ReactJS.

Fundamentals

Agenda

- **Background.**
- **The V in MVC**
- **JSX (JavaScript Extension Syntax).**
- **Developer tools..**
- **React 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

Components

- **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.**

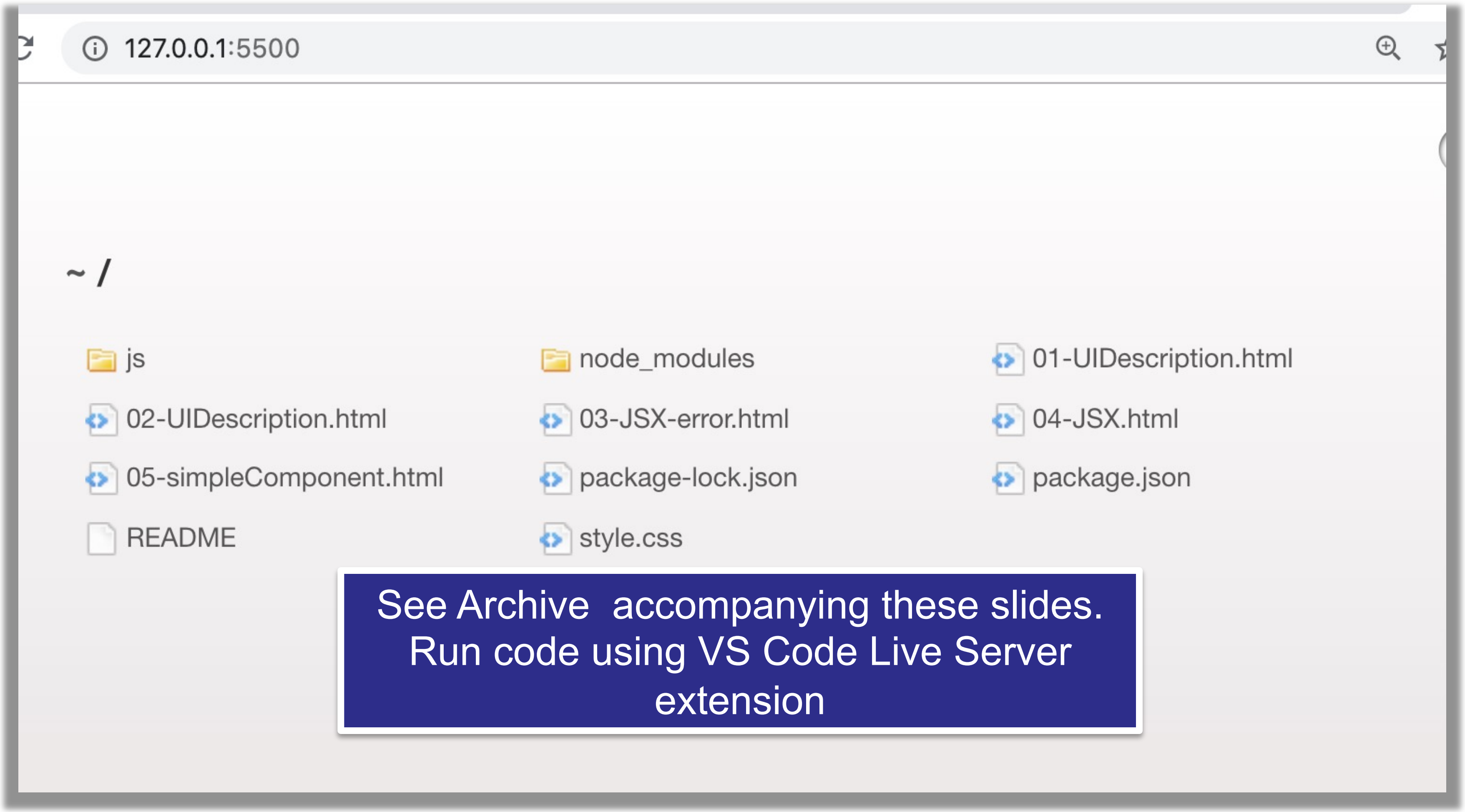
Creating the UI description

- React.createElement() **– create a HTML element.**
- ReactDOM.render() – **attach an element to the DOM.**

- React.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**.

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

UI description implementation
(the imperative way)

- **See the demos:**
  - **Ref.** 01-UIDescription.html.
  - **Nesting** createElement**() calls - Ref**. 02-UIDescription.html

----

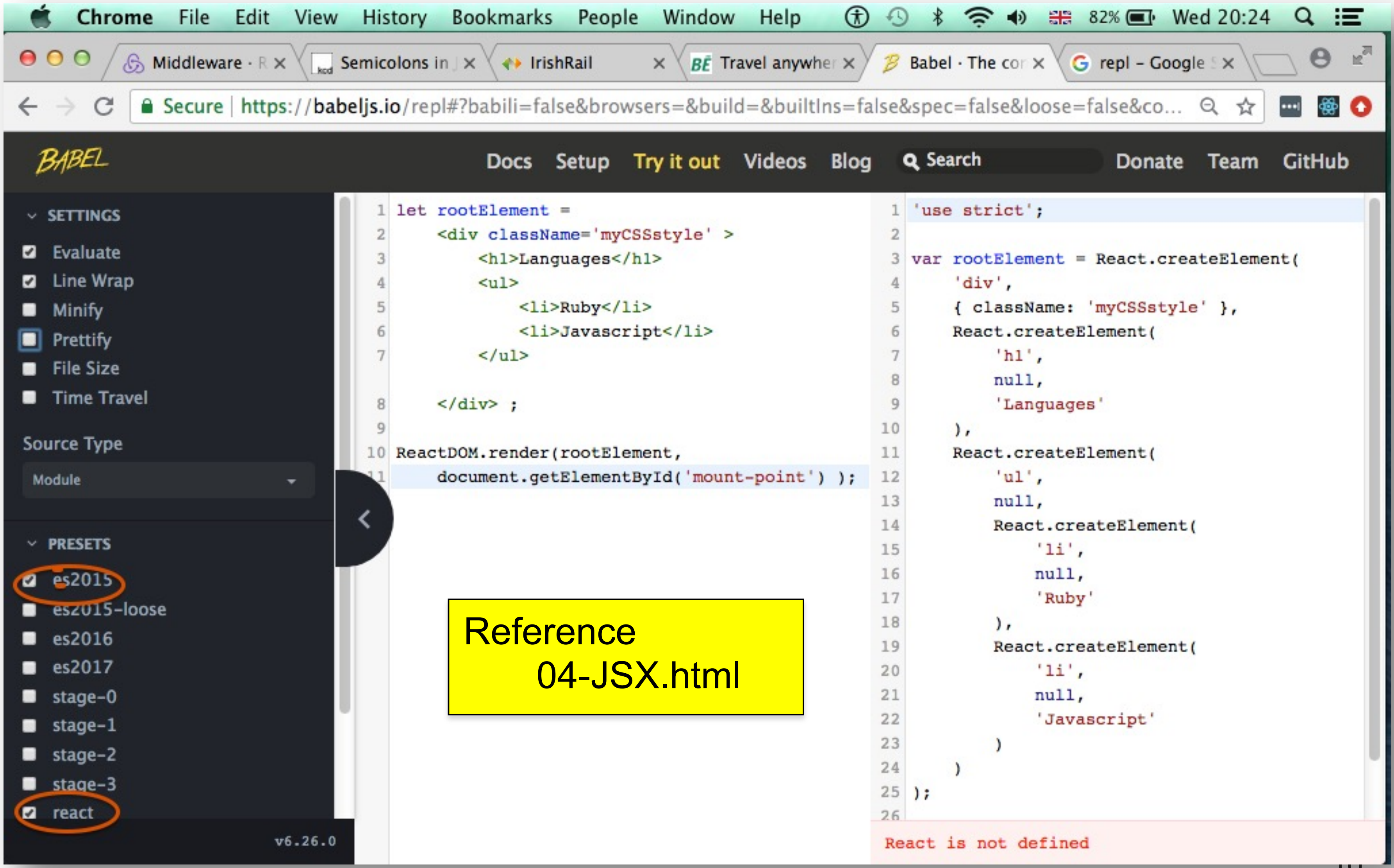
***Imperative programming** is a programming paradigm that uses statements that change a program's state*

