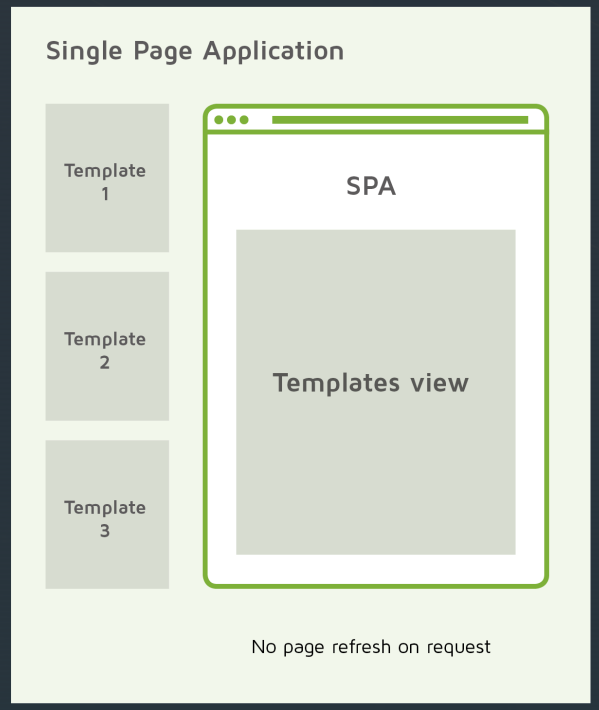
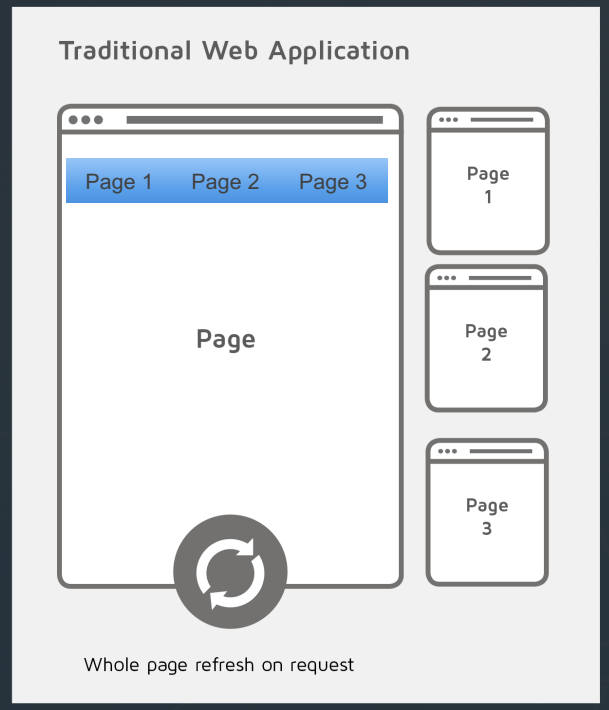
Single Page Application

What is a single page application?

Single page application is a kind of web apps, that dynamically generate the content of the page instead of loading differernt web pages. Therefore we don‘t need any page reload in this type of applications. (e.g Gmail, Facebook, …)





Pros of Single-Page Application

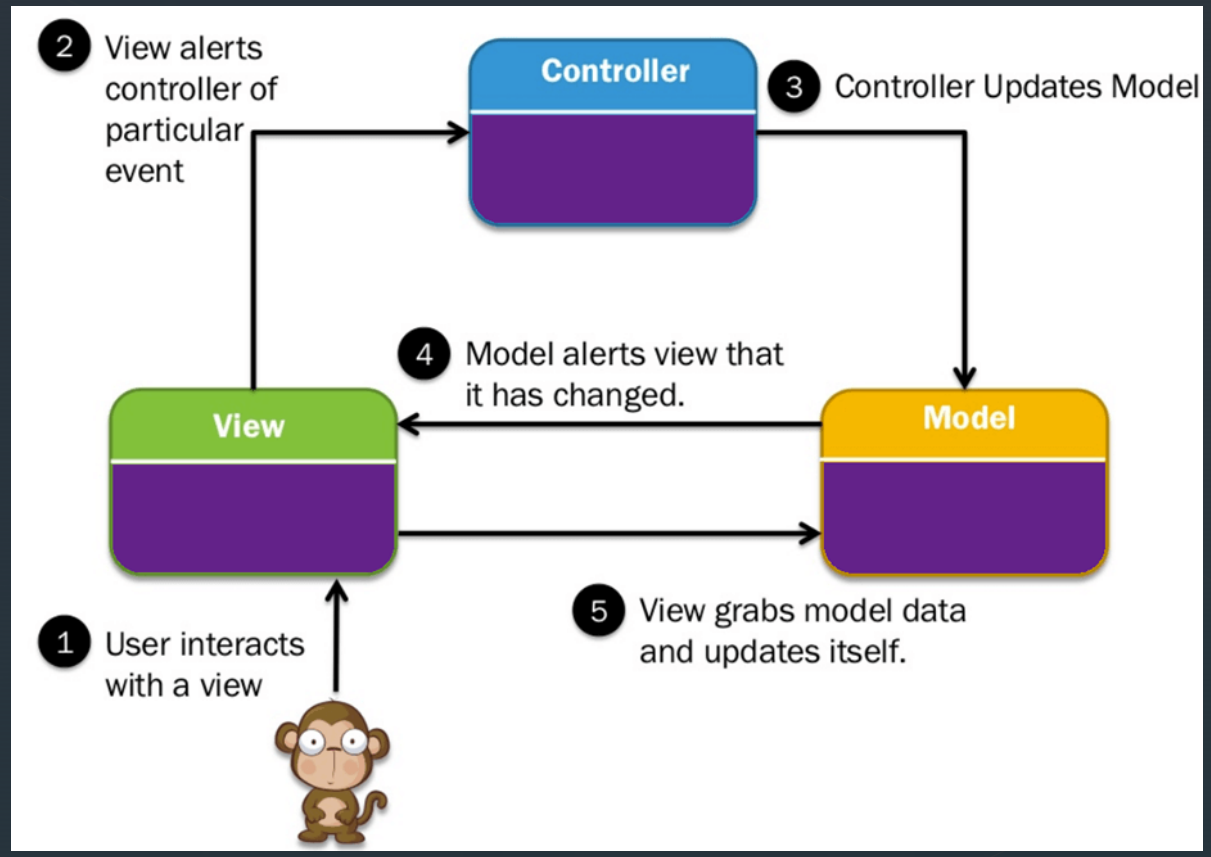
- SPA is fast becuase most of page content will be loaded once and only data will transmitted back and forth
- The development is simplified, because no need to write code to render page on the server.
- Easy to debugging, because chrome web development tools let you to moniter the networks operations.
- Easier to develop mobile applications, because same backend can be used for web and mobile app.
- SPA can cache any local storage effectively, therefore can be used even offline.

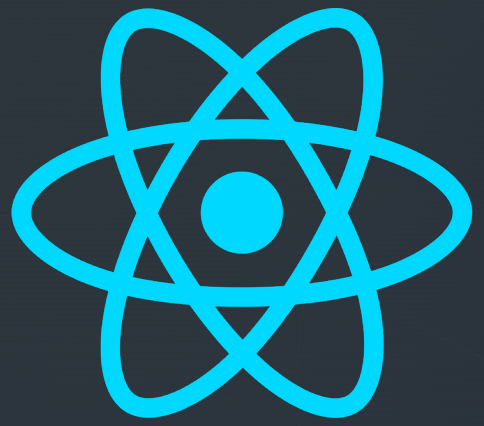
Cons of Single-Page Application

- It is very tricky and not easy to implement
- It is slow to download because of heavy client framework need to be loaded to the client.
- It requires Javascript to be present and enabled.

Model-View-Controller (MVC)

- It‘s a software architecture pattern for implementing user interfaces and split the application into three interconnected parts.



