<epam>

React:
Core Concepts

By Pavel Yukhnovich

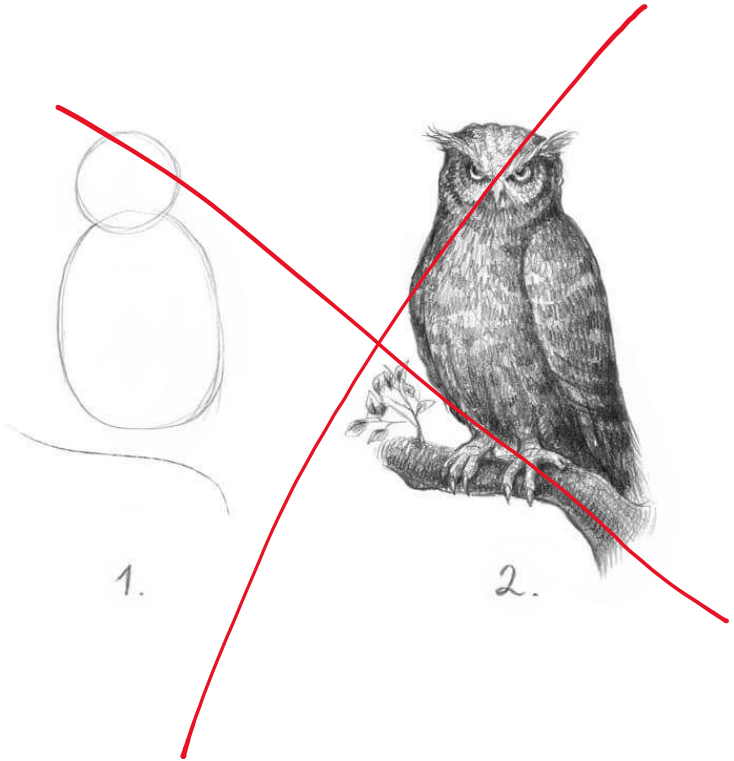
July 2020

Agenda

- INTRODUCTION TO REACT
- JSX
- COMPONENTS CORE CONCEPTS
- PROPS
- STATE
- VIRTUAL DOM

Disclaimer Slide

- PRESENTATION WILL BE SHARED TODAY
- ALL LINKS WILL BE SHARED AS WELL
- Q&A SESSION IN THE END
- CONTACT VIA TEAMS ONCE YOU HAVE QUESTIONS



MUST HAVE THINGS BEFORE GETTING STARTED

JAVASCRIPT: ES6+ CONCEPTS

Javascript: es6+ concepts

- ES6 classes
- Arrow functions
- Let/const
- Template strings
- Spread/Rest operators
- Destructuring
- Map, Reduce and Filter
- Ternary operator
- Import and Export statements
- Promises
- Async Functions
- Default parameters etc

PACKAGE MANAGER

Package Manager

or

INTRODUCTION TO REACT

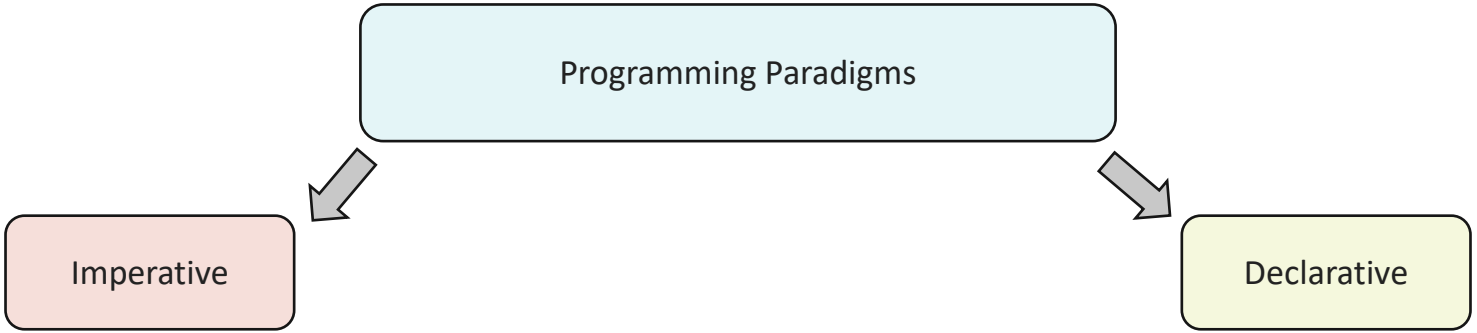
DECLARATIVE PROGRAMMING PARADIGM

Programming Paradigms

Programming Paradigms

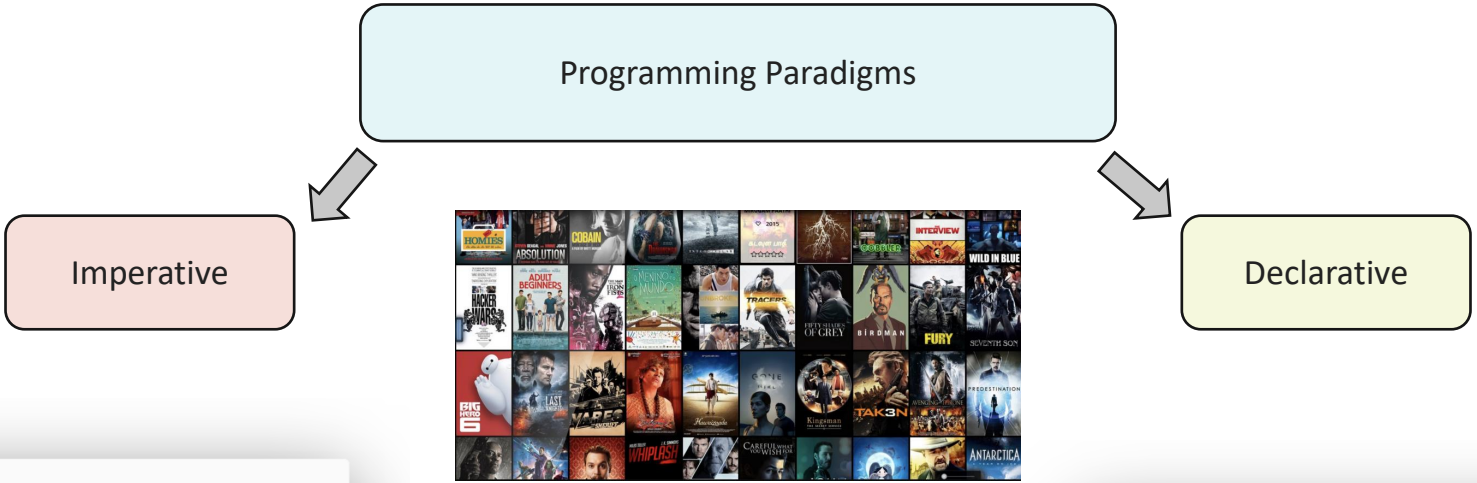
Task: Find best movies.

Programming Paradigms



Classic / ‘Old approach’	Modern / ‘New approach’
An app based on this paradigm is made up of a **series of instructions** that tell the computer **what it should calculate/do and in what order**.	An app based on this paradigm is made up of instructions for **how the program should deal with an input**. Calculations are performed by manipulation of values, with the procedure controlled by the process of recursion.
How?	**What?**
The desired **solution path** is specified.	The desired **result** is specified.

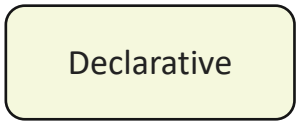
Programming Paradigms




```
const bestMovies = []
for (let i = 0; i < movies.length; i++) {
  let movie = movies[i]
  if (movie.rating >= 5 && movie.year > 2019) {
    bestMovies.push(movie)
  }
}
```

```
const bestMovies = movies.filter(function (movie) {
  return movie.rating >= 5 && movie.year > 2019
})
```

Programming Paradigms



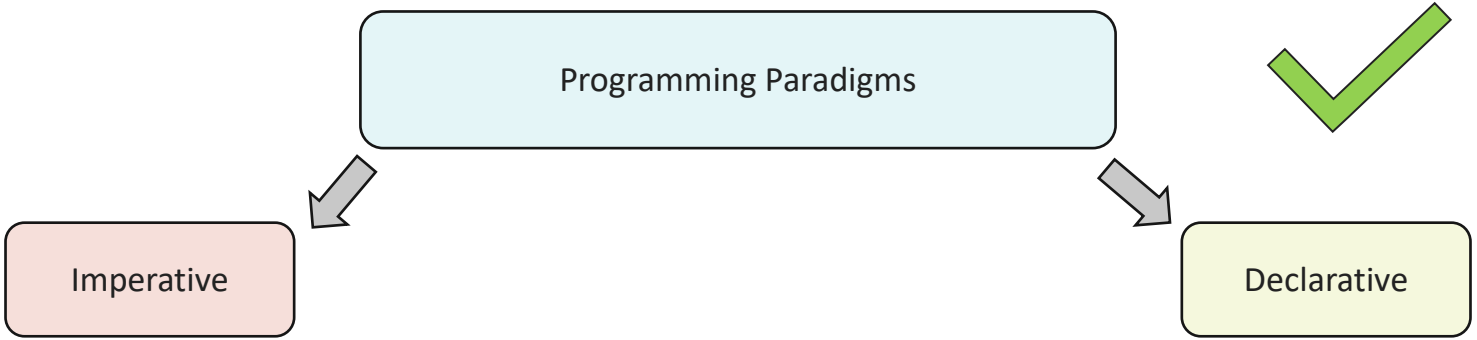
- Easy to read
- Easy to learn
- Solution path is clear
- Easy to modify

- More code
- Might be too complicated
- High risk of errors
- Hard to scale/optimize
- Becomes less popular

- Algorithm optmizations
- Less code
- Shorter code
- Efficient code
- Low risk of errors
- ‘Black box’ coding
- Easy to scale
- Popular

- ‘Black box’ coding
- Hard to scale/optimize

Programming Paradigms



- Easy to read
- Easy to learn
- Solution path is clear
- Easy to modify

- More code
- Might be too complicated
- High risk of errors
- Hard to scale/optimize
- Becomes less popular

- Algorithm optmizations
- Less code
- Shorter code
- Efficient code
- Low risk of errors
- ‘Black box’ coding
- Easy to scale
- Popular

- ‘Black box’ coding
- Hard to scale/optimize

Programming Paradigms

Imperative

- Easy to read
- Easy to learn
- Solution path is clear
- Easy to modify

- More code
- Might be too complicated
- High risk of errors
- Hard to scale/optimize
- Becomes less popular

```
const bestMovies = []
for (let i = 0; i < movies.length; i++) {
  let movie = movies[i]
  if (movie.rating >= 5 && movie.year > 2019) {
    bestMovies.push(movie)
  }
}
```

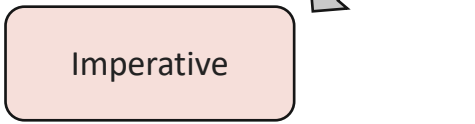
Declarative

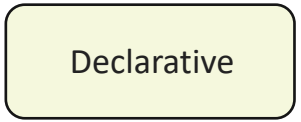
- Algorithm optmizations
- Less code
- Shorter code
- Efficient code
- Low risk of errors
- ‘Black box’ coding
- Easy to scale
- Popular

- ‘Black box’ coding
- Hard to scale/optimize

```
const bestMovies = movies.filter(function (movie) {
  return movie.rating >= 5 && movie.year > 2019
})
```

Programming Paradigms





- Easy to read
- Easy to learn
- Solution path is clear
- Easy to modify

- More code
- Might be too complicated
- High risk of errors
- Hard to scale/optimize
- Becomes less popular

- Algorithm optmizations
- Less code
- Shorter code
- Efficient code
- Low risk of errors
- ‘Black box’ coding
- Easy to scale
- Popular

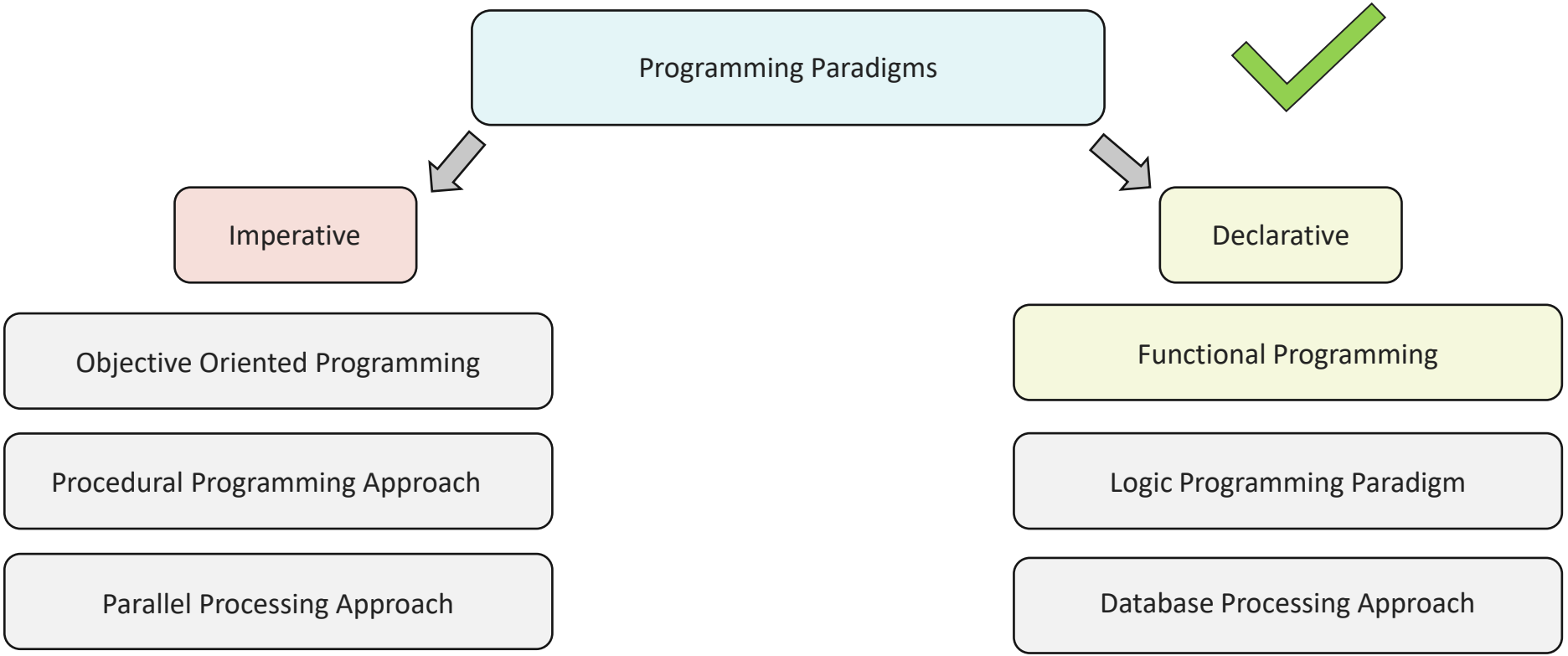
- ‘Black box’ coding
- Hard to scale/optimize

```
const bestMovies = []
for (let i = 0; i < movies.length; i++) {
  let movie = movies[i]
  if (movie.rating >= 5 && movie.year > 2019) {
    bestMovies.push(movie)
  }
}
```

```
const bestMovies = getBestMovies(movies)
```

FUNCTIONAL PROGRAMMING

Programming Paradigms



Functional Programming

Functional Programming principles:

- Purity
- Higher-Order Functions etc

Functional Programming Principles: Purity

Pure functions:

- Pure functions **must take arguments**.
- The same input (arguments) will **always** produce the same output (return).
- Pure functions rely only on local state and **do not mutate external state** (note: console.log changes global state).
- Pure functions **do not produce side effects**.
- Pure functions **cannot call** impure functions.

```
function priceAfterTax(productPrice) {
  return productPrice * 0.2 + productPrice
}
```

```
let pureSum = (a, b) => a + b
```

Functional Programming Principles: Purity

Impure functions:

- Impure functions **might take no arguments**.
- The same input (arguments) **will not produce** the same output (return).
- Impure functions **mutate external state**.
- Impure functions might **call** pure functions.
- Impure functions **produce side effects**.