***Declarative programming** is a programming paradigm … that expresses the logic of a computation without describing its control flow.*

UI description implementation
(the declarative way)

- **JSX – JavaScript extension syntax.**
- **Declarative syntax 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
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 **to 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. **Using specially instrumented web server during development mode - the Webpack library..**
  3. **Using** bundler **tools as part the build process before deployment – Webpack again.**

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(<ComponentX />, ...)

- **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. **A web server.**

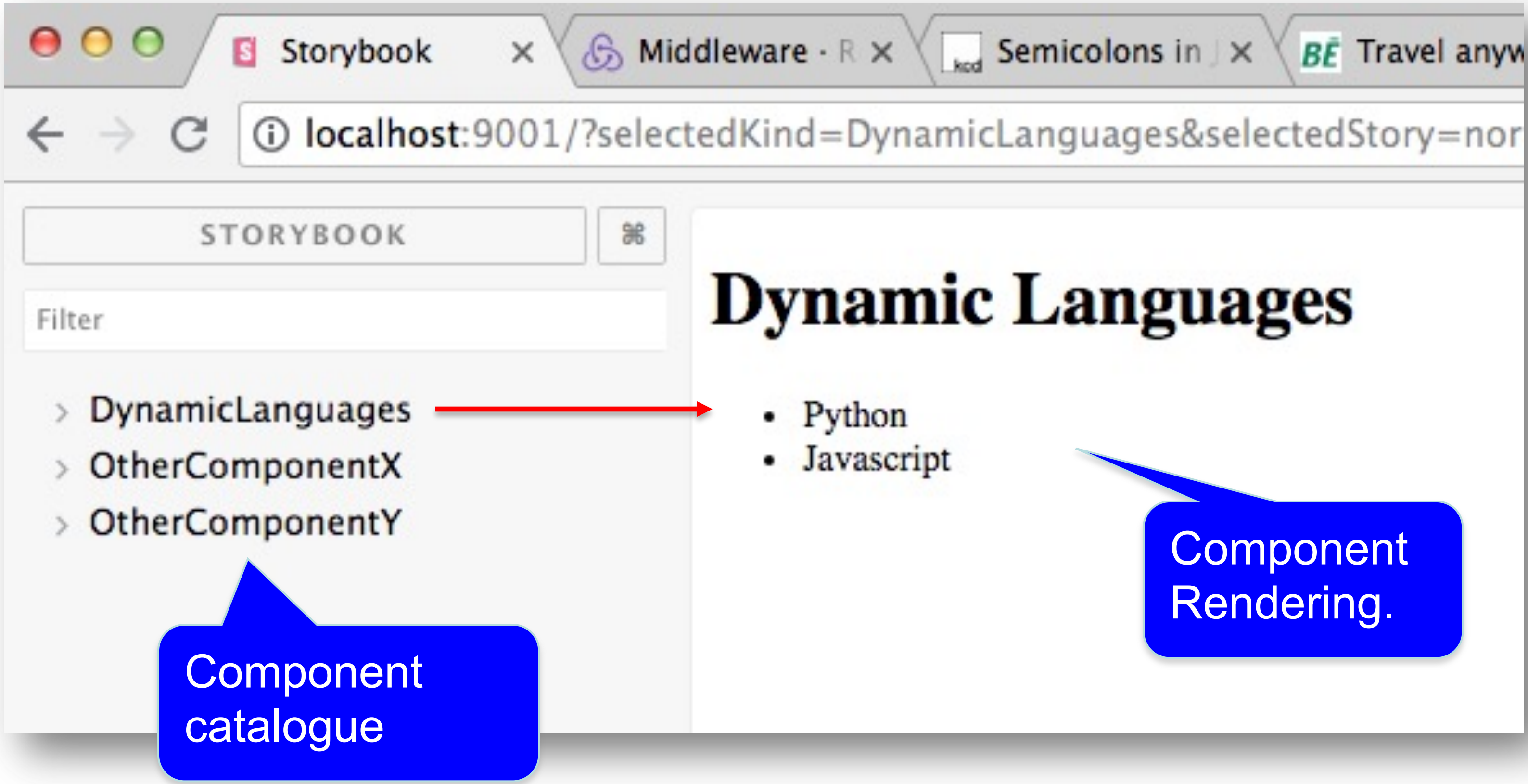
```
$ ./node_modules/.bin/start-storybook -p 6006 -c ./.storybook
```

