ReactJS.

Fundamentals

Agenda

- **Background.**
- **JSX (JavaScript Extension Syntax).**
- **Developer tools.**
  - **Storybook.**
- **Component basics.**

ReactJS.

- **A Javascript framework for building dynamic Web** User Interfaces**.**
  - **Single Page Apps technology.**
  - **Open-sourced in 2012**

- **Client-side framework.**
  - **More a library than a framework.**

Before ReactJS.

- MVC pattern **– The convention for** app design. **Promoted by market leaders**, e.g. **AngularJS (1.x), EmberJS, BackboneJS.**
- **React is not MVC, just V.**
  - **It challenged established best practice (MVC).**
- Templating **– widespread use in the V layer;**
  - **React based on** components.

	Templates	**React components**
Separation of concerns	Technology (JS, HTML)	Responsibility
Semantic	New concepts and micro-languages	HTML and Javascript
Expressiveness	Underpowered	Full power of Javascript

ReactJS

- **Philosophy:** *Build components, not templates.*
- **All about the** User Interface **(UI).**
  - **Not about business logic or the data model (Mvc)**

- **Component - A unit comprised:**

  UI description (HTML) + UI behavior (JS)
  - **Two aspects are tightly coupled and co-located.**
    - **Other frameworks decoupled them.**
  - **Benefits:**
    1. **Improved Composition.**
    2. **Greater Reusability.**
    3. **Better Performance..**

Creating the UI description

- React.createElement() **– create a HTML element.**
- ReactDOM.render() – **attach an element to the DOM.**
- createElement() **arguments**:
  1. **type (h1, div, span etc).**
  2. **properties (style, event handler etc).**
  3. **children.**
  - **We never use** createElement**() directly – far too cumbersome.**
- ReacrDOM.render() **arguments:**
  1. **element to be displayed;.**
  2. **DOM node on which to** mount **the element**.

- **Ref. 01-**UIDescription.html
  - **Ref**. 02-UIDescription.html - using nested method invocation

JSX.

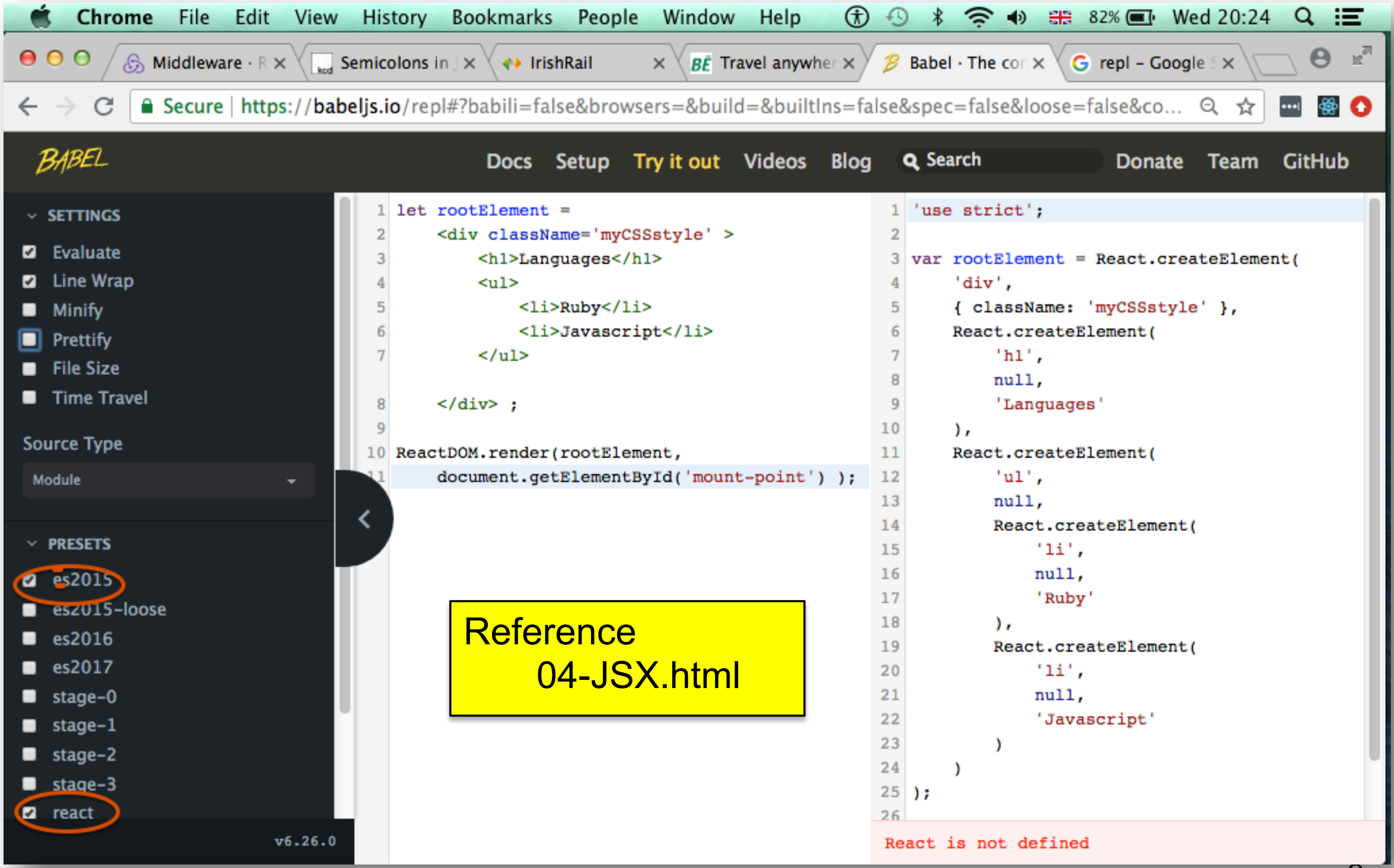
- **JSX – JavaScript extension syntax.**

- **Declarative syntax for coding UI descriptions.**

- **Retains the full power of Javascript.**
- **Allows tight coupling between UI behavior and description.**

- **Must be transpiled (Babel) for browser execution compatibility.**
  - **Reference** 03-JSX-error.html
  - **Reference** 04-JSX.html

REPL (Read-Evaluate-Print-Loop) transpiler.