Side effects:

- Making a HTTP request
- Mutating data
- Printing to a screen or console
- DOM Query/Manipulation
- Math.random()
- Getting the current time

```
// impure function producing a side effect
function showAlert() {
  alert('This is a side effect!')
}



// some variable that is mutated
let someNum = 8;

// this is NOT a pure function
function impureHalf() {
  return someNum / 2;
}

// impure function
//that resembles a pure function,
// but returns different results
given the same inputs
function getRandomRange(min, max) {
  return Math.random() * (max - min) + min
}
```

Functional Programming Principles: Higher Order Functions

Higher Order Function:

- accepts another function as an argument, or
- returns a function as a result.

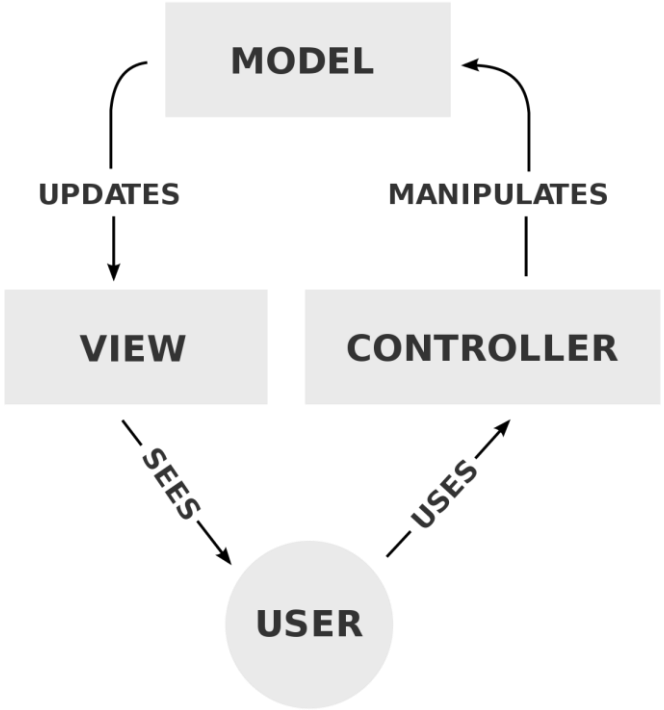
```
const myBtn = document.getElementById('myButton')

// anonymous callback function
myBtn.addEventListener('click', function (e) {
  console.log(`Click event: ${e}`)
})

// named callback function
function btnHandler(e) {
  console.log(`Click event: ${e}`)
}
myBtn.addEventListener('click', btnHandler)
```

```
let sayHi = () => alert('Hi!')

let greet = (greeting) => greeting()

greet(sayHi) // alerts "Hi!"
```

```
const double = (x) => x * 2

const timesTwo = double

timesTwo(4) // result: returns 8
```

WHY REACT

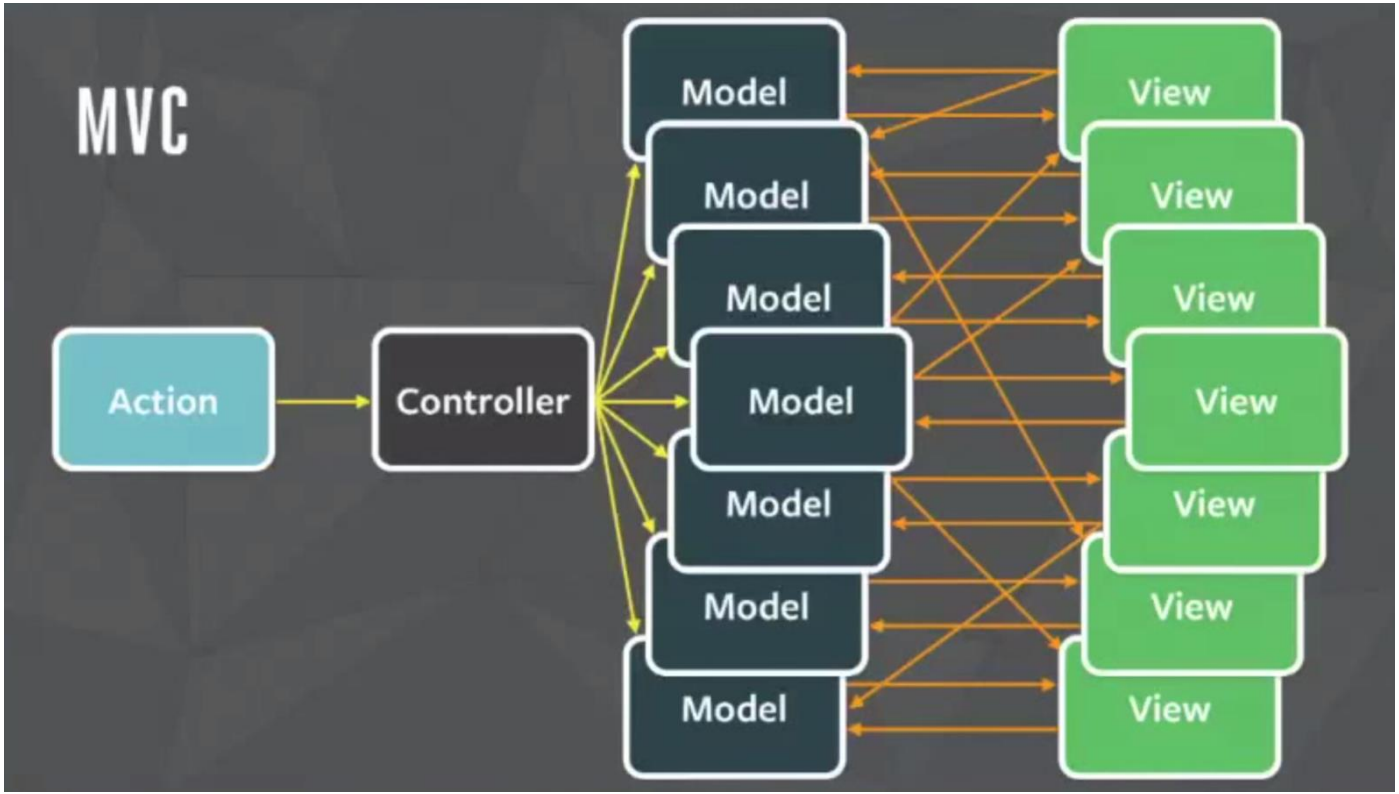
Good old MVC: 2011-2012 ..



ANGULARJS by Google

BACKBONE.JS

Good old MVC


MVC
Action
Controller
Model
Model
Model
Model
Model
Model
Model
View
View
View
View
View
View
View

Why React was created by Facebook?

facebook

99+ messages

3 messages

0 messages

Why React was created by Facebook?

facebook

99+ messages

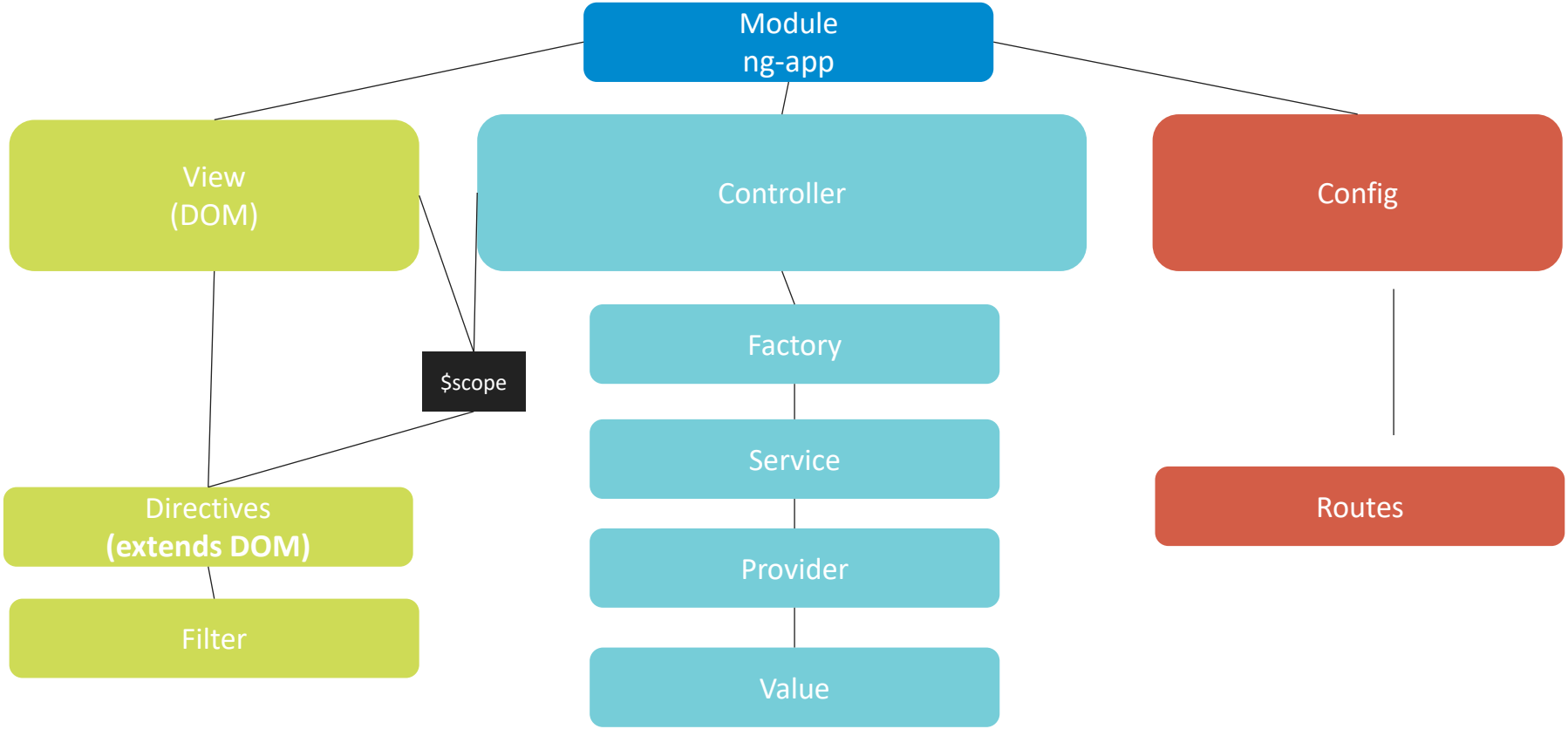
3 messages

0 messages

Wish List

- Unified state of the app
- Components reuse
- Parallel development
- Declarative approach
- Modularity
- Secure app
- Performant app

Why don’t you want to use AngularJS?



Why don’t you want to use AngularJS?

Templates
- ng-disabled
- ng-cloak
- ng-hide
- ng-if
- ng-repeat
- ng-show
- ng-switch
- ng-view

Data Binding
- ng-bind
- ng-href
- ng-init
- ng-model
- ng-src
- ng-style

Forms
- ng-maxlength
- ng-minlength
- ng-pattern
- ng-required
- ng-submit

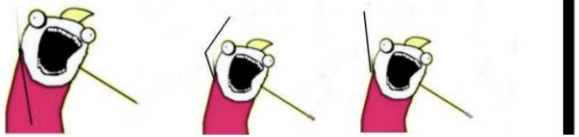
Application
- ng-app
- ng-controller

Key AngularJS Directives

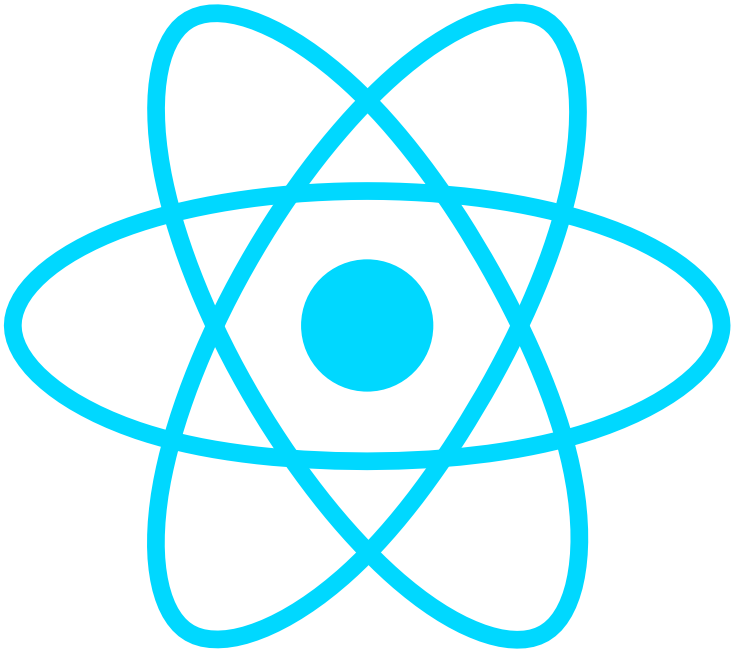
Behavior
- ng-blur
- ng-change
- ng-checked
- ng-click
- ng-key*
- ng-mouse*

Dev community in any year

- Pure Javascript
- Pure HTML
- Simple data flow
- No problems in learning
- No issues while using
- Reusable components
- Great Performance
- Huge Ecosystem



React: Main info



- May 29, 2013
- Original Author: Jordan Walke
- Current version: 16.13.1
- MIT Licence
- https://reactjs.org/

React: Main info

- Virtual DOM;
- One-direction data flow;
- Reusable components;
- Great Developer Tools;
- Huge and growing ecosystem;
- Easy to learn, easy to use
- Dev Community

React: Releases - NPM

- 59 116 Dependents
- 7.5 mln downloads weekly
- 3 dependencies

Almost any new module at npm these days at first thinks about React support and then might go to any other framework.