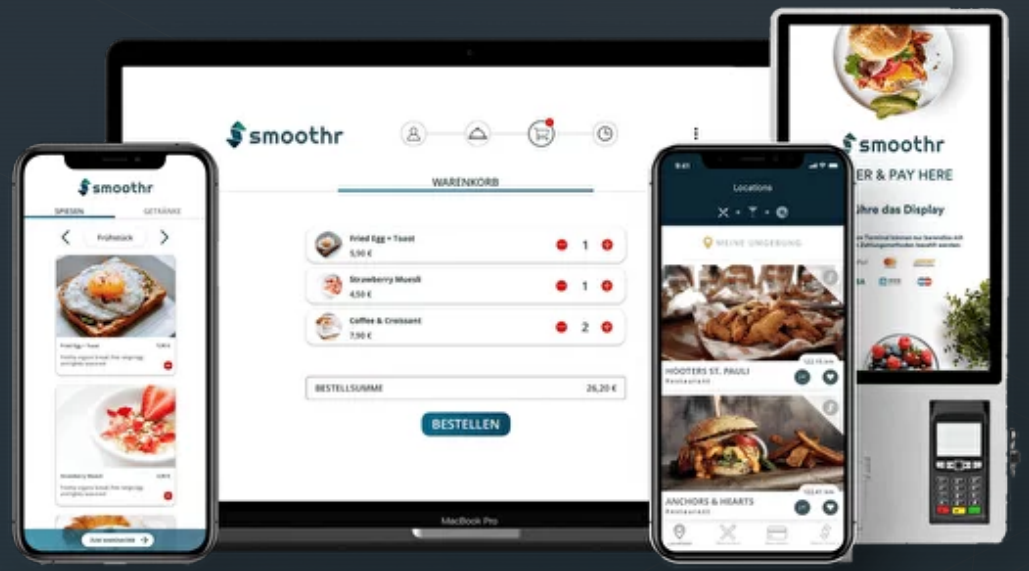


React

ReactJS Introduction

What is React?

- React is a Javascript library for building fast and interactive user interfaces for the web as well as mobile applications

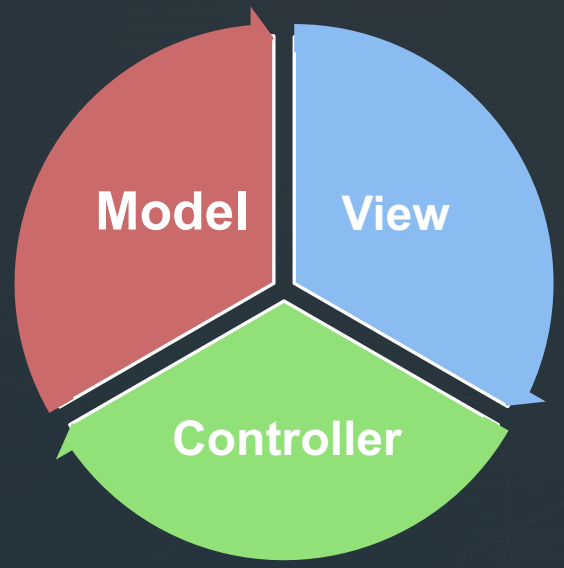


What is React?

- React is a Javascript library for building fast and interactive user interfaces for the web as well as mobile applications
- It is an open-source reusable component-based front-end library

What is React?

- React is a Javascript library for building fast and interactive user interfaces for the web as well as mobile applications
- It is an open-source reusable component-based front-end library
- In a Model View Controller architecture, React is the View which is responsible for how the app looks and feels.



What is

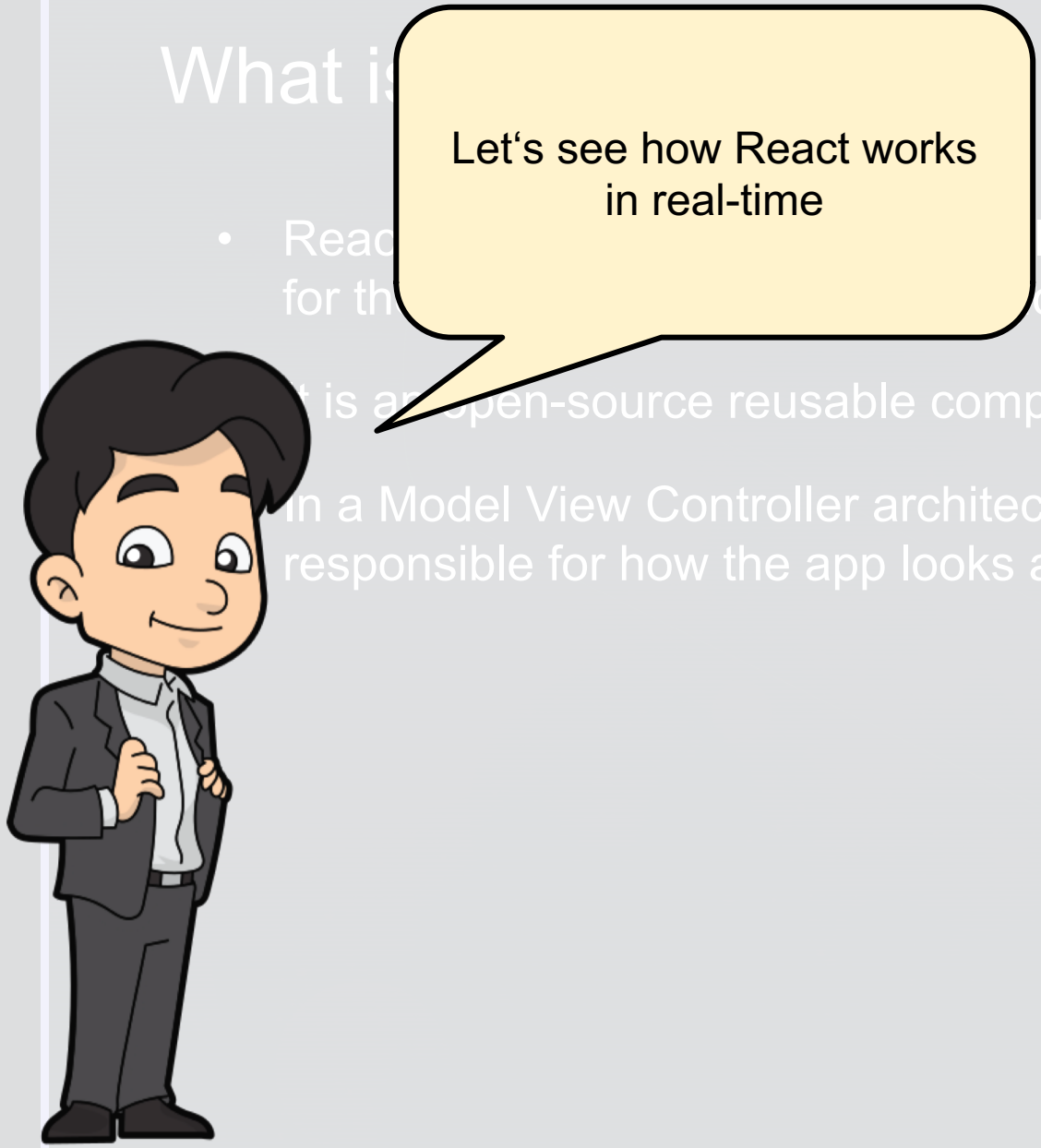
Did You Know?
React was created by **Jordan Walke,** who was a software engineer at Facebook

- Reac ding fast and interactive user interfaces for th cations

t is an open-source reusable component-based front-end library

In a Model View Controller architecture, React is the View which is responsible for how the app looks and feels.

What is



- Reac ding fast and interactive user interfaces for th cations

t is an open-source reusable component-based front-end library

In a Model View Controller architecture, React is the View which is responsible for how the app looks and feels.

Take an **example** of an **Instagram** webpage which is entirely built using React

018 Mockup

PROFILE PSD

jordan Follow ...

