

ReactJS (v18+).

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.**

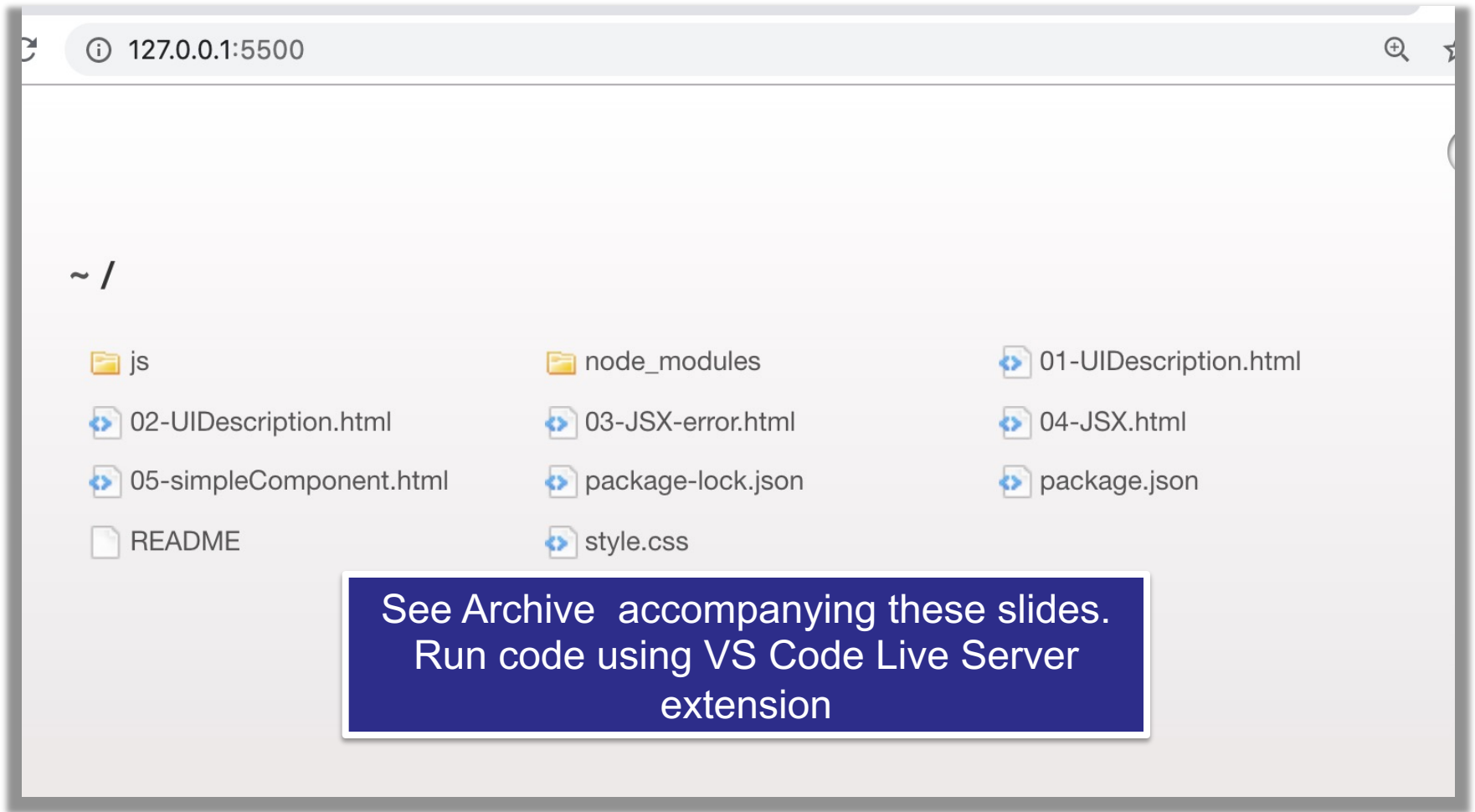
Creating the UI description.

(Vanilla React)

- `React.createElement()` – **create a HTML element.**
- `ReactDOM.createRoot()` – **which existing DOM node to attach the created element.**
- `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.**
- `ReactDOM.createRoot()` **arguments:**
 1. **DOM node on which to mount a new element.**

Code Demos.

(See lecture archive)



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.

The screenshot shows the Babel REPL interface in a Chrome browser. The left sidebar contains settings and presets. The main area displays the transpiled code for a JSX snippet.

