ReactJS.

Fundamentals

Agenda

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

ReactJS.

- **A Javascript framework for building dynamic Web** User Interfaces**.**
  - **A 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 of:**

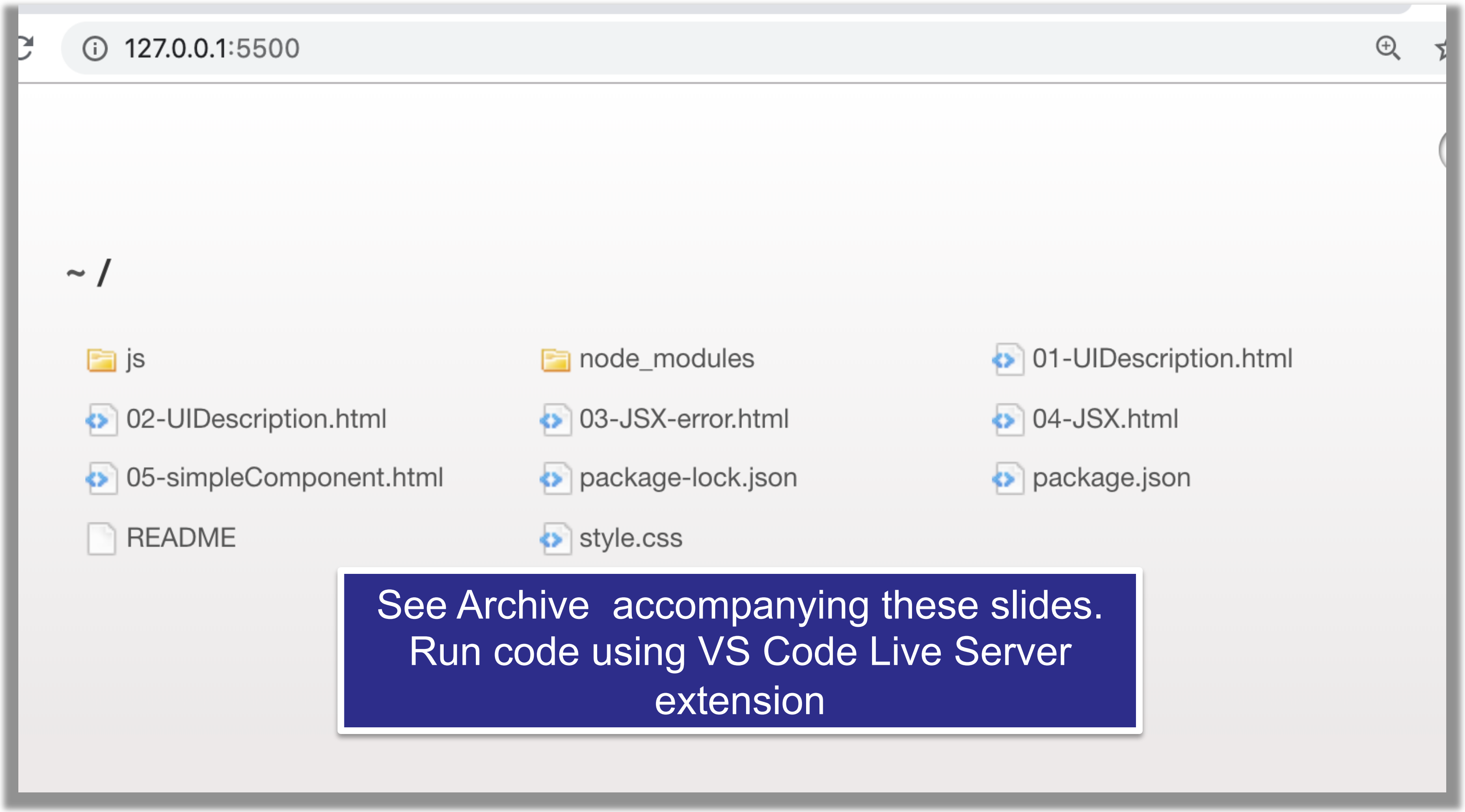
  UI description (HTML) + UI behavior (JS)
  - **Two aspects are tightly coupled and co-located.**
    - **Pre-React frameworks decoupled them.**
  - **Benefits:**
    1. **Improved Composition.**
    2. **Greater Reusability.**
    3. **Better Performance.**

Creating the UI description

- React.createElement() **– create a HTML element.**
- ReactDOM.render() – **attach an element to the DOM.**
- createElement() **arguments**:
  1. **type (h1, div, button etc).**
  2. **properties (style, event handler etc).**
  3. **children (0 -> M).**
  - **We never use** createElement**() directly – too cumbersome.**
- ReacrDOM.render() **arguments:**
  1. **element to be displayed.**
  2. **DOM node on which to** mount **the element**.

- **Ref.** 01-UIDescription.html.
- **Nesting** createElement**() calls - Ref**. 02-UIDescription.html

