[Lesson 2]

[DAN.IT EDUCATION]

[Что мы рассмотрим:]

- Вложенные компоненты
- Структура проекта и правила именования
- Что такое props и как с ними работать
- Изменение родительского state из дочернего компонента
- Зачем нужны keys
- Что такое refs

[DAN.IT EDUCATION]

[Props]

У каждого компонента могут быть свойства. Они хранятся в this.props, и передаются компоненту как атрибуты.

```
let value1 = {name: Garry, surname: Potter};
<MyComponent data={value1} someOtherProp={[1,2,3,4,5]} />
```

В свойство можно передать любой javascript примитив, объект, переменную и даже выражение. Значение свойства должно быть взято в фигурные скобки.
Значения доступны через **this.props.ИМЯ_СВОЙСТВА**
В нашем случае, мы получим:

```
this.props.data - объект {name: Garry, surname: Potter}
this.props.someOtherProp - массив [1,2,3,4,5]
```

[DAN.IT] EDUCATION

Вложенные компоненты

```
import React from 'react'

const FancyBorder = (props) => {
     const { children, color } = props;
    return (
      <div className={'fancyBorder fancyBorder-' + color}>
        {children}
      </div>
    );
}
```

Вложенные компоненты

```
import React from 'react'
import FancyBorder from '../FancyBorder'

const WelcomeDialog = (props) => {
    return (
      <FancyBorder color="blue">
        <h1 className="dialog-title">
          Welcome
        </h1>
        <p className="dialog-message">
          Thank you for visiting our spacecraft!
        </p>
      </FancyBorder>
    );
}
```

Структура проекта и правила именования

Имя файла компонента должно быть в PascalCase

Имена компонентов должны именоваться в виде TabSwitcher, а не tabSwitcher, tab-switcher и т. д. Называть другие типы файлов в такой нотации не нужно, поскольку CamelCase сигнализирует нам, что файл является компонентом React.

Структура проекта и правила именования

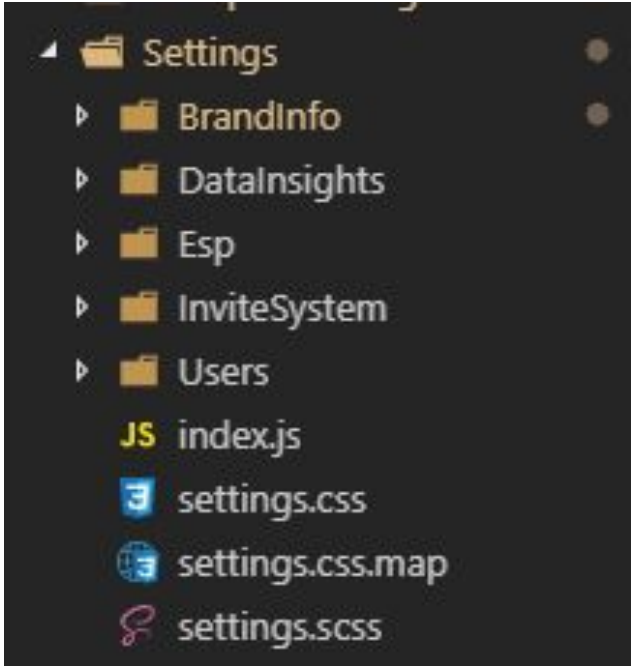
Все компоненты должны находиться в каталоге components

Держите компоненты внутри выделенного каталога компонентов. Храните каждый компонент в отдельной папке соответствующей названию компонента. Если вам нужен файл стилей – храните его рядом с реализацией самого компонента, тут же будут хранится и тесты.

Главное преимущество в том, что компоненты не спрятаны где-то глубоко внутри и все составные части (стили, тесты и т.д.) находятся в одном месте из-за чего вы не будете "бегать" по всему проекту в поисках всех связанных файлов.