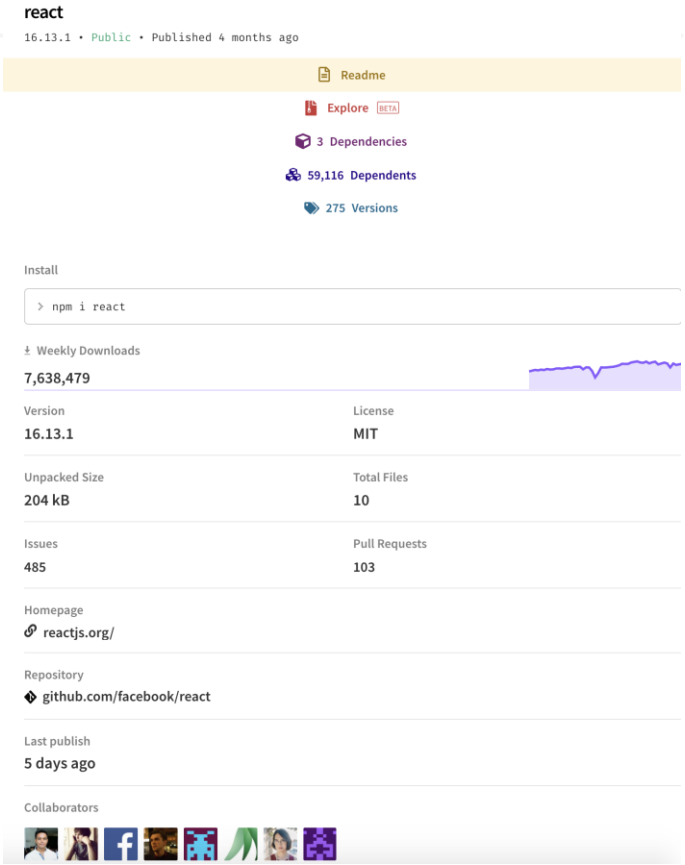
i.e. If there is a library in npm, it won't be much of a trouble to use it with React

React: 204 kB

Angular: 2.08 Mb

Vue.JS: 2.98 Mb

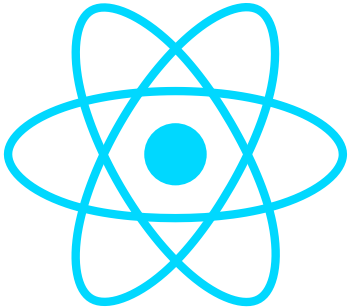
Available:
59016 packages
17174 packages
17627 packages



React: Competitors

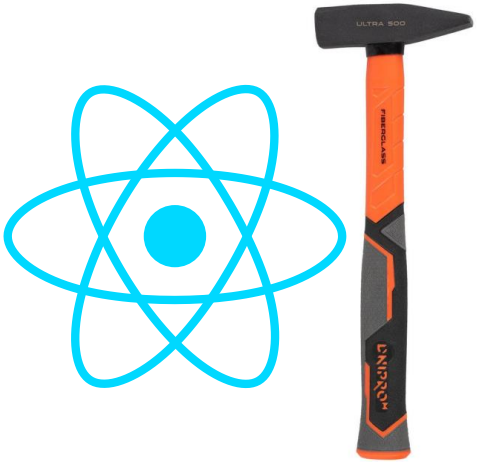
Angular

Vue.JS

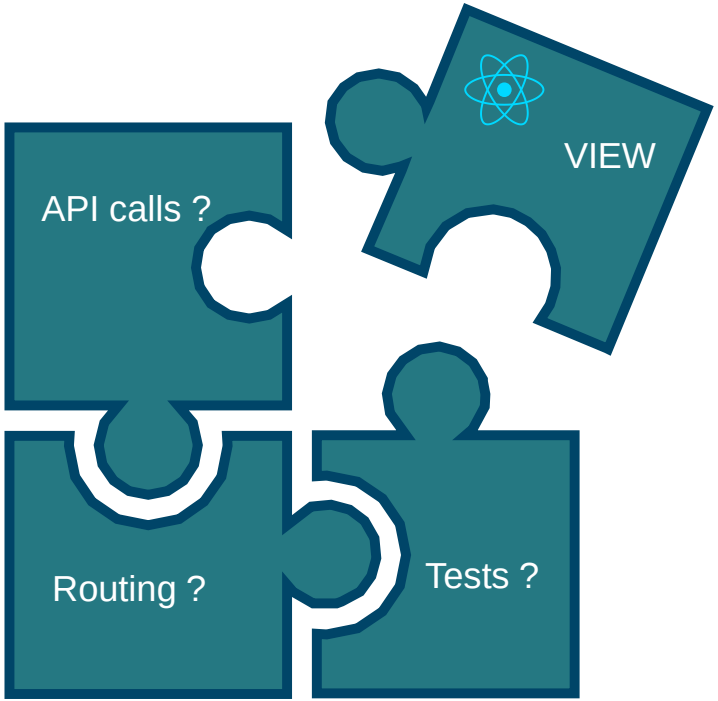


is a **VIEW** library

React vs AngularJS



React: convention over configuration



React: Releases - GitHub

- 16.13.1 (semantic versioning)
- Small and often releases

Breaking changes are inconvenient for everyone, so we try to minimize the number of major releases – for example, React 15 was released in April 2016 and React 16 was released in September 2017; React 17 isn't expected until 2019.

Instead, we release new features in minor versions. That means that minor releases are often more interesting and compelling than majors, despite their unassuming name.

- 2.9 mln repos use React
- 141 000 stars
- True open source – 1346 contributors

* Open Source Code of Conduct

16.13.1 (March 19, 2020)

React DOM

- Fix bug in legacy mode Suspense where effect clean-up functions are not fired. This only affects users who use Suspense for data fetching in legacy mode, which is not technically supported. (@acdlite in #18238)
- Revert warning for cross-component updates that happen inside class render lifecycles (`componentWillReceiveProps` , `shouldComponentUpdate` , and so on). (@gaearon in #18330)

16.13.0 (February 26, 2020)

React

- Warn when a string ref is used in a manner that's not amenable to a future codemod (@lunaruan in #17864)
- Deprecate `React.createFactory()` (@trueadm in #17878)

React DOM

- Warn when changes in `style` may cause an unexpected collision (@sophiebits in #14181, #18002)
- Warn when a function component is updated during another component's render phase (@acdlite in #17099)
- Deprecate `unstable_createPortal` (@trueadm in #17880)
- Fix `onMouseEnter` being fired on disabled buttons (@AlfredoGJ in #17675)
- Call `shouldComponentUpdate` twice when developing in `StrictMode` (@bvaughn in #17942)
- Add `version` property to ReactDOM (@ealush in #15780)
- Don't call `toString()` of `dangerouslySetInnerHTML` (@sebmarkbage in #17773)
- Show component stacks in more warnings (@gaearon in #17922, #17586)

Concurrent Mode (Experimental)

- Warn for problematic usages of `ReactDOM.createRoot()` (@trueadm in #17937)
- Remove `ReactDOM.createRoot()` callback params and added warnings on usage (@bvaughn in #17916)
- Don't group Idle/Offscreen work with other work (@sebmarkbage in #17456)
- Adjust `SuspenseList` CPU bound heuristic (@sebmarkbage in #17455)
- Add missing event plugin priorities (@trueadm in #17914)
- Fix `isPending` only being true when transitioning from inside an input event (@acdlite in #17382)
- Fix `React.memo` components dropping updates when interrupted by a higher priority update (@acdlite in #18091)
- Don't warn when suspending at the wrong priority (@gaearon in #17971)
- Fix a bug with rebasing updates (@acdlite and @sebmarkbage in #17560, #17510, #17483, #17480)

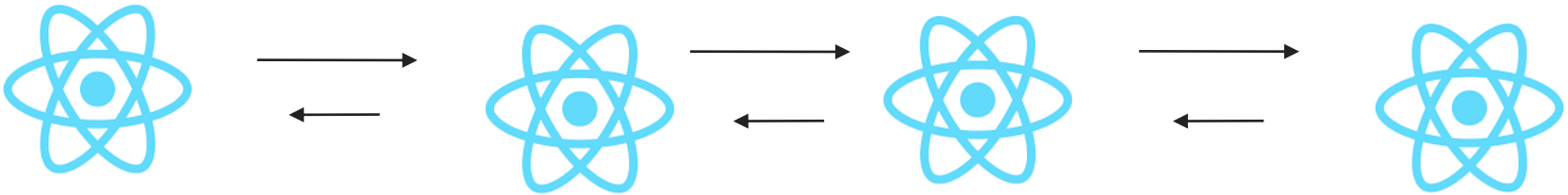
16.12.0 (November 14, 2019)

React DOM

- Fix passive effects (`useEffect`) not being fired in a multi-root app. (@acdlite in #17347)

Backward compatibility

React 16 maintains the exact same public API as in the previous versions (15.x and earlier) so it’s not like you have to re-learn the whole thing from scratch. As with most updates to the library, there are some breaking changes but these are manageable and shouldn’t cause any serious problems.



15 15.6 16.0 16.8 hooks

Supported Browsers

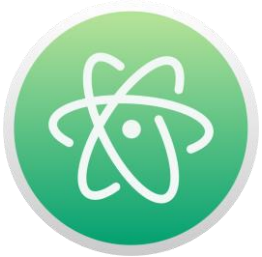
By default, the generated project supports all modern browsers. Support for Internet Explorer 9, 10, and 11 requires polyfills. For a set of polyfills to support older browsers, use react-app-polyfill.