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

v6.26.0

Input Code (Left):

```
1 let rootElement =
2   <div className='myCSSstyle' >
3     <h1>Languages</h1>
4     <ul>
5       <li>Ruby</li>
6       <li>Javascript</li>
7     </ul>
8   </div> ;
9
10 ReactDOM.render(rootElement,
11   document.getElementById('mount-point') );
```

Output Code (Right):

```
1 'use strict';
2
3 var rootElement = React.createElement(
4   'div',
5   { className: 'myCSSstyle' },
6   React.createElement(
7     'h1',
8     null,
9     'Languages'
10  ),
11  React.createElement(
12    'ul',
13    null,
14    React.createElement(
15      'li',
16      null,
17      'Ruby'
18    ),
19    React.createElement(
20      'li',
21      null,
22      'Javascript'
23    )
24  )
25 );
```

Error Message (Bottom Right): 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.**
 - **The Vite library.**
- **How?**
 - 1. Manually, via REPL or command line.**
 - **When experimenting only.**
 - 2. Using an instrumented web server – Vite library instrumentation.**
 - **Ideal for development.**
 - 3. Using bundler tools as part of the build process – Vite again.**
 - **Production standard.**

React Components.

- **We develop COMPONENTS.**
 - **A JS function that returns a UI description, i.e. JSX.**
- **We reference a component like a HTML tag.**

e.g.

```
const rootElement =
```

```
  ReactDOM.createRoot(document.getElementById("mount-point"));
```

```
rootElement.render( <DynamicLanguages> );
```

- **Reference** 05-simpleComponent.html

React Developer tools.

- **Vite - Features:**
 - **Scaffolding/Generator.**
 - **Development web server: auto-transpilation on file change + live reloading (HMR – Hot Module Replacement).**
 - **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.**



- **Installation:**

\$ npm install @storybook/react

- **The tool has two aspects:**

1. **A web server.**

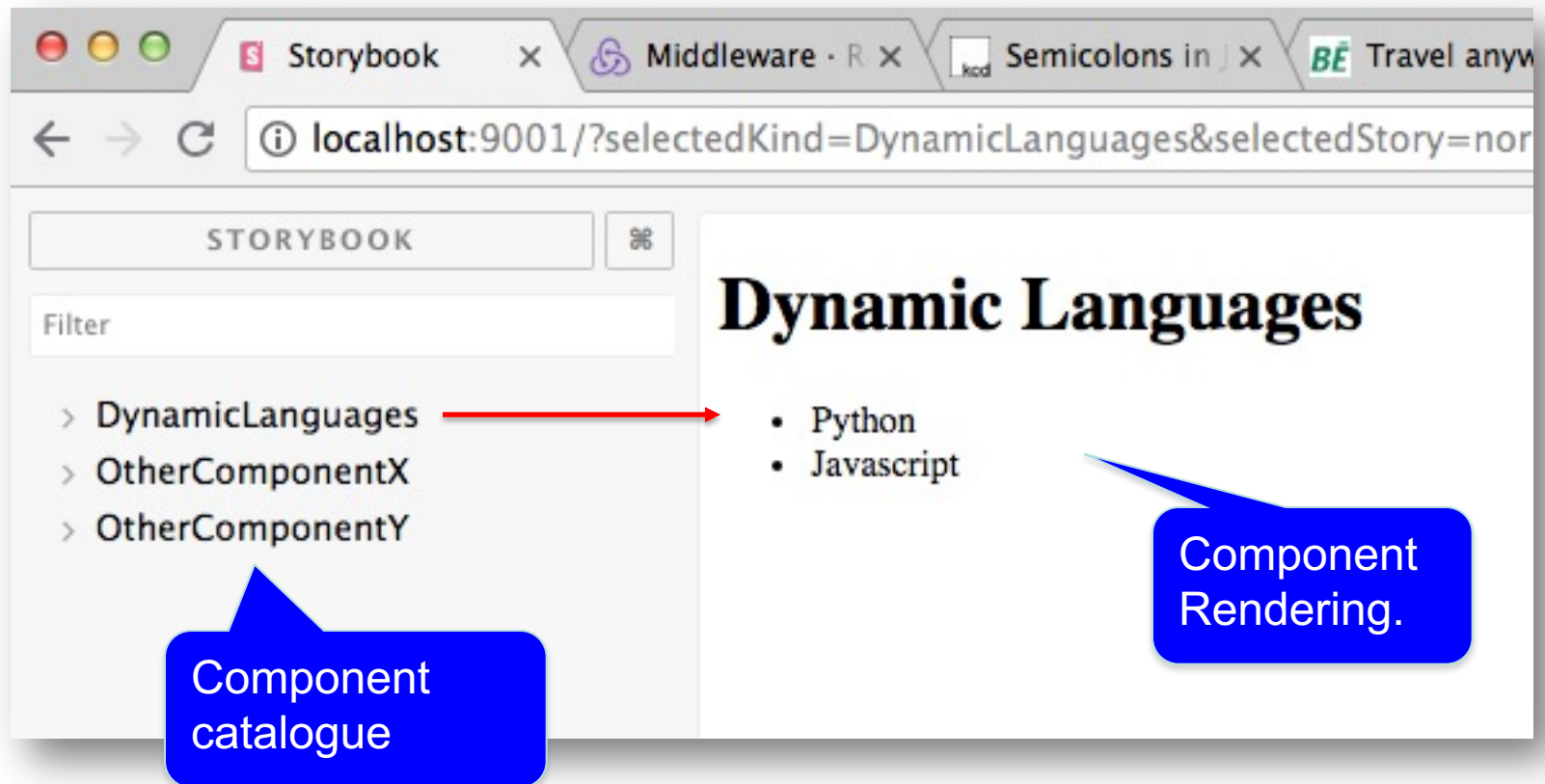
\$./node_modules/.bin/***start-storybook -p 6006 -c ./storybook***