Chrome File Edit View History Bookmarks People Window Help
Wed 20:24
https://babeljs.io/repl#?babili=false&browsers=&build=&builtIns=false&spec=false&loose=false&co...
BABEL
Docs Setup Try it out Videos Blog Search Donate Team GitHub
v6.26.0

SETTINGS

- [✓] Evaluate
- [✓] Line Wrap
- [ ] Minify
- [ ] Prettify
- [ ] File Size
- [ ] Time Travel

Source Type: Module

PRESETS

- [✓] es2015
- [ ] es2015-loose
- [ ] es2016
- [ ] es2017
- [ ] stage-0
- [ ] stage-1
- [ ] stage-2
- [ ] stage-3
- [✓] react

```
let rootElement =
    <div className='myCSSstyle' >
        <h1>Languages</h1>
        <ul>
            <li>Ruby</li>
            <li>Javascript</li>
        </ul>

    </div> ;

ReactDOM.render(rootElement,
    document.getElementById('mount-point') );
```

```
'use strict';

var rootElement = React.createElement(
    'div',
    { className: 'myCSSstyle' },
    React.createElement(
        'h1',
        null,
        'Languages'
    ),
    React.createElement(
        'ul',
        null,
        React.createElement(
            'li',
            null,
            'Ruby'
        ),
        React.createElement(
            'li',
            null,
            'Javascript'
        )
    )
);
```

```
React is not defined
```

Reference
04-JSX.html

JSX.

- **HTML-like markup.**
  - **It’s actually XML code.**

- **Must be transformed (transpiled) into ES5.**
  - **The Babel tool suite.**

- **Some minor HTML tag attributes differences, e.g. className (class), htmlFor (for).**

- **Allows** declarative description **of the UI inlined in JavaScript.**

- **Combines the ease-of-use of templates with the power of JS.**

Transpiling JSX.

- **What?**
  - **The Babel platform**

- **How?**
  1. **Manually, via REPL or command line.**
     - **When experimenting only.**
  2. **By the web server (using special tooling, i.e.Webpack).**
     - **Suitable for app development mode.**
  3. **As part of the build process for an app.**
     - **When deploying app for production.**

React Components.

- **We develop COMPONENTS.**
  - **A JS** class **that extend** React.Component
  - The render() **method:**
    - **Mandatory.**
    - **Returns the component** UI description**.**

- **Can reference components as <u>HTML tags.</u>**
  - **E.g.** ReactDOM.render(<ComponentName />, ...)

- **Reference** 05-simpleComponent.html

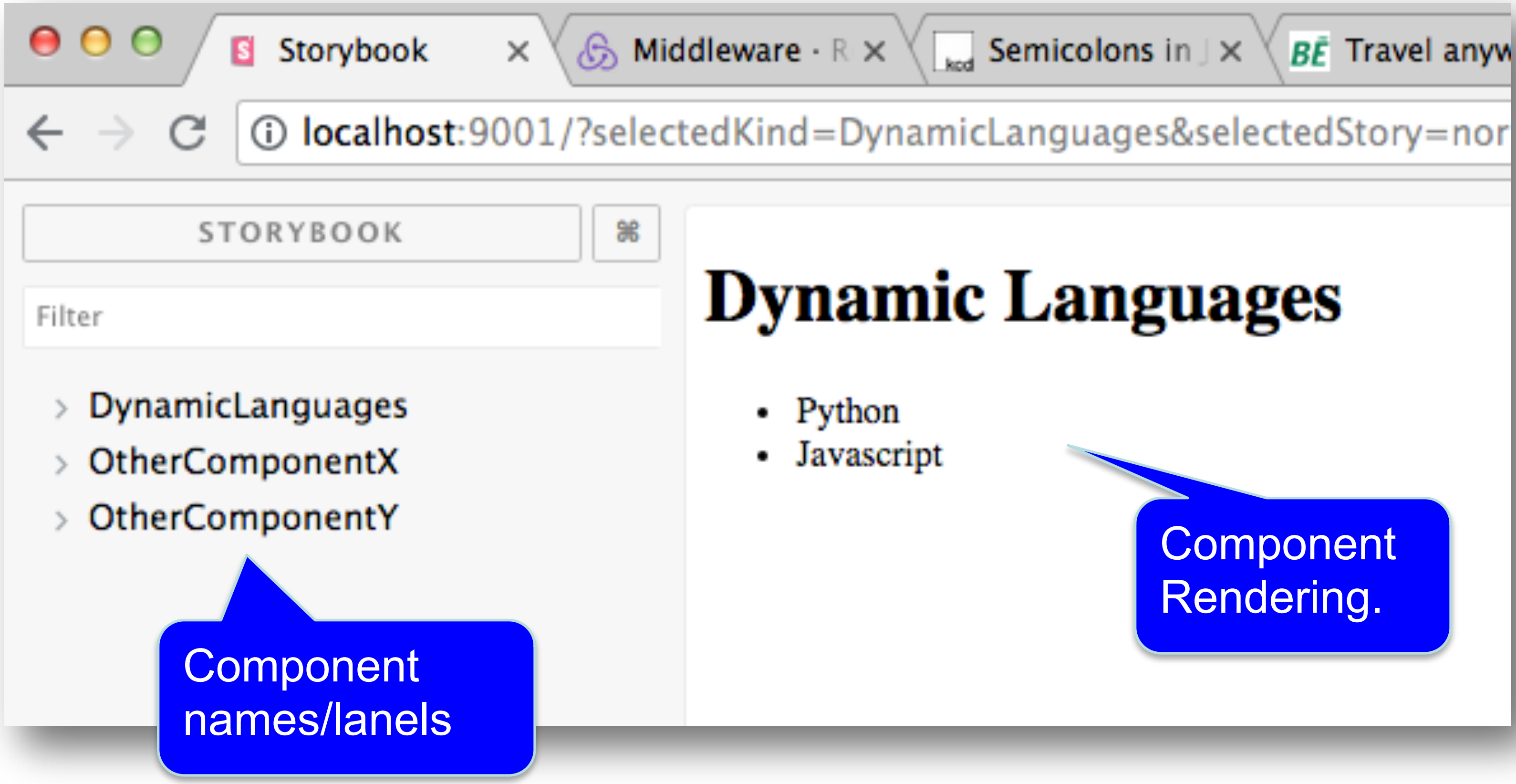
React Developer tools.