React is IDE and code editor agnostic

Visual Studio Code

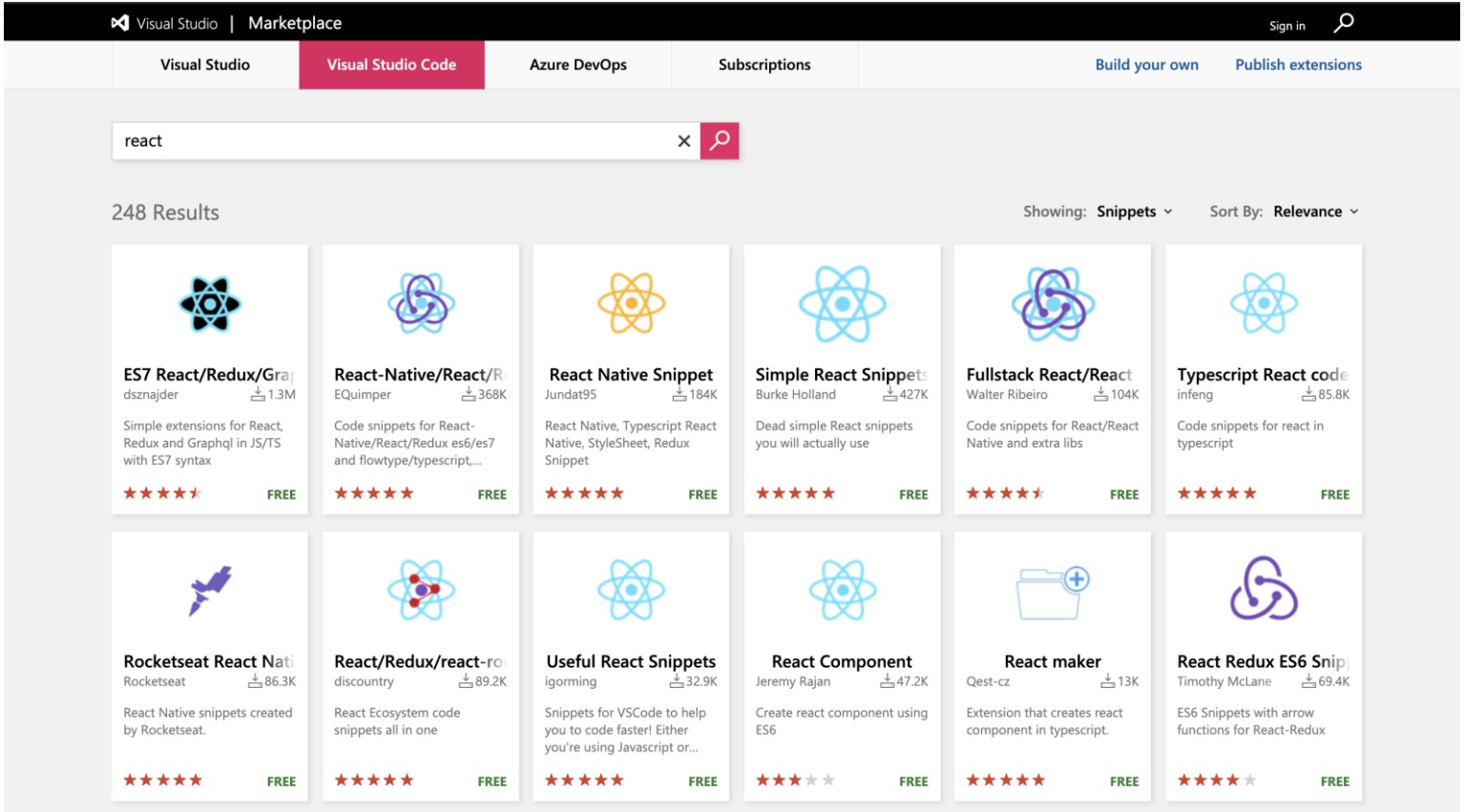
WebStorm



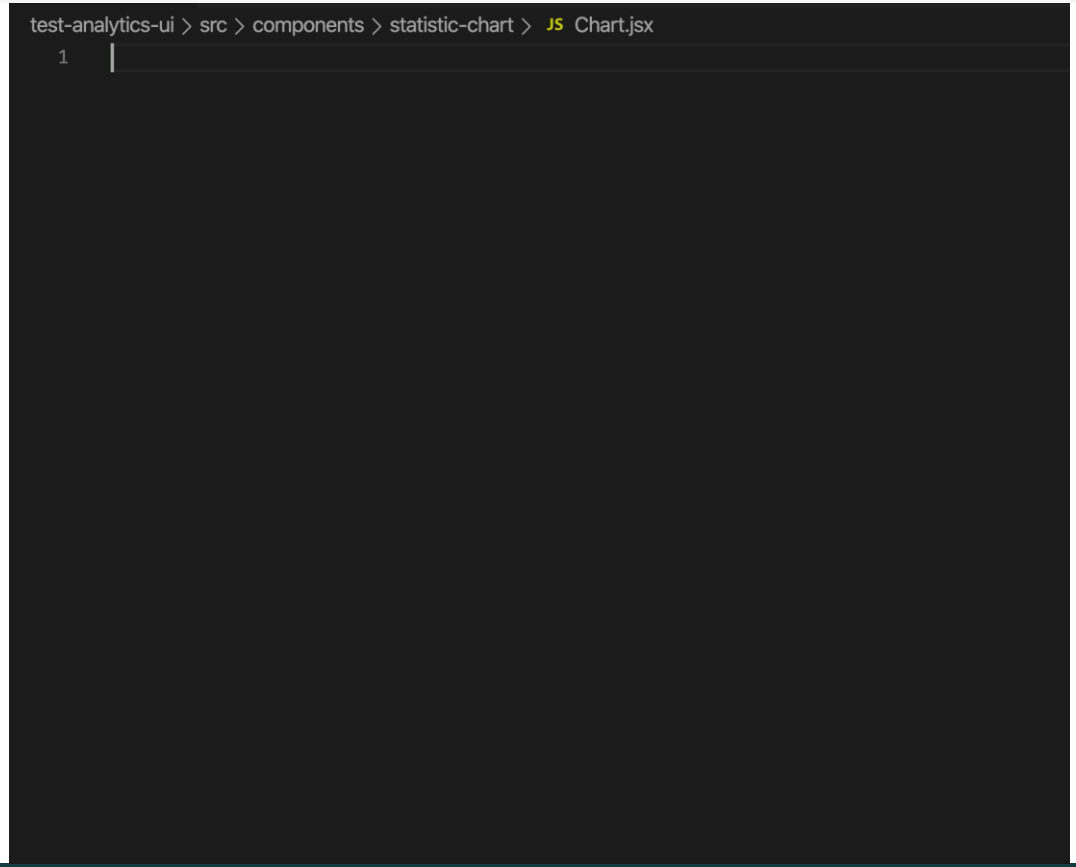
Atom

Sublime

VS Code extensions



VS Code extension: React/Redux/GraphQL/React-Native snippets



COMPONENTS-DRIVEN APPROACH OR REACT EVERYWHERE

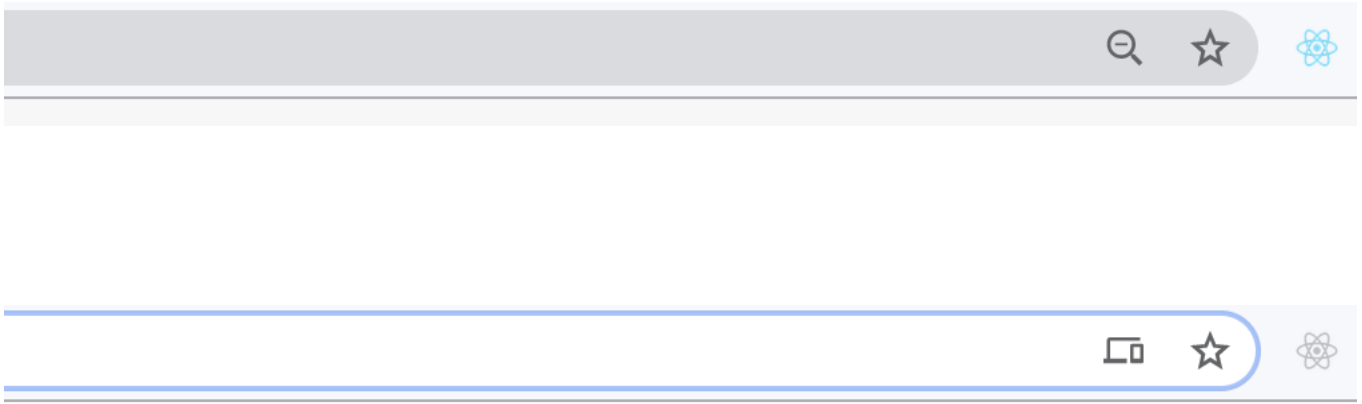
Who uses React today?

Useful Chrome extension for React

React Detector

Web app uses React library

Web app uses React library

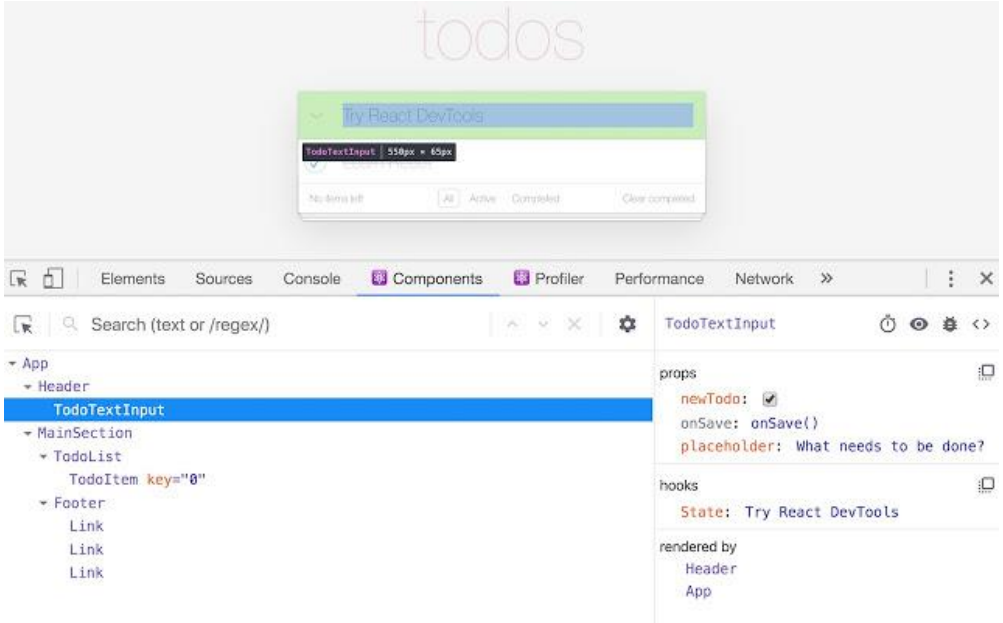


Web app doesn’t use React library

*might be triggered for any React widget on the page

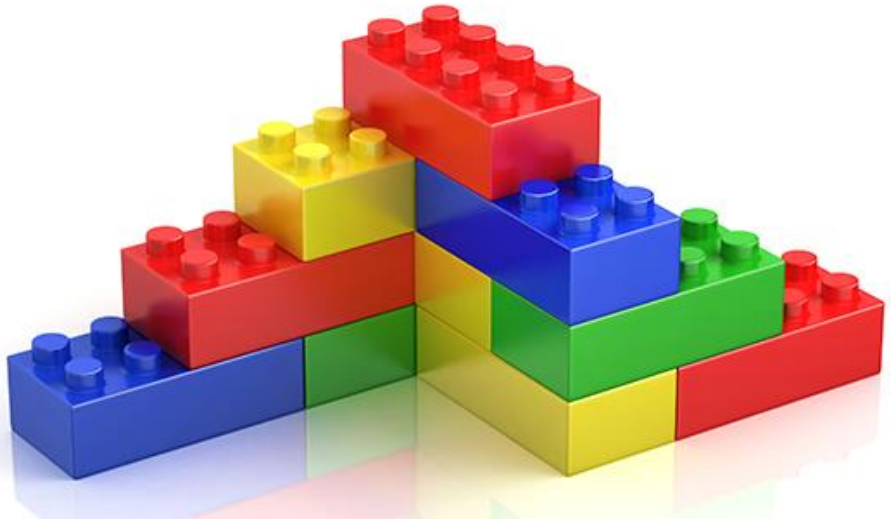
Useful Chrome extension for React

React Developer Tools (v4)
One of the most powerful tool for React development

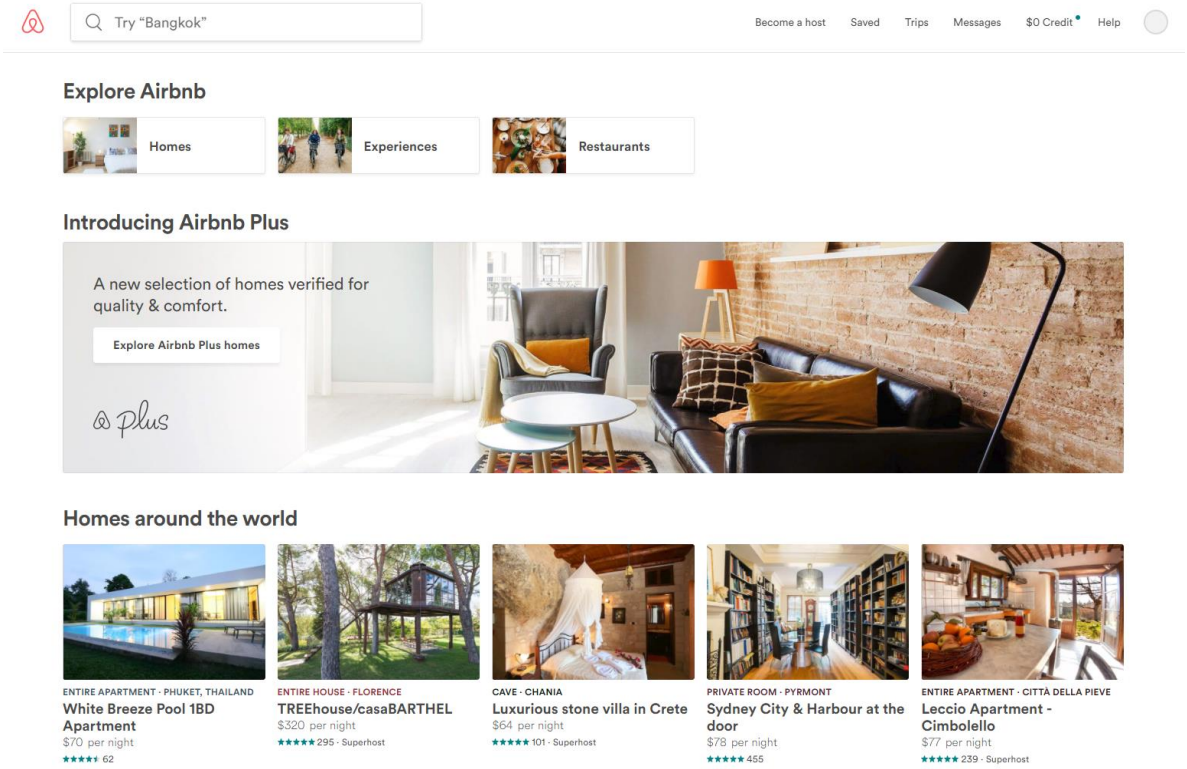


Components

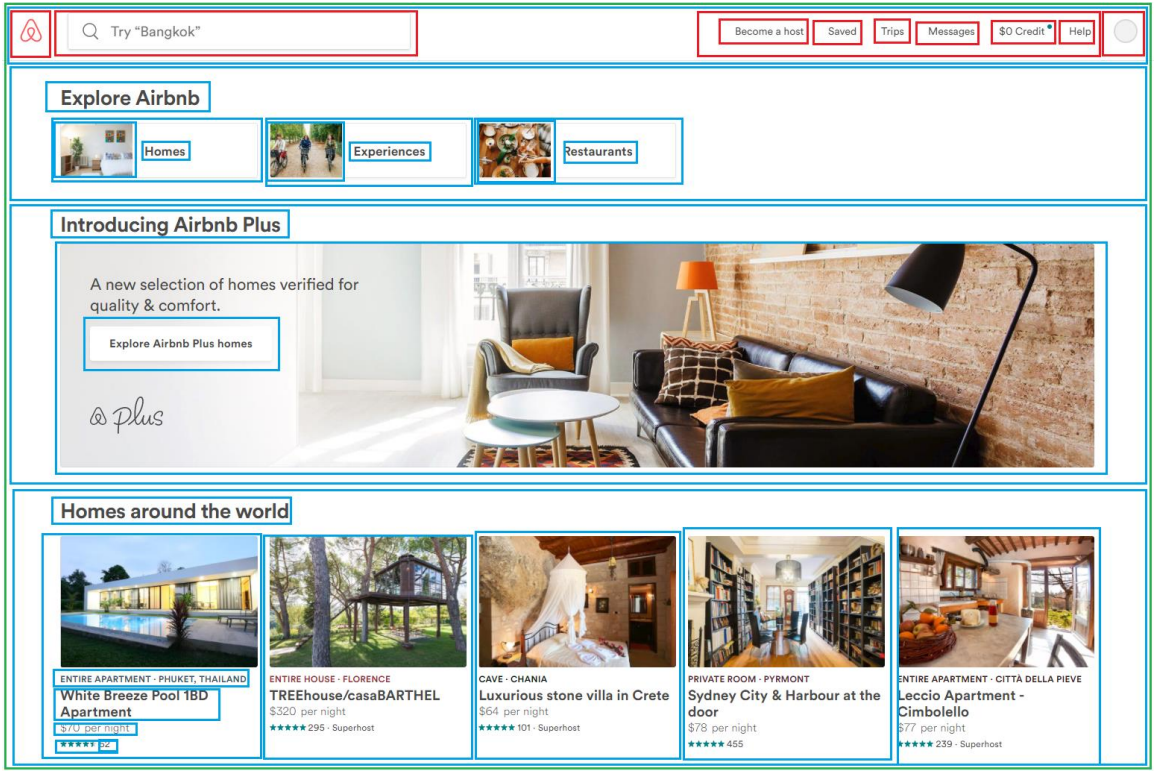
Components let you split the UI into independent, reusable pieces, and think about each piece in isolation.