Code Demos



127.0.0.1:5500

~ /

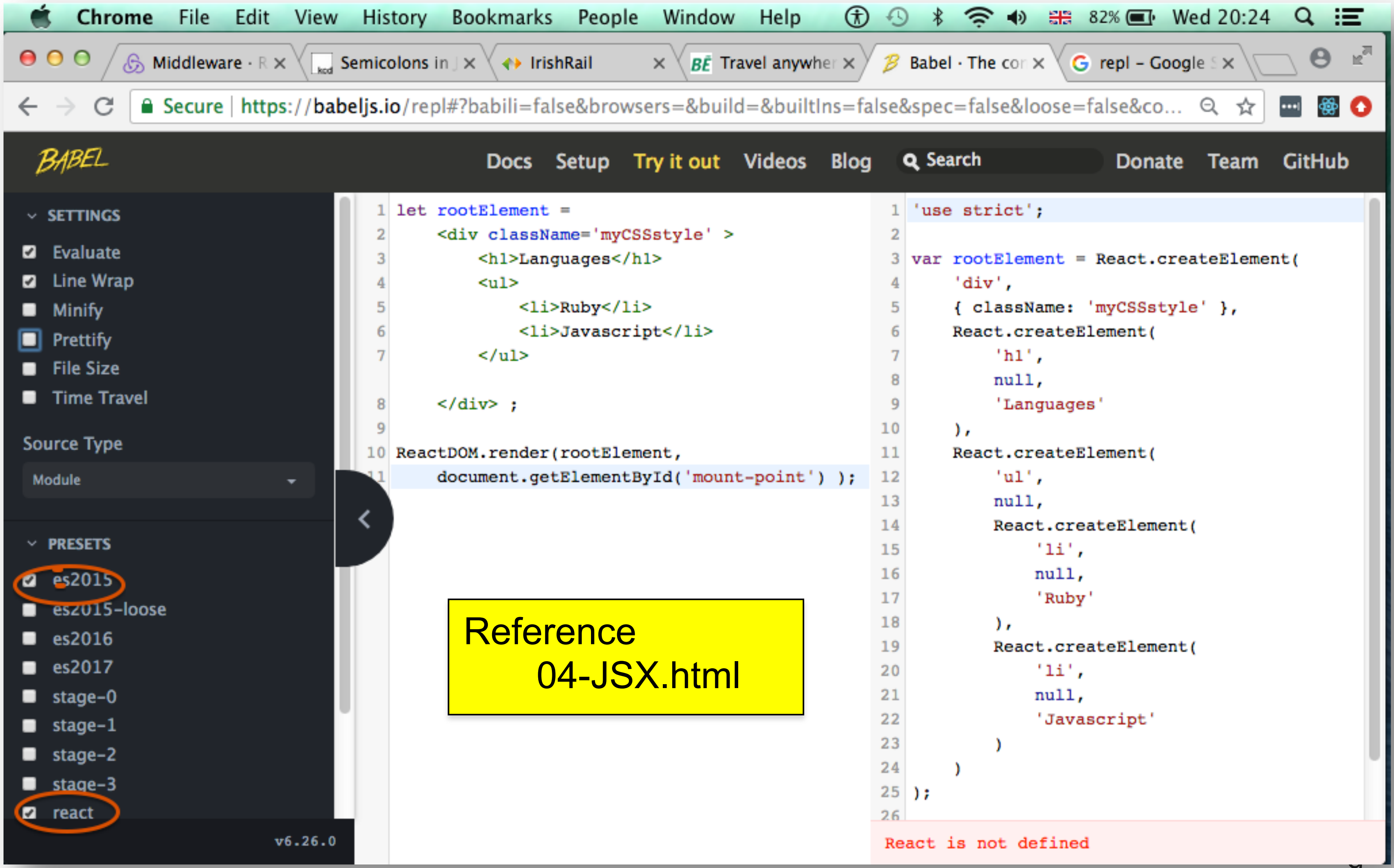
js
node_modules
01-UIDescription.html
02-UIDescription.html
03-JSX-error.html
04-JSX.html
05-simpleComponent.html
package-lock.json
package.json
README
style.css

See Archive accompanying these slides.
Run code using VS Code Live Server extension

JSX.

- **JSX – JavaScript extension syntax.**
- **<u>Declarative</u> <u>syntax</u> for coding UI descriptions.**
- **Retains the full power of Javascript.**
- **Allows tight coupling between UI behavior and UI description.**
- **Must be transpiled before being sent to browser.**
  - **The Babel tool**
- **Reference** 03-JSX-error.html and 04-JSX.html

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


Chrome File Edit View History Bookmarks People Window Help
Wed 20:24
Middleware · R
Semicolons in J
IrishRail
Travel anywhere
Babel · The cor
repl – Google S
Secure 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.**
- **Some minor HTML tag attributes differences, e.g. className (class), htmlFor (for).**
- **Allows** UI description **be coded in a** declarative style **and be 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. **During development by the web server (using special tooling, i.e.Webpack).**
  3. **Before deployment as part of the build process for an app.**

React Components.

- **We develop COMPONENTS.**
  - **A JS** function that **returns a** UI description, i.e. JSX**.**

- **Can reference a component like a <u>HTML tag.</u>**

  e.g. ReactDOM.render(<ComponentName />, ...)

- **Reference** 05-simpleComponent.html

React Developer tools.

- create-react-app (CRA) **- 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 - **Features:**
  - **A** development environment **for React components.**
  - **Allows components be developed in isolation.**
  - **Promotes more reusable, testable components.**
  - **Quicker development – ignore app-specific dependencies.**

STORYBOOK

- **Installation:**

```
$ npm install @storybook/react
```

- **Tool has two aspects:**

1. **The server:**

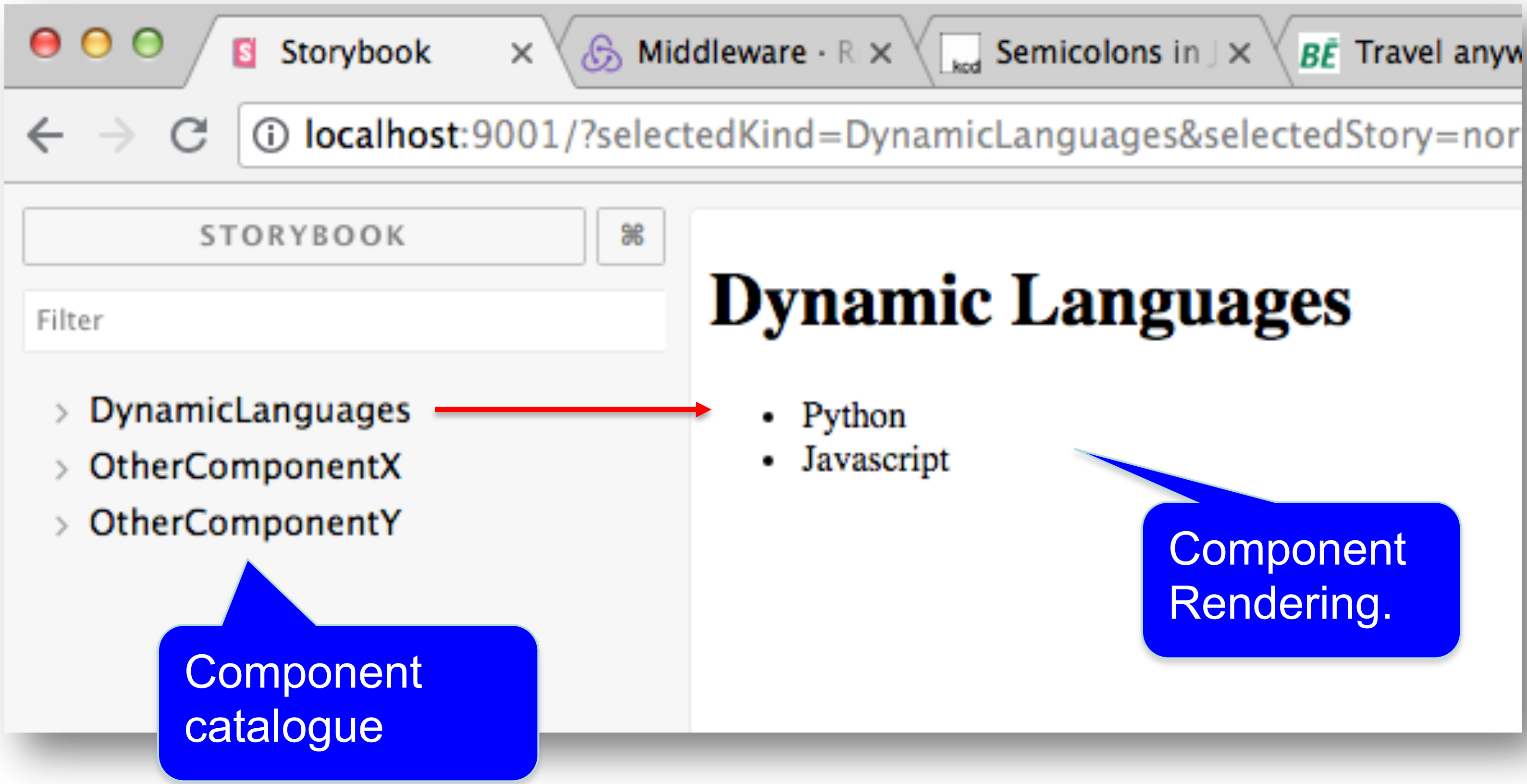
```
$ ./node_modules/.bin/start-storybook -p 6006 -c ./.storybook
```