- create-react-app (CRA) **tool. Features:**
  - **Scaffolding/Generator.**
  - **Development web server: auto-transpilation on file change + live reloading.**
  - **Builder: build production standard version of app, i.e. minification, bundling.**

- Storybook **tool:**
  - **A** development environment **for UI components.**
  - **Runs outside of your app → develop component in isolation.**
  - **Leads to more reusable, testable components.**
  - **Quicker development – ignore app-specific dependencies.**

STORYBOOK

- **Tool interface.**



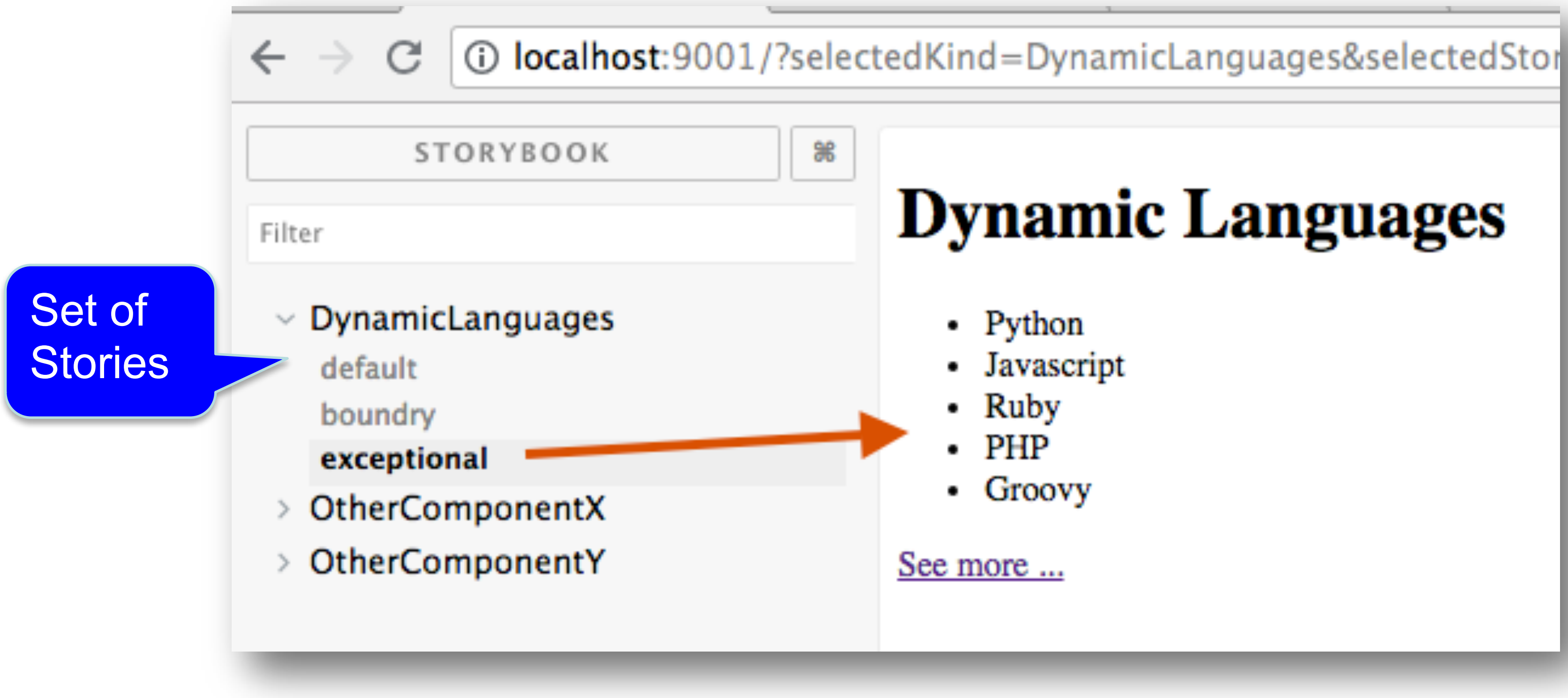
Dynamic Languages

- Python
- Javascript

- **Component design considerations:**
  - **A component may have several STATES → State effects what it renders to the screen.**
- EX.: DynamicLanguages **component.**
  - **States might be:**
    - Default **– less than 5 languages → Render full list**
    - Boundary **– empty list → Render 'No languages' message**
    - Exceptional **– More than 5 languages → Render first 5 and a 'See More…' link to display next 5.**
- **Each state case termed a STORY.**

STORYBOOK

- Document **a component's stories using** nesting**:**



localhost:9001/?selectedKind=DynamicLanguages&selectedStor

STORYBOOK

Filter

- DynamicLanguages
  - default
  - boundry
  - **exceptional**
- OtherComponentX
- OtherComponentY