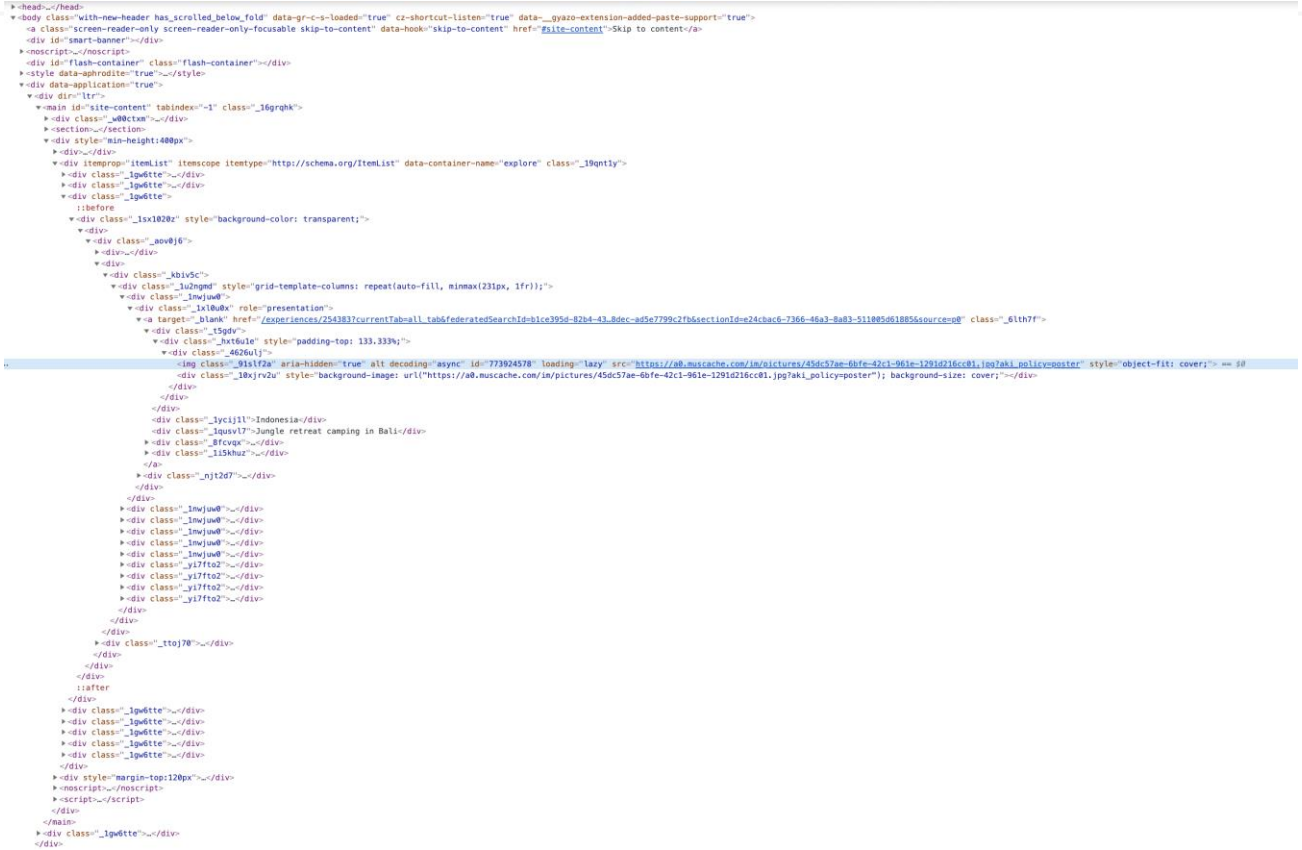
Components


Try “Bangkok”
Become a host
Saved
Trips
Messages
$0 Credit
Help
Explore Airbnb
Homes
Experiences
Restaurants
Introducing Airbnb Plus
A new selection of homes verified for quality & comfort.
Explore Airbnb Plus homes
Homes around the world
ENTIRE APARTMENT · PHUKET, THAILAND
White Breeze Pool 1BD Apartment
$70 per night
62
ENTIRE HOUSE · FLORENCE
TREEhouse/casaBARTHEL
$320 per night
295 · Superhost
CAVE · CHANIA
Luxurious stone villa in Crete
$64 per night
101 · Superhost
PRIVATE ROOM · PYRMONT
Sydney City & Harbour at the door
$78 per night
455
ENTIRE APARTMENT · CITTÀ DELLA PIEVE
Leccio Apartment - Cimbolello
$77 per night
239 · Superhost

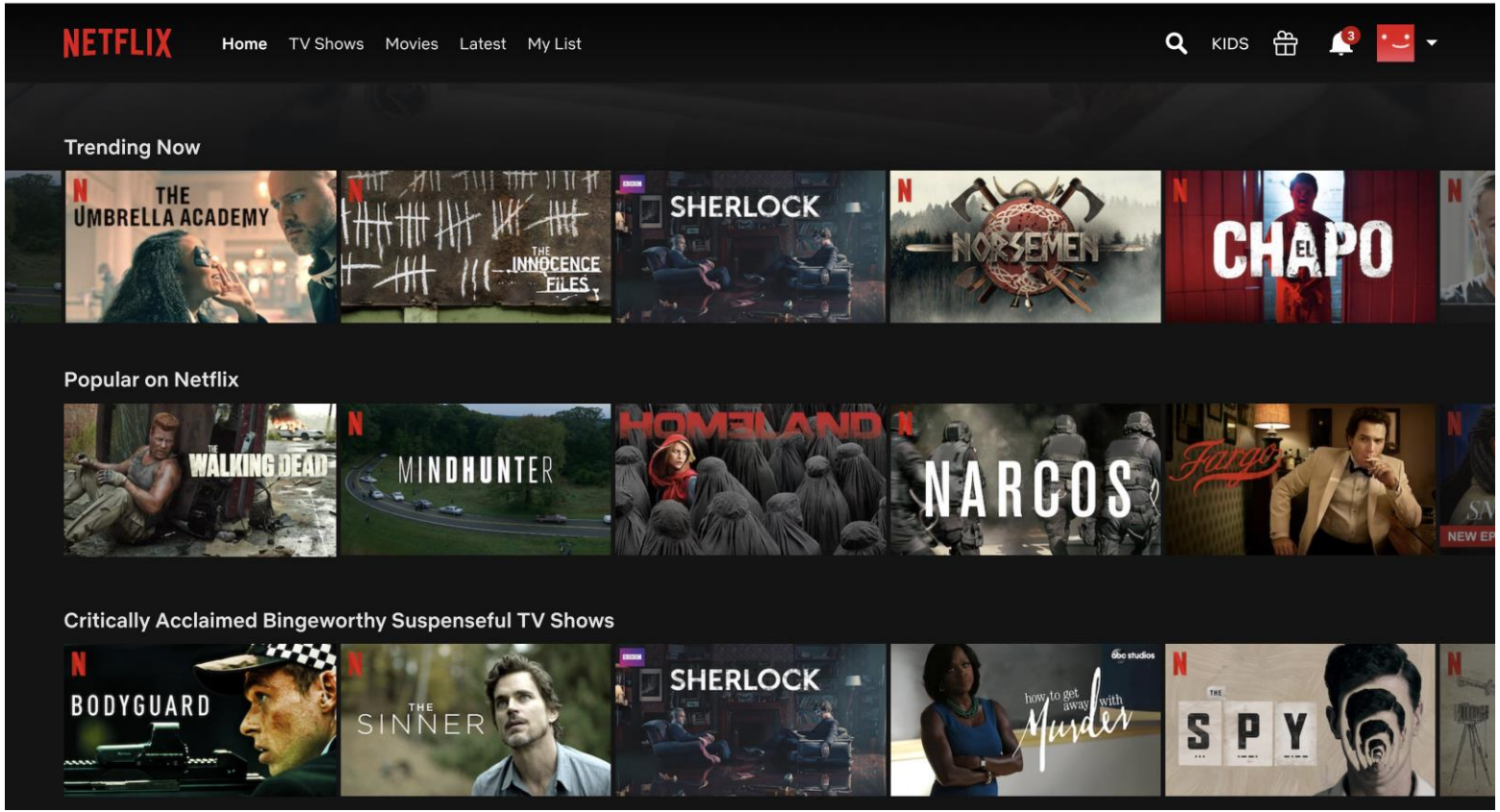
Components


Try "Bangkok"
Become a host
Saved
Trips
Messages
$0 Credit
Help
Explore Airbnb
Homes
Experiences
Restaurants
Introducing Airbnb Plus
A new selection of homes verified for quality & comfort.
Explore Airbnb Plus homes
plus
Homes around the world
ENTIRE APARTMENT · PHUKET, THAILAND
White Breeze Pool 1BD Apartment
$70 per night
ENTIRE HOUSE · FLORENCE
TREEhouse/casaBARTHEL
$320 per night
295 · Superhost
CAVE · CHANIA
Luxurious stone villa in Crete
$64 per night
101 · Superhost
PRIVATE ROOM · PYRMONT
Sydney City & Harbour at the door
$78 per night
455
ENTIRE APARTMENT · CITTÀ DELLA PIEVE
Leccio Apartment - Cimbolello
$77 per night
239 · Superhost

Components



Almost all web apps in … year



Almost all web apps in … year

Nav Bar

Main

List

List

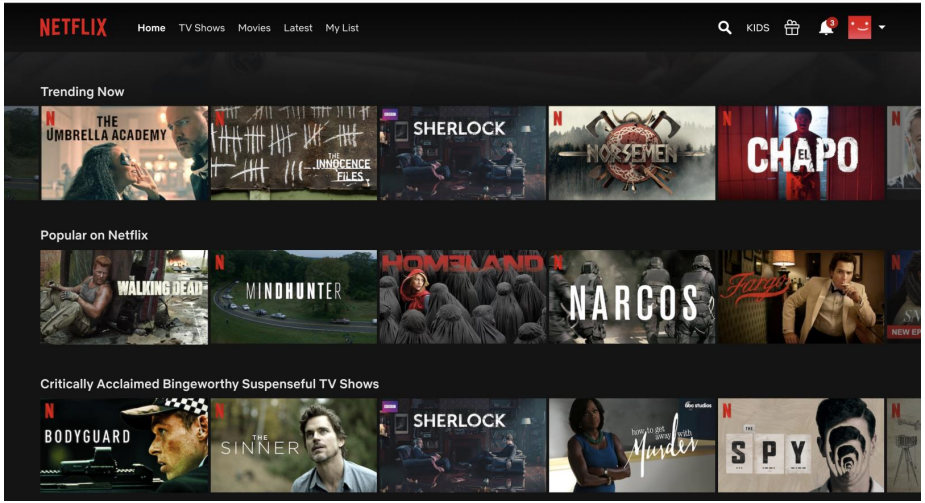
List

List Title

List Item

List Item

List Item

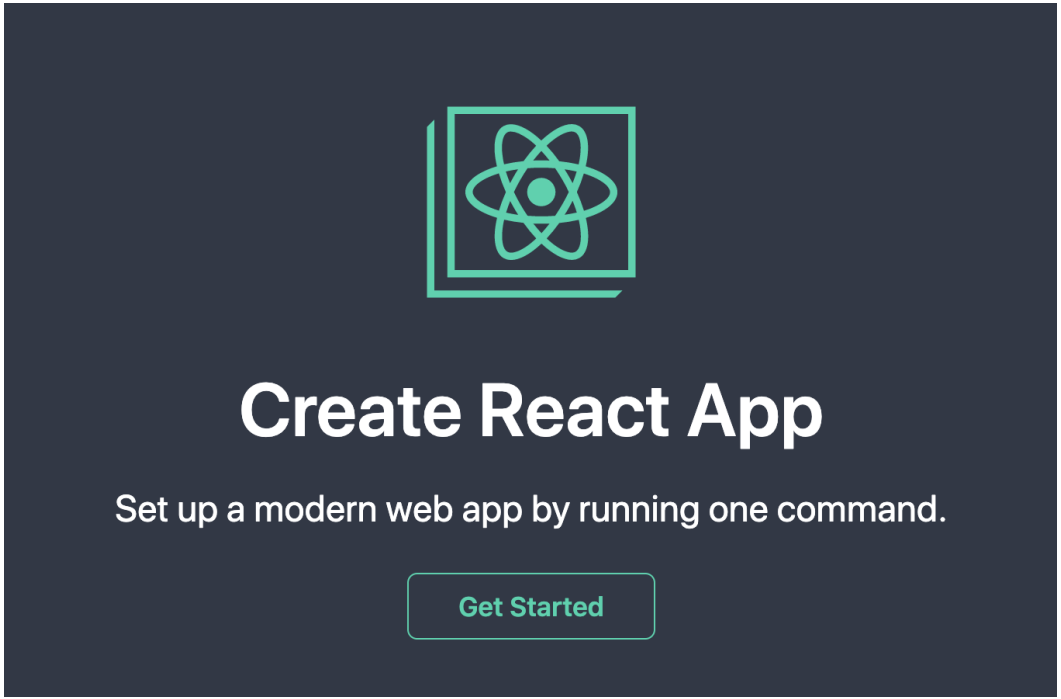


SET UP YOUR FIRST REACT APP

How to set up React app: HTML page

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>React Samples</title>
  </head>
  <body>
    <!-- Target container -->
    <div id="root"></div>

    <!-- React library & ReactDOM (Development Version)-->
    <script
  src="https://unpkg.com/react@16/umd/react.development.js">
  </script>
    <script
  src="https://unpkg.com/react-dom@16/umd/react-dom.development.js">
  </script>