   - **Performs live re-transpilation and re-loading.**

2. **The User interface.**

STORYBOOK

- **Storybook UI (User interface).**

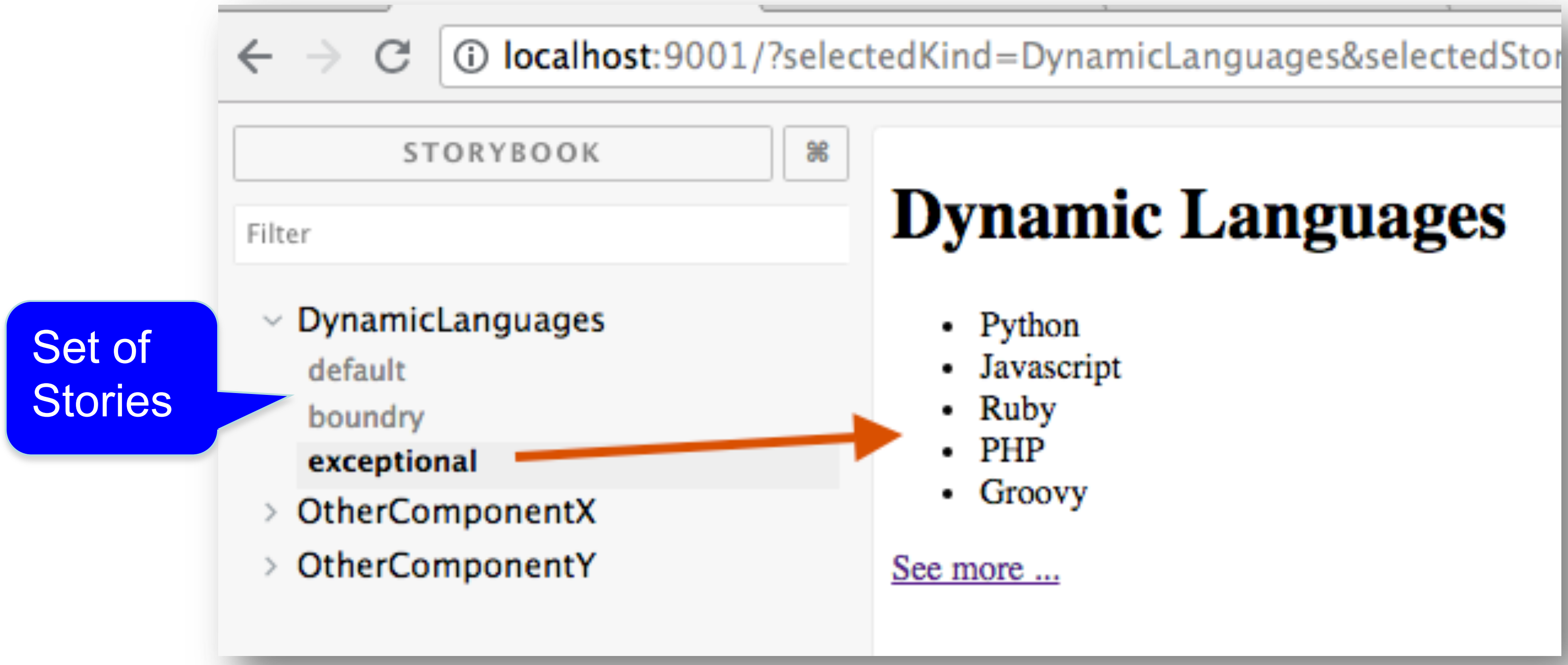


STORYBOOK

- **What is a Story?**
- **A component may have several STATES → State effects how it renders.**
  - **Each state case termed a STORY.**
  - **Stories are a design consideration.**
- EX.: DynamicLanguages **component.**
  - **States might be:**
    - Default **– 5 or less 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.**