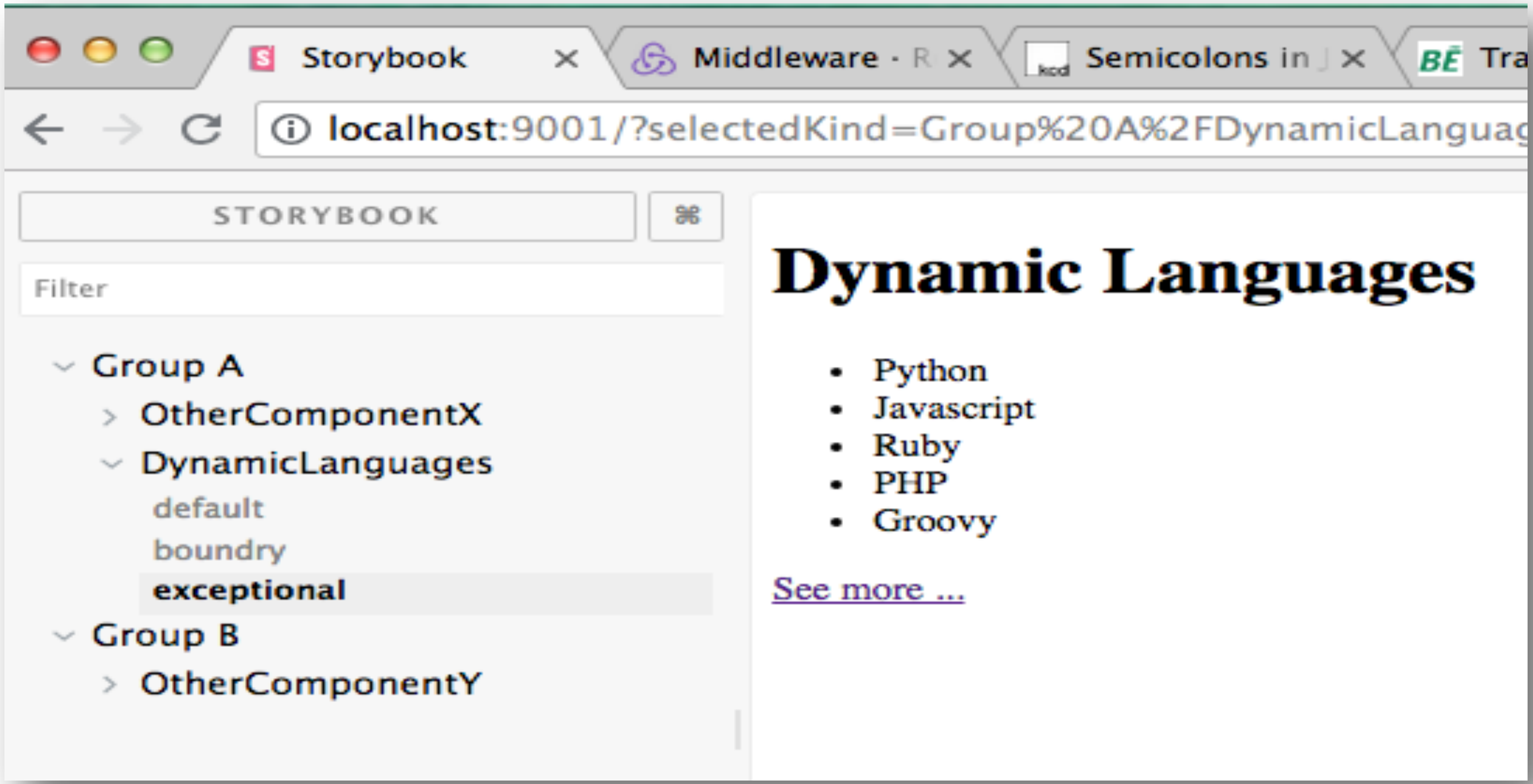
Dynamic Languages

- Python
- Javascript
- Ruby
- PHP
- Groovy

See more ...

STORYBOOK

- **For large component libraries define component** groups.
  - **helps others understand the catalogue**.



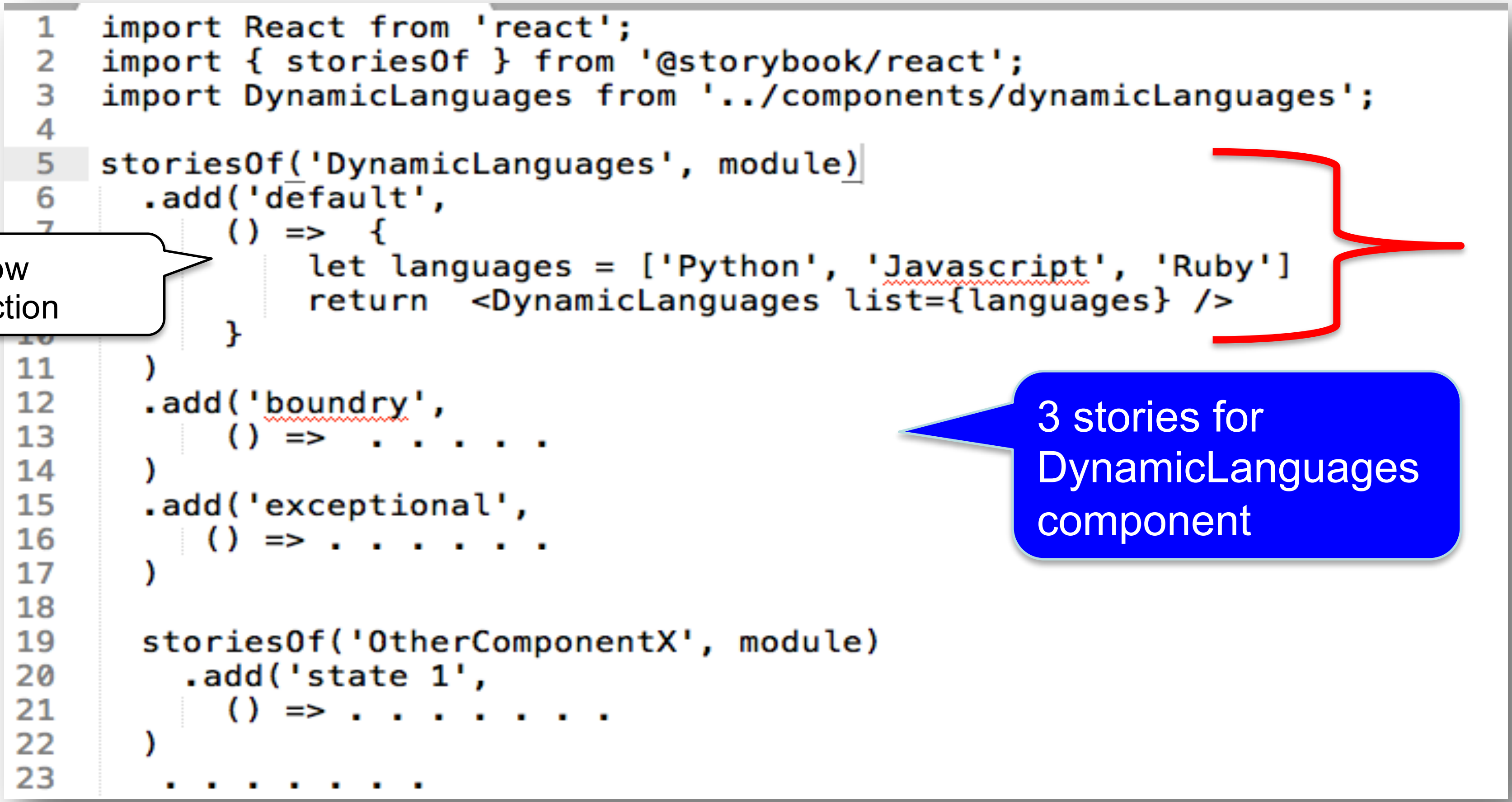
Dynamic Languages

- Python
- Javascript
- Ruby
- PHP
- Groovy

See more ...