    <script>
      // Pure React and JavaScript code
    </script>
  </body>
</html>
```

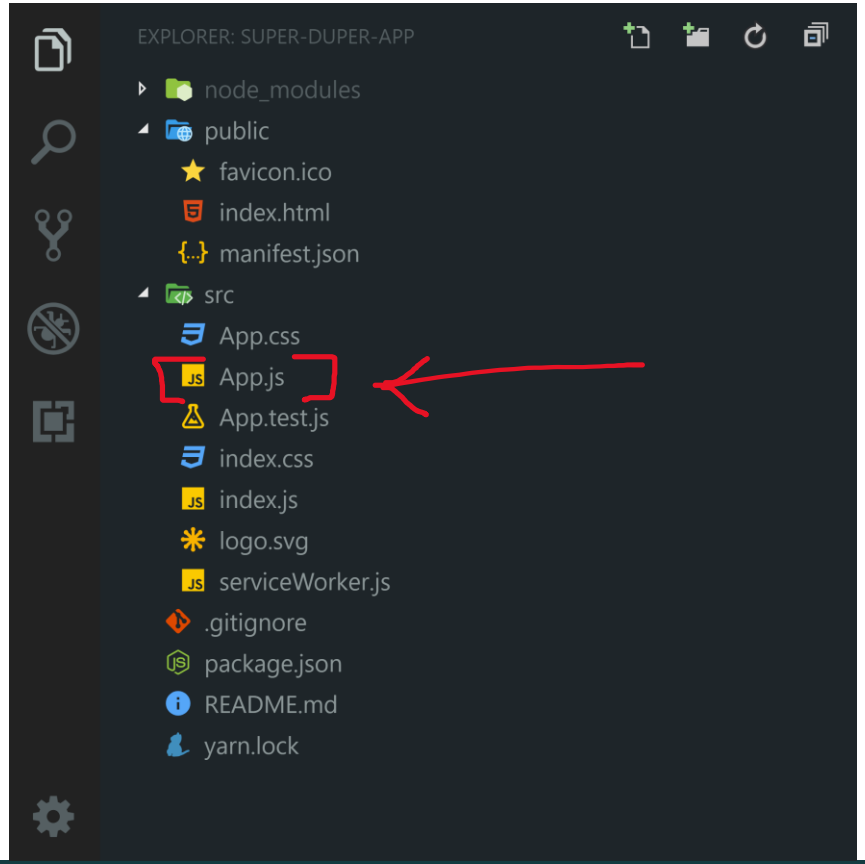
How to set up React app: Create React App




```
npx create-react-app my-app
cd my-app
npm start
```

https://create-react-app.dev/

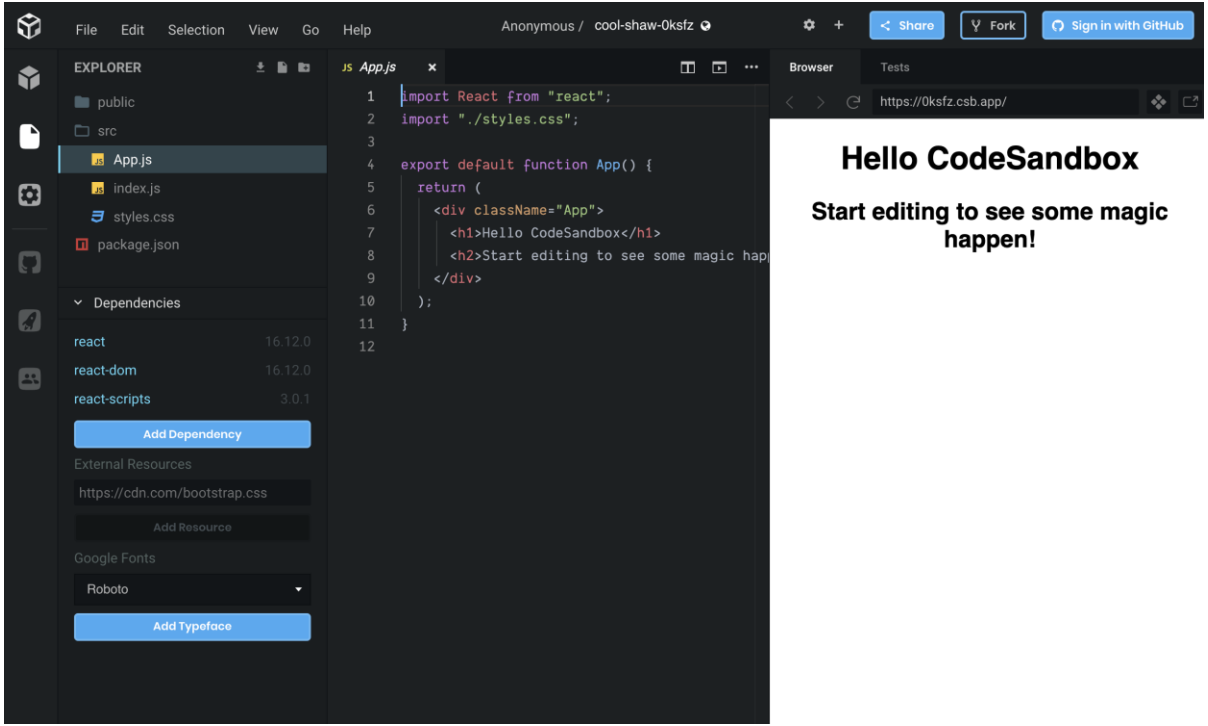
How to set up React app: Create React App



Benefits of Create React App

- React, JSX, ES6, Typescript and Flow syntax support.
- Autoprefixed CSS
- CSS Reset/Normalize
- A live development server
- A fast interactive unit test runner with built-in support for coverage reporting
- A build script to bundle JS, CSS, and images for production, with hashes and sourcemaps
- An offline-first service worker and a web app manifest, meeting all the Progressive Web App criteria.

How to set up React app: Online IDE



https://codesandbox.io/s/new

JSX

REACT ELEMENTS

Hello React: React Element

Hello React!

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

const myHeader = React.createElement(
  'h1',
  { className: 'header_title' },
  'Hello React!'
)

const rootElement = document.getElementById('root')

ReactDOM.render(myHeader, rootElement)
```

Hello React: React Element

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

const myHeader = React.createElement(
  'h1',
  { className: 'header_title' },
  'Hello React!'
)

const rootElement = document.getElementById('root')

ReactDOM.render(myHeader, rootElement)
```

Hello React!

```
React.createElement (type, config, children);

/*
* @param {string || function || class} type element type.
* @param {Object || null} config element properties.
* @param {string || Array} children element children.
*/
```

```
ReactDOM.render (nextElement, container, [callback]);

/*
* @param {ReactElement} nextElement element to render.
* @param {DOMElement} container DOM element to render into.
* @param {function} callback function triggered on completion.
*/
```

Hello React: React Element

Hello React!

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

const myHeader = React.createElement(
  'h1',
  { className: 'header_title' },
  'Hello React!'
)

const rootElement = document.getElementById('root')

ReactDOM.render(myHeader, rootElement)
```

```
<h1 class='header_title'>Hello React!</h1>
```

Hello React: React Element

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

const myHeader = React.createElement(
  'h1',
  { className: 'header_title' },
  'Hello React!'
)

const rootElement = document.getElementById('root')

ReactDOM.render(myHeader, rootElement)
```

Hello React!

```
React.createElement (type, config, children);

/*
* @param {string || function || class} type element type.
* @param {Object || null} config element properties.
* @param {string || Array} children element children.
*/
```

```
ReactDOM.render (nextElement, container, [callback]);

/*
* @param {ReactElement} nextElement element to render.
* @param {DOMElement} container DOM element to render into.
* @param {function} callback function triggered on completion.
*/
```

```
<h1 class='header_title'>Hello React!</h1>
```

Hello React: React Element

```
const myHeader = React.createElement (
  "h1",
  {
    id: "id_001",
    className: "header_title",
    key : "001"
  },
  "Hello React!"
);
```

Hello React!

```
console.log(myHeader);

{
  type: "h1",
  props: {
    id: "id_001",
    className: "header_title",
    children: "Hello React!"
  },
  key: "001",
  ref: ""
}
```

Hello React: React Element

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

const elem1 = React.createElement(
    "h1",
    {id: "id_001", className: "my_title"},
    "Hello World!“
);
const elem2 = React.createElement(
    "div",
    {id: "id_002", className: "my_desc"},
    "Some description...“
);

const root = React.createElement("div", null, elem1, elem2);

ReactDOM.render(root, document.getElementById("container"));
```

```
<!DOCTYPE html>
<html> == $0
  <head>…</head>
  <body>
    <div id="root">
      <div id="container">
        <div data-reactroot>
          <h1 id="id_001" class="my_title">Hello World!</h1>
          <div id="id_002" class="my_desc">Some description...</div>
        </div>
      </div>
    </div>
```

```
{
    type: "div",
    key: "001",
    props: {
        children:[
            {…},
            {
                type: "h1",
                key: "002",
                props: {
                    //..
                    children: […]
                    //..
            }
            }
        ]
    }
}
```

Hello React: HTML vs React.createElement

```
let elem = React.createElement(
  'div',
  null,
  React.createElement('h1', null, 'Title'),
  React.createElement(
    'div',
    { className: 'child1' },
    React.createElement('div', { className: 'child1_1' }, 'Sometext'),
    React.createElement('div', { className: 'child1_2' }, 'Sometext')
  ),
  React.createElement(
    'div',
    { className: 'child2' },
    React.createElement(
      'ul',
      null,
      React.createElement('li', null, '1'),
      React.createElement('li', null, '2'),
      React.createElement('li', null, '3')
    )
  )
)
```

Title

Some text
Some text

- 1
- 2
- 3

```
<div>
  <h1>Title</h1>
  <div class="child1">
    <div class="child1_1">Some text</div>
    <div class="child1_2">Some text</div>
  </div>
  <div class="child2">
    <ul>
      <li>1</li>
      <li>2</li>
      <li>3</li>
    </ul>
  </div>
</div>
```

Hello React: JSX to JS to HTML

```jsx
let elem = (
  <div>
    <h1>Title</h1>
    <div className="child1">
      <div className="child1_1">Some text</div>
      <div className="child1_2">Some text</div>
    </div>
    <div className="child2">
      <ul>
        <li>1</li>
        <li>2</li>
        <li>3</li>
      </ul>
    </div>
  </div>
)
```

JSX

```js
let elem = React.createElement(
  'div',
  null,
  React.createElement('h1', null, 'Title'),
  React.createElement(
    'div',
    { className: 'child1' },
    React.createElement('div', { className: 'child1_1' }, 'Sometext'),
    React.createElement('div', { className: 'child1_2' }, 'Sometext')
  ),
  React.createElement(
    'div',
    { className: 'child2' },
    React.createElement(
      'ul',
      null,
      React.createElement('li', null, '1'),
      React.createElement('li', null, '2'),
      React.createElement('li', null, '3')
    )
  )
)
```

JS

BABEL

```html
<div>
  <h1>Title</h1>
  <div class="child1">
    <div class="child1_1">Some text</div>
    <div class="child1_2">Some text</div>
  </div>
  <div class="child2">
    <ul>
      <li>1</li>
      <li>2</li>
      <li>3</li>
    </ul>
  </div>
</div>
```

Title

Some text
Some text

- 1
- 2
- 3

JSX

JSX is smart


```
repl: Expected corresponding JSX closing tag for <div> (1:5)
> 1 | <div></span>
    |      ^
```

JSX is an Expression

```
const getGreeting = (admin) => {
  if (admin) {
    return <h1>Hello Admin</h1>
  }
  return <h1> Hello User</h1>
}
```

Javascript Expression in JSX

```
const planet = 'Earth'

<div>Hello {planet}!</div>
```

```
const planet = 'Earth'
React.createElement('div', null, 'Hello', planet, '!')
```

JSX (common mistake)

```jsx
//JSX


<div>
    {
    comments.forEach((comment)=>{
        return
        <div>
            {comment}
        </div>
    })
}
</div>
```

```jsx
//JS


React.createElement(
    "div",
    null,
    comments.forEach(
        function(comment){
            return;
            React.createElement(
                "div",
                null,
                comment
            );
        }
    )
);
```

JSX (fix for a common mistake)

```
<div>
    {
    comments.forEach((comment)=>{
        return (
            <div>
                {comment}
            </div>
        );
    })
}
</div>
```

```
//JS

React.createElement(
    "div",
    null,
    comments.forEach(
        function(comment){
            return React.createElement(
                "div",
                null,
                comment
            );
        })
);
```

Javascript Statement in JSX

```
<div>
        { if (true) return "Hello"}
    </div>
```

```
repl: Unexpected token (2:4)
  1 |           <div>
> 2 |                   {if (true) return "Hello"}
    |                    ^
  3 |           </div>
  4 |
```

Javascript Conditional Operator in JSX

```jsx
<div>{true ? 'Hello Admin' : 'Hello user'}</div>
```

```jsx
React.createElement('div', null, true ? 'Hello Admin' : 'Hello user')
```

String literals and default value

```jsx
//JSX

<div className = {"myClass"}></div>

<div className = "myClass" ></div>

<div className></div>
```

```jsx
//JS

React.createElement('div', { className: 'myClass' })

React.createElement('div', { className: 'myClass' })

React.createElement('div', { className: true })
```

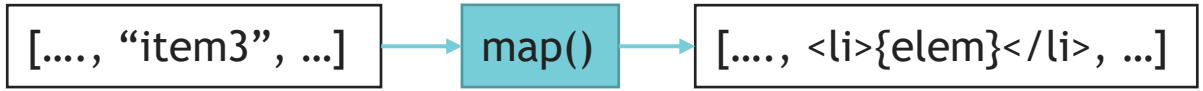
Booleans, Null, and Undefined Are Ignored

```
<div/>

<div></div>

<div>{false}</div>

<div>{null}</div>

<div>{true}</div>


<div>
    {showHeader && <div>Header</div> }
    <div>Content</div>
</div>
```

Rendering a list of JSX expressions

```
let data = ['item1', 'item2', 'item3', 'item4', 'item5']

let list = (
  <ul>
    {data.map((elem) => (
      <li>{elem}</li>
    ))}
  </ul>
)

ReactDOM.render(list, document.getElementById('container'))
```

```
<!DOCTYPE html> == $0
<html>
▶<head>…</head>
▼<body>
  ▼<div id="root">
    ▼<div id="container">
      ▼<ul data-reactroot>
          <li>item1</li>
          <li>item2</li>
          <li>item3</li>
          <li>item4</li>
          <li>item5</li>
        </ul>
      </div>
    </div>
```



Console

```
▶Warning: Each child in an array or iterator should have a unique "key"    react.js:20478
prop. Check the top-level render call using <ul>. See https://fb.me/react-warning-keys for
more information.
    in li
```

JSX & events

```jsx
<button onClick={likeItem}>LIKE</button>
```

```jsx
<button onclick = “likeItem( )”>
    LIKE
</button>
```

JSX & events

- Clipboard Events
- Composition Events
- Keyboard Events
- Focus Events
- Form Events
- Generic Events
- Mouse Events
- Pointer Events
- Selection Events
- Touch Events
- UI Events
- Wheel Events
- Media Events
- Image Events
- Animation Events
- Transition Events
- Other Events

All Supported HTML Attributes

htmlFor

Since for is a reserved word in JavaScript, React elements use htmlFor instead.

className

To specify a CSS class, use the className attribute.

