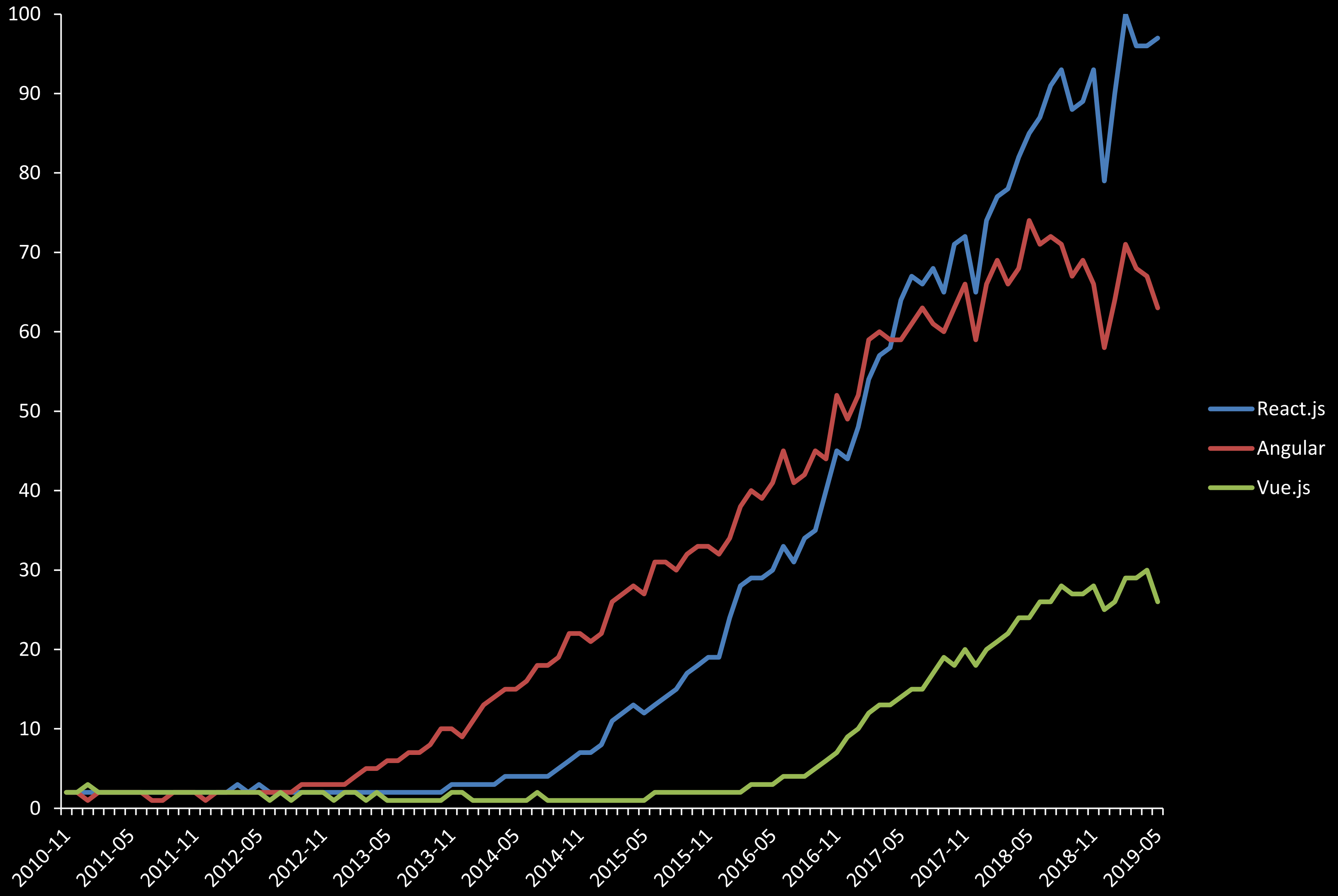
Internet Applications

React

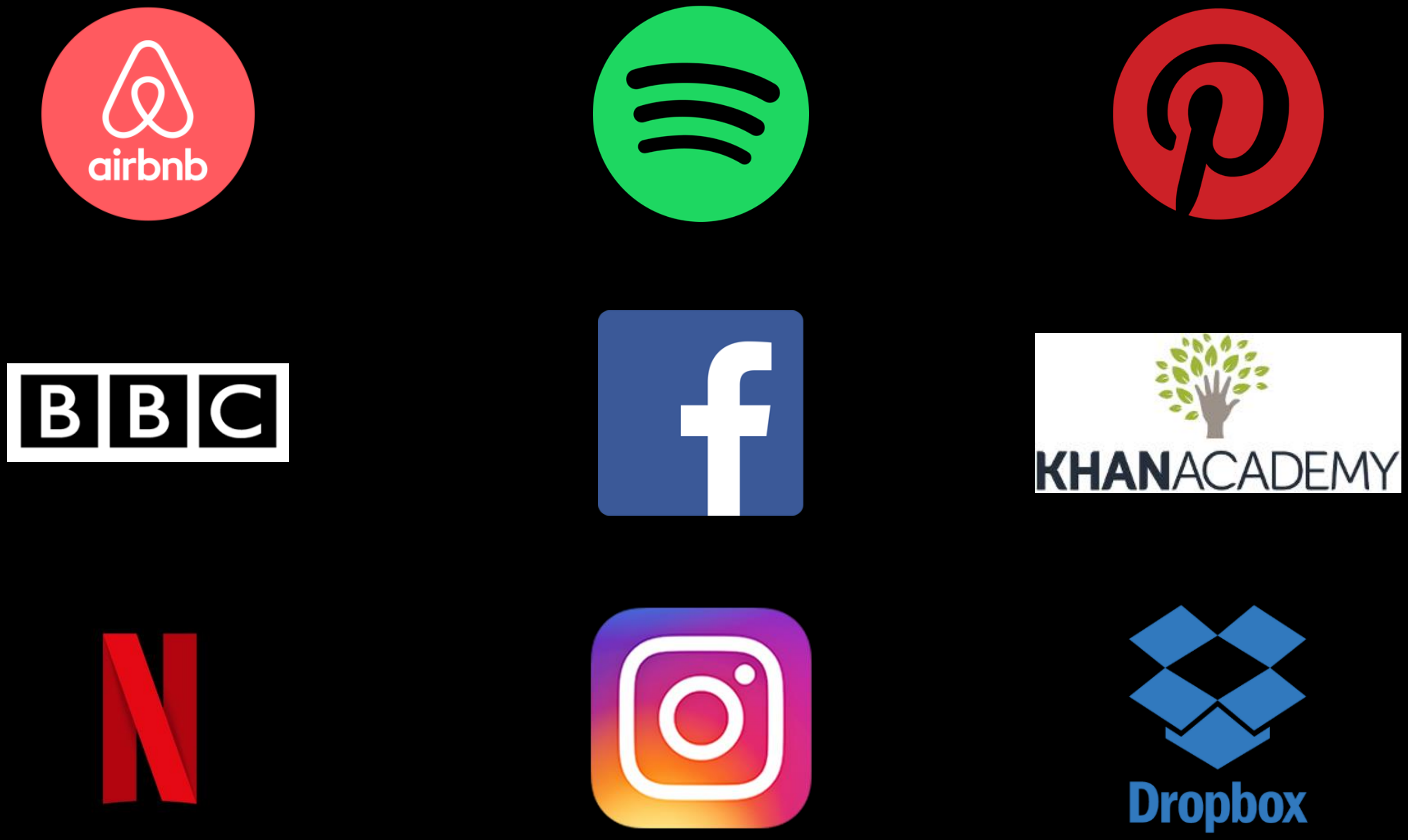
JavaScript frameworks landscape



Key characteristics

- A library for creating UIs
- Created and sustained by Facebook
- Declarative
- Based on components
- Everything is written in JavaScript

Who uses React?



as well as **hundreds** of other companies…

Hello, world!

```
<div id="root"></div>
<script type="text/babel">
  ReactDOM.render(
    <h1>Hello, world!</h1>,
    document.getElementById('root')
  );
</script>
```

JSX

- JavaScript expression

```
const element = <h1>Hello, world!</h1>;
```

- Expressions in curly braces

```
const element = <h1>2 + 2 = {2 + 2}</h1>;
```

- Attributes

```
const element = <img src={user.avatarUrl}></img>;
```

- camelCase notation for attributes (e.g., className)

JSX

- Text displayed as string
- Transpilation

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

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

JSX

- Immutability
- Only the things that change are being updated

```
function tick() {
  const element = (
    <div>
      It’s{' '}
      {new Date().toLocaleTimeString()}.
    </div>
  );
  ReactDOM.render(
    element,
    document.getElementById('root')
  );
}

setInterval(tick, 1000);
```