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

2. **Web browser user interface.**

STORYBOOK

- **Storybook User interface.**



STORYBOOK

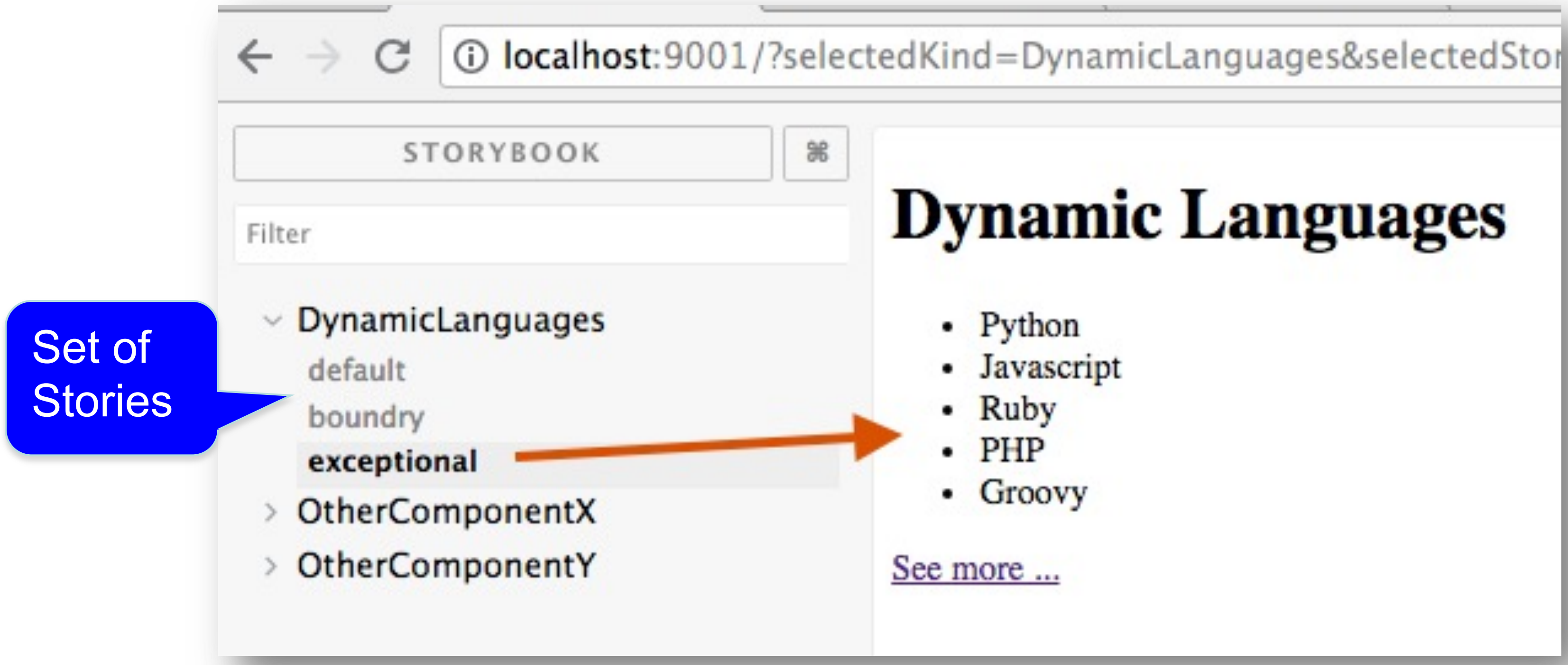
- **What is a Story?**

- **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

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



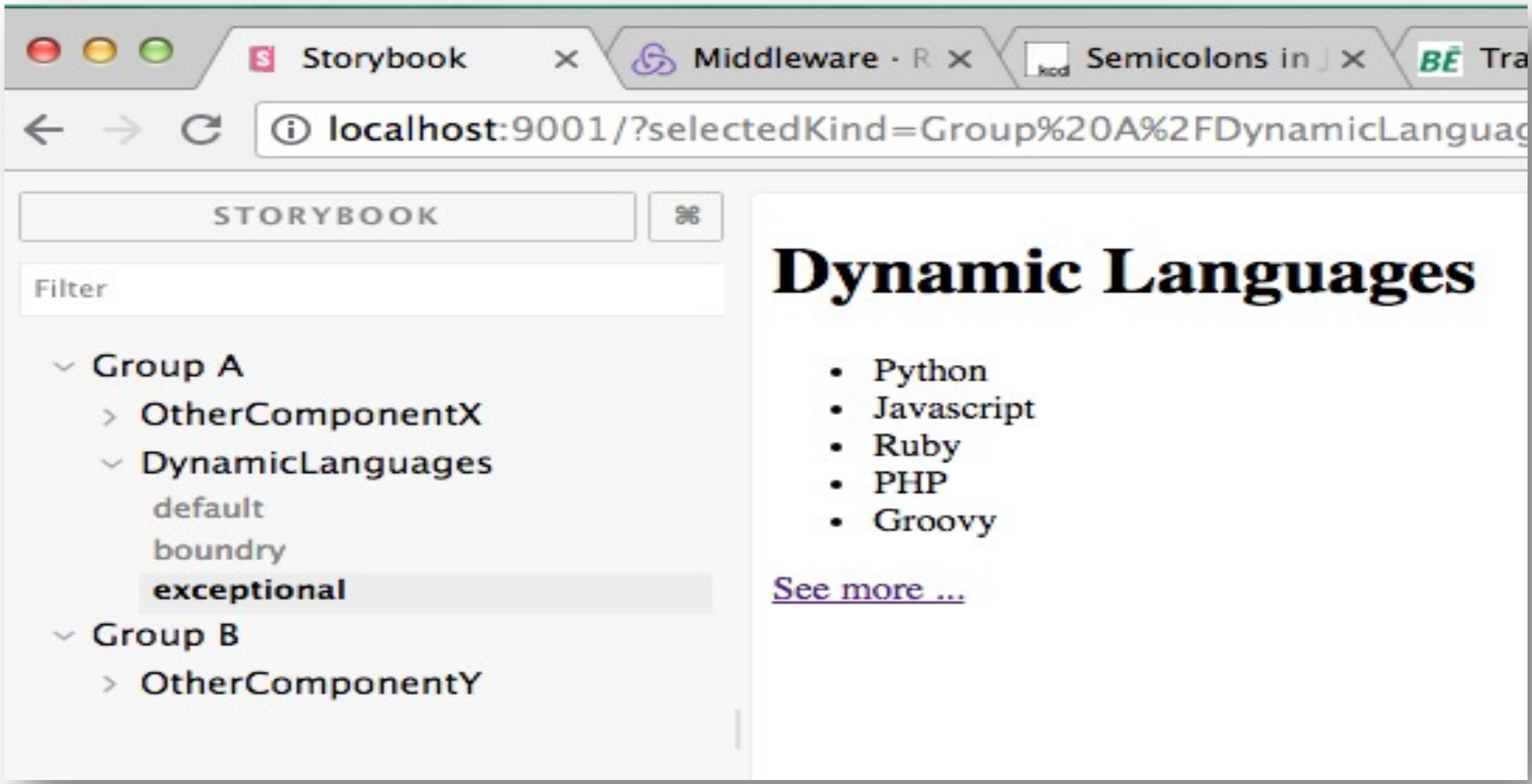
Dynamic Languages

- Python
- Javascript
- Ruby
- PHP
- Groovy

See more ...

STORYBOOK

- **Define component** groups when **component library is large.**
  - **helps others team members with searching**.