1,048 posts 13.5k followers 22 following

Jordan A. Graphic designer and photographer

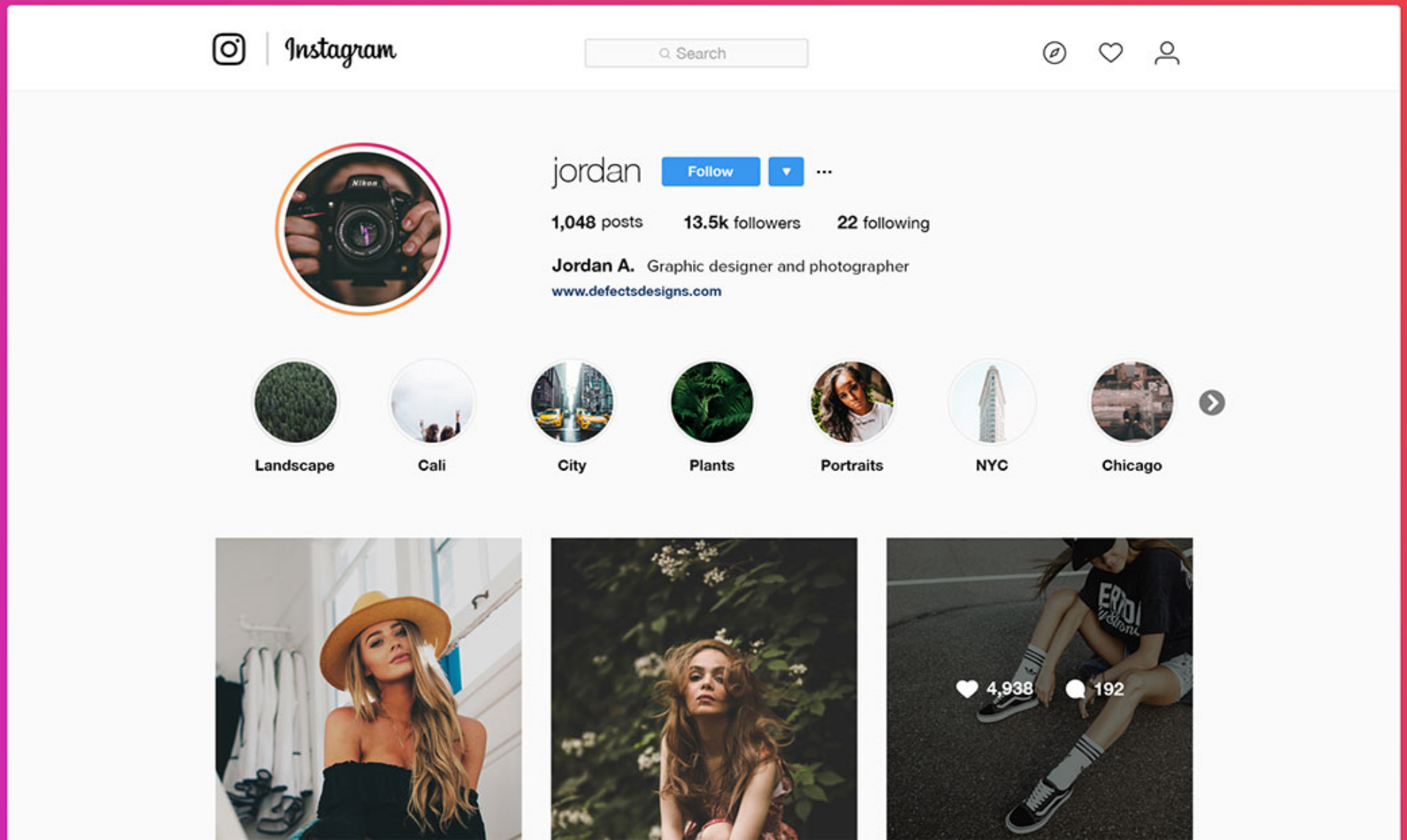
www.defectsdesigns.com

Landscape Cali City Plants Portraits NYC Chicago

4,938 192

Instagram 2018 Mockup

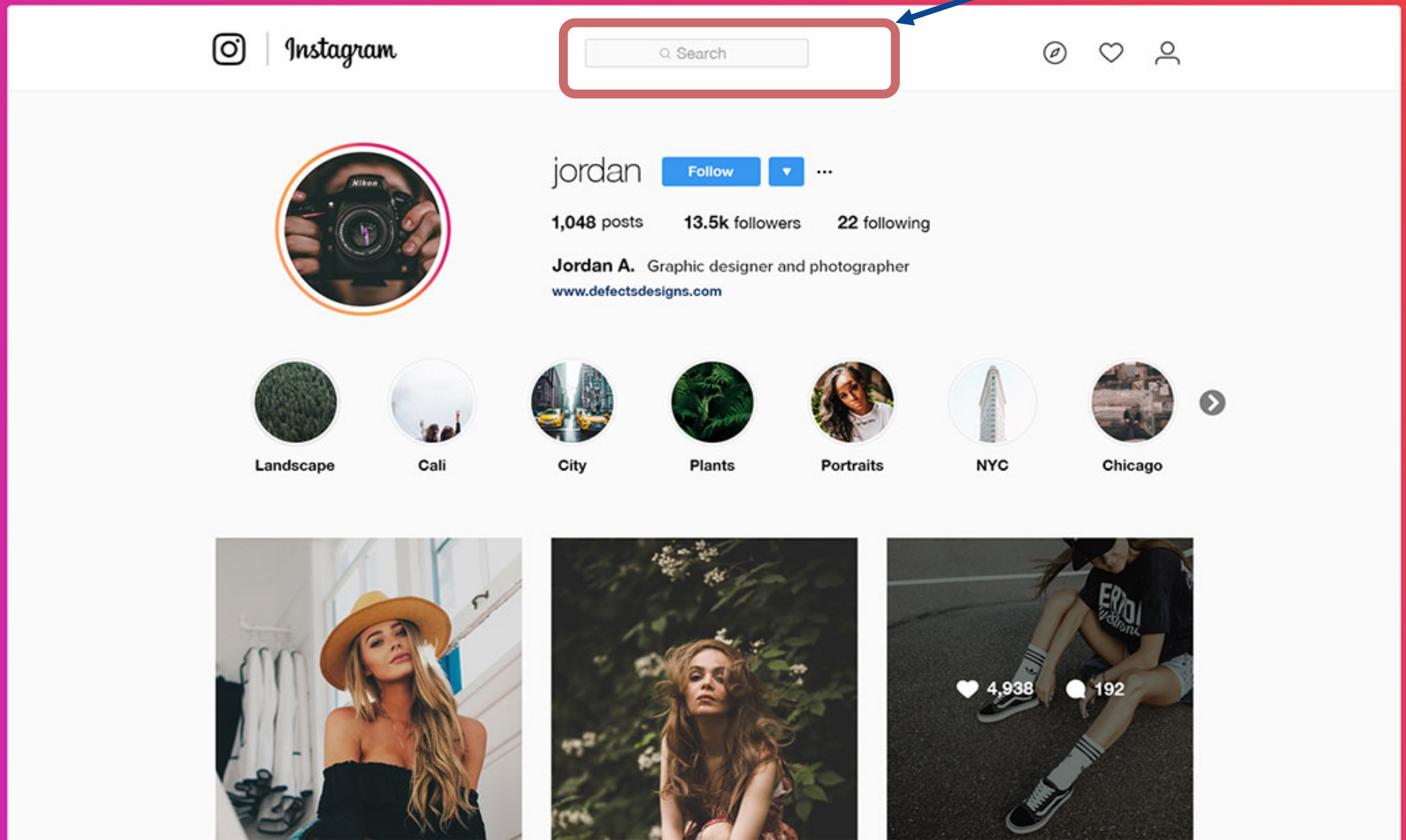
FREE WEB PROFILE PSD



Instagram 2018 Mockup

FREE WEB PROFILE PSD

Search Component

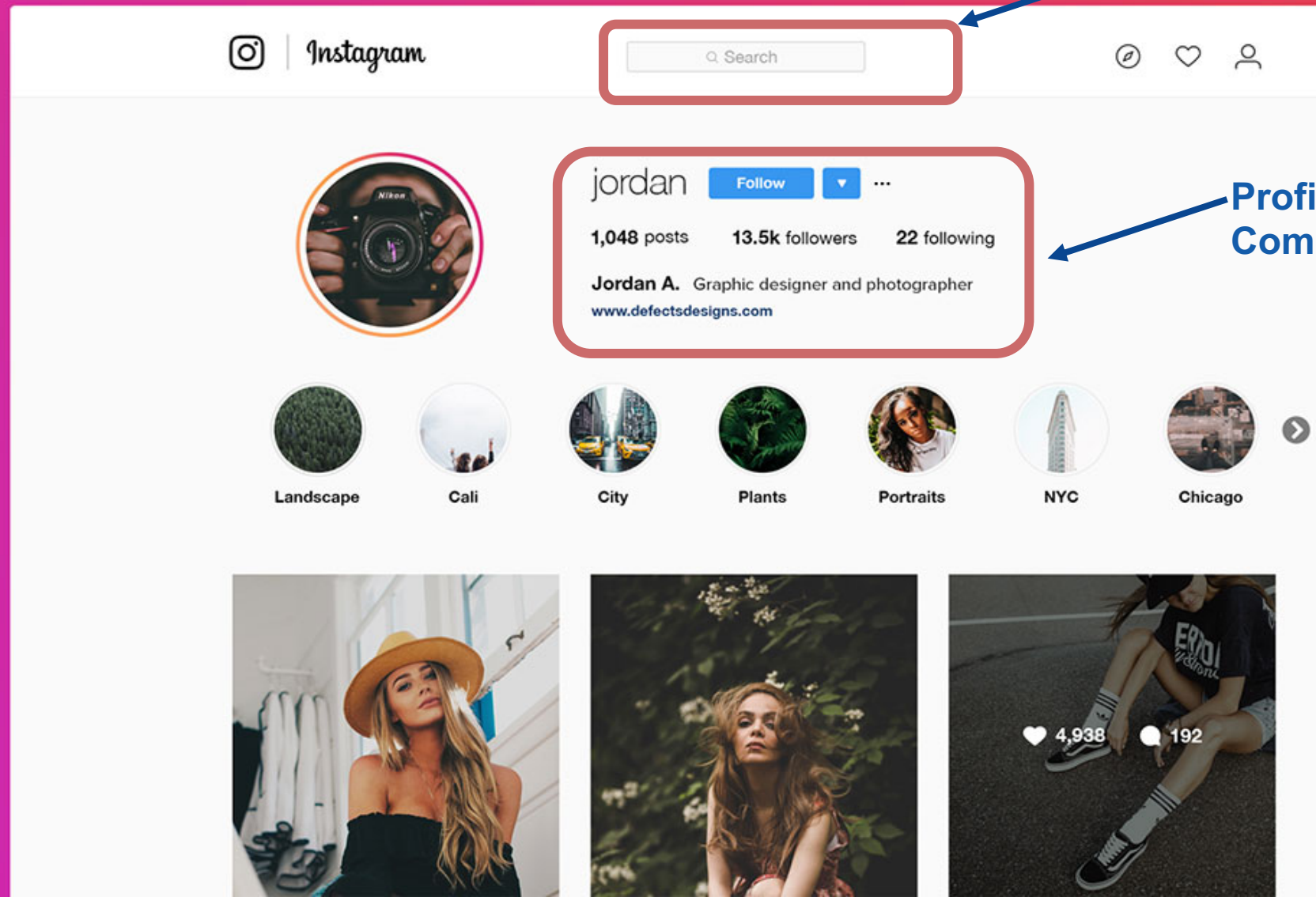


Instagram 2018 Mockup

FREE WEB PROFILE PSD

Search Component

Profile Description Component



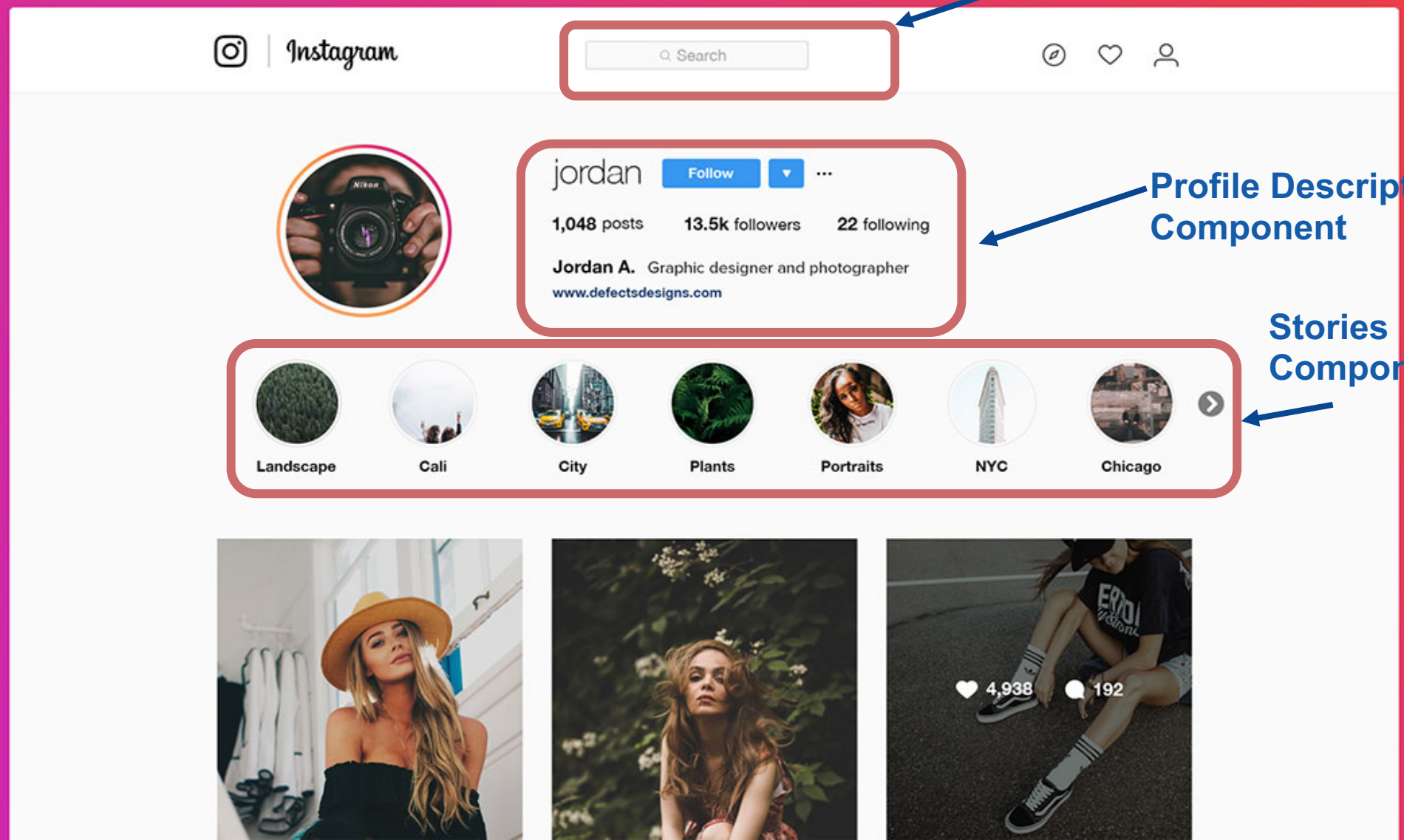
Instagram 2018 Mockup

FREE WEB PROFILE PSD

Search Component

Profile Description Component