Writing stories

- **Fluent-style API for writing stories.**
  - **Method chaining programming style.**



```
import React from 'react';
import { storiesOf } from '@storybook/react';
import DynamicLanguages from '../components/dynamicLanguages';

storiesOf('DynamicLanguages', module)
  .add('default',
       () =>  {
           let languages = ['Python', 'Javascript', 'Ruby']
           return  <DynamicLanguages list={languages} />
       }
  )
  .add('boundry',
       () =>  . . . . . 
  )
  .add('exceptional',
      () => . . . . . . 
  )

  storiesOf('OtherComponentX', module)
    .add('state 1',
       () => . . . . . . .
  )
   . . . . . . .
```

Arrow function

3 stories for DynamicLanguages component

Grouping stories.

- **Use directory pathname pattern –** *dir / subdir / subsubdir*,

```
storiesof('Group A/Component 1')
    .add('…'), () => {………}
    .add('…'), () => {………}
storiesof('Group A/Component 2')
   .add('…'), () => {………}
   .add('…'), () => {………}
storiesof('Group B/Component X')
    .add('…'), () => {………}
   .add('…'), () => {………}
   .add('…'), () => {………}
```

Grouping storoes

- **Lots of flexibility with grouping approach. Ex:**

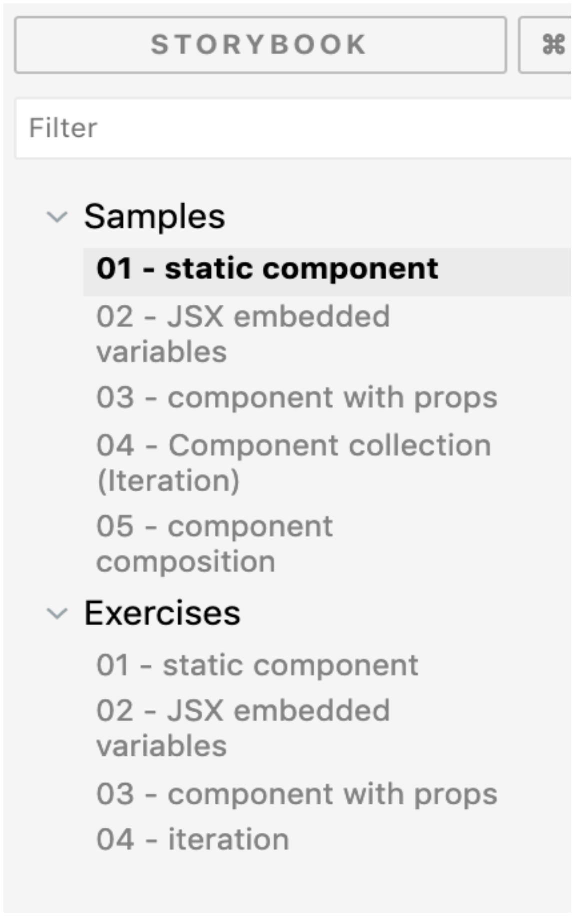
```
storiesof('Group A/Component 1/')
    .add('story1'), () => {………}
storiesof('Group A/Component 1/')
    .add('story2'), () => {………}
storiesof('Group A/Component 1/')
    .add('story3'), () => {………}

storiesof('Group A/Component 2/')
    .add('story1'), () => {………}
storiesof('Group A/Component 2/')
    .add('story1'), () => {………}
```

Storybook groups these 3 stories – same label

… back to components . . .

Samples

- **Samples to demonstrate Component features.**
  - **Basis for this week's lab..**

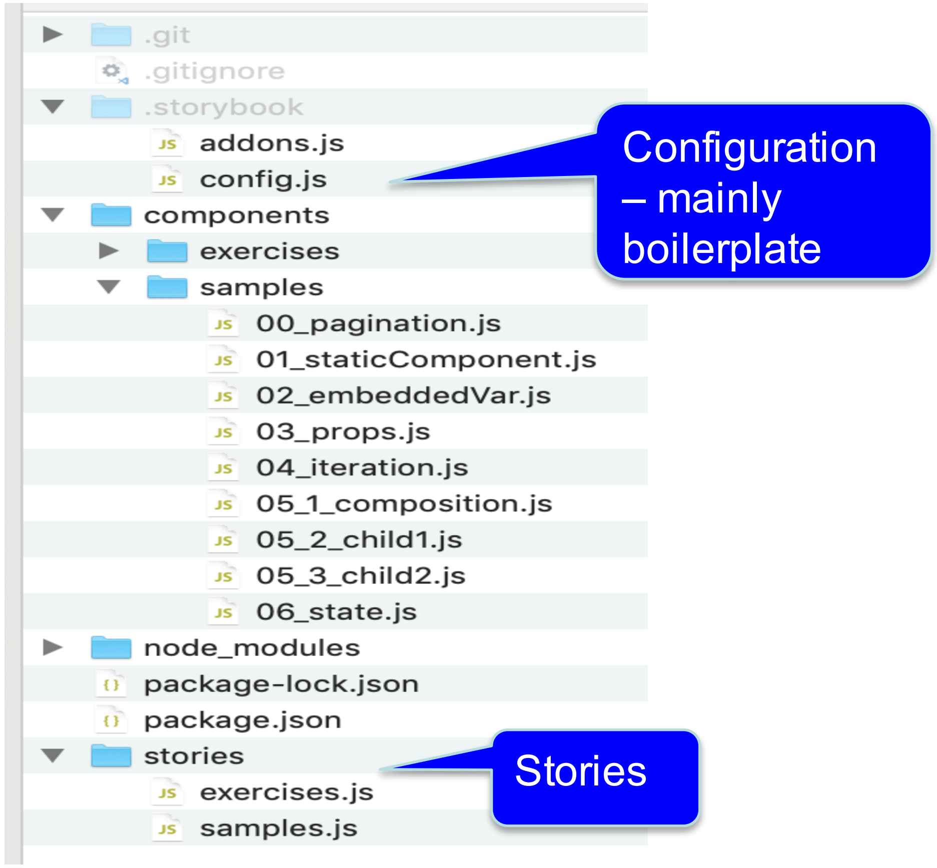


STORYBOOK ⌘

Filter

Samples
- **01 - static component**
- 02 - JSX embedded variables
- 03 - component with props
- 04 - Component collection (Iteration)
- 05 - component composition