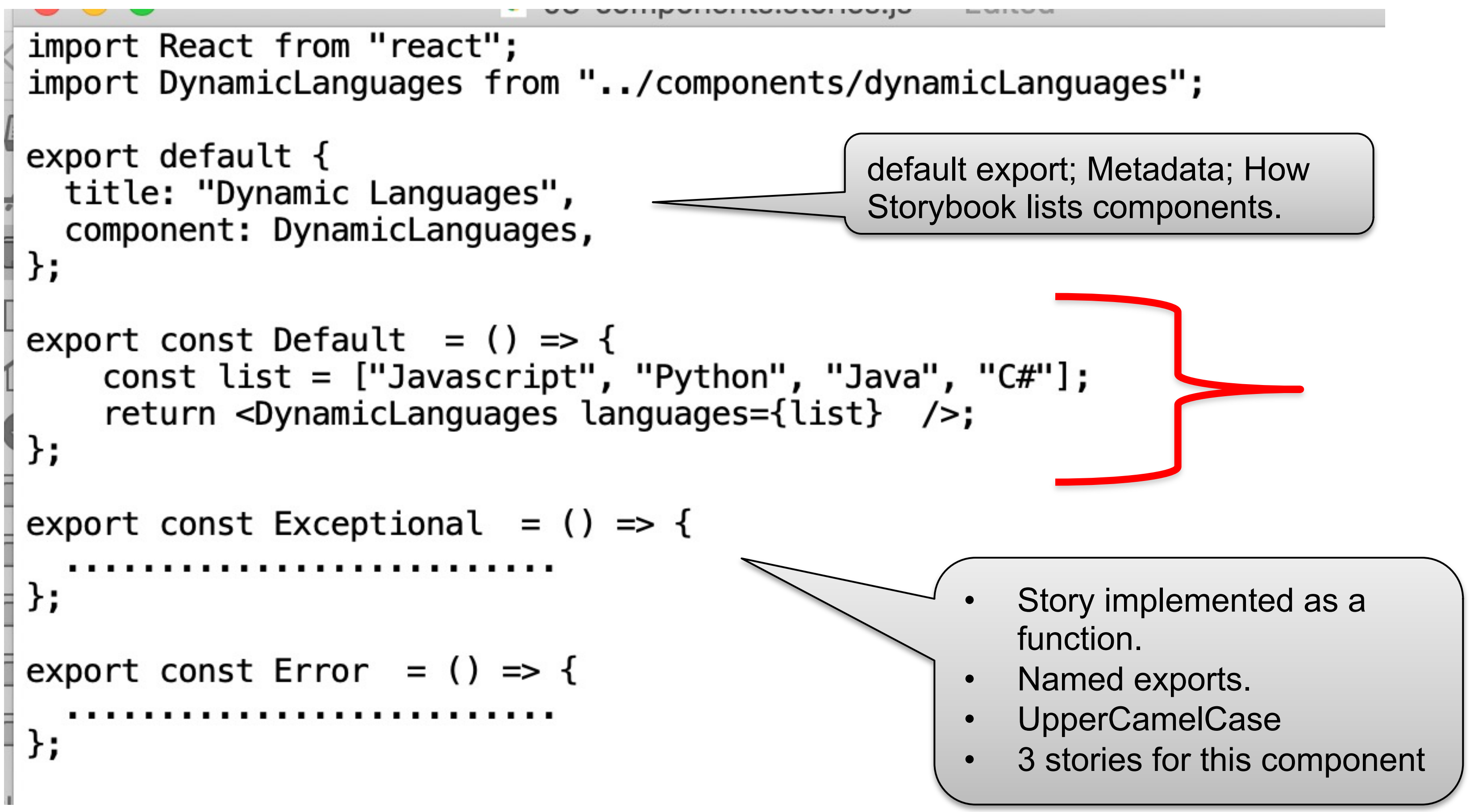
Dynamic Languages

- Python
- Javascript
- Ruby
- PHP
- Groovy

See more ...

Writing stories

- .stories.js **file extension (convention)**
- **1 Stories file per coponent**



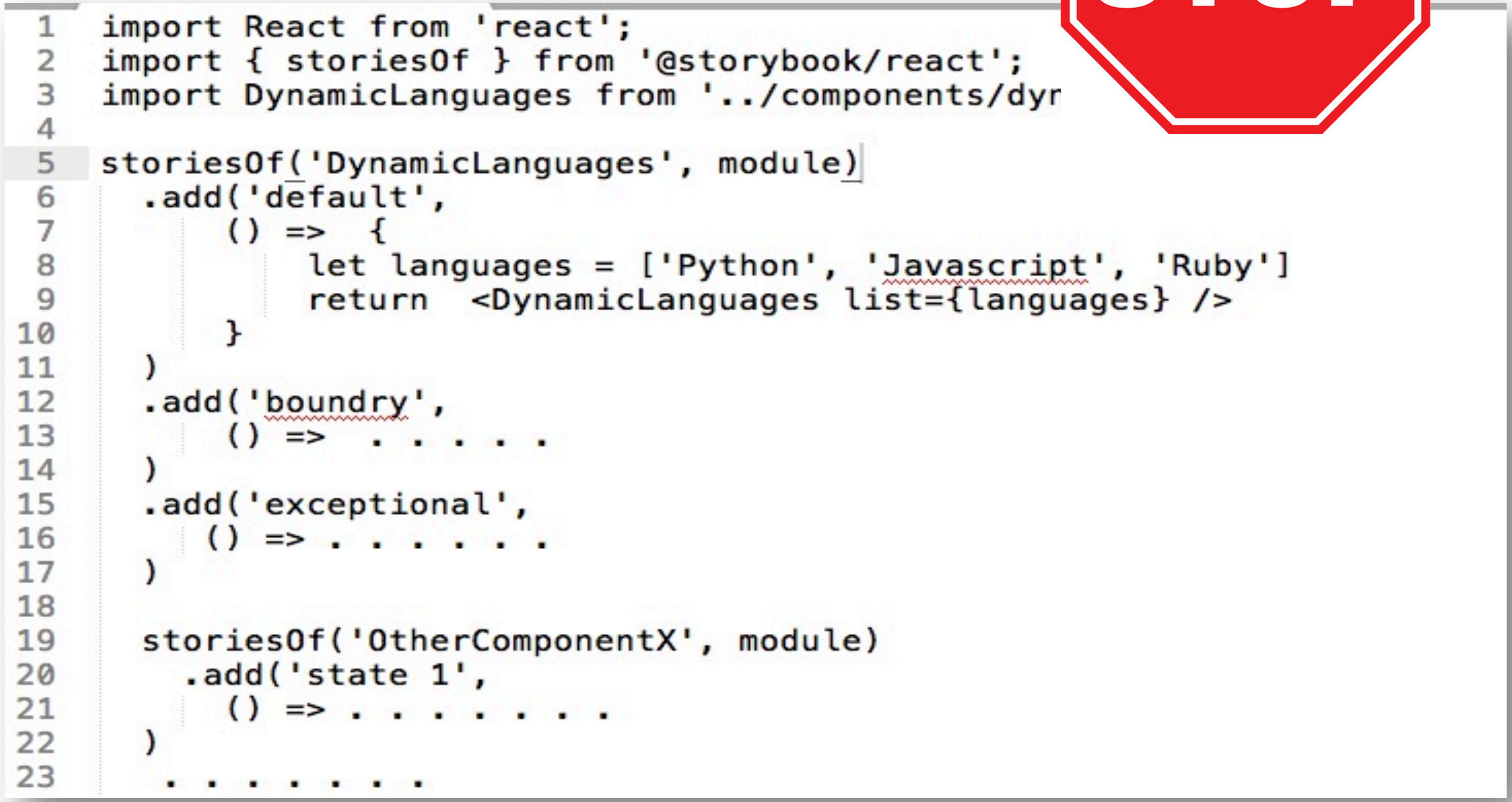
```
import React from "react";
import DynamicLanguages from "../components/dynamicLanguages";

export default {
  title: "Dynamic Languages",
  component: DynamicLanguages,
};

export const Default  = () => {
    const list = ["Javascript", "Python", "Java", "C#"];
    return <DynamicLanguages languages={list}  />;
};

export const Exceptional  = () => {
  .........................
};

export const Error  = () => {
  .........................
};
```

default export; Metadata; How Storybook lists components.