Stories Component



Instagram 2018 Mockup

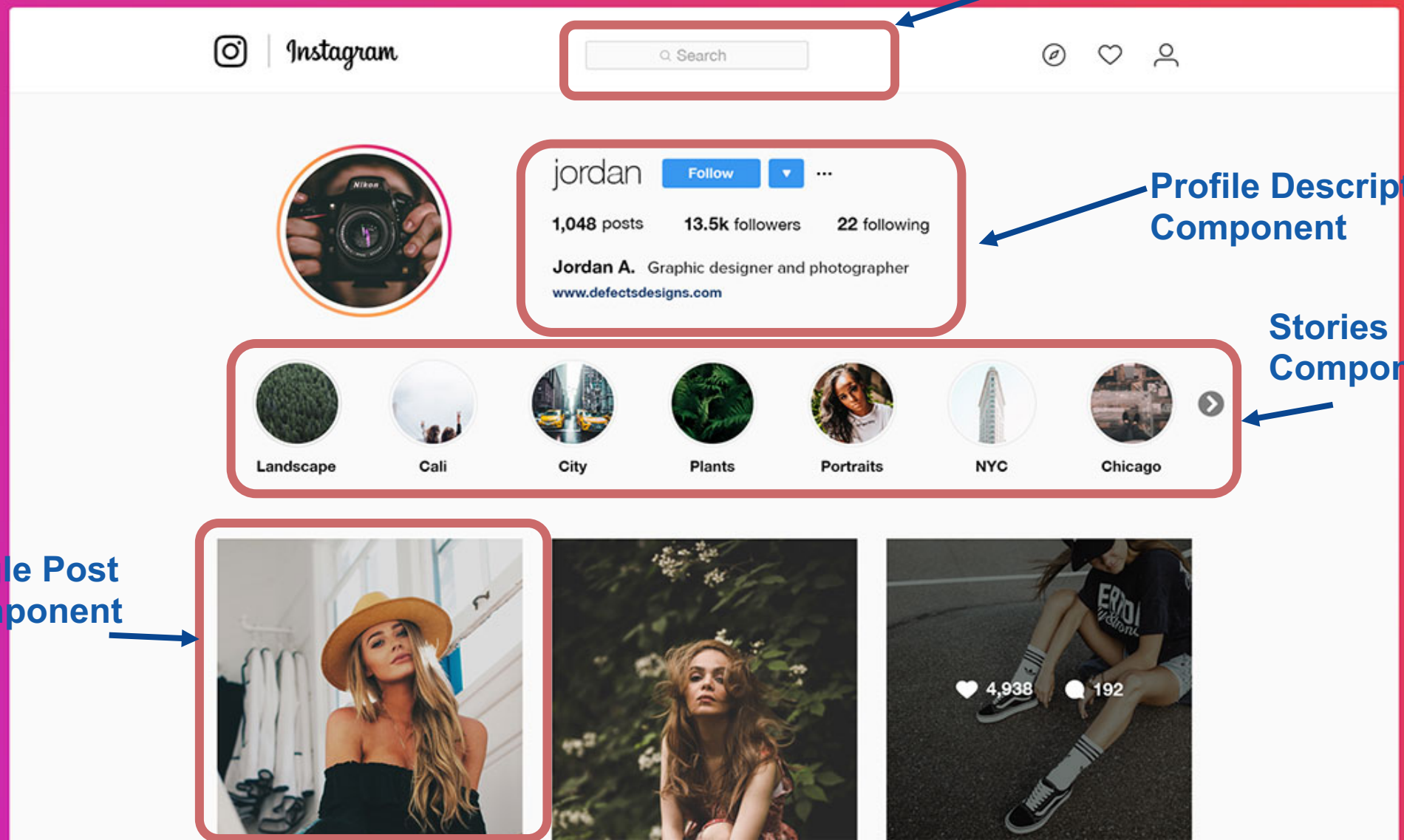
FREE WEB PROFILE PSD

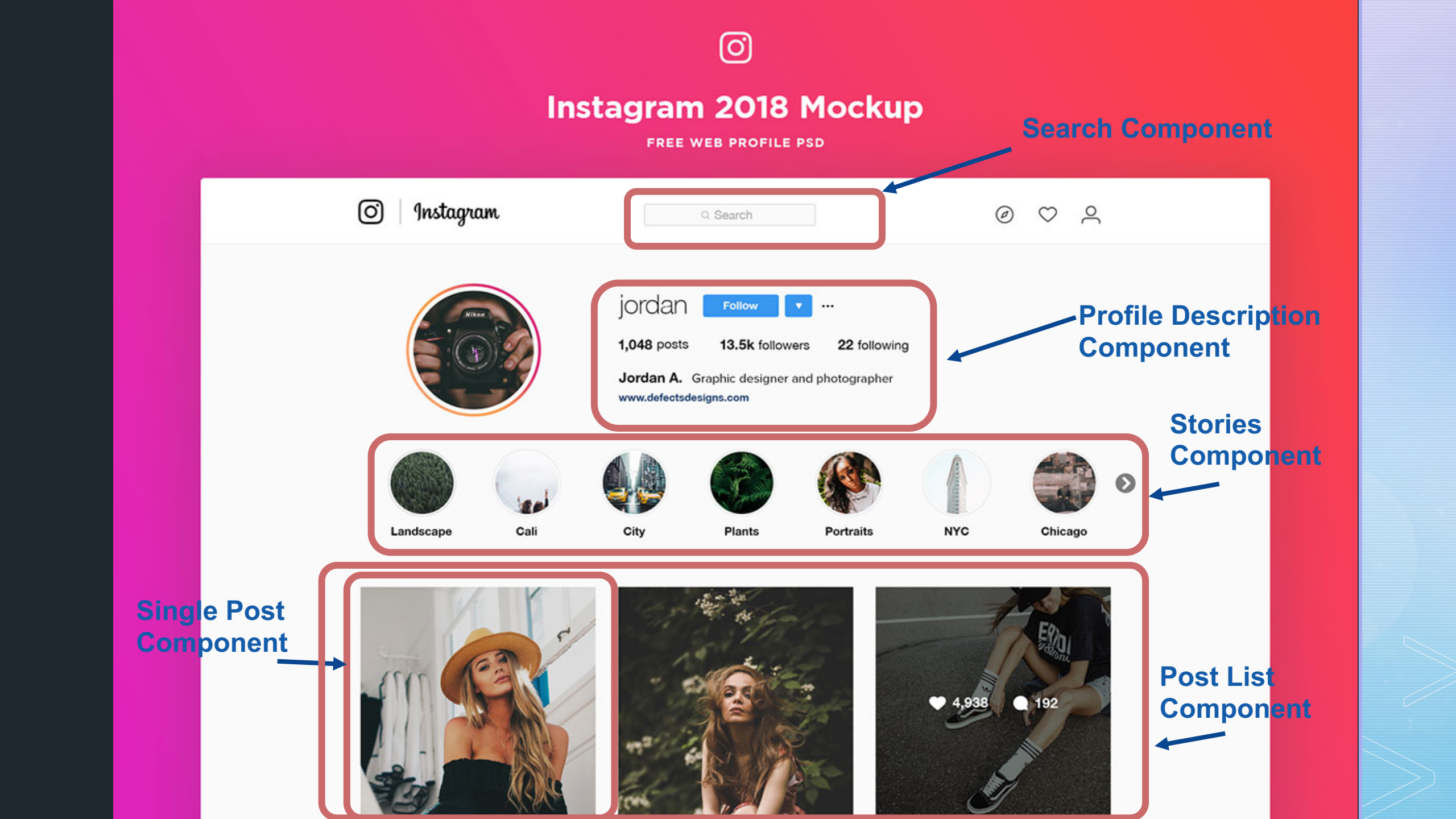
Search Component

Profile Description Component

Stories Component

Single Post Component




Instagram 2018 Mockup
FREE WEB PROFILE PSD
Search Component
Instagram
Search
jordan
Follow
1,048 posts
13.5k followers
22 following
Jordan A. Graphic designer and photographer
www.defectsdesigns.com
Profile Description Component
Landscape
Cali
City
Plants
Portraits
NYC
Chicago
Stories Component
Single Post Component
4,938
192
Post List Component