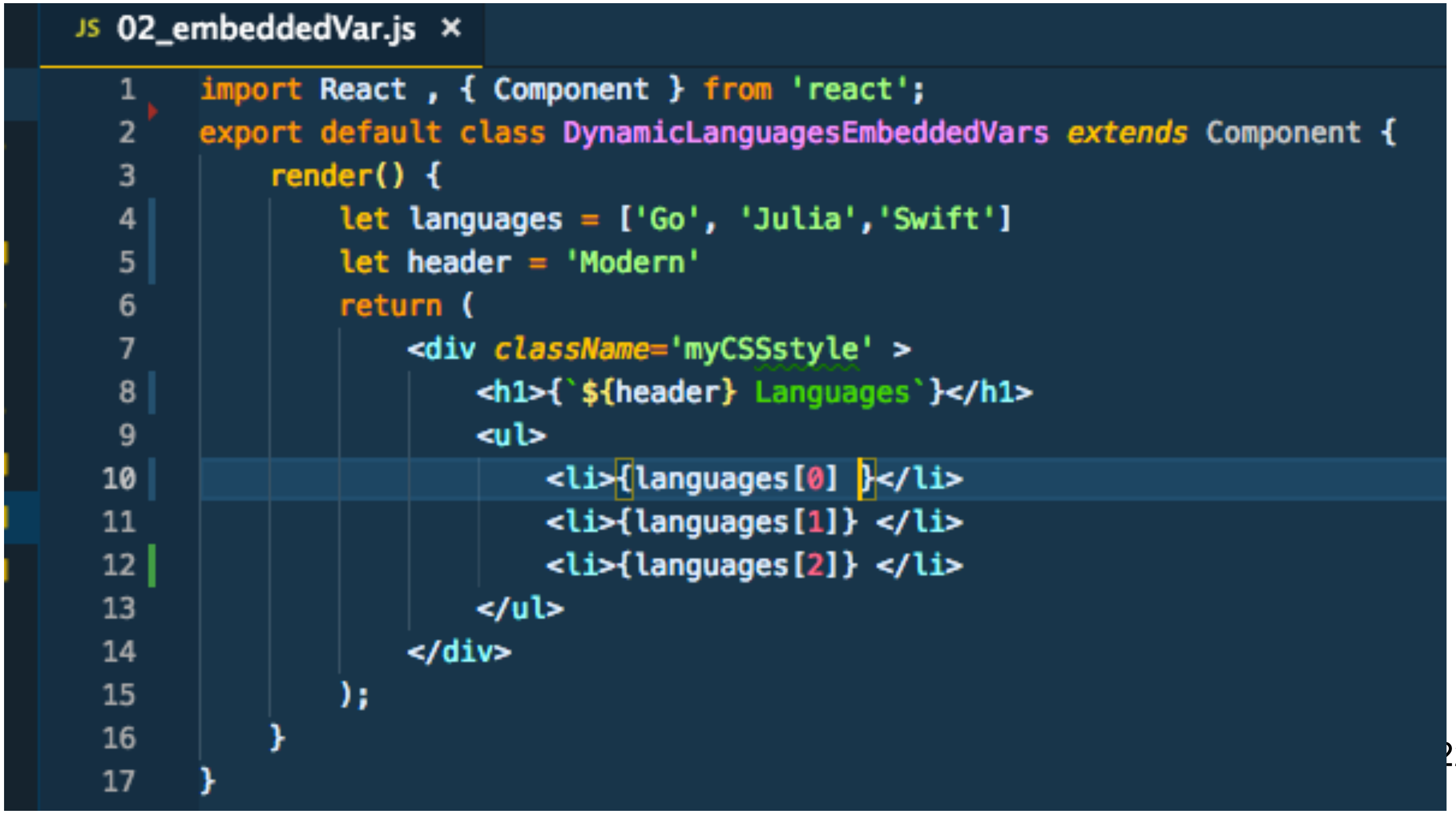
Exercises
- 01 - static component
- 02 - JSX embedded variables
- 03 - component with props
- 04 - iteration



```
▶ .git
  .gitignore
▼ .storybook
    addons.js
    config.js
▼ components
  ▶ exercises
  ▼ samples
      00_pagination.js
      01_staticComponent.js
      02_embeddedVar.js
      03_props.js
      04_iteration.js
      05_1_composition.js
      05_2_child1.js
      05_3_child2.js
      06_state.js
▶ node_modules
  package-lock.json
  package.json
▼ stories
    exercises.js
    samples.js
```

JSX embedded variables.

- **Dereference variable embedded in JSX code using { } braces.**
  - **Braces can contain any valid JS expression.**
- **Reference** samples/02_embeddedVariables.js



02_embeddedVar.js

```
import React , { Component } from 'react';
export default class DynamicLanguagesEmbeddedVars extends Component {
    render() {
        let languages = ['Go', 'Julia','Swift']
        let header = 'Modern'
        return (
            <div className='myCSSstyle' >
                <h1>{`${header} Languages`}</h1>
                <ul>
                    <li>{languages[0] }</li>
                    <li>{languages[1]} </li>
                    <li>{languages[2]} </li>
                </ul>
            </div>
        );
    }
}
```

Reusability.

- **Achieve reusability through** parameterization**.**
- props **– Component properties/attribute;**
  - **Passing props to a component:**

    <CompName prop1Name={value} prop2Name={value . . . . />
  - **Access inside component via** this.props **object:**

