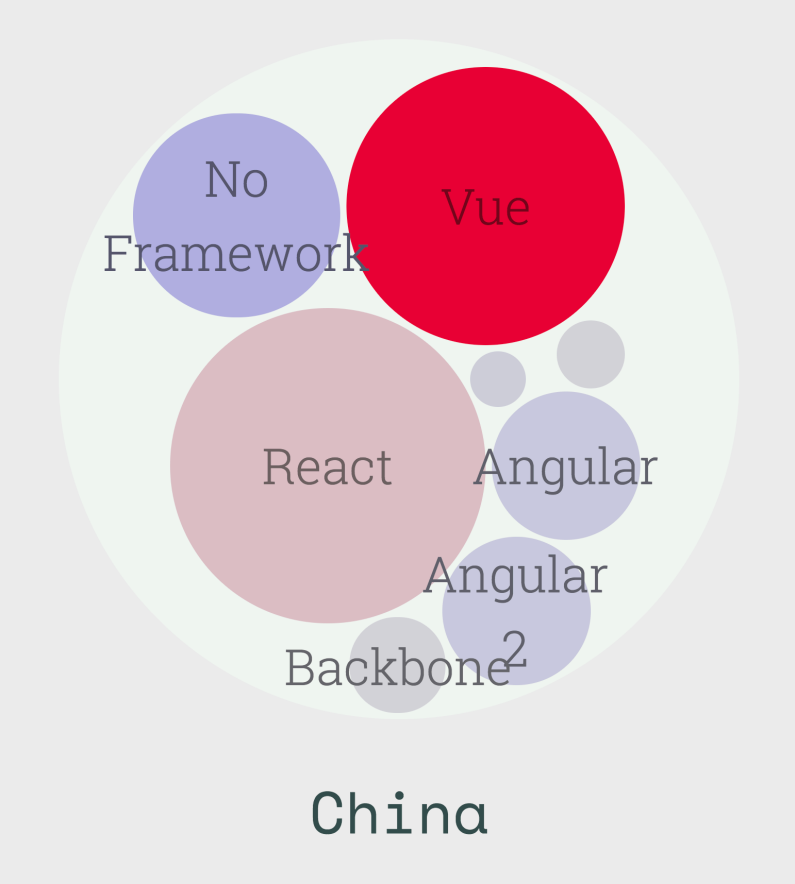

https://unsplash.com/photos/HyZaYuPXyEo

Ognjen Gatalo Follow
Software engineer / Web developer / Maker https://ognjengatalo.com
Aug 30, 2017 · 3 min read