React divides the UI into multiple components, which makes the code easier to debug, and each component has it‘s own property and function

jordan Follow ...

1,048 posts 13.5k followers 22 following

Jordan A. Graphic designer and photographer

www.defectsdesigns.com

Landscape Cali City Plants Portraits NYC Chicago

4,938 192

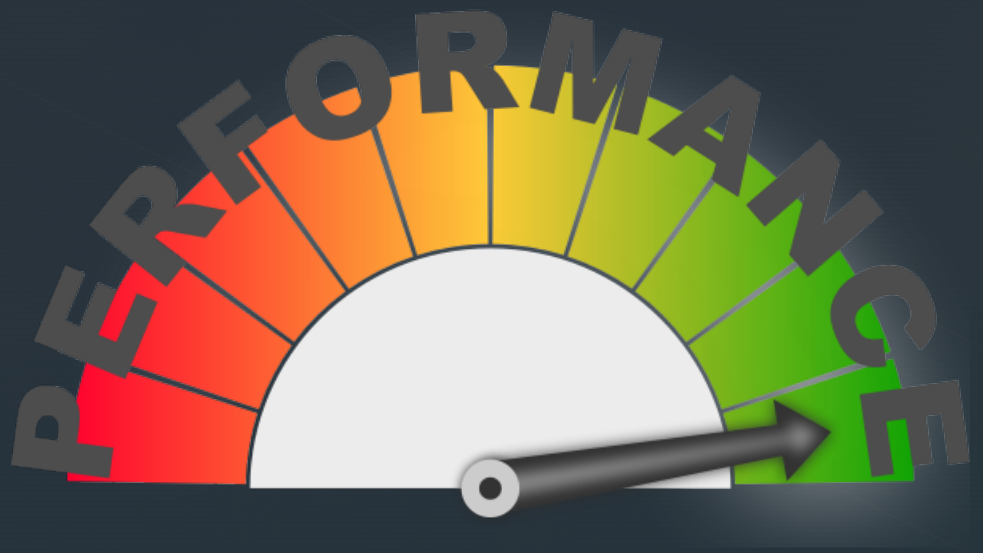
Why React?

Why React?

- Easy creation of dynamic web applications

Why React?

- Easy creation of dynamic web applications
- Performance enhancement



Why React?

- Easy creation of dynamic web applications
- Performance enhancement
- Reusable components

Why React?