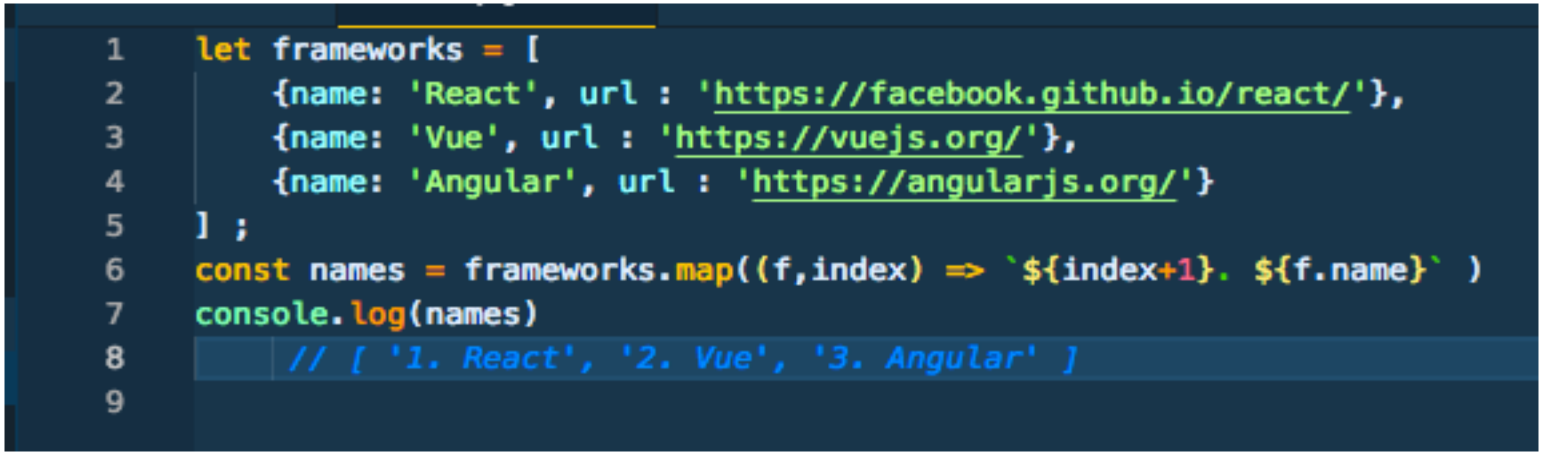
    let p1 = this.props.prop1Name
  - **Immutable.**

- **Reference** samples/03_props.js **and related story.**

Aside – Some JS issues

- **When an arrow function has only ONE statement, which is its return value, then you may omit:**
  - **Body curly braces; ‘return’ keyword; Semi-colon**

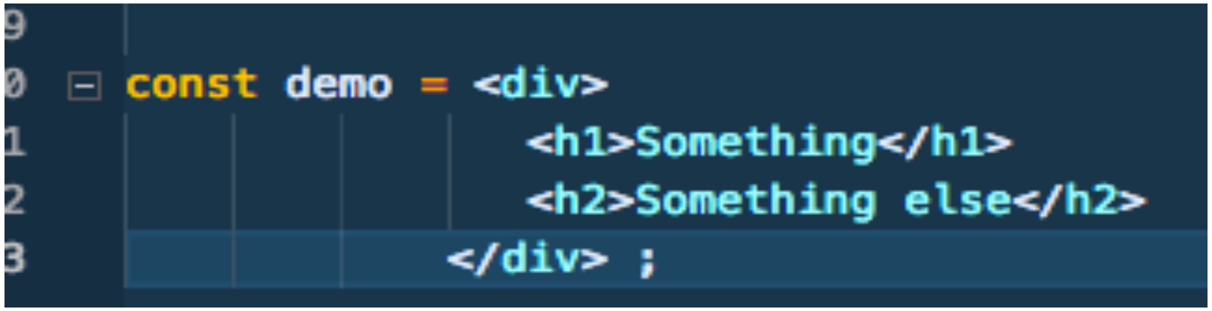
- **The Array** map **method – returns a new array based on applying the function argument to each element of the source array.**



```
let frameworks = [
    {name: 'React', url : 'https://facebook.github.io/react/'},
    {name: 'Vue', url : 'https://vuejs.org/'},
    {name: 'Angular', url : 'https://angularjs.org/'}
] ;
const names = frameworks.map((f,index) => `${index+1}. ${f.name}` )
console.log(names)
    // [ '1. React', '2. Vue', '3. Angular' ]

```

Aside – Some JS issues

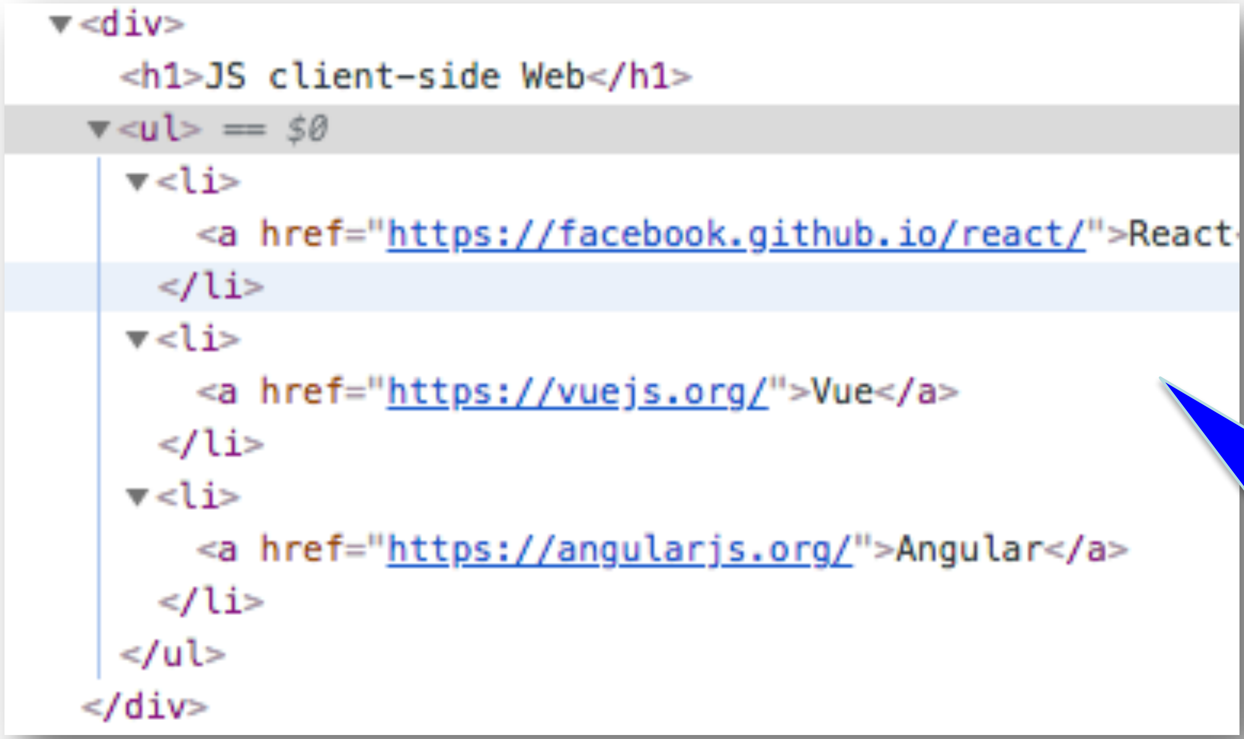
- **We can assign a** single **JSX element to a variable.**



```
const demo = <div>
                <h1>Something</h1>
                <h2>Something else</h2>
             </div> ;
```

Component collection - Iteration

- **Obj.: Generate a collection of component instances.**
- **Reference** samples/04_iteration.js



```
▼<div>
   <h1>JS client-side Web</h1>
 ▼<ul> == $0
   ▼<li>
      <a href="https://facebook.github.io/react/">React
    </li>
   ▼<li>
      <a href="https://vuejs.org/">Vue</a>
    </li>
   ▼<li>
      <a href="https://angularjs.org/">Angular</a>
    </li>
  </ul>
 </div>
```

Real DOM produced by story (From Chrome Dev Tools)

The render() return value.