STORYBOOK

- **Storybook UI – List a component's states/stories under its name:**



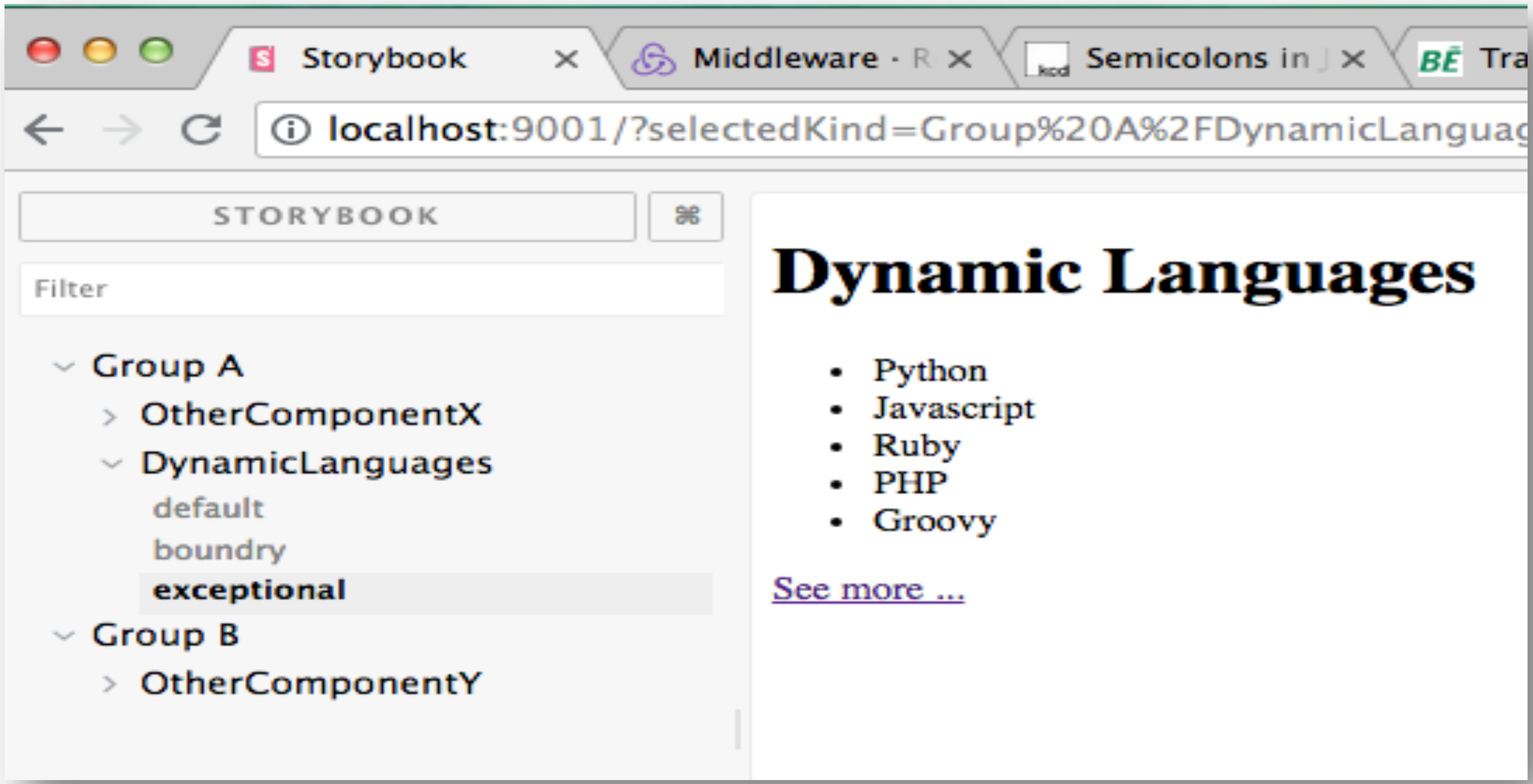
Dynamic Languages

- Python
- Javascript
- Ruby
- PHP
- Groovy

See more ...

STORYBOOK

- **Define component** groups when **component library is large.**
  - **helps others understand the structure**.



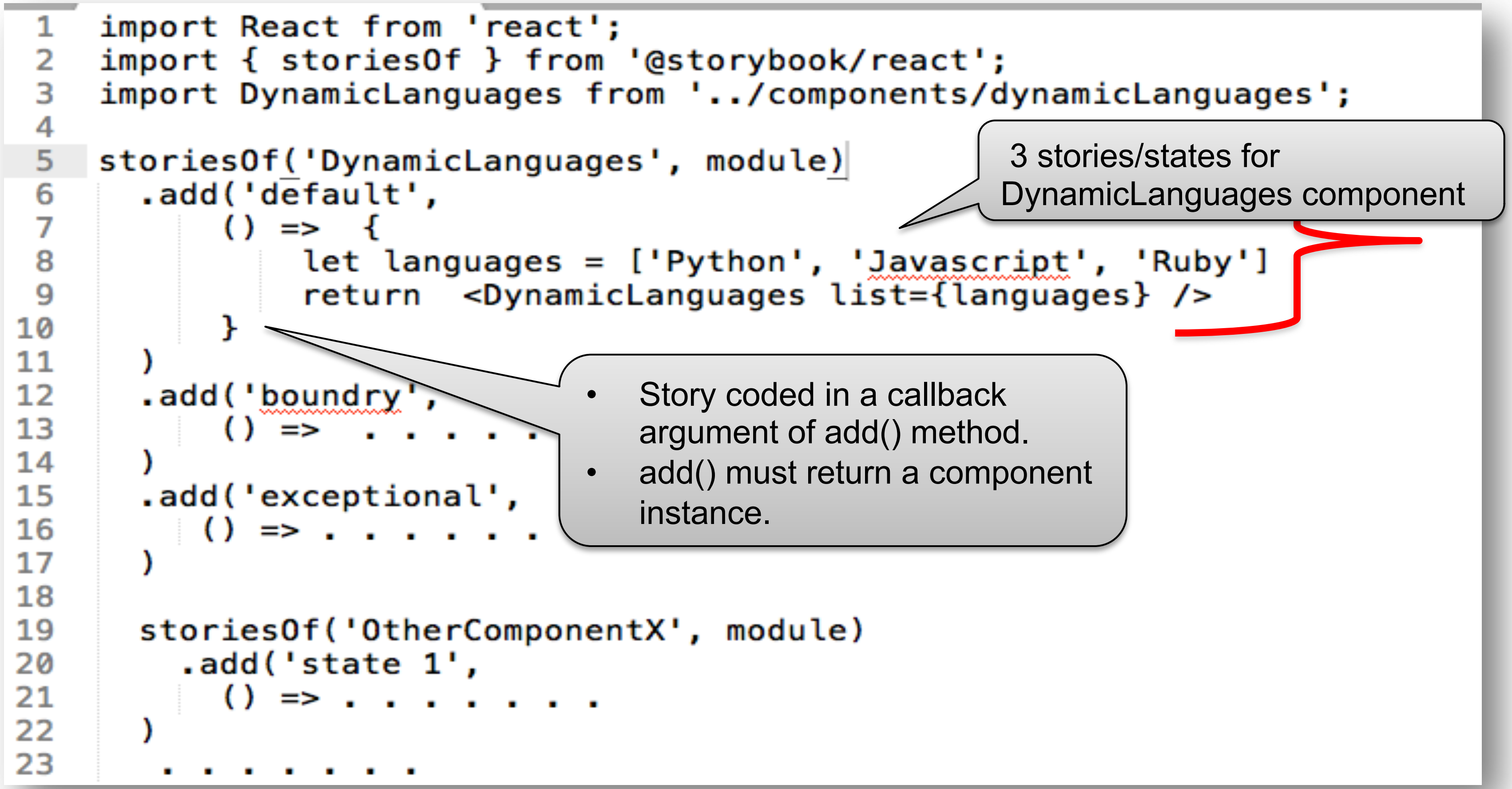
Dynamic Languages

- Python
- Javascript
- Ruby
- PHP
- Groovy

See more ...