- Story implemented as a function.
- Named exports.
- UpperCamelCase
- 3 stories for this component

Writing stories

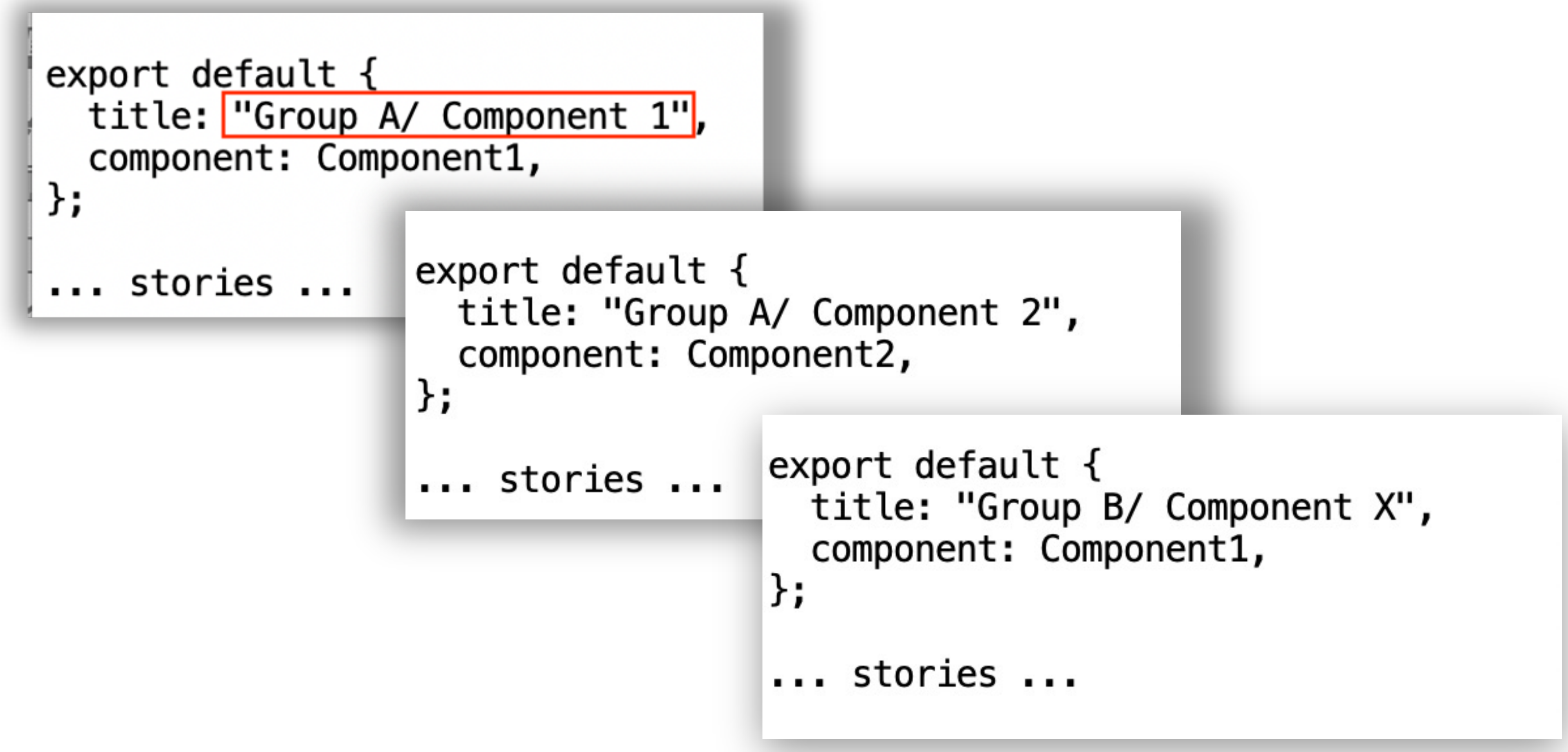
- **Fluent-style syntax for writing stories.**
  - **Method chaining programming style.**



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

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

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

Grouping stories.

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



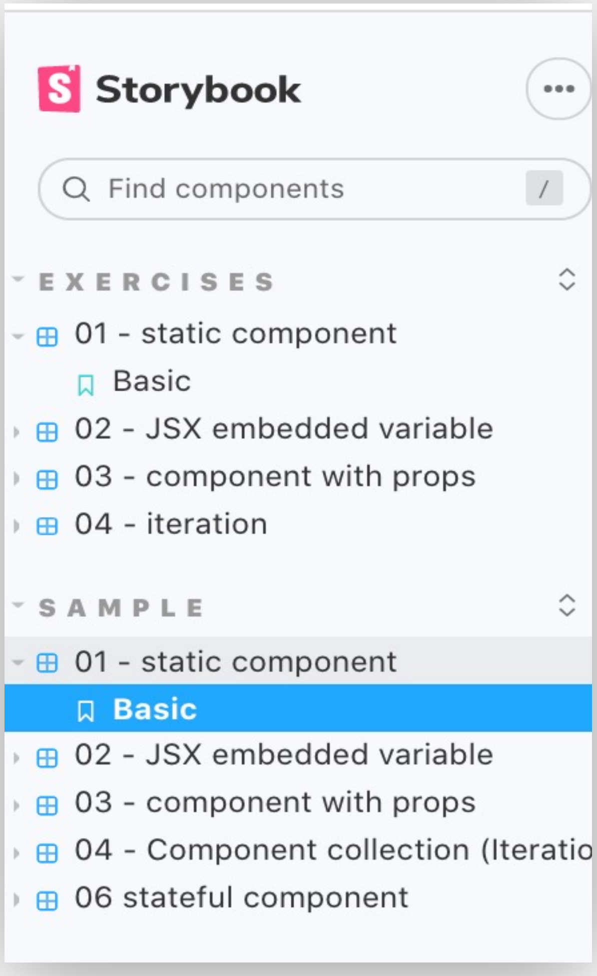
```
export default {
  title: "Group A/ Component 1",
  component: Component1,
};

... stories ...
```

```
export default {
  title: "Group A/ Component 2",
  component: Component2,
};

... stories ...
```

```
export default {
  title: "Group B/ Component X",
  component: Component1,
};

... stories ...
```

… back to components . . .