- **Examples:**
  - return <h1>Something</h1> ;
  - return <MyComponent prop1={…..} prop2={……} /> ;
  - return (

```
        <div>
          <h1>{this.props.type}</h1>
          <ul>
              . . . . . . .
          </ul>
        </div>
  ) ;
```

- **Must enclose in ( ) when multiline.**

The render() return value.

- **Must return only ONE element.**
- **Examples:**
  - return (

    ```
    return (
            <h1>{this.props.type}</h1>
            <ul>
                . . . . . . . .
            </ul>
        ) ;
    ```

  - **Error** – ‘Adjacent JSX elements must be wrapped in an enclosing tag**’**
  - **Solution: Wrap elements in a div tag.**

The render() return value.

- **Old solution:**

```
return (
    <div>
      <h1>{this.props.type}</h1>
      <ul>
          . . . . . . . .
      </ul>
    </div>
    ) ;
```

- **Added unnecessary depth to DOM → effected performance.**

- **New solution:**

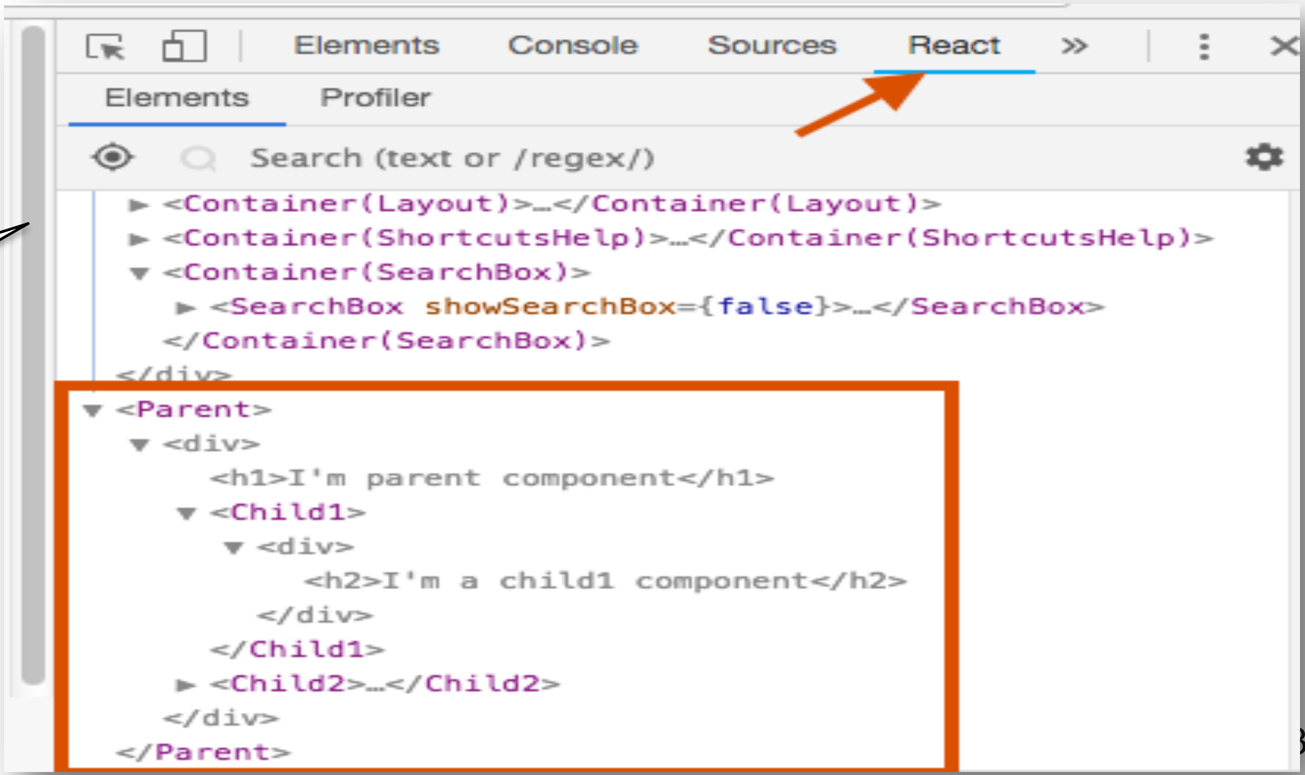
```
return (
    <Fragment>
      <h1>{this.props.type}</h1>
      <ul>
          . . . . . . . .
      </ul>
    </Fragment>
    ) ;
```

- Fragment **– special React element.**
  - **No DOM presence.**

Component ***Composition***.

A React application is designed as <u>a hierarchy of components</u>.

- **Components have** children **– nesting.**
- ***See*** *05_1_composition.js.*



```
▶ <Container(Layout)>…</Container(Layout)>
▶ <Container(ShortcutsHelp)>…</Container(ShortcutsHelp)>
▼ <Container(SearchBox)>
   ▶ <SearchBox showSearchBox={false}>…</SearchBox>
  </Container(SearchBox)>
</div>
▼ <Parent>
   ▼ <div>
        <h1>I'm parent component</h1>
      ▼ <Child1>
         ▼ <div>
              <h2>I'm a child1 component</h2>
           </div>
        </Child1>
      ▶ <Child2>…</Child2>
     </div>
  </Parent>
```

From Chrome Dev Tools React extension

Summary.

- **JSX.**
  - **UI** description **and** behaviour **tightly coupled.**

- **All about components.**
  - **A class that extends** React.Component.
  - **The** render **method.**
  - **Parameterization via props.**

- **Storybook tool.**
  - **Develop components in isolation.**
  - **Story – the state (data values) of a component can effect its rendering (and behaviour).**