67 useful tools, libraries and resources for saving your time as a web developer



source: stateofjs.com



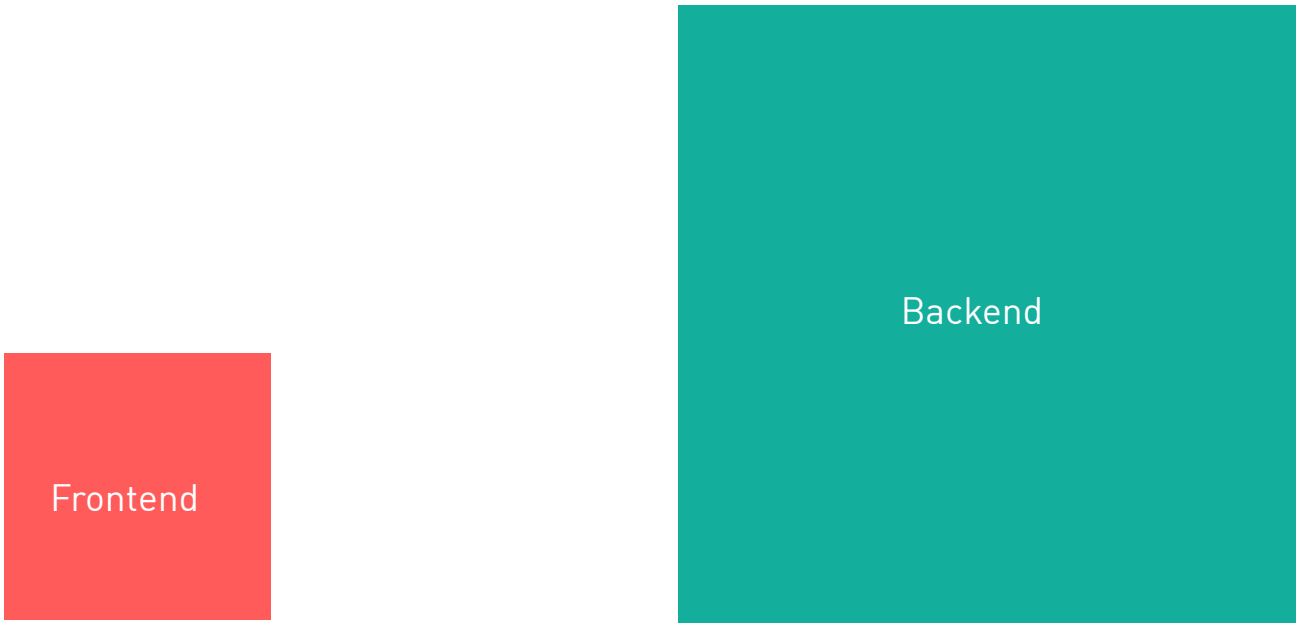
China

source: stateofjs.com

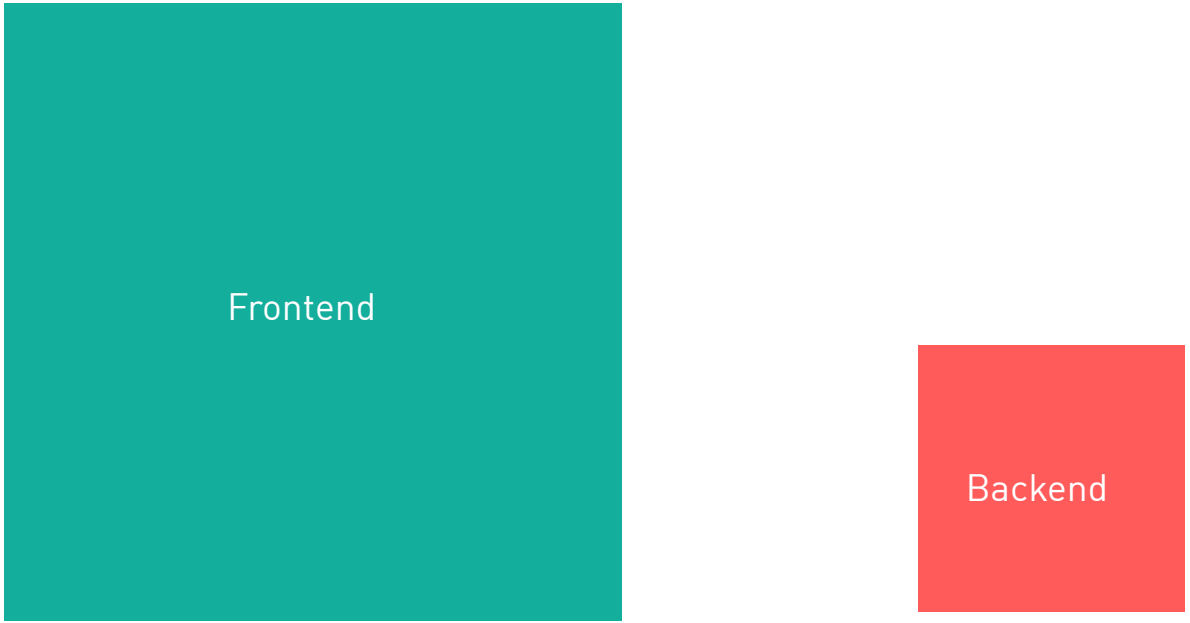
Worldwide Average

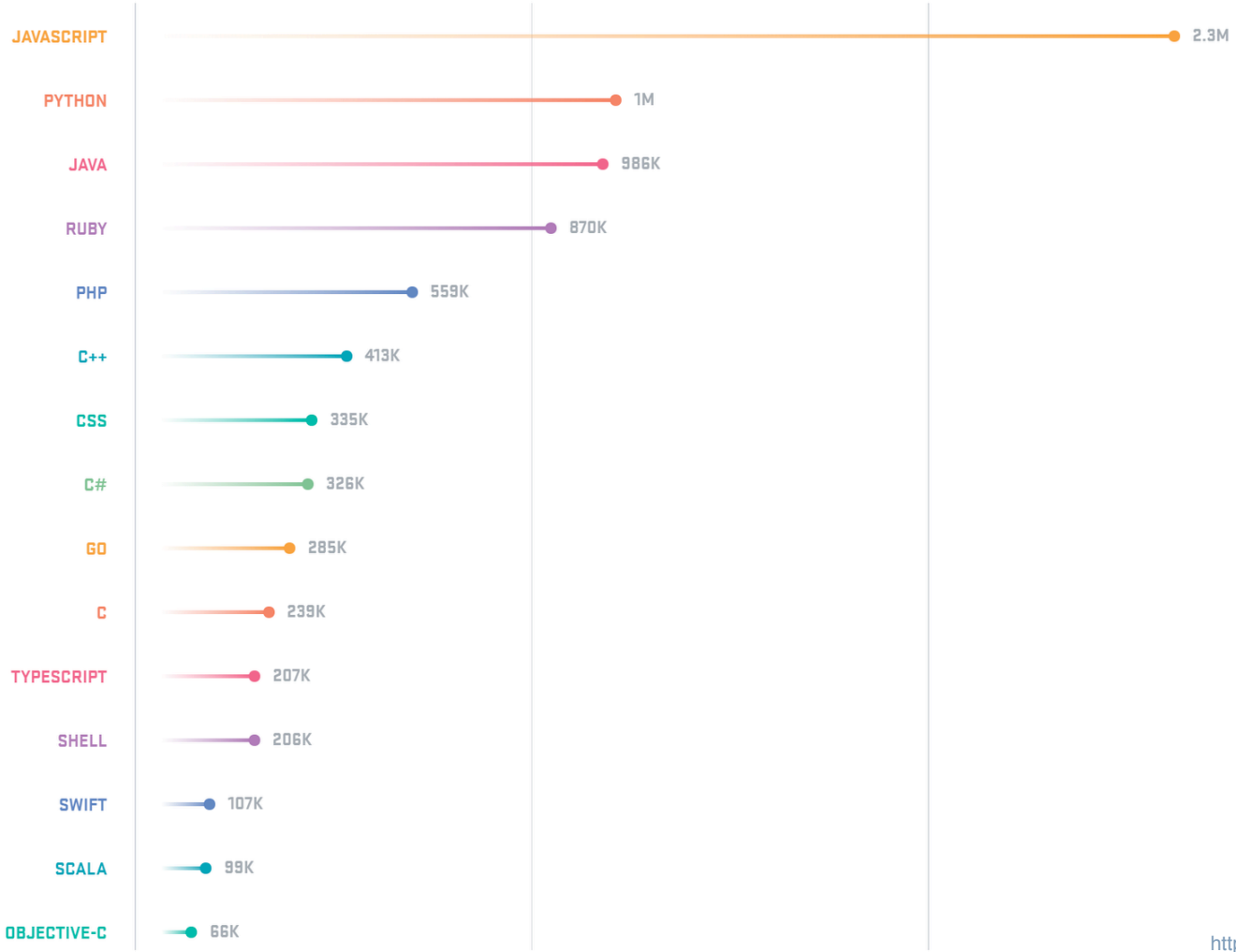
source: stateofjs.com

Early 2000's



Now?





https://octoverse.github.com/

What is React?

«A JavaScript library for building user interfaces»

– reactjs.org

Why React.js?

Composable, reusable & scalable

https://unsplash.com/photos/JRVxgAkzIsM

Declarative

https://unsplash.com/photos/9kvo3qoHJKw

JSX

<MyComponent />

DOM efficiency

https://unsplash.com/photos/kN7zlz9oDxE

https://unsplash.com/photos/BdvkiLfnNgs

Maintainers

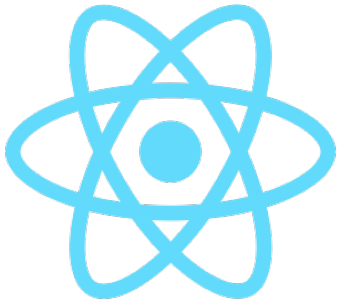
Why React.js?