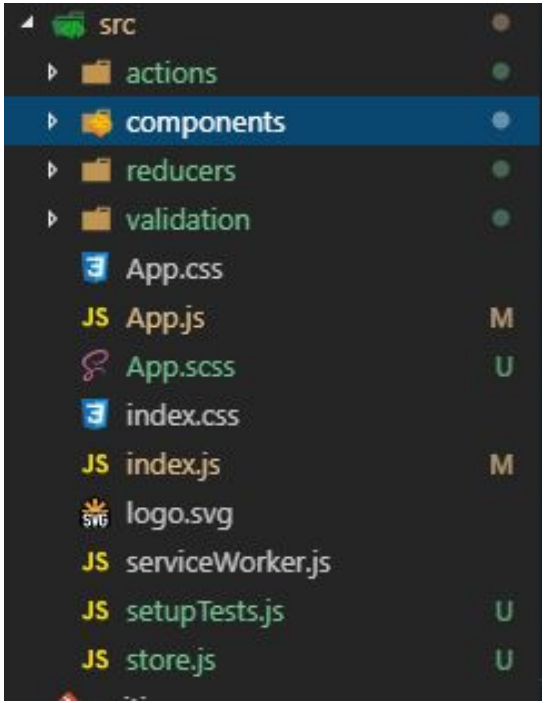
Структура проекта и правила именования

Если компонент является составной частью другого компонента и более нигде не используется и/или функционально завязан на родительском компоненте, то такой компонент помещается в родительский, то есть:



Структура проекта и правила именования

Обычно, структура реального проекта выглядит примерно так:



Изменение родительского state из дочернего компонента

./src/App.js

```
...
state = {
     showModal: false
}

showToggle = () => {
     this.setState({showModal: !this.state.showModal})
}
...
render(){
     ...
     <Header showModalWindow={this.showToggle} />
     <Dashboard modalWindowStatus={this.state.showModal}>
     ...
}
```

Изменение родительского state из дочернего компонента

./src/components/Header/index.js

```
...
render(){
     ...
     <Button onClick={this.props.showToggle} />
     ...
}
```

./src/components/Dashboard/index.js

```
...
render(){
     ...
     {this.props.modalWindowStatus ? <Modal /> : null}
     ...
}
```

[keys]

./src/components/MailList/index.js

```
import {mails} from '../../dummy.js'

export default function MailList({mails}){
     const mailsElements = mails.map((mail) => <li>{mail.from} - {mail.subject}</li>);
     return(<ul>mailsElements</ul>);
}
```

./src/dummy.js

```
export const mailList = [
                 {id: 1,
                  from: 'bender@futurama.com',
                  subject: 'Kill all humans'
                 },
                 {id: 2,
                  from: 'fray@futurama.com',
                  subject: 'Shut up? and take my money!'
                 }
                ]
```