COMPONENTS

Components in React

What is **a React Component?**

React components are reusable chunks of JavaScript that output (via JSX) virtual HTML elements

Components via JSX

Function(al) component

```
import React from 'react'

let Header = () => <h1>Hello React</h1>

export default Header
```

```
import React from 'react'

function Header() {
  return <h1>Hello React!</h1>
}

export default Header
```

Class component (via Component)

```
import React from 'react'

class Header extends React.Component {
  render() {
    return <h1>Hello React</h1>
  }
}

export default Header
```

Class component (via PureComponent)

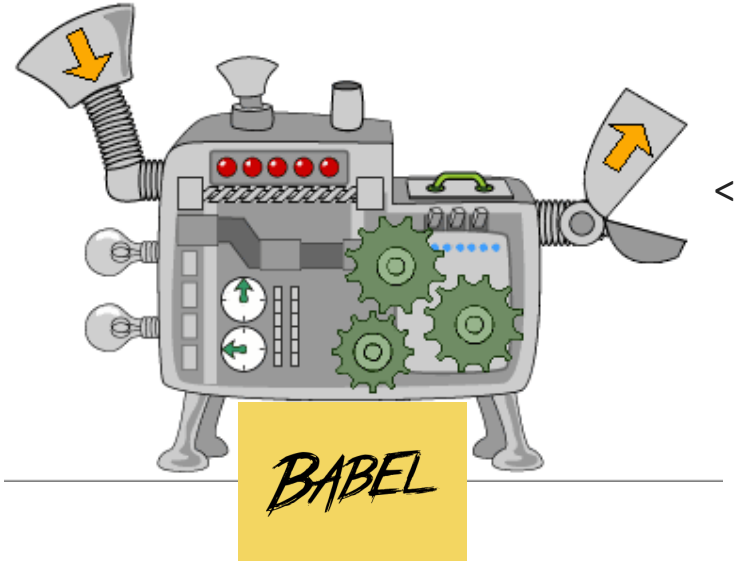
```
import React from 'react'

class Header extends React.PureComponent {
  render() {
    return <h1>Hello React</h1>
  }
}

export default Header
```

Components

Components are functions that accept arbitrary inputs (props) and return React elements describing what should appear on the screen

JSX

<Header/>



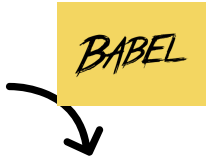
JS

<**h1**>Hello React</**h1**>;

JSX to JS via Babel

```jsx
let App = () => {
  return (
    <div className="App">
      <h1>React Mentoring Program</h1>
      <h2>Start editing to see some magic happen!</h2>
    </div>
  )
}
```

JSX



JS

```js
var App = function App() {
  return React.createElement(
    'div',
    {
      className: 'App',
    },
    React.createElement('h1', null, 'React Mentoring Program'),
    React.createElement('h2', null, 'Start editing to see some magic happen!')
  )
}
```

Components Composition. Props. Stateless

```
function MovieCard(props) {
  return (
    <div className="component">
      <h3>{props.title}</h3>
    </div>
  )
}
```

```
function App() {
  return (
    <div className="App component">
      <HeaderSection />
      <MovieCard title="Stranger Things" />
      <MovieCard title="True Detective" />
      <MovieCard title="Breaking Bad" />
      <Footer />
    </div>
  )
}
```

Components Composition. State & Props. Stateful

```
class HeaderSection extends Component {
  state = {
    isPremiumUser: false,
    title: 'Hello React',
  }

  render() {
    return (
      <div className="component">

        <Header title={this.state.title} />

        {this.state.isPremiumUser ? (
          <h2>Let's watch 4k some movies!</h2>
        ) : (
          <h2>Let's watch some movies!</h2>
        )}
      </div>
    )
  }
}

export default HeaderSection
```

```
let Header = ({ title }) => <h1 className="component">{title}</h1>

export default Header
```

Components. Hot questions – Detailed answers in the next training

When to use a Class Component over a Function Component?

If the component needs *lifecycle methods* then use class component otherwise use function component.

OR at this moment try use function components more often.

What is PureComponent?

React.PureComponent is exactly the same as React.Component except that it handles the shouldComponentUpdate() method.

OR at this moment - PureComponent is almost exactly the same as React.Component.

The detailed analysis of difference between state and props – next training as well.

VIRTUAL DOM

Virtual DOM

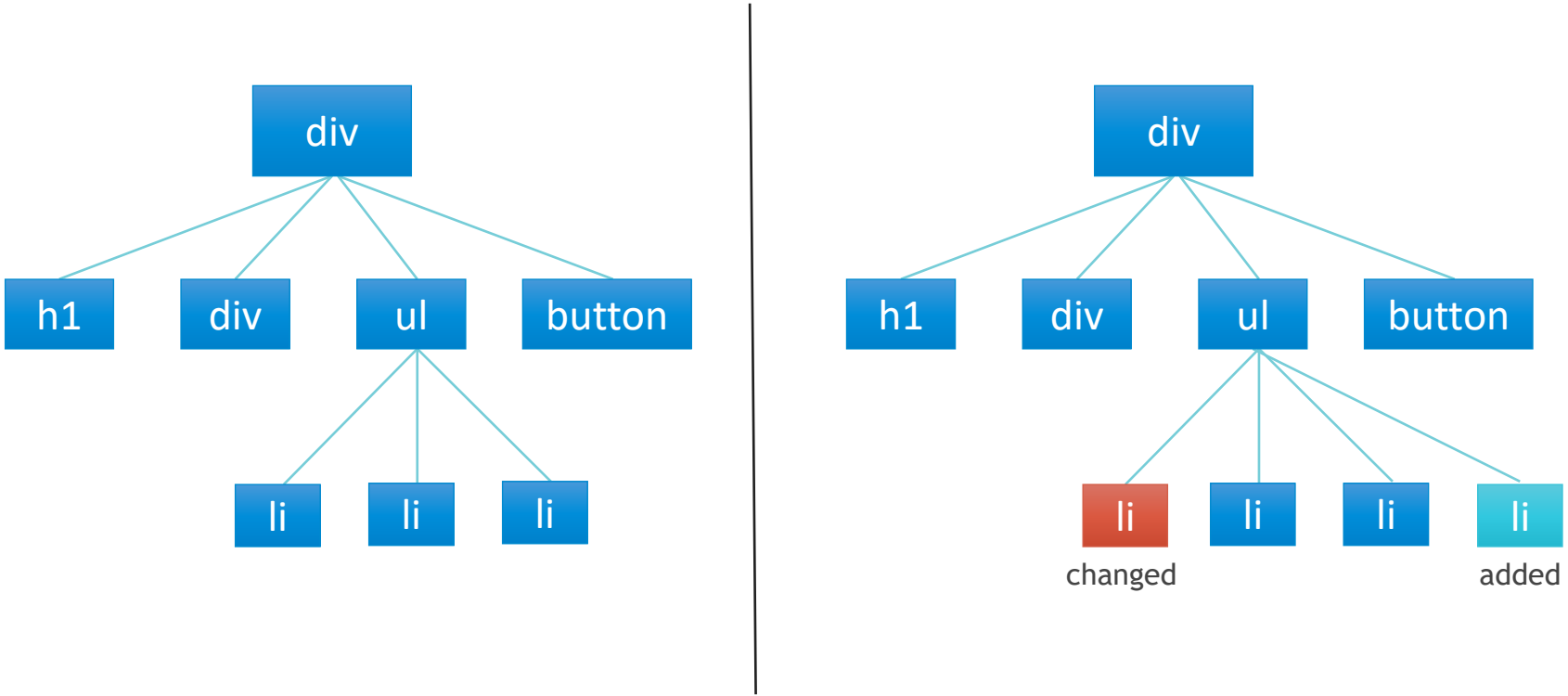
The Virtual DOM is an abstraction of the HTML DOM.

Virtual DOM is a JavaScript representation of the DOM.

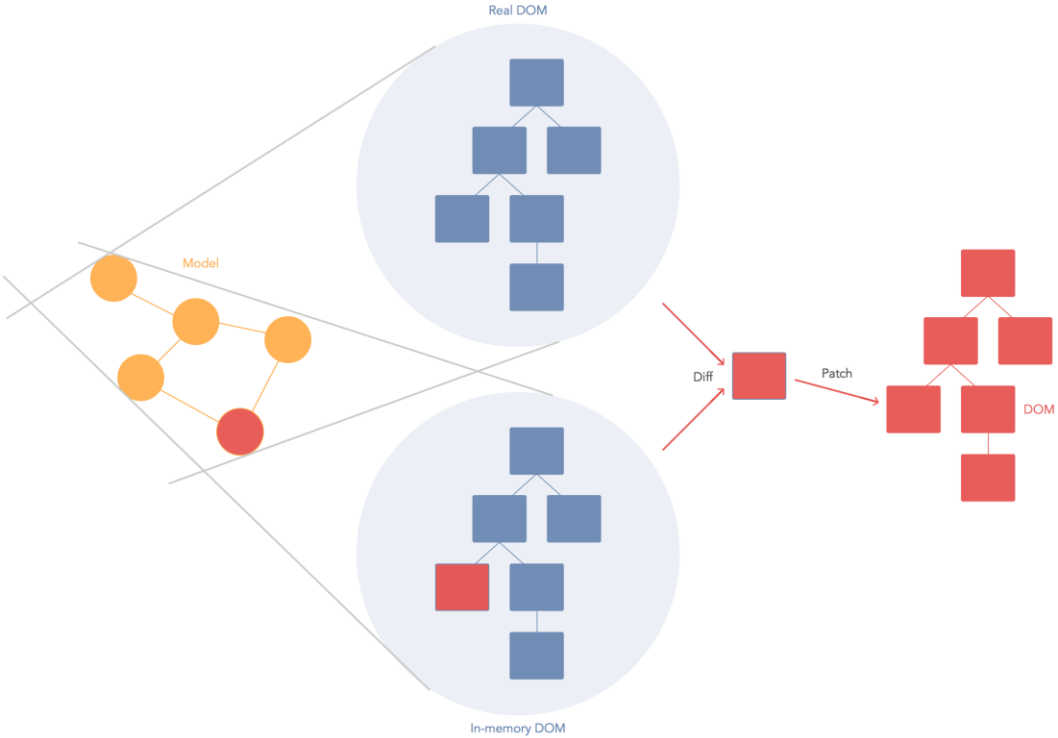
It is lightweight and detached from the browser-specific implementation details.

Since the DOM itself was already an abstraction, the virtual DOM is, in fact, an abstraction of an abstraction.

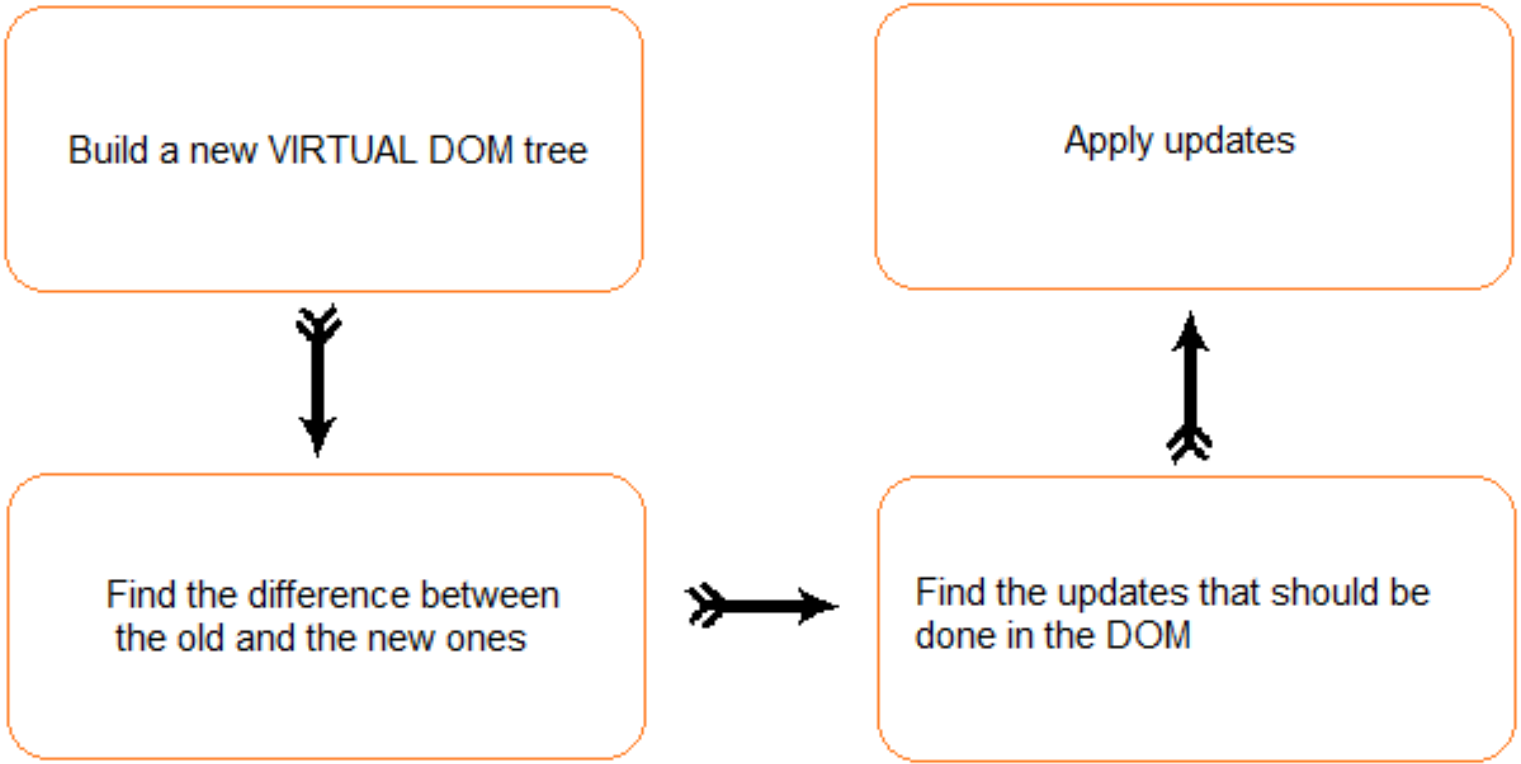
Virtual DOM


div
h1
div
ul
button
li
li
li
div
h1
div
ul
button
li
li
li
li
changed
added

Virtual DOM



Virtual DOM


Build a new VIRTUAL DOM tree
Apply updates
Find the difference between the old and the new ones
Find the updates that should be done in the DOM

Virtual DOM

Reconciliation

The process of updating your UI to match your application state

Accurate Diff algorithm

Accurate Diff algorithm generates the **minimum number of operations** to transform one tree into another.