Writing stories

- **Fluent-style syntax 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',
      () => . . . . . . .
  )
   . . . . . . .
```

3 stories/states for DynamicLanguages component

- Story coded in a callback argument of add() method.
- add() must return a component instance.

Grouping stories.

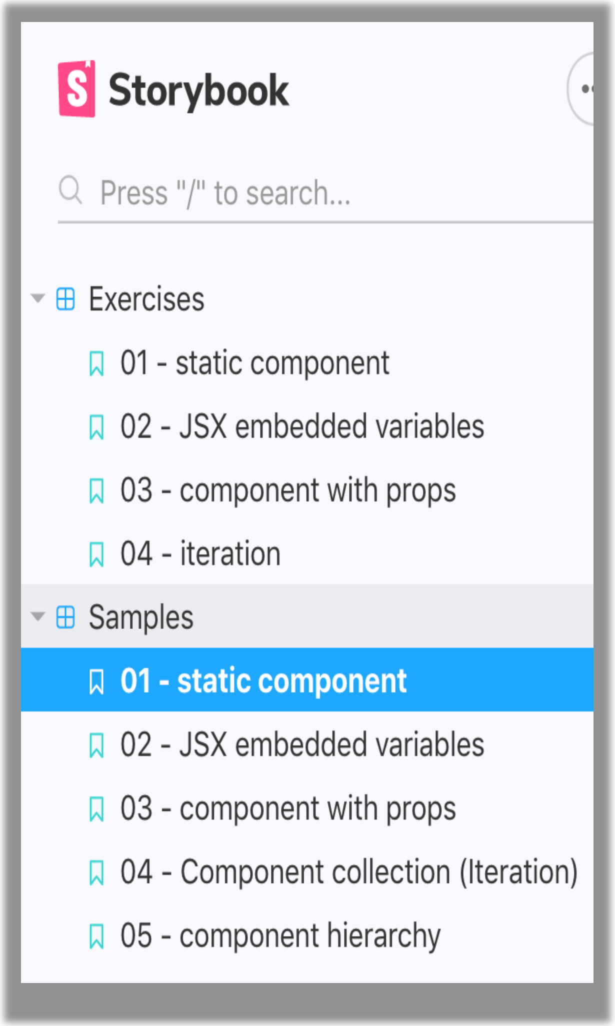
- **Use directory pathname symbol (/) to indicate component grouping (i.e.** group*/subgroup/….).* **EX.:**

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

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

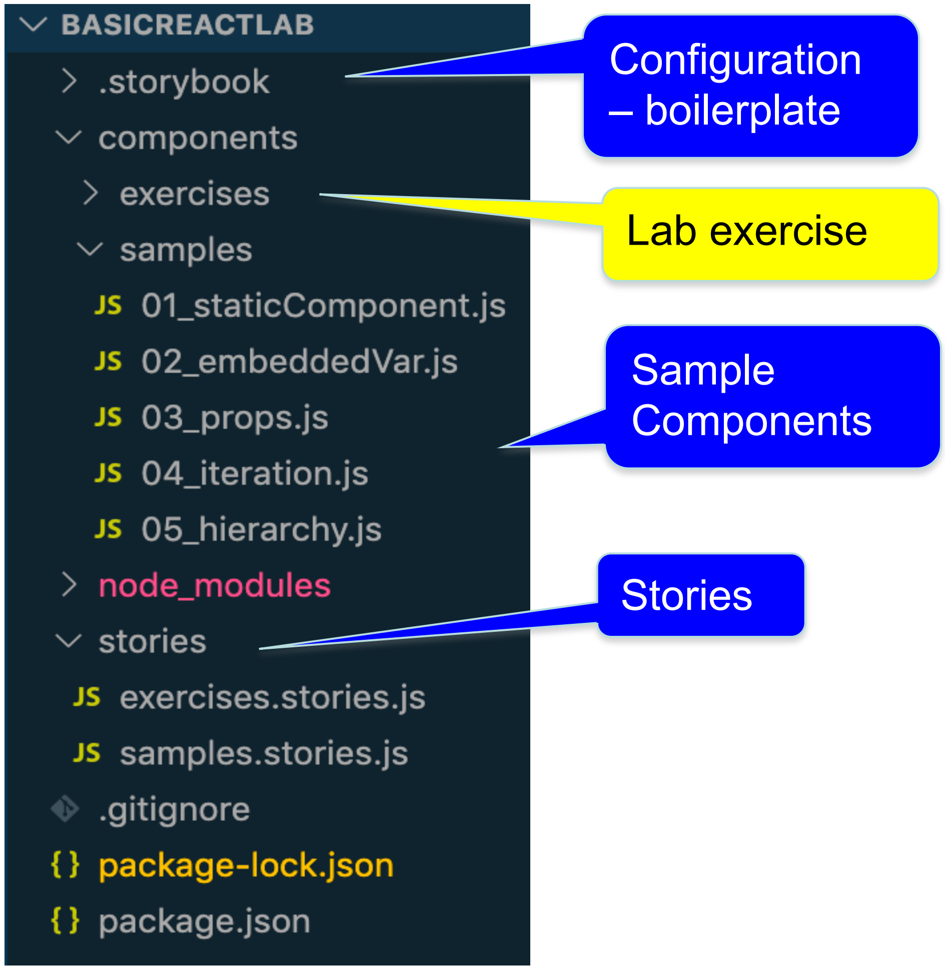
… back to components . . .

Demo Samples


Storybook

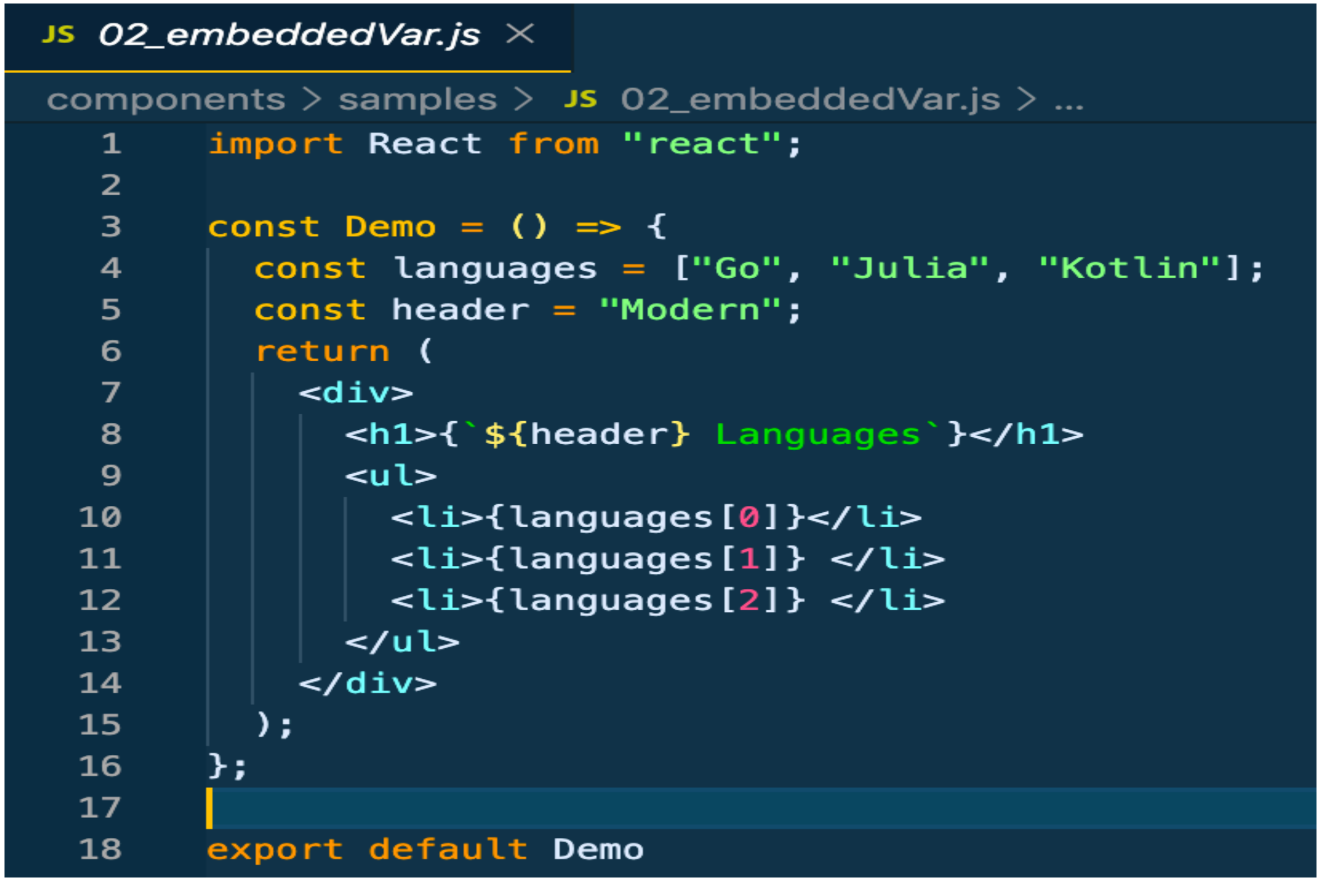
Press "/" to search...

- Exercises
  - 01 - static component
  - 02 - JSX embedded variables
  - 03 - component with props
  - 04 - iteration
- Samples
  - 01 - static component
  - 02 - JSX embedded variables
  - 03 - component with props
  - 04 - Component collection (Iteration)
  - 05 - component hierarchy