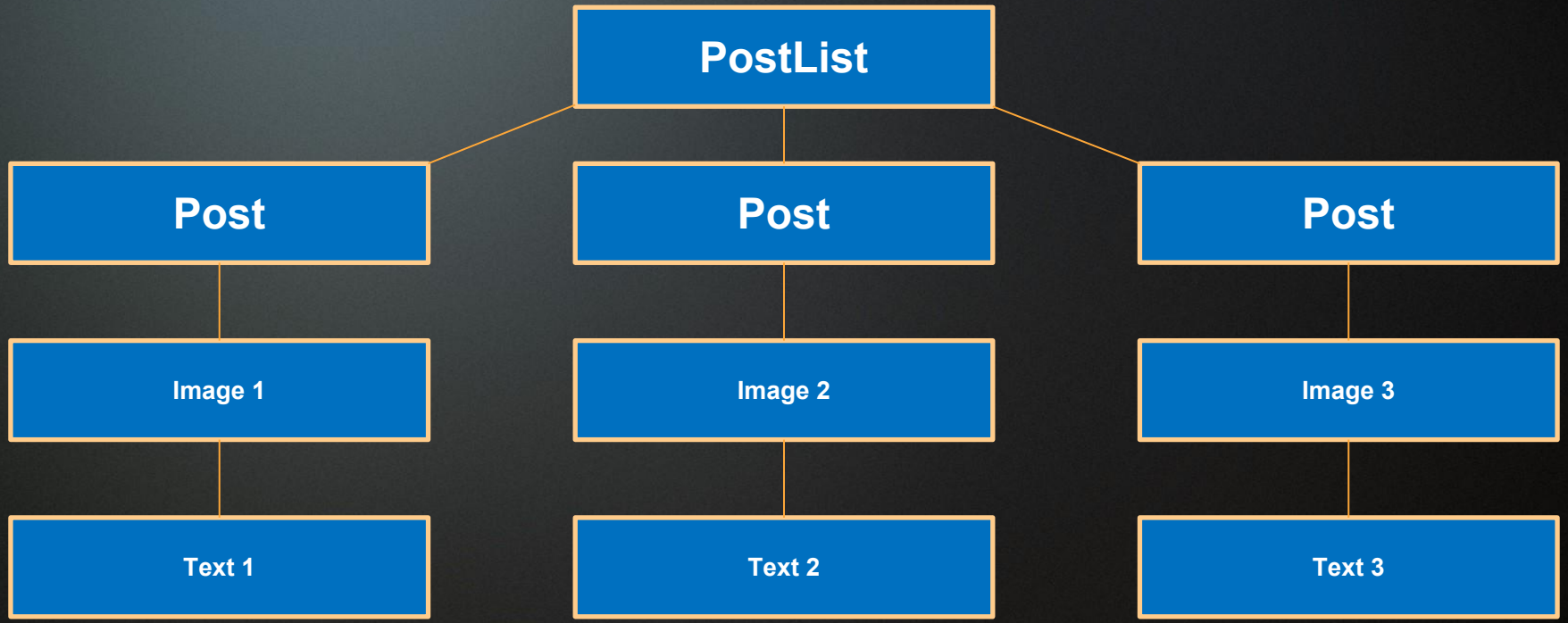
[keys]

./src/components/MailList/index.js

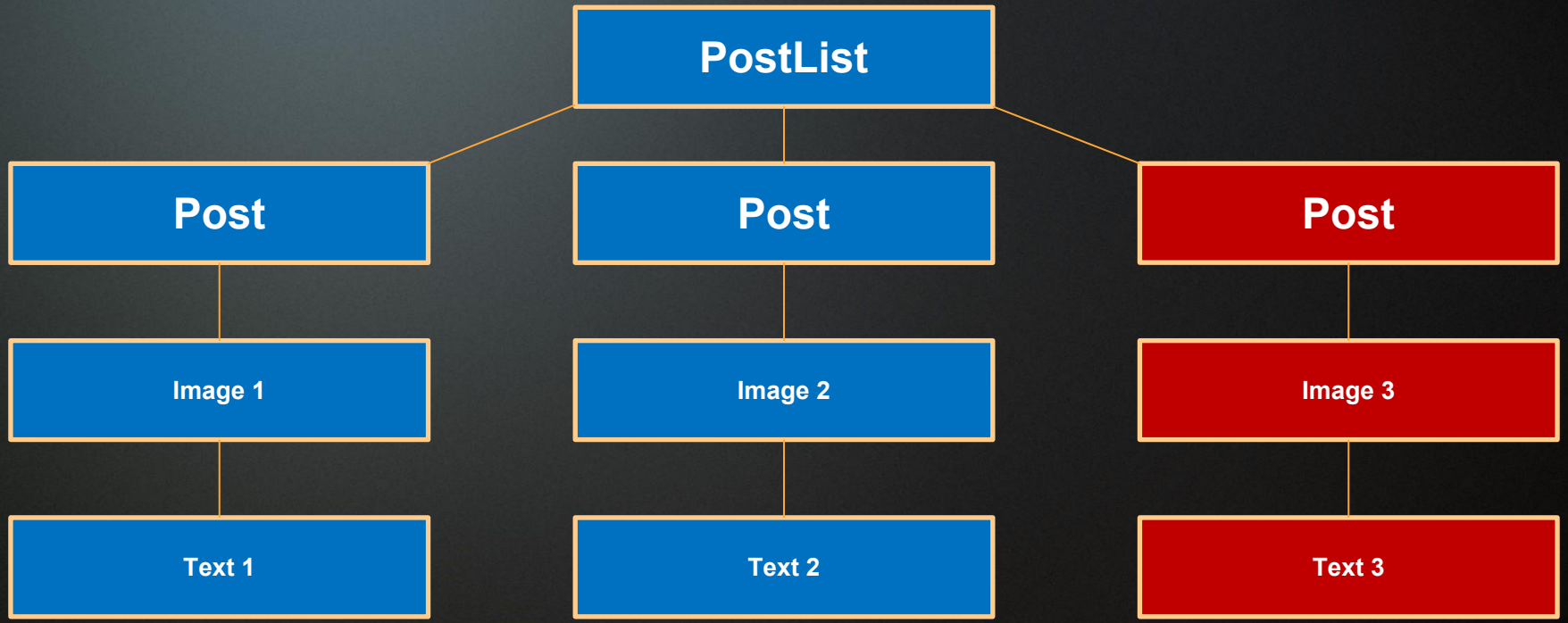
```
import {mails} from '../../dummy.js'

export default function MailList({mails}){
     const mailsElements = mails.map((mail) => {
          <li key={mail.id}>
               {mail.from} - {mail.subject}
          </li>
     });
     return(<ul>{mailsElements}</ul>);
}
```