Accurate Diff algorithm

Accurate Diff algorithm has a complexity of

O(n^3)

Accurate Diff algorithm

Accurate Diff algorithm has a complexity of

O(n^3)

10000 nodes in a tree

10000^3 = 1000 * 10^9

≈ 1000 sec! at 1GHz

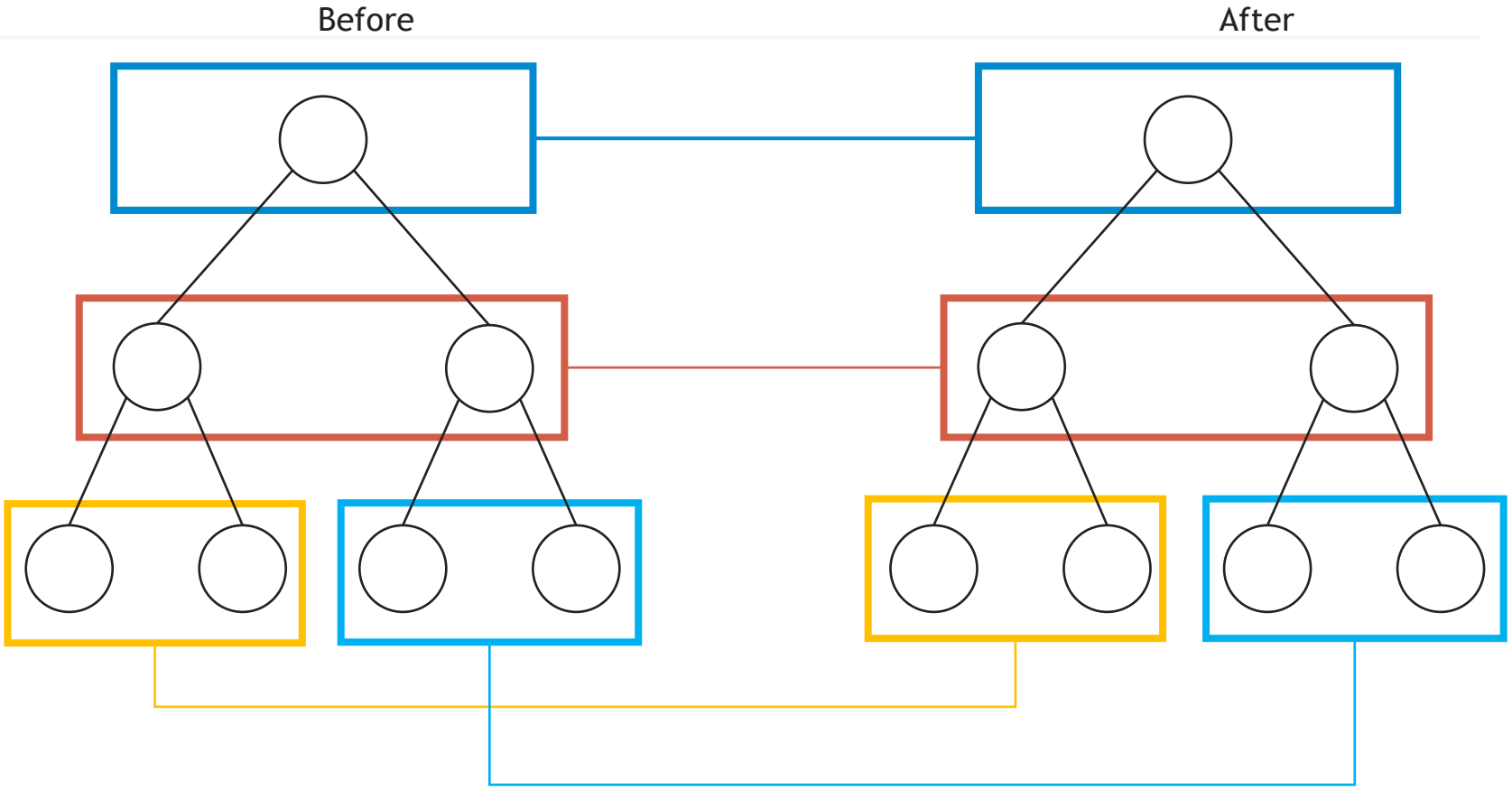
≈ **17 minutes!!!**

React Diff algorithm

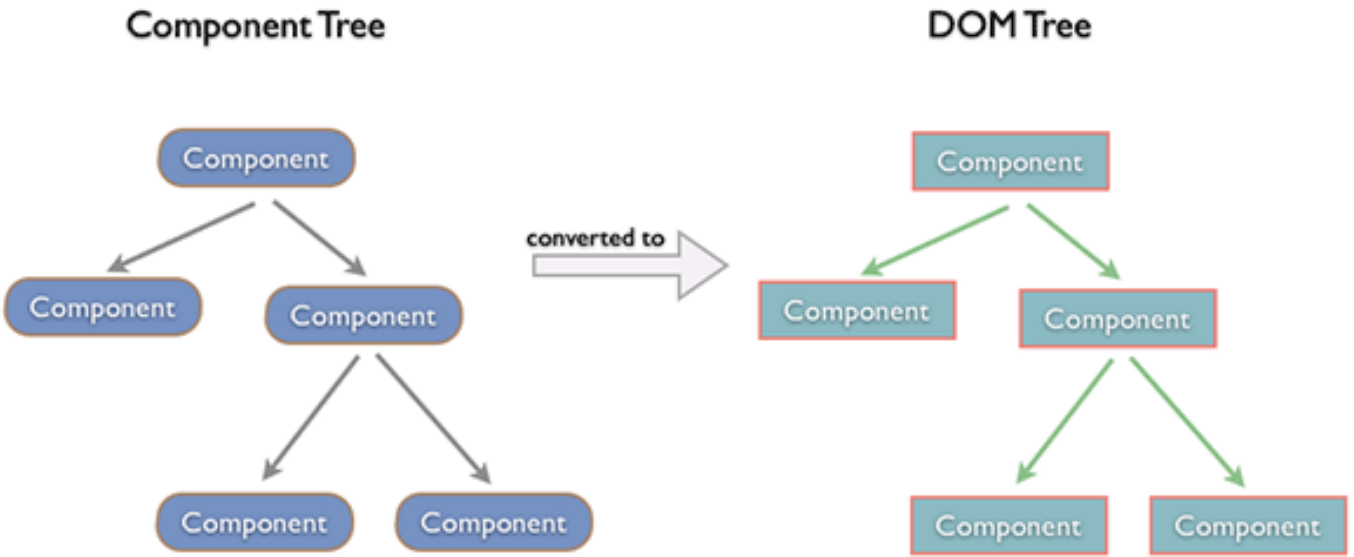
Instead of optimal algorithm, React implements a **heuristic algorithm** which has a complexity of

O(n)

React Diff Algorithm



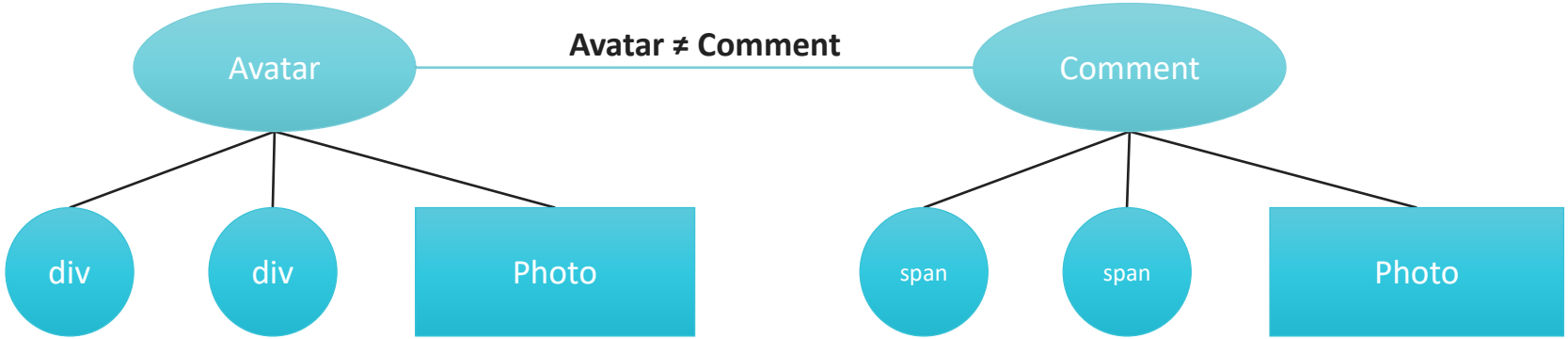
Virtual DOM


Component Tree
Component
Component
Component
Component
Component
converted to
DOM Tree
Component
Component
Component
Component
Component

React Diff Algorithm

```
<Avatar>
  <div>Some complex DIV</div>
  <div>One more complex DIV</div>
  <Photo/>
</Avatar>
```

```
<Comment>
  <span>Some span</div>
  <span>One more span</div>
  <Photo/>
</Comment>
```



Virtual DOM: React Diff Algorithm

Assumption #1:

Two elements of different types will produce different trees.

Reconciliation

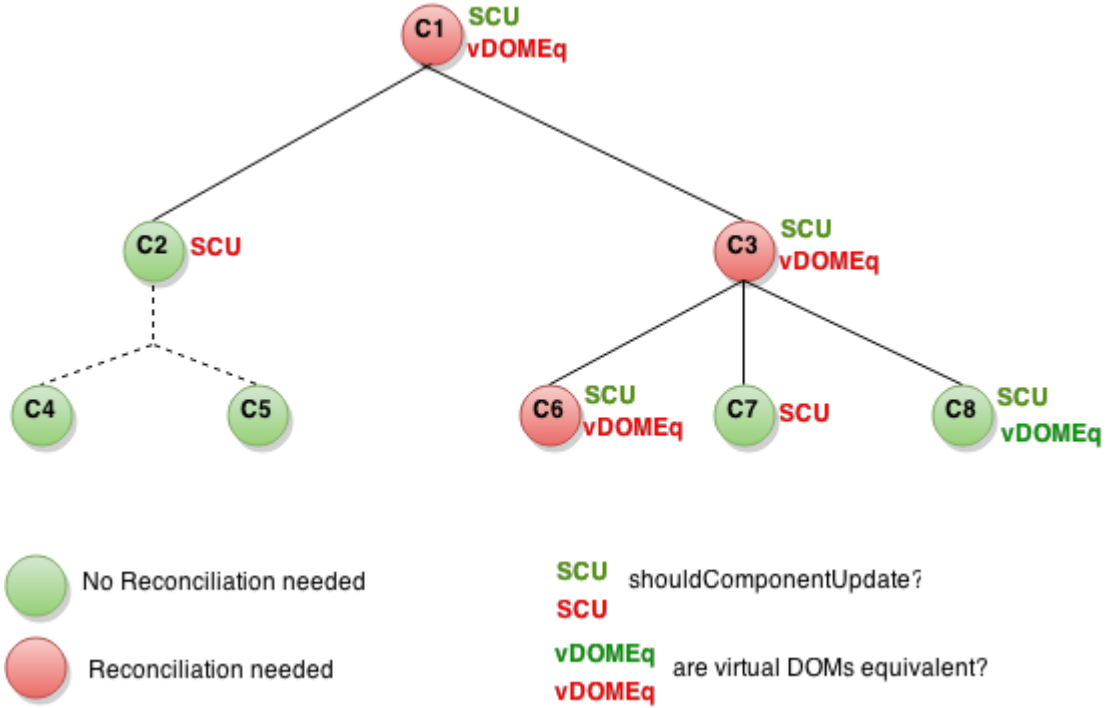
DEMO Reconciliation in Action 1

stackblitz.com/edit/react-z5obkc?file=TasksList.js

Virtual DOM

But how does React iterates through a list of children?

Virtual DOM: React Diff Algorithm



Virtual DOM: React Diff Algorithm

Assumption #2:

The developer can hint at which child element may be stable across different renders with a **key** prop.

Virtual DOM: React Diff Algorithm

- React iterates through the new set of children.
- For each child, React checks whether there is an old child that has the same key as the new child. If an explicit key is not provided, React uses its position.

Virtual DOM: React Diff Algorithm

If there is **NO new child with the same key as an old child**, the old child is **unmounted**.

Virtual DOM: React Diff Algorithm

If there is **NO old child with the same key as a new child**, the new child is **mounted**.

Virtual DOM: React Diff Algorithm

If there is an **old and new child with the same key**, React decides whether it is necessary to **update** the instance vs doing a clean **unmount/mount**.

Virtual DOM: React Diff Algorithm

No key provided – React uses position of an item.

```
<ul>
  <li>first data</li>          //1
  <li>second data</li>         //2
</ul>
```

```
<ul>
  <li>first data</li>          //1
  <li>second data</li>         //2
  <li>third data</li>          //3
</ul>
```

Virtual DOM: React Diff Algorithm

No key provided – React uses position of an item.

```
<ul>
  <li>first data</li>          //1
  <li>second data</li>         //2
</ul>
```

```
<ul>
  <li>zero data</li>           //1
  <li>first data</li>          //2
  <li>second data</li>         //3
</ul>
```

Inserting an element at the beginning affects the performance.

React will **mutate** items instead of reusing it.

Virtual DOM: React Diff Algorithm

When children have keys, React uses the key to match children in the original tree with children in the subsequent tree.

```
<ul>
  <li key="a1">first data</li>
  <li key="a2">second data</li>
</ul>
```

```
<ul>
  <li key="a0">zero data</li>
  <li key="a1">first data</li>
  <li key="a2">second data</li>
</ul>
```

Inserting and an element at the beginning **doesn't** affect the performance.

React will **reuse** items instead of mutating it.

Reconciliation

DEMO Reconciliation in Action 2

stackblitz.com/edit/react-z5obkc?file=TasksList.js

THANKS