```
BASICREACTLAB
  > .storybook
  v components
    > exercises
    v samples
        01_staticComponent.js
        02_embeddedVar.js
        03_props.js
        04_iteration.js
        05_hierarchy.js
  > node_modules
  v stories
      exercises.stories.js
      samples.stories.js
    .gitignore
    package-lock.json
    package.json
```

JSX embedded variables.

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



JS 02_embeddedVar.js

components > samples > JS 02_embeddedVar.js > ...

```
import React from "react";

const Demo = () => {
  const languages = ["Go", "Julia", "Kotlin"];
  const header = "Modern";
  return (
    <div>
      <h1>{`${header} Languages`}</h1>
      <ul>
        <li>{languages[0]}</li>
        <li>{languages[1]} </li>
        <li>{languages[2]} </li>
      </ul>
    </div>
  );
};

export default Demo
```

Reusability.

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

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

     ```
     const ComponentName = (props) => {
          const p1 = props.prop1Name
          . . . . . . . . .
     ```
  3. **Props are** Immutable**.**
  4. **Part of a component's design.**

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

Aside – Some JS issues

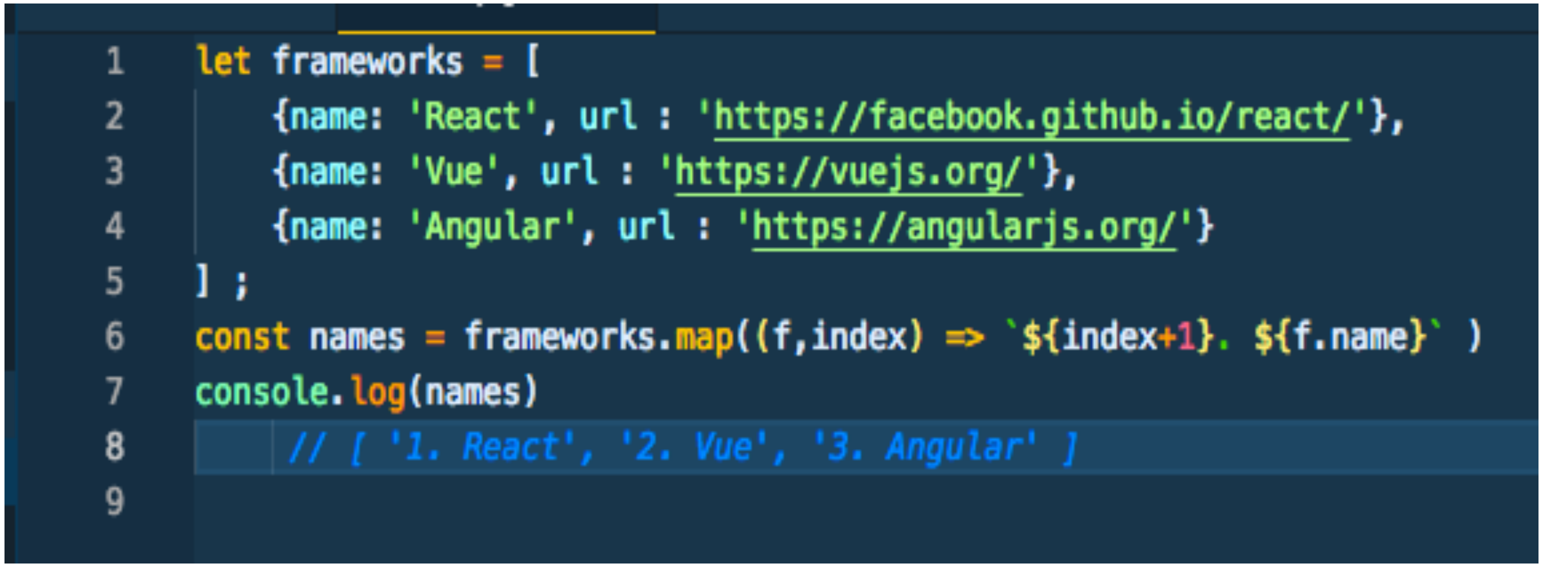
- **When an arrow function has only ONE statement, which is its return value, then you may omit:**
  - **Body curly braces; ‘return’ keyword.**

```
const increment = (num) => {
    return num + 1
}
```

```
const increment = (num) => num + 1
```

Aside – Some JS issues

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

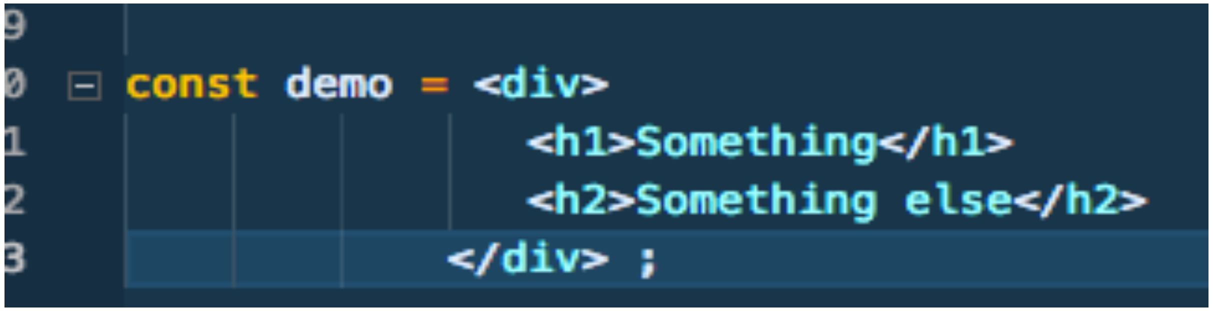