Technical

- Composability, reusability, scalability
- Declarative
- JSX
- DOM efficiency

Non-technical

- Enormous community
- Resourceful maintainers



Core technical concepts

DOM and rendering

https://unsplash.com/photos/lJrTJHzQj0s

Technical concepts: Components

- Building blocks
- Composable
- Defines a part of UI to be shown

  ...or behaviour

«Components let you split the UI into independent, reusable pieces, and think about each piece in isolation»

– reactjs.org

The component and its elements

```
import React from 'react';

function MyComponent(){
  return (
    <h1>
      Hello World!
    </h1>
  );
}

export default MyComponent;
```

JSX

```
function MyComponent() {
    return (
      <div id='my-id'>
        <span>
          Hello world!
        </span>
      </div>
    );
}
```

JSX

```jsx
const element = (
  <h1 className="greeting">
    Hello, world!
  </h1>
);
```

```jsx
const element = React.createElement(
  'h1',
  {className: 'greeting'},
  'Hello, world!'
);
```

https://reactjs.org/docs/introducing-jsx.html

Switching from JSX to JS

```
Hello {name}!
```

JSX

```
let name = 'John';

function MyComponent(){
  return (
    <div id='my-id'>
      <span>
        Hello {name}!
      </span>
    </div>
  );
}
```

JSX

Prevents injection attacks

The component

...with arrow functions

```
const MyComponent = () => (
  <h1>
    Hello World!
  </h1>
);
```

Component trees and composition

```
const App = () => (
  <div>
    <CoolHeader />
    <MainPageContent />
    <AwesomeFooter />
  </div>
);
```

Rendering a full app!

```
import React from 'react';
import ReactDOM from 'react-dom';

function App(){
  return (
    <div>
      <h1>My first app!</h1>
    </div>
  );
}

ReactDOM.render(
  <App />,
  document.querySelector('main')
);
```

Technical concepts: Props

- Receive values from other Components
- Read-only
- Changes will re-render the component

https://unsplash.com/photos/CDDM_Wkcmks

Passing props

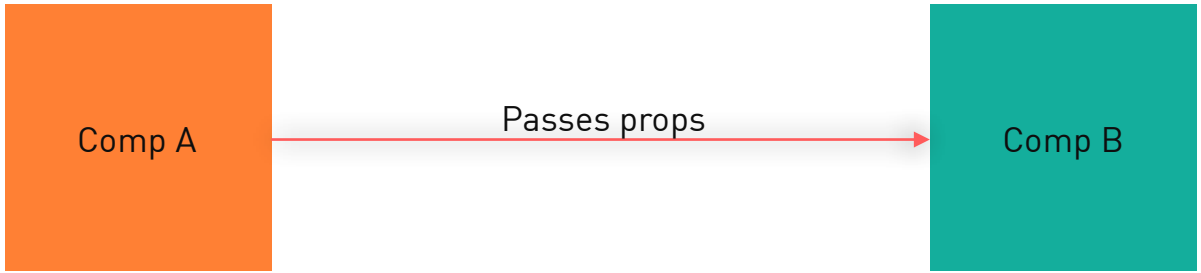
```
import React from 'react';

function Greeter(props){
  return (
    <h1>
      Hello { props.name }!
    </h1>
  );
}

function App(){
  return <Greeter name='Dan' />
}
```

Passing props

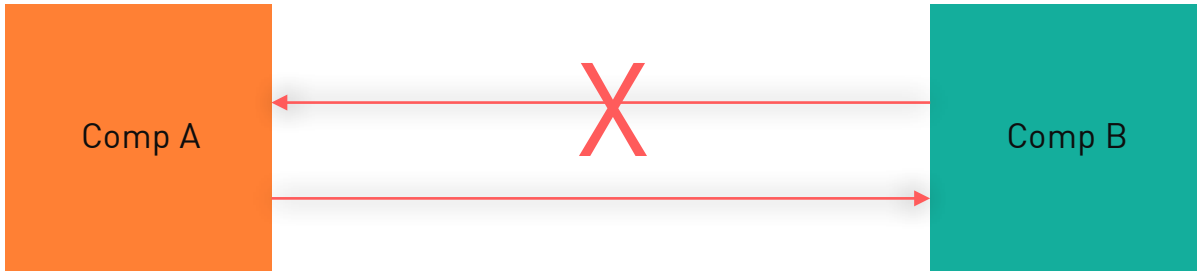
... destructured

```
import React from 'react';

function Greeter({ name }){
  return (
    <h1>
      Hello { name }!
    </h1>
  );
}

function App(){
  return <Greeter name='Dan' />
}
```

Props:
Unidirectional data flow



Props:
Unidirectional data flow



Technical concepts: State (and lifecycle)

- Values that may change over time in a Component
- Used in Class Components
- Changes will re-render the component

The class component

```
class MyComponent extends React.Component {
  render(){
    return (
      <h1>
        Hello World!
      </h1>
    );
  }
}
```