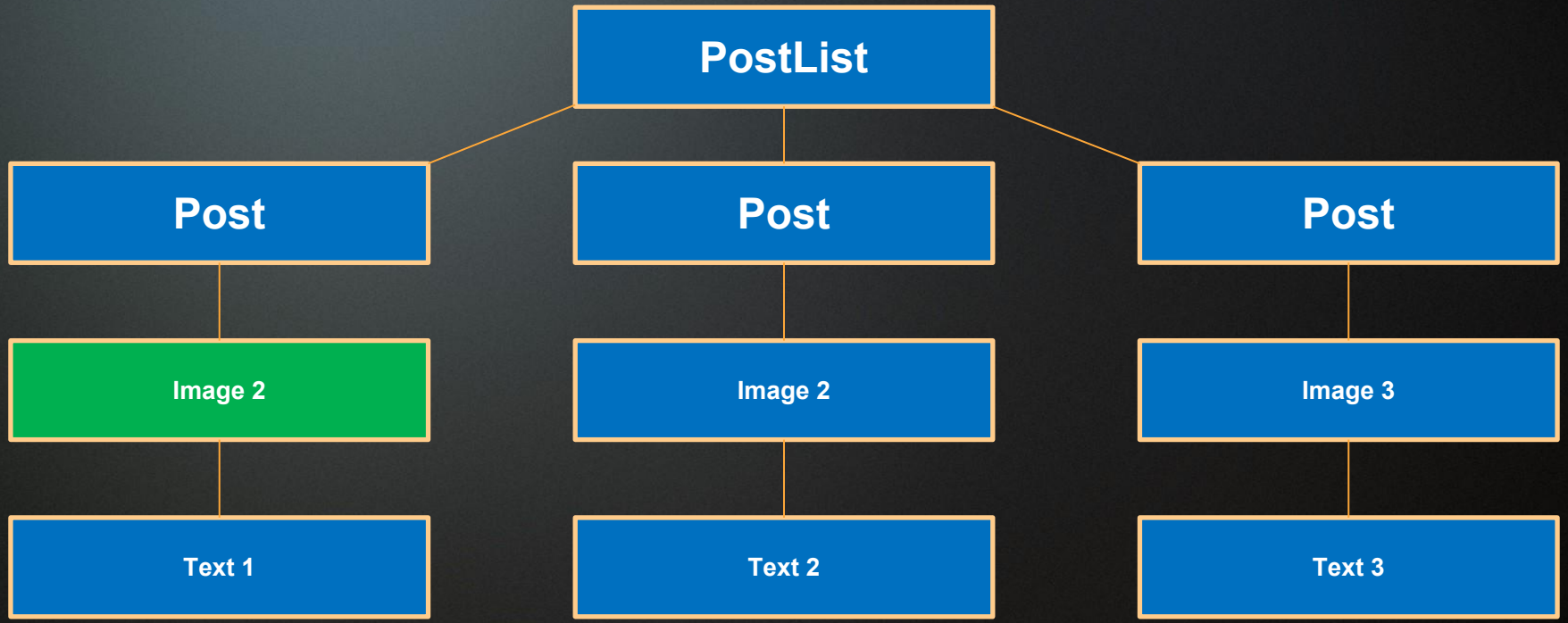
Virtual DOM


PostList
Post
Post
Post
Image 1
Image 2
Image 3
Text 1
Text 2
Text 3

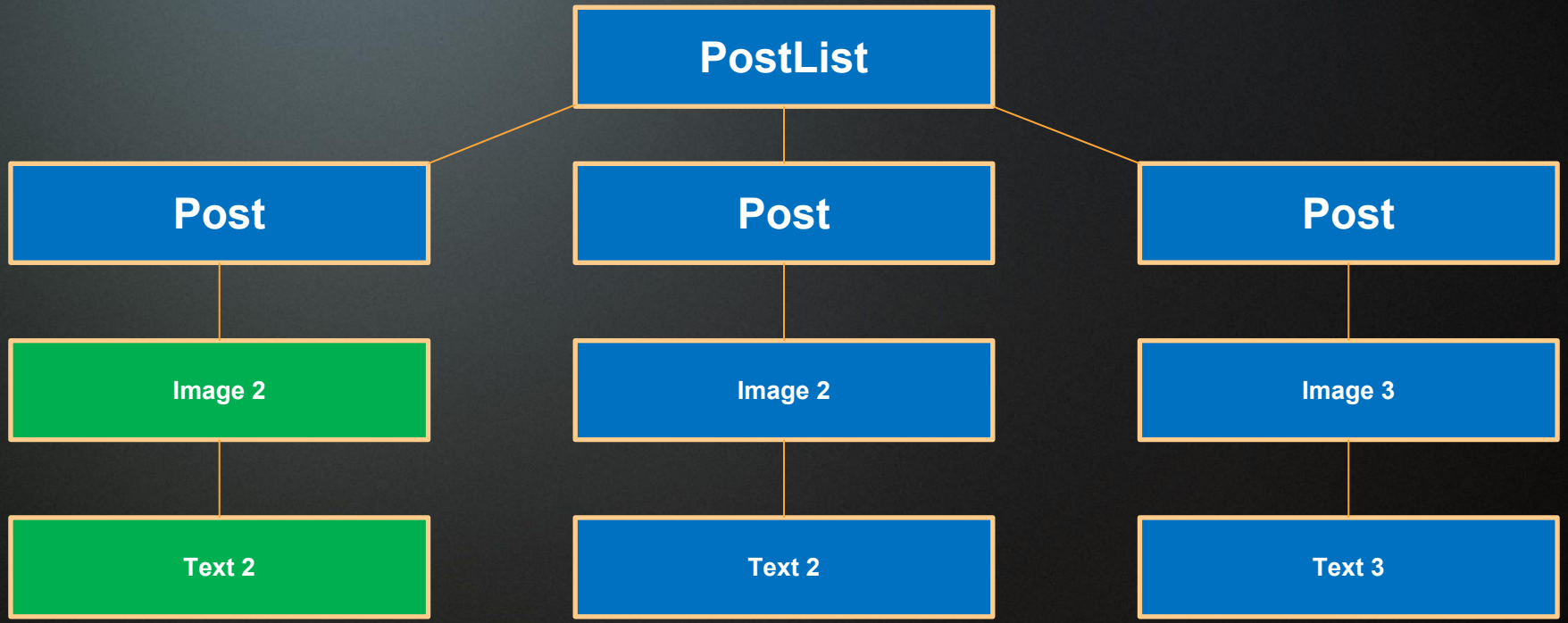
Virtual DOM


PostList
Post
Post
Post
Image 1
Image 2
Image 3
Text 1
Text 2
Text 3

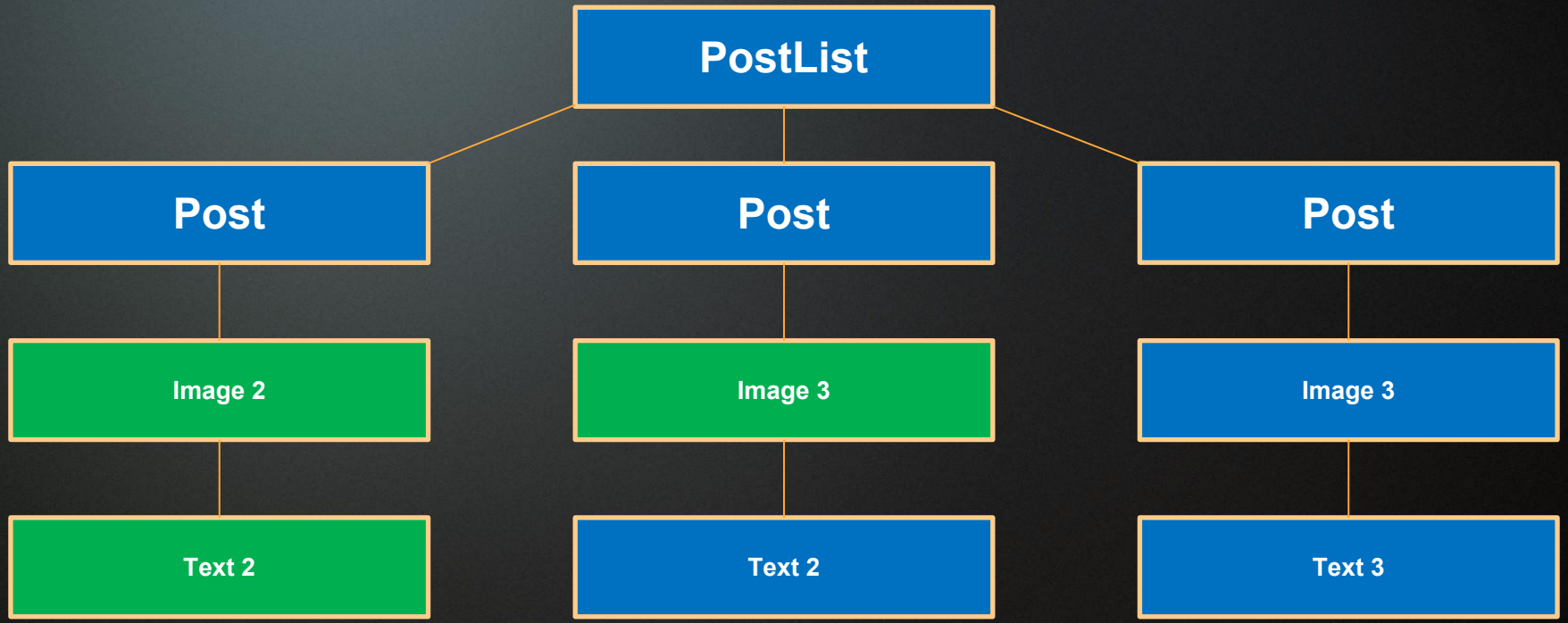
Virtual DOM


PostList
Post
Post
Post
Image 2
Image 2
Image 3
Text 1
Text 2
Text 3

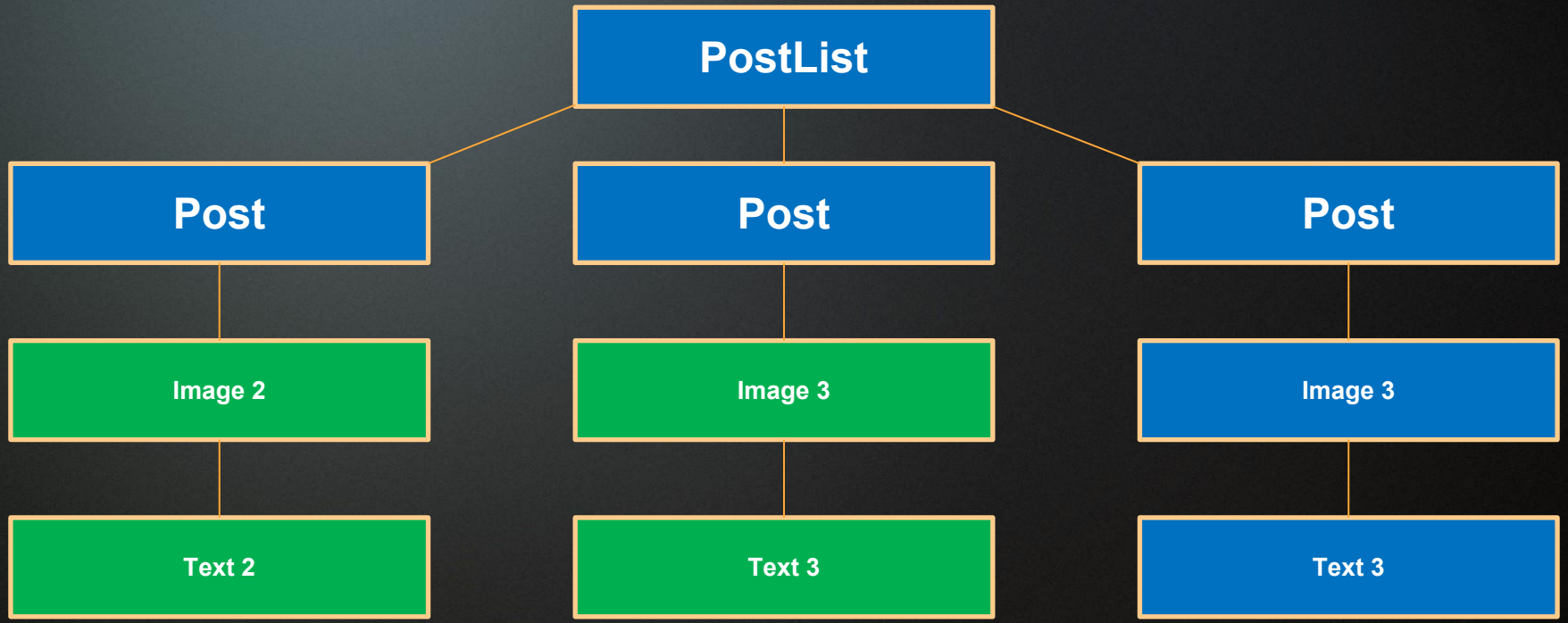
Virtual DOM