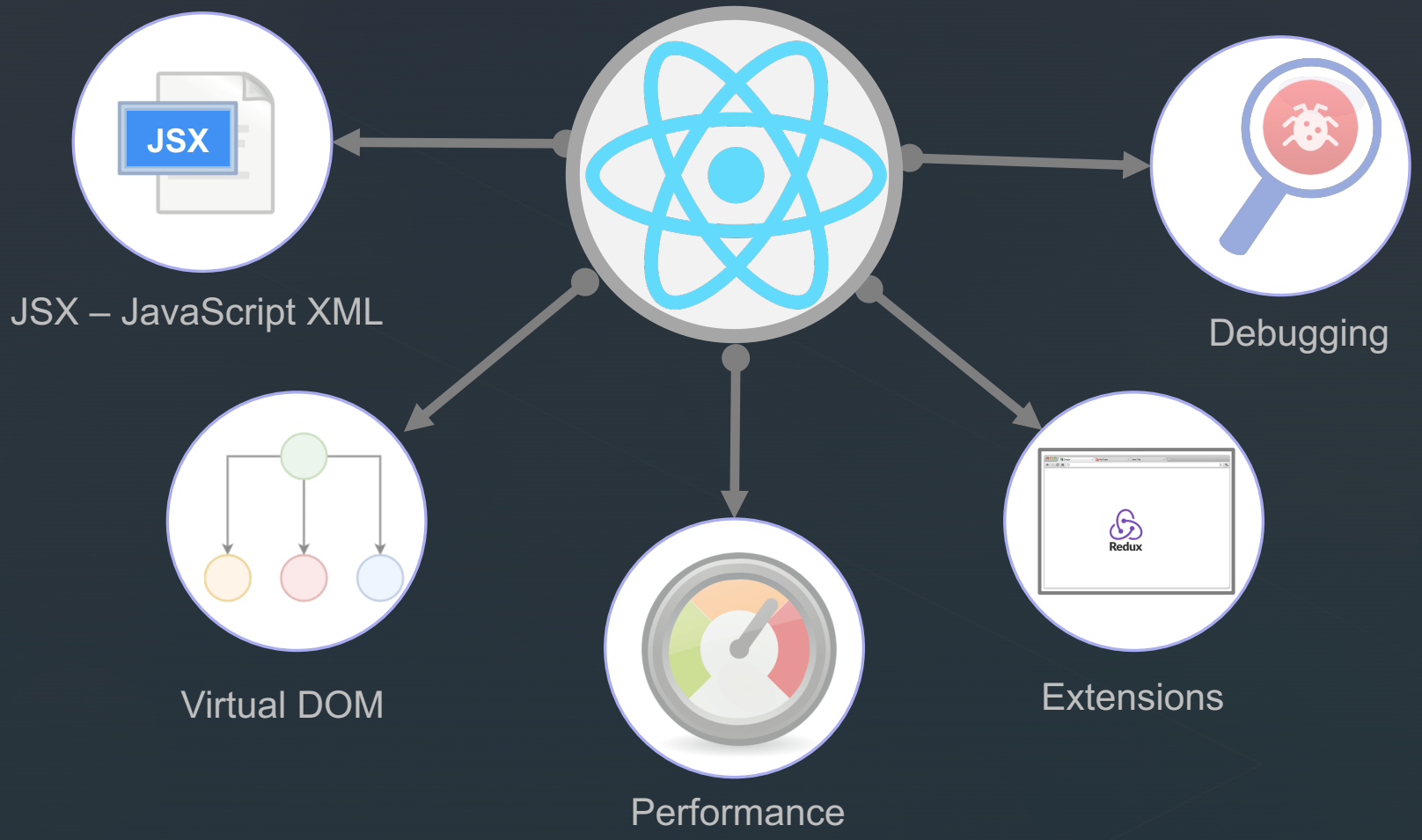
- Easy creation of dynamic web applications
- Performance enhancement
- Reusable components
- Can be used for mobile apps

Why React?

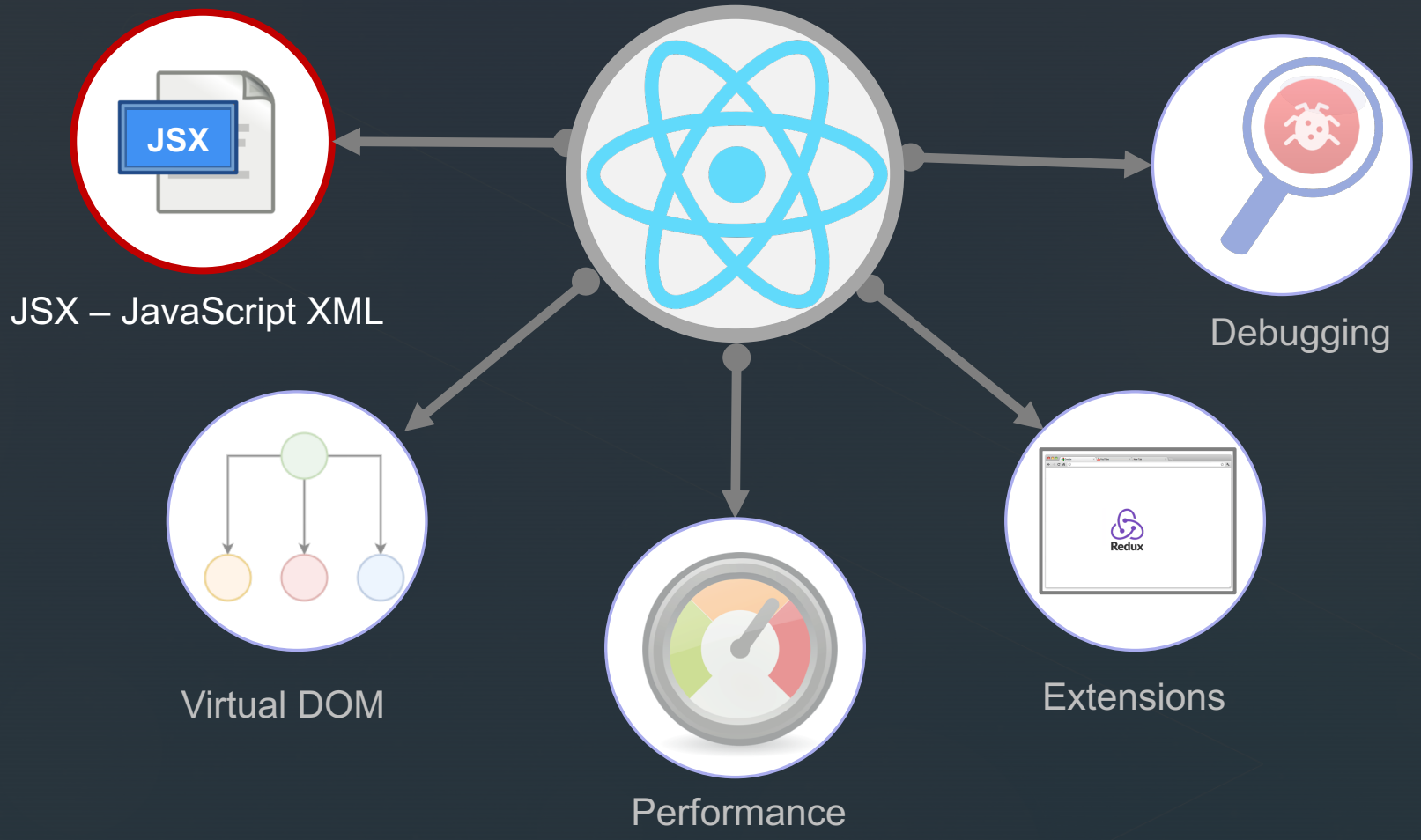
- Easy creation of dynamic web applications
- Performance enhancement
- Reusable components
- Can be used for mobile apps
- Dedicated tools for debugging

Features of React

Features of React



Features of React



What is JSX?

- JSX is a syntax exntension to JavaScript. It is used with React to describe what the User Interface should look like.
- By Using JSX, you can write HTML structure in the same file that contains JavaScript code.
- JSX helps in making the code easier to understand and debug as it avoids usage of JS DOM structures which are rather complex.

JS + HTML = JSX

What is

You got it! JSX is basically a combination of Javascript and HTML

- JSX Script. It is used he User Interface ould look like.

y Using JSX, you can write HTML structure in the same file that contains JavaScript code.

JSX helps in making the code easier to understand and debug as it avoids usage of JS DOM structures which are rather complex.