Demo Samples

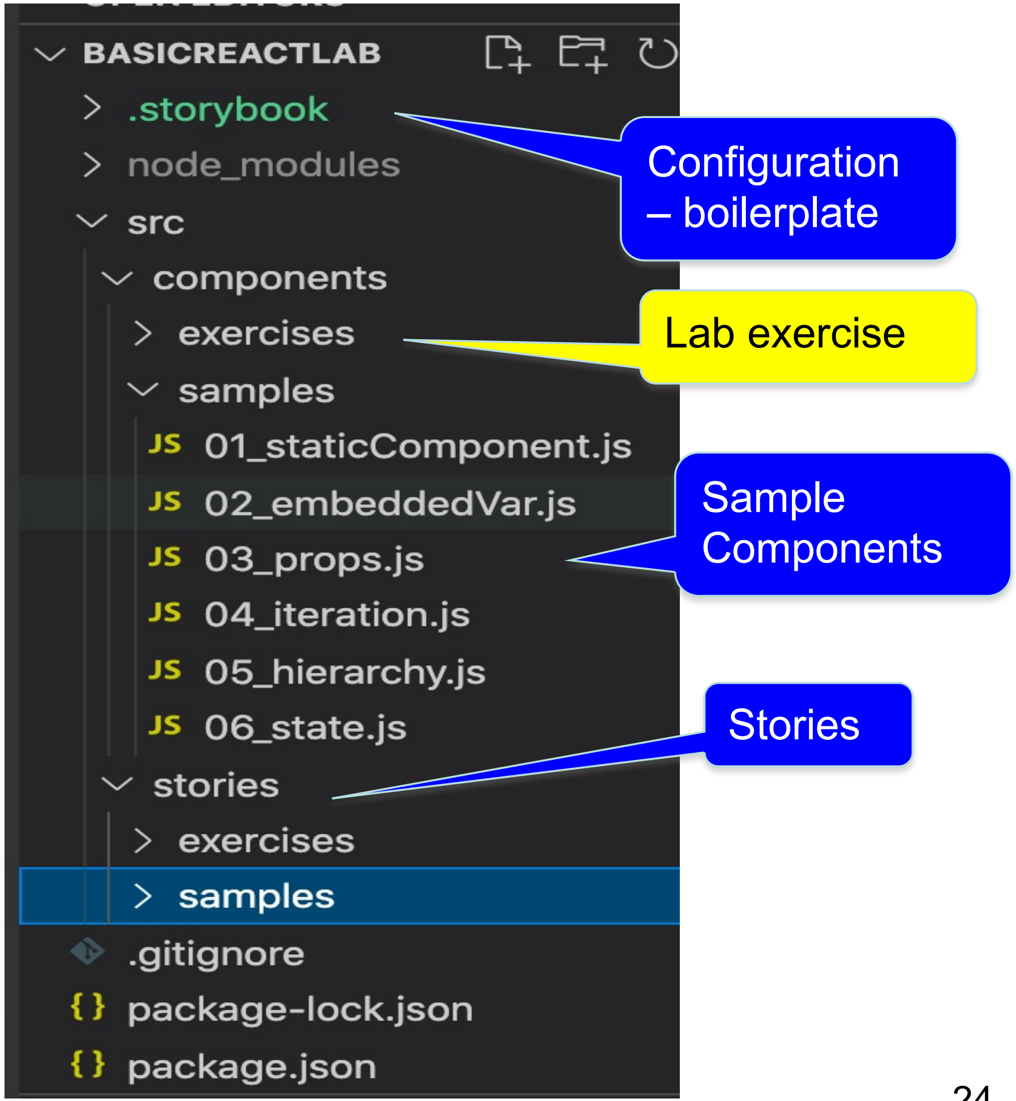


Storybook

Find components /

EXERCISES
- 01 - static component
  - Basic
- 02 - JSX embedded variable
- 03 - component with props
- 04 - iteration