Component

- A function returning a React element

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}


class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Component

- Usage

  ```
  const element = <Welcome name="Maciej" />;
  ```

- Names begin with capital letters
- Passing data using props parameter
- Has to be a pure function w.r.t. props parameter (props is read-only)

Pure functions

- Always return the same result for the same parameter values
- Leave no side effects

State management

```
class User extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      counter: 0
    };
  }
  render() {
    return (
      <div>
        Welcome, {this.props.userName}!
      </div>
    );
  }
}

ReactDOM.render(
  <User userName="Maciej" />,
  document.getElementById('root')
);
```

State management

- Changing state

  ```
  this.setState({counter: 0});
  ```

- Impacts only selected fields
- Updates view
- When relying on previous state

  ```
  this.setState((prevState, props) => ({
    counter: prevState.counter + 1
  }));
  ```

- Top-down state propagation

Events

```
class User extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      userName: this.props.userName,
      counter: 0
    };
  }
  count() {
    this.setState({
      counter: this.state.counter + 1
    });
  }
  render() {
    return (
      <div>
        <div>
          Welcome, {this.state.userName}!
          This is your {this.state.counter} click.
        </div>
        <button onClick={this.count}>Count</button>
      </div>
    );
  }
}
```

This doesn’t work!

Events

```
class User extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      userName: this.props.userName,
      counter: 0
    };
  }
  count = () => {
     this.setState({
      counter: this.state.counter + 1
    });
  }
  render() {
    return (
      <div>
        <div>
          Welcome, {this.state.userName}!
          This is your {this.state.counter} click.
        </div>
        <button onClick={this.count}>Count</button>
      </div>
    );
  }
}
```

Events

```
class User extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      userName: this.props.userName,
      counter: 0
    };
  }
  count() {
     this.setState({
      counter: this.state.counter + 1
    });
  }
  render() {
    return (
      <div>
        <div>
          Welcome, {this.state.userName}!
          This is your {this.state.counter} click.
        </div>
        <button onClick={this.count.bind(this)}>Count</button>
      </div>
    );
  }
}
```

Events

```
class User extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      userName: this.props.userName,
      counter: 0
    };
    this.count = this.count.bind(this);
  }
  count() {
    this.setState((prevState, props) => ({
      counter: prevState.counter + 1
    }));
  }
  render() {
    return (
      <div>
        <div>
          Welcome, {this.state.userName}!
          This is your {this.state.counter} click.
        </div>
        <button onClick={this.count}>Count</button>
      </div>
    );
  }
}
```

Lists

```
function User(props) {
  return <li>{props.name}</li>
}

function Users(props) {
  const users = props.list;
  return (
    <div>
      <h1>List of users</h1>
      <ol>{users.map(user =>
            <User key={user}
                  name={user} />
      )}</ol>
    </div>
  )
}

ReactDOM.render(
  <Users list={["Dante", "Patrokles", "Mieciu", "Heniu"]} />,
  document.getElementById('root')
);
```

Forms

- The problem with state
- Controlled components

Forms

```
class UserForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {name: ''};
  }
  handleChange(event) {
    this.setState({name: event.target.value});
  }
  handleSubmit(event) {
    event.preventDefault();
    ...
  }
  render() {
    return (
      <form onSubmit={this.handleSubmit.bind(this)}>
        <input type="text" value={this.state.name}
               onChange={this.handleChange.bind(this)} />
        <input type="submit" value="Login" />
      </form>
    );
  }
}
```

Component lifecycle – mounting

- constructor()
- componentWillMount()
- render()
- componentDidMount()

Componentn lifecycle – update

- componentWillReceiveProps()
- shouldComponentUpdate()
- componentWillUpdate()
- render()
- componentDidUpdate()

Component lifecycle – other

- componentWillUnmount()
- componentDidCatch()

Virtual DOM

1. Virtual DOM gets updated.
2. Comparing current V-DOM with previous version.
3. React looks for changes.
4. Changes are propagated to true DOM.
5. Changes in DOM cause change in the application interface.

Configuration

- Babel
- Webpack
- React Developer Tools

Quickstart

- https://codepen.io/
- Single html file with react, react-dom, and babel scripts attached
- Create React App

```
npx create-react-app my-app

cd my-app
npm start
```