JS + HTML = JSX

Let‘s look at this basic JSX syntax and understand how it works

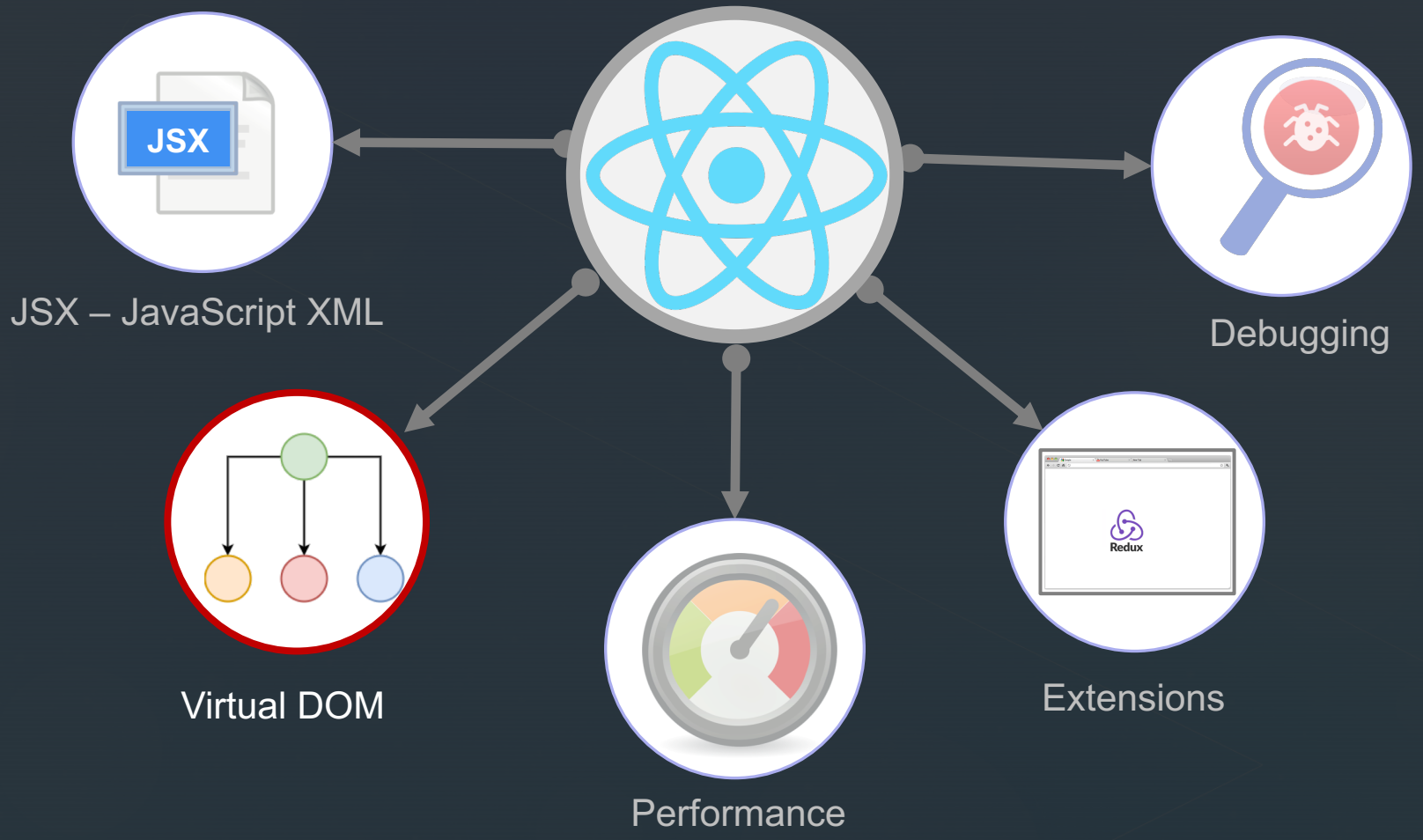
```
const simple = <h1>Hello World!</h1>;
```

“Const simple =“ is a JavaScript code

```
const simple = <h1>Hello World!</h1>;
```

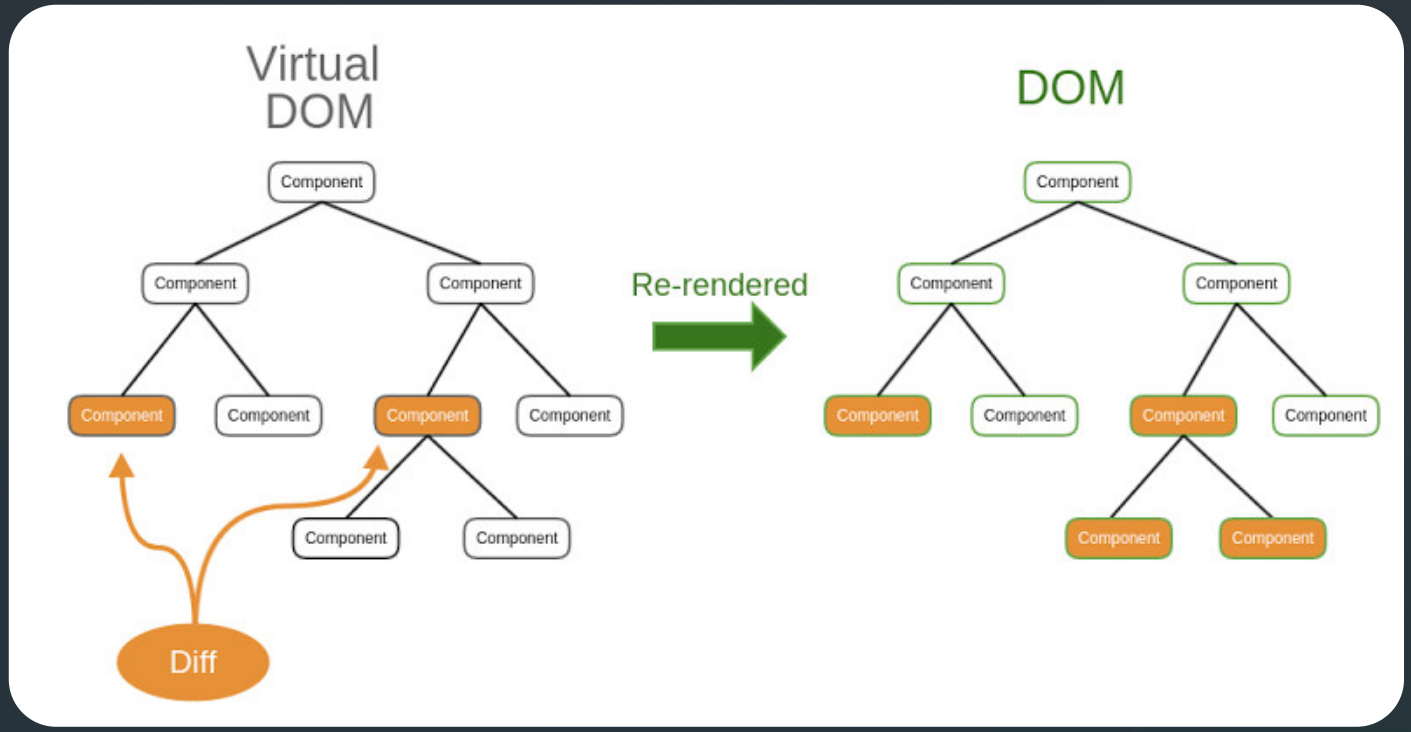
```
const simple = <h1>Hello World!</h1>;
```

Features of React



Virtual DOM (React DOM)

- React Keeps a lightweight representation of the Real DOM in the memory, and that is known as the Virtual DOM



Virtual

- Reac
  Real
  the Virtual

DOM treats an XML or HTML document as a tree structure in which each node is an object representing a part of the document

Virtual)

- Reac ntation of the Real is known as the Virtu M

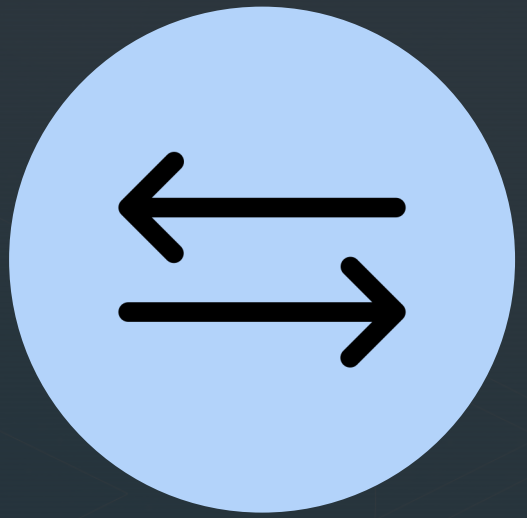
Notice that Virtual DOM is the exact copy of the real DOM

Virtual DOM (React DOM)