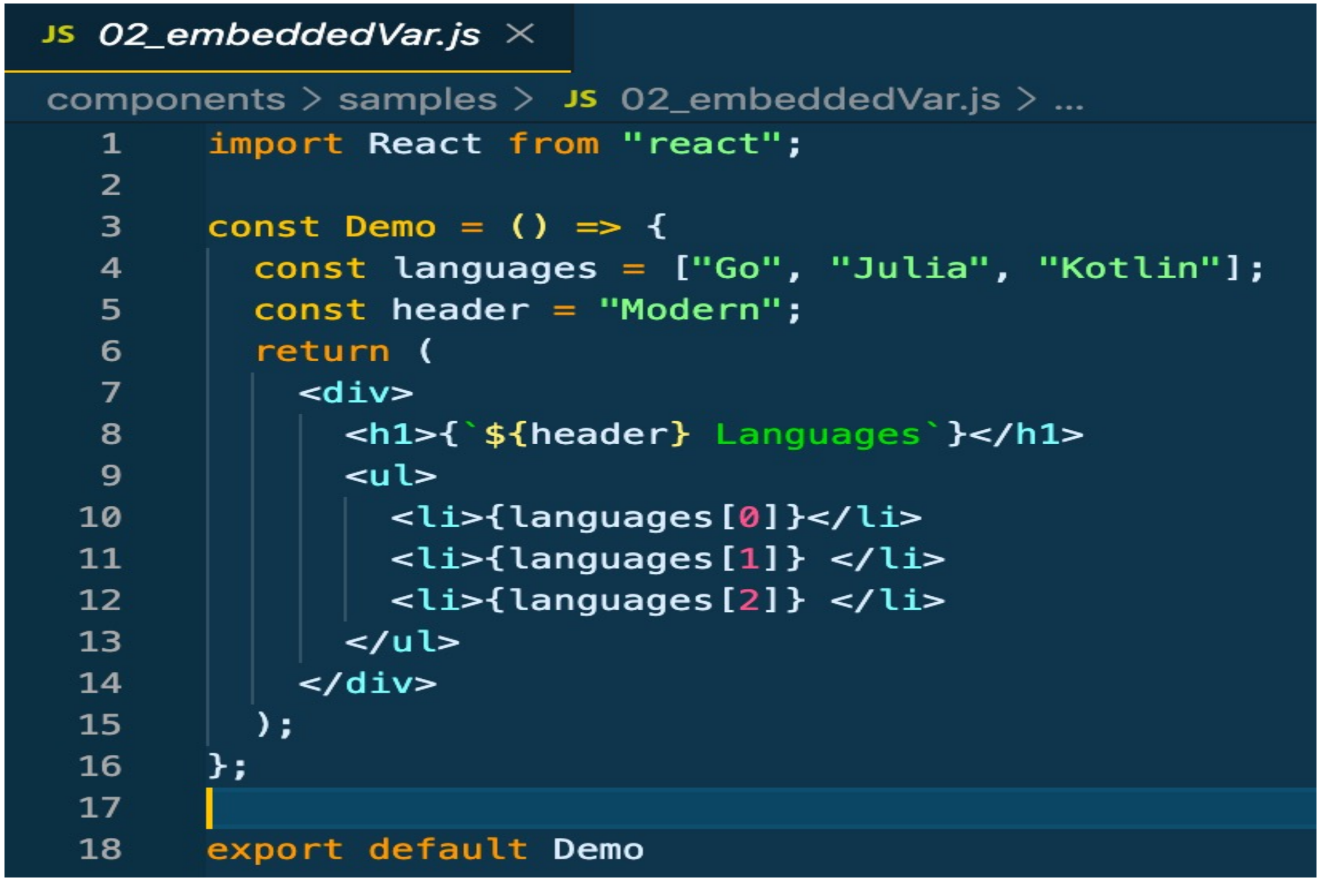
SAMPLE
- 01 - static component
  - Basic
- 02 - JSX embedded variable
- 03 - component with props
- 04 - Component collection (Iteratio
- 06 stateful component



```
BASICREACTLAB
  > .storybook
  > node_modules
  v src
    v components
      > exercises
      v samples
        JS 01_staticComponent.js
        JS 02_embeddedVar.js
        JS 03_props.js
        JS 04_iteration.js
        JS 05_hierarchy.js
        JS 06_state.js
    v stories
      > exercises
      > samples
  .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



02_embeddedVar.js

components > samples > 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.

- **We 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).**

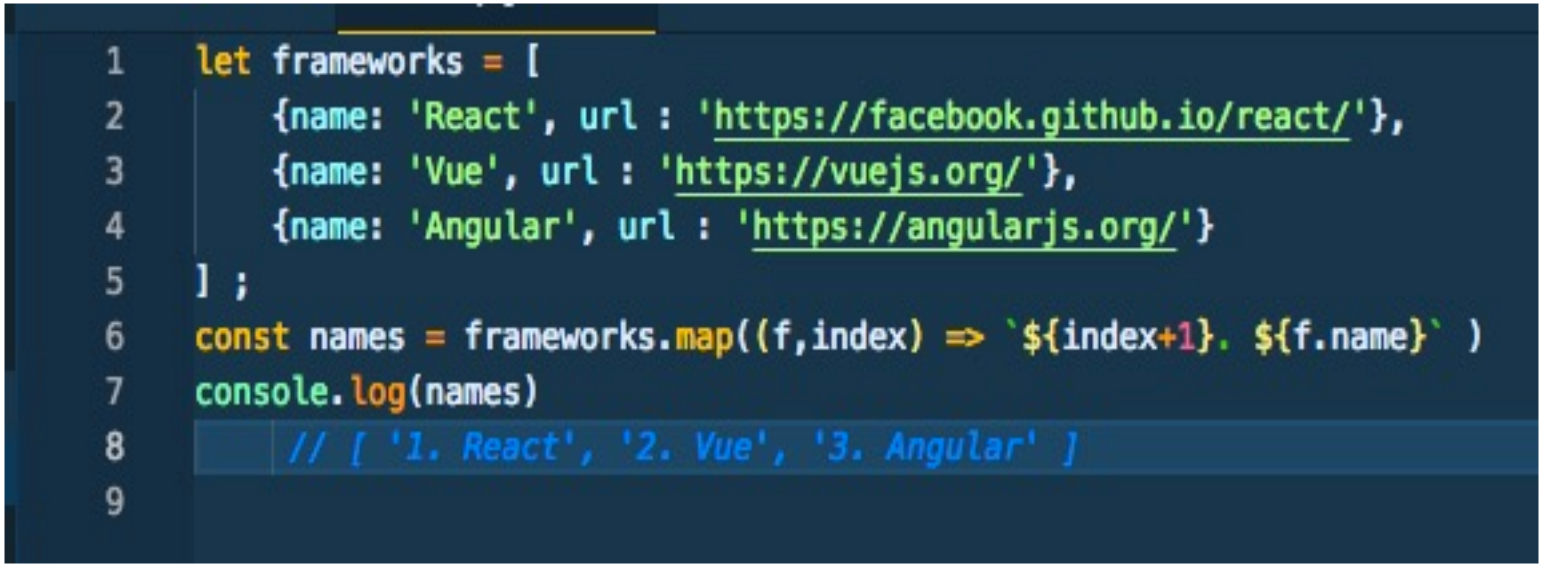
Aside – Some JS features

- **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 features

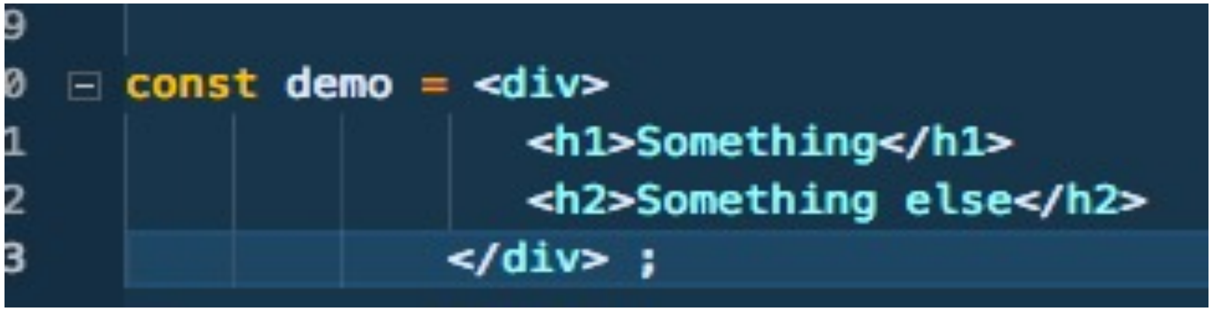
- **The Array** map **method – returns a new array based on applying the function argument to each element of the source array.**



```
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' ]

```

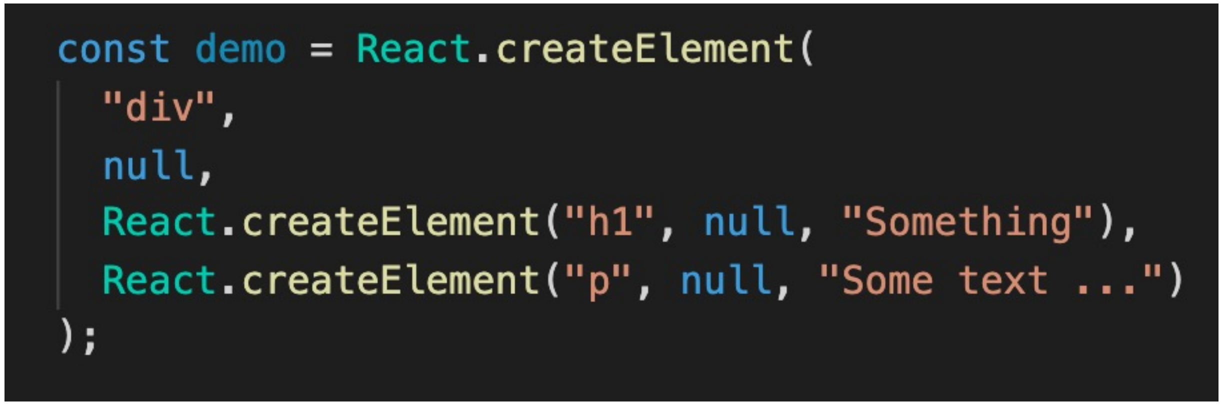
Aside – Some JS features.

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



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

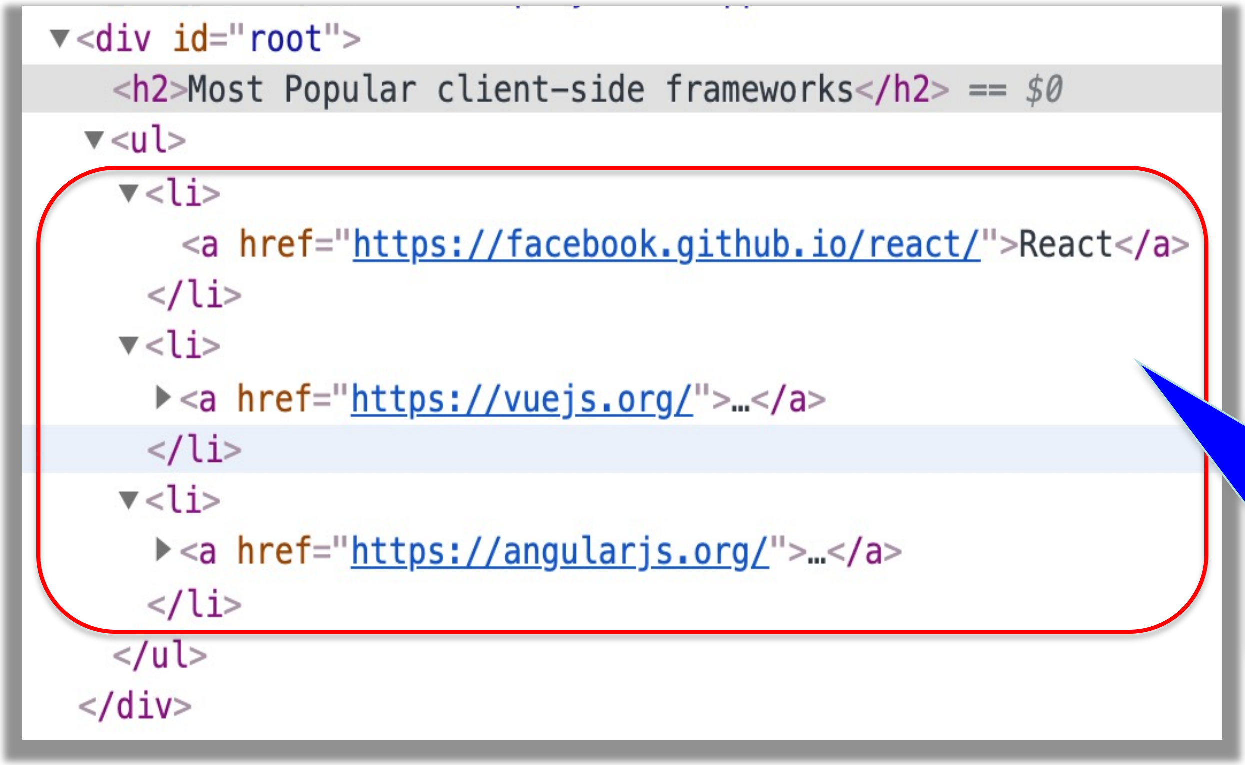
- **Why?**



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

Component collection - Iteration

- **Use case: We want to generate an array of (similar) component from a data array.**
- **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 <MyComponent prop1={…..} prop2={……} /> ;
  2. return (