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

2. **Web browser user interface.**



- **Storybook User interface.**

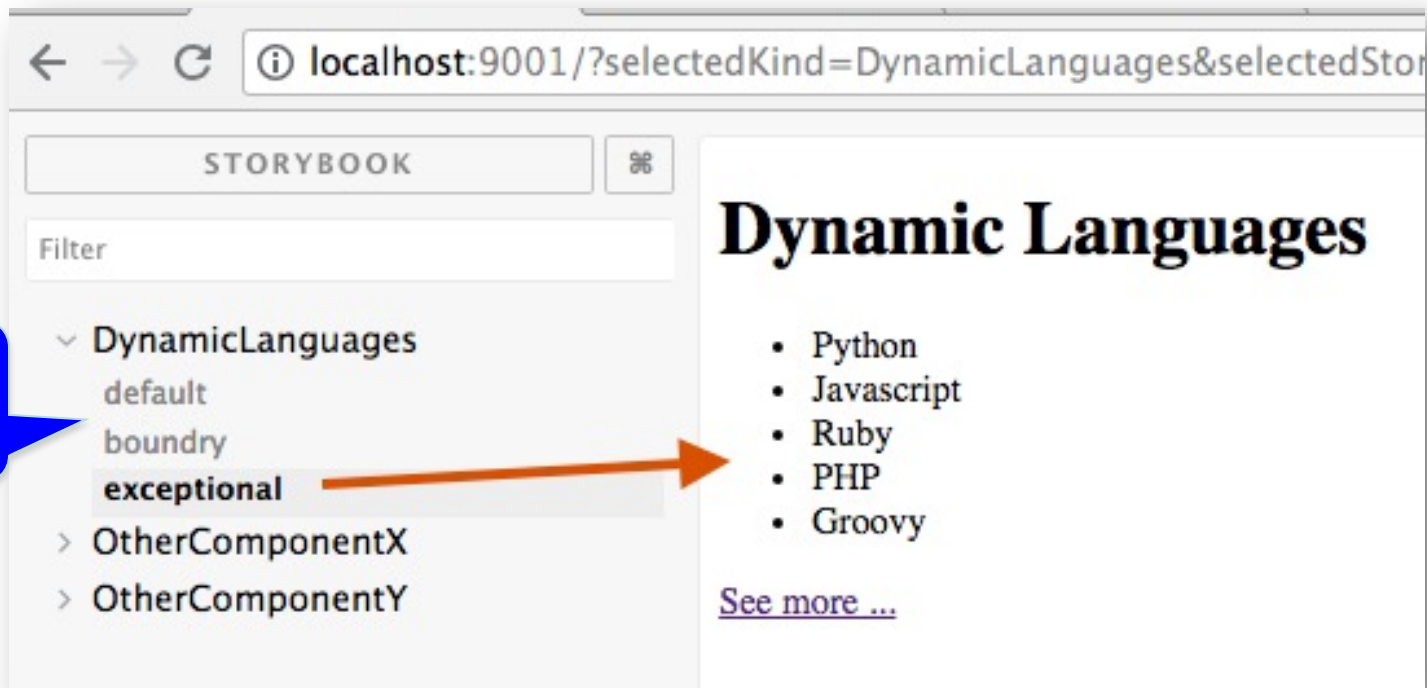




- **What is a Story?**
- **A component may have several STATES → State affects how it renders.**
 - **Each state case termed a STORY.**
 - **Stories are a design consideration.**
- **EX.: DynamicLanguages component.**
 - **States might be:**
 - **Default – 5 or less languages → Render full list**
 - **Boundary – empty list → Render ‘No languages’ message**
 - **Exceptional – More than 5 languages → Render first 5 and a ‘See More...’ link to display next 5.**