```
let frameworks = [
    {name: 'React', url : 'https://facebook.github.io/react/'},
    {name: 'Vue', url : 'https://vuejs.org/'},
    {name: 'Angular', url : 'https://angularjs.org/'}
] ;
const names = frameworks.map((f,index) => `${index+1}. ${f.name}` )
console.log(names)
    // [ '1. React', '2.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

```

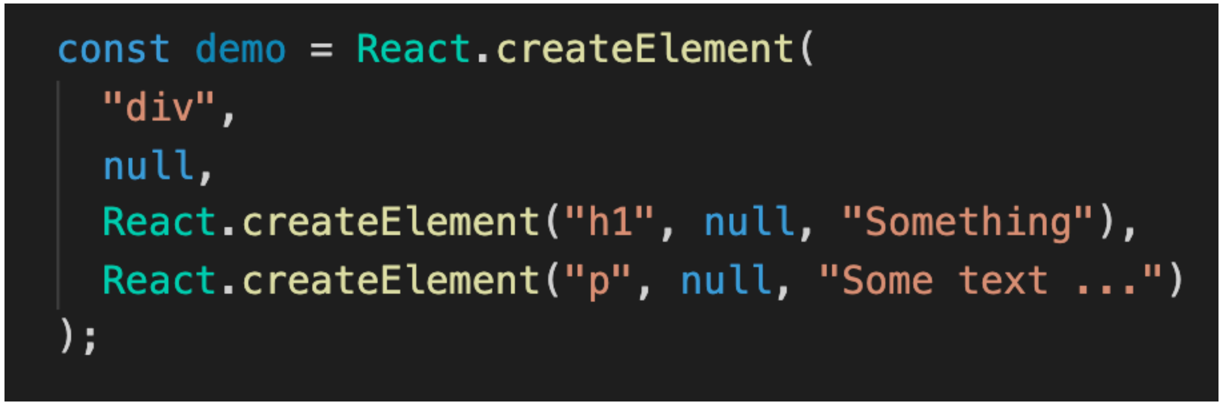
Aside – Some JS issues

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



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

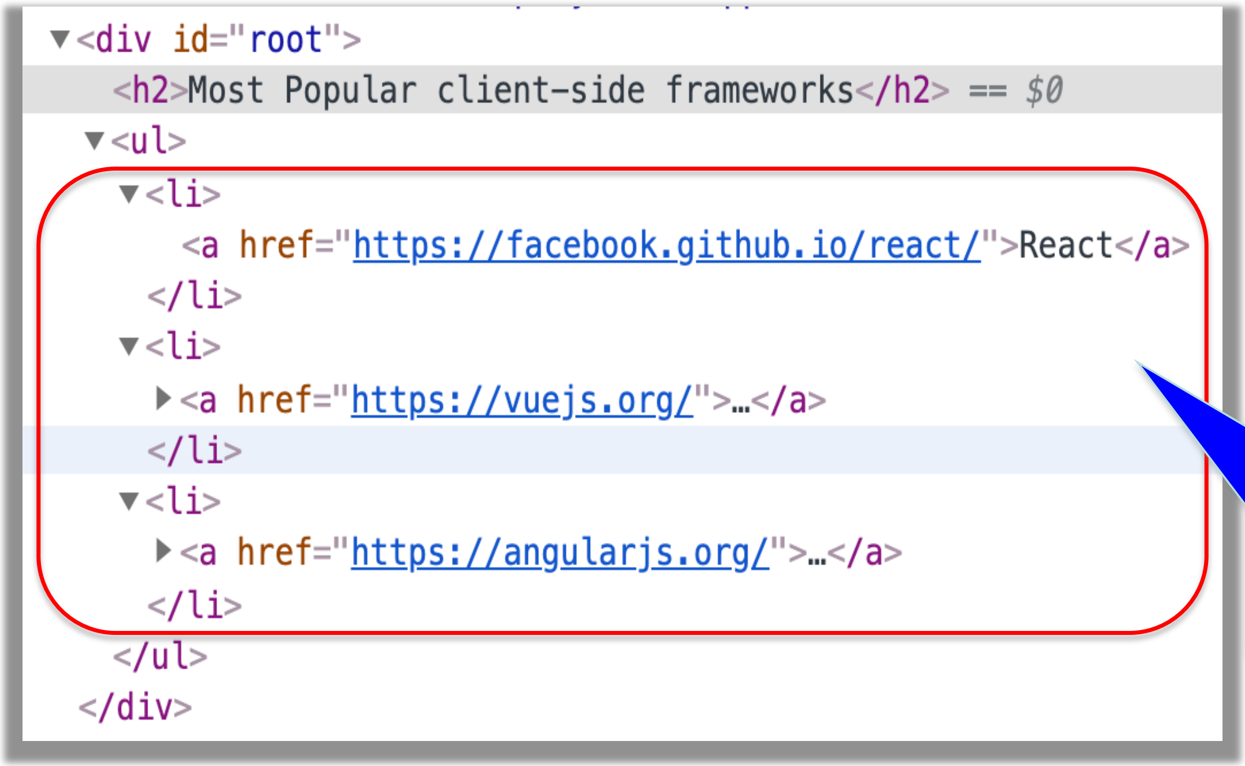
- **Why?**



```
const demo = React.createElement(
  "div",
  null,
  React.createElement("h1", null, "Something"),
  React.createElement("p", null, "Some text ...")
);
```

Component collection - Iteration

- **Use case: A component prop is an array which it uses to generate a collection of JSX elements.**
- **Reference** samples/04_iteration.js