PostList
Post
Post
Post
Image 2
Image 2
Image 3
Text 2
Text 2
Text 3

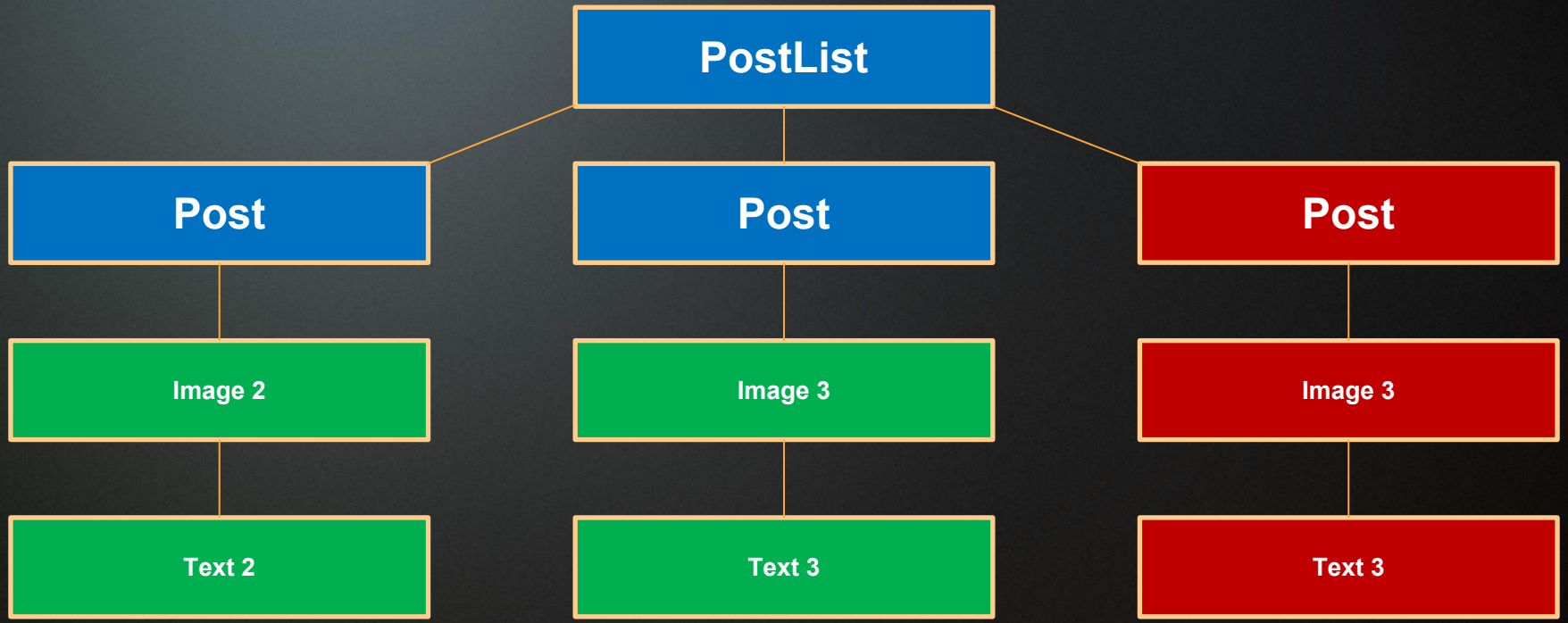
Virtual DOM


PostList
Post
Post
Post
Image 2
Image 3
Image 3
Text 2
Text 2
Text 3

Virtual DOM


PostList
Post
Post
Post
Image 2
Image 3
Image 3
Text 2
Text 3
Text 3

Virtual DOM


PostList
Post
Post
Post
Image 2
Image 3
Image 3
Text 2
Text 3
Text 3

[Refs]

refs используются для получения ссылки на узел DOM (Document Object Model) или компонента в React

Когда использовать:

- Управление фокусом, выделением текста или воспроизведением мультимедиа
- Анимация
- Интеграция реакт приложения с библиотеками работающими с DOM

[DAN.IT] EDUCATION

[Refs]

```
class CustomTextInput extends Component {
  focus = () => {
    // Установка фокуса на поле текстового ввода (input) с явным использованием
    // исходного API DOM
    this.textInput.focus();
  }
  render() {
    // Использование обратного вызова `ref` для сохранения ссылки на поле
     // текстового ввода (input) как элемента DOM в this.textInput.
    return (
      <div>
        <input type="text"
          ref={(input) => { this.textInput = input; }} />
        <input type="button" value="Focus the text input" onClick={this.focus} />
      </div>
    );
  }
}
```

[DAN.IT] EDUCATION