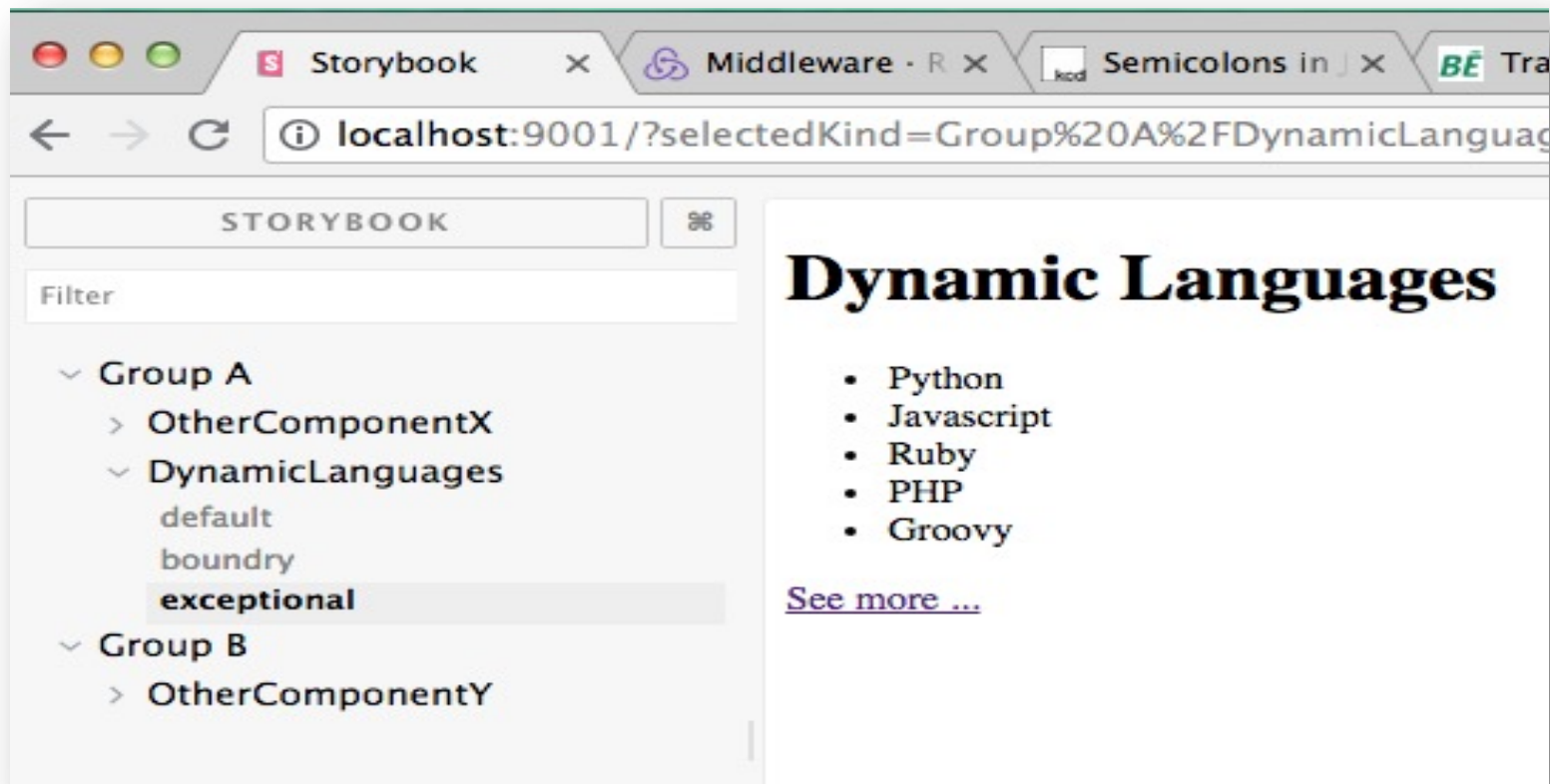


- List a component's states/stories under its name:





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



Writing stories

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

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

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

default export; Metadata; How Storybook lists components.

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

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

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

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

Grouping stories.