```
▼<div id="root">
    <h2>Most Popular client-side frameworks</h2> == $0
  ▼<ul>
    ▼<li>
        <a href="https://facebook.github.io/react/">React</a>
      </li>
    ▼<li>
      ▶<a href="https://vuejs.org/">…</a>
      </li>
    ▼<li>
      ▶<a href="https://angularjs.org/">…</a>
      </li>
    </ul>
  </div>
```

Required HTML produced by component. (From Chrome Dev Tools)

Component return value.

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

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

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

Component return value.

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

Component return value.

- **Old solution:**

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

- **Adds unnecessary depth to DOM → effects performance.**

- **New solution:**

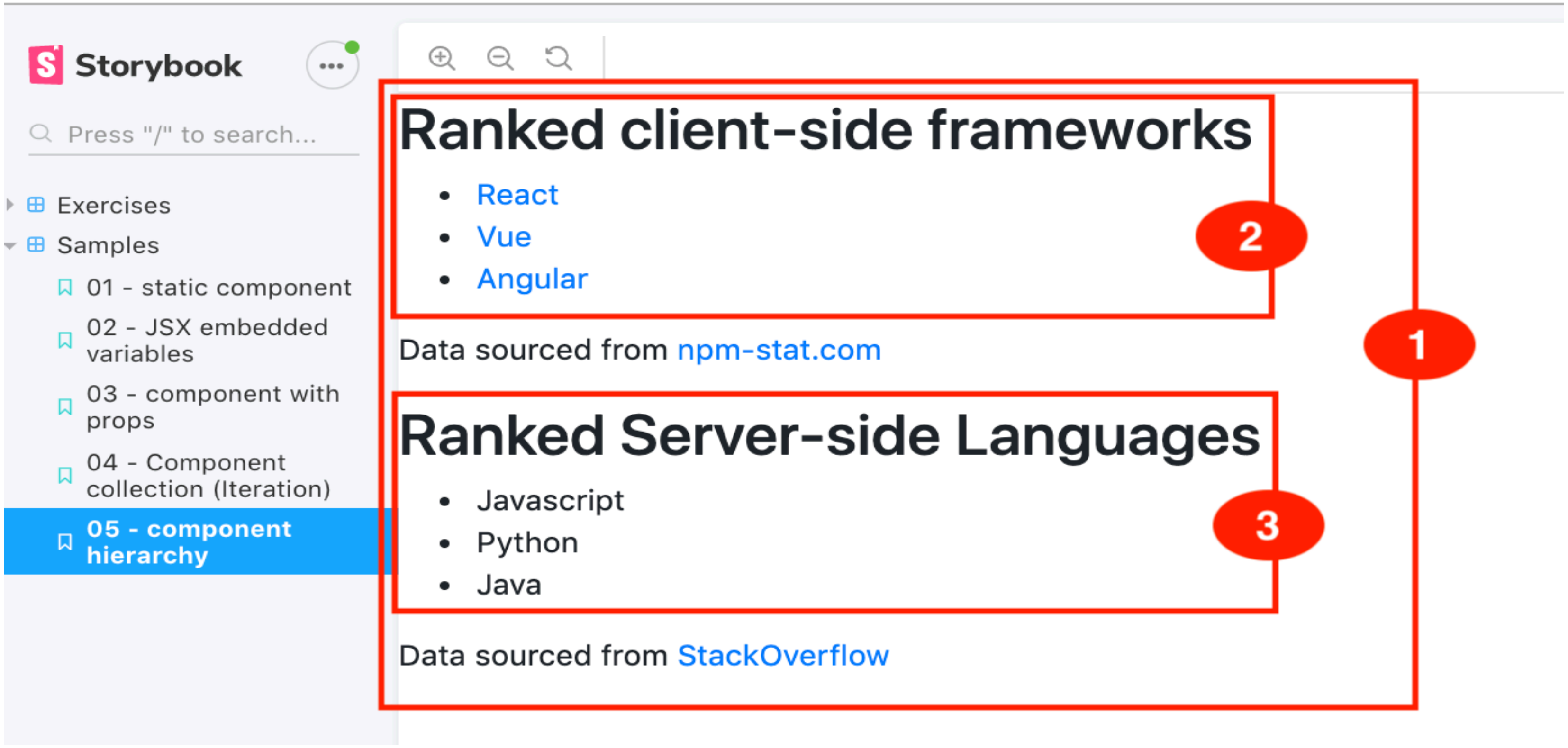
```
return (
    <>
      <h1>{this.props.type}</h1>
      <ul>
          . . . . . . . .
      </ul>
    </>
    ) ;
```

- **<> </> – special React element, termed** Fragment**.**
  - **No DOM presence.**

Component *Hierarchy*.

A React application is designed as a hierarchy of components.

- **Components have** children **– nesting.**
- **Ref.** 05_hierarchy.js.



Ranked client-side frameworks

- React
- Vue
- Angular

Data sourced from npm-stat.com

Ranked Server-side Languages

- Javascript
- Python
- Java

Data sourced from StackOverflow

Material UI.

Material Design.

- Material **is a** design system **created by Google to help teams build** high-quality digital experiences **for Android, iOS, and web.**
- **A** visual lang**uage that synthesizes classic principles of good design with the innovation and possibility of technology and science.**
- **Inspired by:**
  - **the physical world and its textures, including how they reflect light and cast shadows.**
  - **the study of paper and ink.**

- **Material is a metaphor.**
  - **Material surfaces reimagine the mediums of paper and ink.**

Material Components.

- Material Components **are interactive building blocks for creating a user interface.**
- **They cover a range of interface needs, including:**
  1. **Display: Placing and organizing content using components like cards, lists, and grids.**
  2. **Navigation: Allowing users to move through the product using components like navigation drawers and tabs.**
  3. **Actions: Allowing users to perform tasks using components such as the floating action button.**
  4. **Input: Enter information or make selections using components like text fields and selection controls.**
  5. **Communication: Alerting users to key information and messages using snackbars, banners and dialogues.**

Theming.

- **Material Design does not mean copy Google design.**
- Material Theming **makes it easy to customize Material Design to match the look and feel of your** brand**, with built-in support and guidance for customizing colors, typography styles, and corner shape.**
- Color **- Material’s color system is an organized approach to applying color to a UI. Global color styles have** semantic names **and defined usage in components – primary, secondary.**
- Typography **- The Material Design type system provides 13** typography styles **for everything from headlines to body text and captions. Each style has a clear meaning and intended application within an interface.**

Material UI.

- **A React component library for faster and easier web development.**
- **<Card />, <Box />, <Grid />, <Menu />, <Button />, <Icon />, <Snackbar />, <Typography /> ……**

- **Build your own design system, or start with Material Design.**

- **The CSS-in-JS model.**

CSS-in-JS

- **Plain CSS**

.my-header {
  background-color: lightblue;
  padding: 10px;
}

- **--------------------**

import ‘app.css’

<header
  className=”my-header">
    ………..
</header>

Must be CamelCase

- **CSS-in-JS**

.import { makeStyles } from
  "@material-ui/core/styles";
const useStyles = makeStyles(({ 1
  myHeader: {
    backgroundColor: "lightblue",
    padding: "10px”
  });
const classes = useStyles(); 2
<header
className={classes.myHeader}>
……….. </header> 3

Summary.

- **JSX.**
  - **UI** description **and** behaviour **tightly coupled.**
  - **Can embed variables/expressions with braces.**
- **All about components.**
  - **A function that takes a props argument and returns a single JSX element .**
  - **Components can be nested.**
- **Storybook tool.**
  - **Develop components in isolation.**
  - **Story – the state (data values) of a component can effect its rendering (and behaviour).**
- **Material Design – The Material UI React library.**