```
        <div>
          <h1>{this.props.type}</h1>
          <MyComponent prop1={…..} prop2={……} />
          <p>
              . . . . . .
          </p>
        </div>
    ) ;
```

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

Component return value.

- **Must return only ONE element.**
- **Error Examples:**
  -
    ```
    return (
            <h1>{this.props.type}</h1>
            <MyComponent prop1={…..} prop2={……} />
            <p>
                . . . . . . .
            </p>
    ) ;
    ```
  - **Error** – ‘Adjacent JSX elements must be wrapped in an enclosing tag’
  - **Solution: Wrap elements in a <div> tag.**

Component return value.

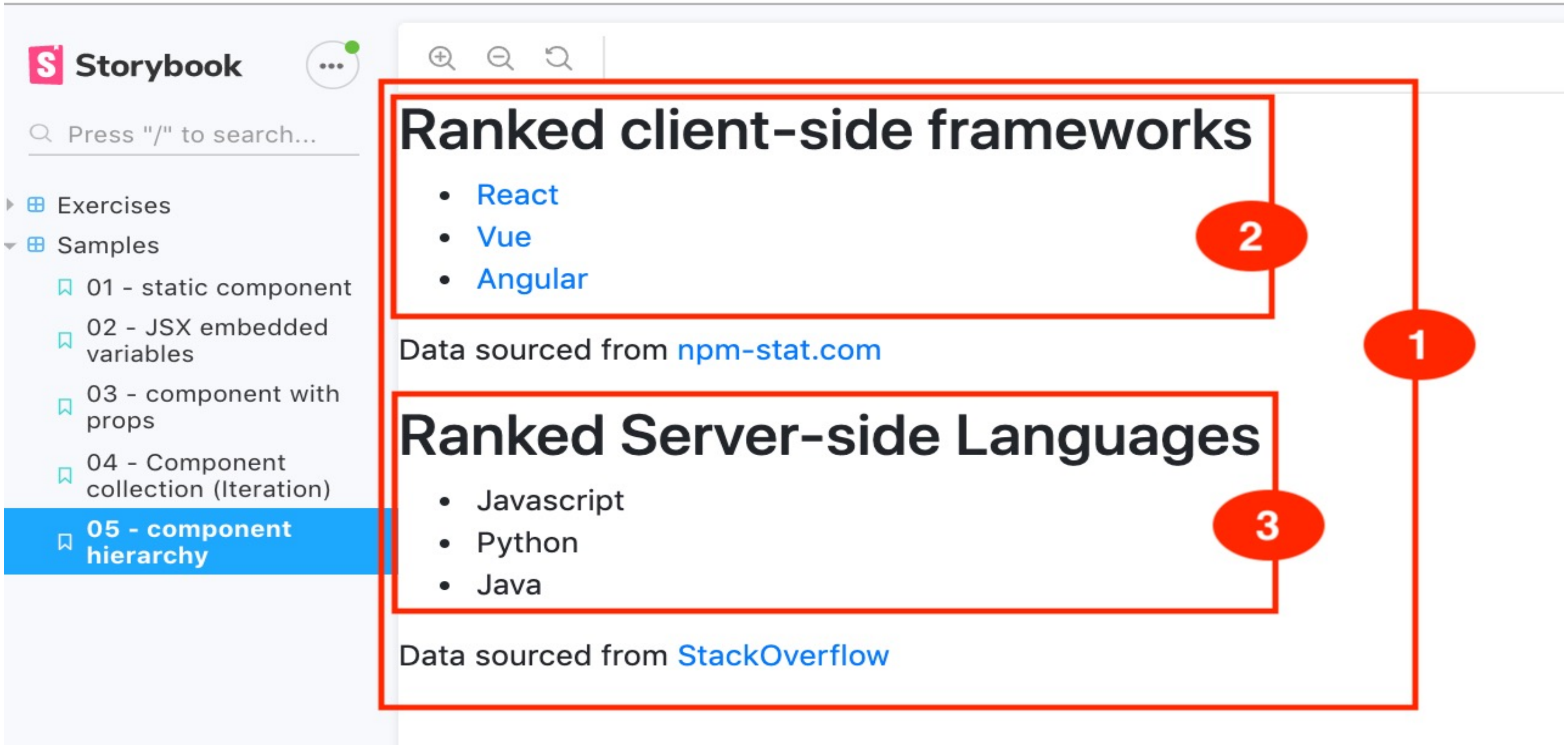
- **Old solution:**
```
return (
    <div>
        <h1> ……</h1>
        <MyComponent …… />
        <p> …….. </p>
    </div>
);
```
- **Adds unnecessary depth to DOM → effects performance.**

- **Alternative solution:**
```
return (
    <>
        <h1> ……</h1>
        <MyComponent …… />
        <p> …….. </p>
    </>
);
```
- **<> </> – special React element, termed** Fragment**.**
  - **No DOM presence.**

Component ***Hierarchy***.

All React application are 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

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).**