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

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

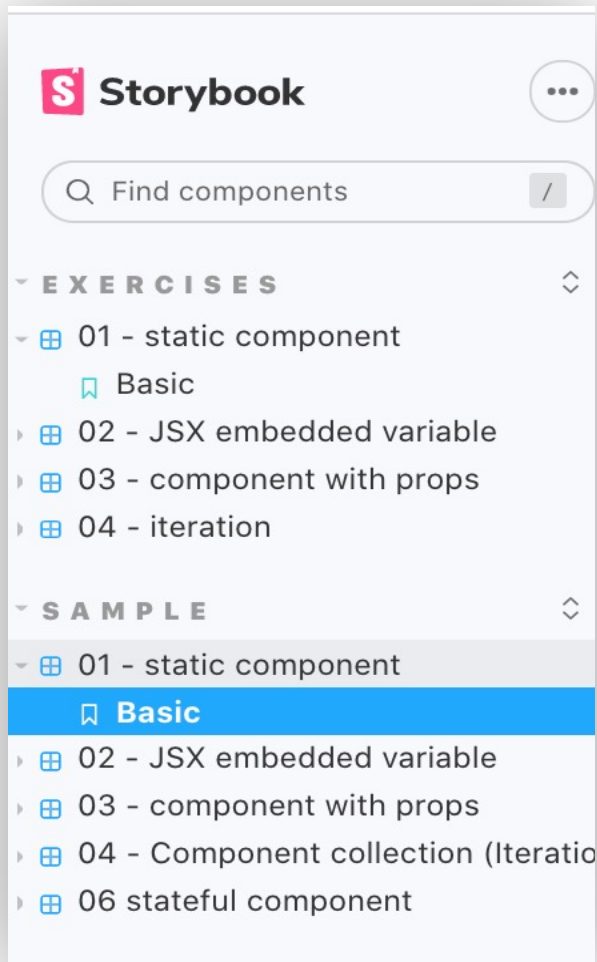
```
export default {  
  title: "Group A/ Component 2",  
  component: Component2,  
};  
  
... stories ...
```

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

... back to components . . .

Demo Samples

(See lab exercise)



JSX - embedded variables.

- Use { } to dereference variable embedded in JSX.
 - Curly braces can contain any valid JS expression.
- **Reference** src/components/samples/02_embeddedVariables.js

```
JS 02_embeddedVar.js ×
components > samples > JS 02_embeddedVar.js > ...
1  import React from "react";
2
3  const Demo = () => {
4    const languages = ["Go", "Julia", "Kotlin"];
5    const header = "Modern";
6    return (
7      <div>
8        <h1>`${header} Languages`</h1>
9        <ul>
10         <li>{languages[0]}</li>
11         <li>{languages[1]} </li>
12         <li>{languages[2]} </li>
13       </ul>
14     </div>
15   );
16 };
17
18 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** `src/components/samples/03_props.js` (and related story).

Aside.

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

```
9  
0  - const demo = <div>  
1      <h1>Something</h1>  
2      <h2>Something else</h2>  
3      </div> ;
```

- **Why?**

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

Component collection - Iteration

- **Use case:** Generate an array of (similar) component from a data array.
- **Reference** src/components/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 → affects 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.** `src/components/samples/05_hierarchy.js`.

Storybook

Press "/" to search...

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

Ranked client-side frameworks

- React
- Vue
- Angular

Data sourced from npm-stat.com

Ranked Server-side Languages

- Javascript
- Python
- Java

Data sourced from [StackOverflow](https://stackoverflow.com)

1

2

3

React version 18

- Released March 2022.
- One significant breaking change.

```
You, 1 second ago | 1 author (You)
1  ✓ import React from 'react'
2    import ReactDOM from 'react-dom'
3    import App from './App'
4    You, 5 months ago • Completed app but using Bootstrap
5    ReactDOM.render(<App />, document.getElementById('root'));
```

```
You, 1 second ago | 1 author (You)
1  import React from 'react'
2  import { createRoot } from 'react-dom/client'
3  import App from './App'
4  You, 3 weeks ago • Initialize project using Create React App
5  const rootEleent = createRoot( document.getElementById('root' ) )
6  rootEleent.render(<App /> );
```

React 18

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 UI.

A 3rd party component library to build
high quality digital UIs

Material Design.

- Material (Design) **is a design system created by Google to help teams build** high-quality digital experiences **for Android, iOS, and web.**
- **A visual language 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 digital 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 an application 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 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.

- **MUI is a React component library based on the Material Design system.**
- **Its React components include: <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>
```

- **CSS-in-JS**

```
const myHeader = {  
  backgroundColor: "lightblue",  
  padding: "10px"  
};
```

```
.....  
<header style={myHeader}>  
..... </header>
```



Must be
CamelCase

Many Alternatives

- **Libraries:**
 - **Tailwind**
 - **Chakra**
 - **React-Bootstrap**
- **Design Systems:**
 - **ANT**
- **CSS models:**
 - **CSS modules**
 - **BEM**