State

```
class NameForm extends React.Component {
  state = {
    value: '',
  };

  handleChange = (event) => {
    this.setState({value: event.target.value});
  };

  handleSubmit = () => {
    const name = this.state.value;
    // handle submit click
  };

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <label>
          Name:
          <input
            type="text"
            value={this.state.value}
            onChange={this.handleChange}
          />
        </label>
        <input type="submit" value="Submit" />
      </form>
    );
  }
}
```

Lifecycle methods

```
class MyComponent extends React.Component {

  componentDidMount(){
    // e.g., fetch data from server
  }

  render(){
    return (
      <h1>
        Hello World!
      </h1>
    );
  }
}
```

State

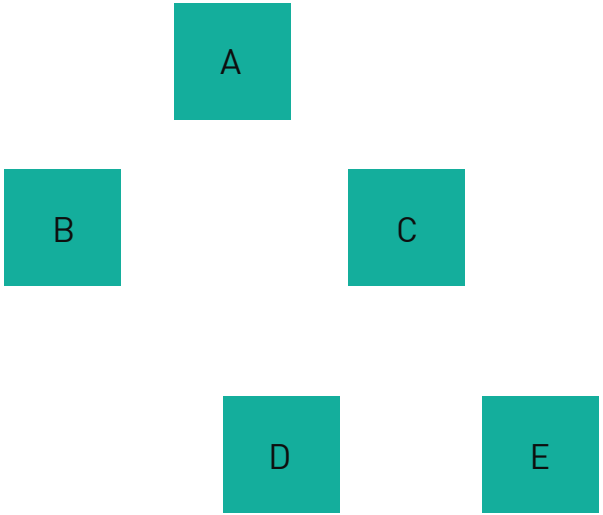
```
class EmployeePage extends React.Component {

  state = {
    pending: false,
    data: null,
  };

  async componentDidMount() {
    this.setState({pending: true});
    const data = await server.get('organisation/23122');
    this.setState({
      pending: false,
      data,
    });
  }

  . . .
```

State

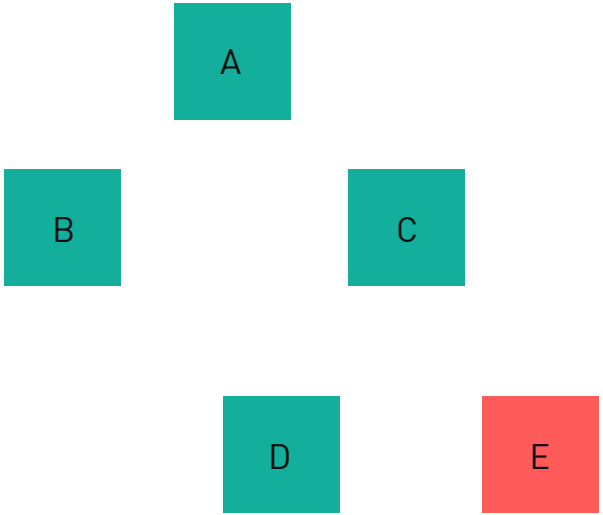
```
  . . .

render(){

  const { pending, data } = this.state;

  if(!data){
    return null;
  }

  if(pending){
    return <Spinner />;
  }

  return (
    <div>
      <h1>
        Change accesses to {data.organisation.name}
      </h1>
      <EmployeeList employees={data.organisation.employees} />
    </div>
  );

}
```

```
class EmployeePage extends React.Component {
  state = {
    pending: false,
    data: null,
  };

  async componentDidMount() {
    this.setState({pending: true});
    const data = await server.get('organisation/23122');
    this.setState({
      pending: false,
      data,
    })
  }

  render() {
    const {pending, data} = this.state;
    if (!data) { return null };
    if (pending) { return <Spinner/> };
    return (
      <div>
        <h1>
          Change accesses to {data.organisation.name}
        </h1>
        <EmployeeList employees={data.organisation.employees}/>
      </div>
    );
  }
}
```

Technical concepts: Virtual DOM

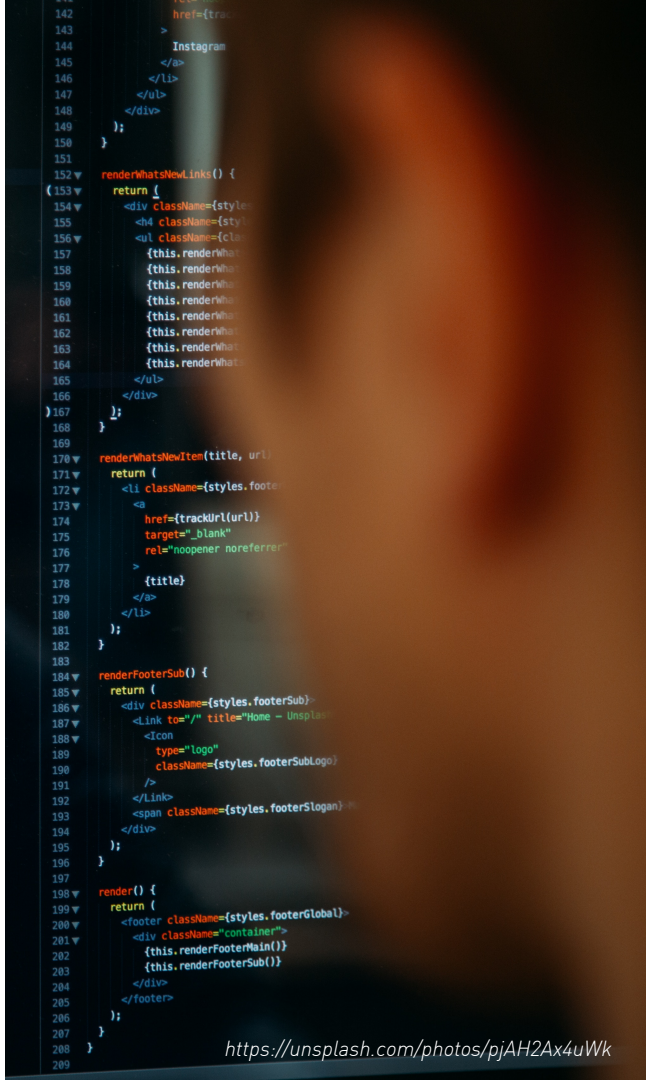


Technical concepts: Virtual DOM



Technical concepts

- Components. The main building blocks
- JSX. HTML markup empowered by JavaScript
- Props. Sharing values between Components
- State. Internal changeable values inside Components
- Props & state. Trigger re-renders when changed
- Lifecycle methods. Controlling updates to Components
- Virtual DOM. Abstraction layer on top of the DOM. Makes partial re-rendering possible.



https://unsplash.com/photos/pjAH2Ax4uWk

Testing declarative code!

given input **x**

you should expect output **y**

Reusability and design systems

SpareBank 1 Designsystem

SpareBank 1 sitt felles språk på tvers av disipliner, for å sikre konsistent design i løsningene våre.

Kom i gang

Stil og tone

Lær om hvordan vi bruker språk til å snakke direkte med kundene våre.

Visuell identitet

Hvordan vi tilnærmer oss utformingen av våre digitale produkter.

Universell utforming

Retningslinjer for design og utvikling av tilgjengelige løsninger.

Komponenter

Vårt komponentbibliotek, implementert i Less og React.

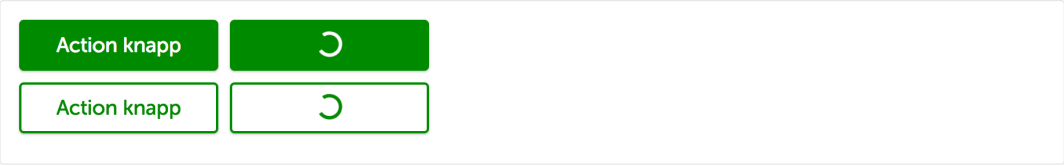
Komponenter

Vårt bibliotek av felles komponenter, implementert i Less og React.

ActionButton

```
import { ActionButton } from '@sb1/ffe-buttons-react';
```

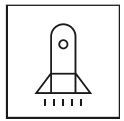
PROPS & METHODS



VIEW CODE

Summarising part 1

The road ahead



Write once, run anywhere

React Native

Today's landscape

Larger frontend web applications

Users demand interactive websites

Demand for fast websites

React.js

Composable, reusable and scalable

Built for creating interactive sites

Virtual DOM

React workshops this week!

Tuesday 12:15 - 14:00 (group 2)

Wednesday 10:15 - 12:00 (group 4)

http://tiny.cc/react-uio

<TodoApp next={true} />

```
break;
```

Who am I?

- Senior consultant & React practice lead at Bekk

- Started with React ~four years ago

Resources

- Create React App!

- Official React.js documentation

- egghead.io

- Babel REPL