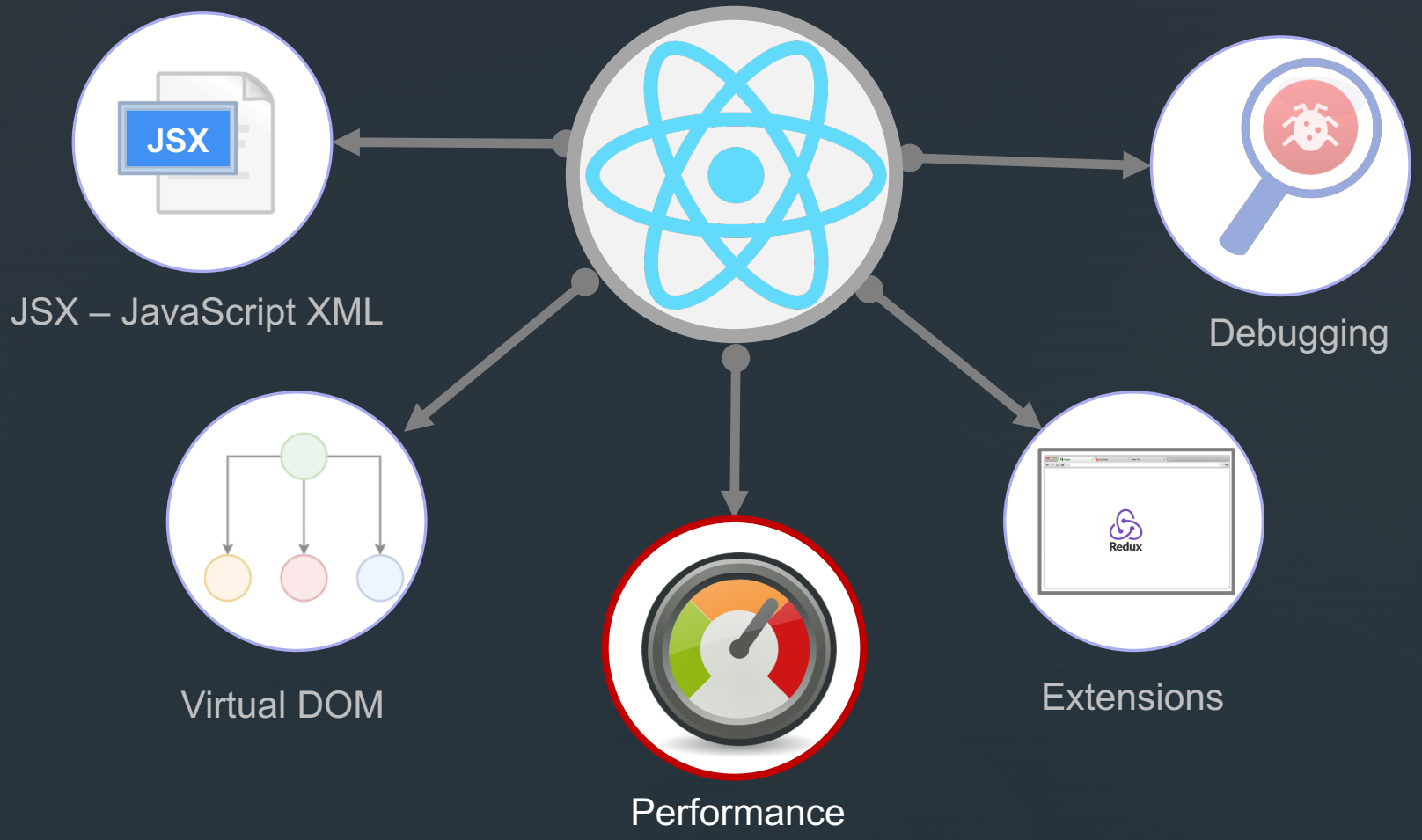
- React Keeps a lightweight representation of the Real DOM in the memory, and that is known as the Virtual DOM
- Manipulating Real DOM is much slower than manipulating virtual DOM, because nothing gets drawn onscreen.

Virtual DOM (React DOM)

- React Keeps a lightweight representation of the Real DOM in the memory, and that is known as the Virtual DOM
- Manipulating Real DOM is much slower than manipulating virtual DOM, because nothing gets drawn onscreen.
- When the state of an object changes, Virtual DOM changes only that object in the real DOM instead of updating all the objects



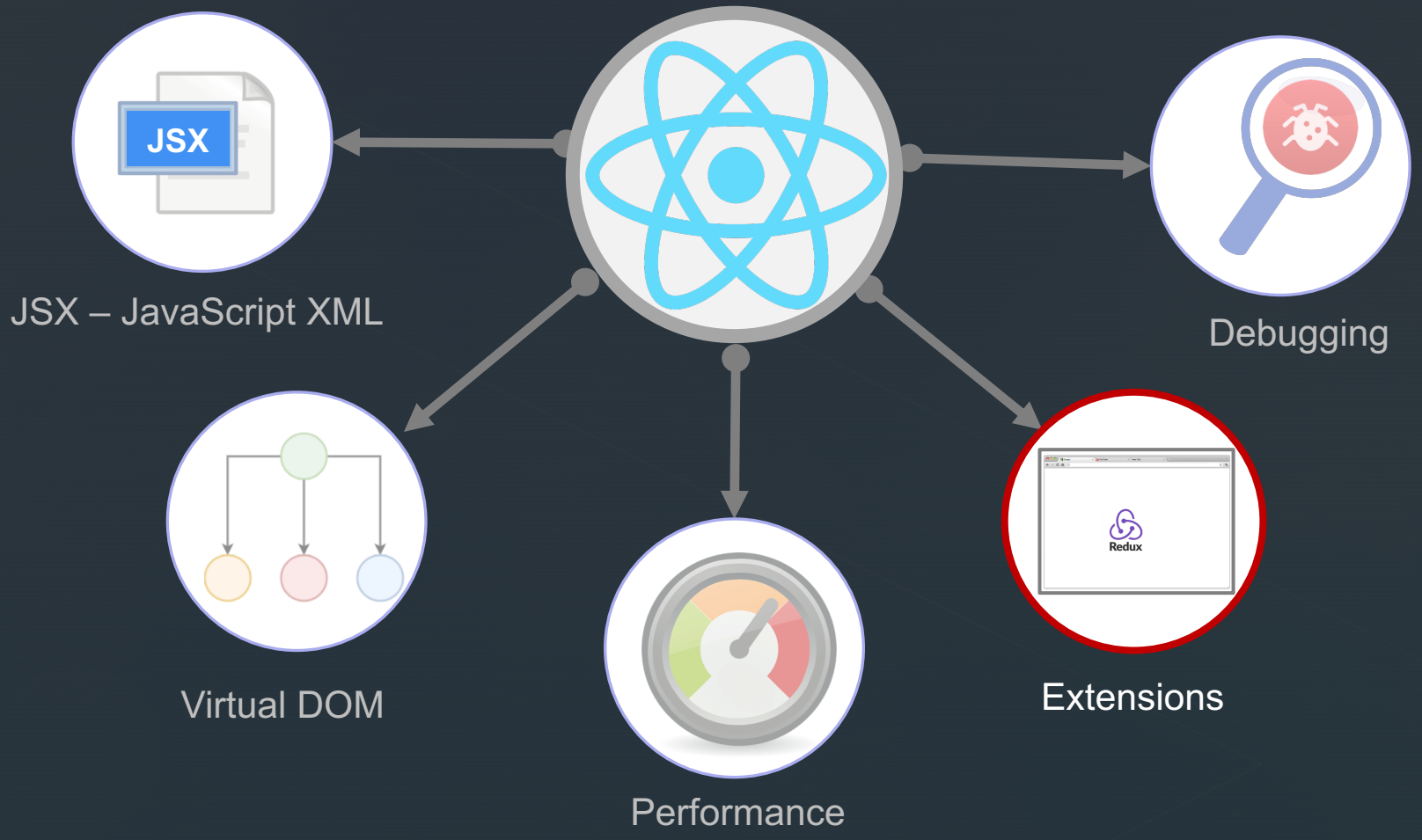
Features of React



Performance

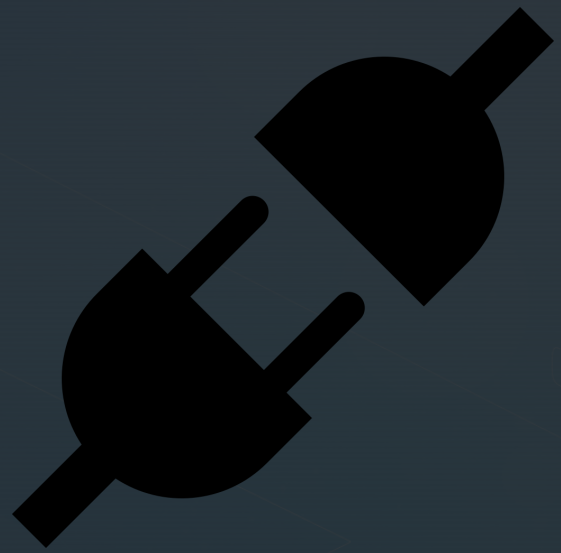
- React uses Virtual DOM that makes the web apps fast.
- Complex User Interface is broken down into individual components allowing multiple users to work on each component simultaneously.

Features of React

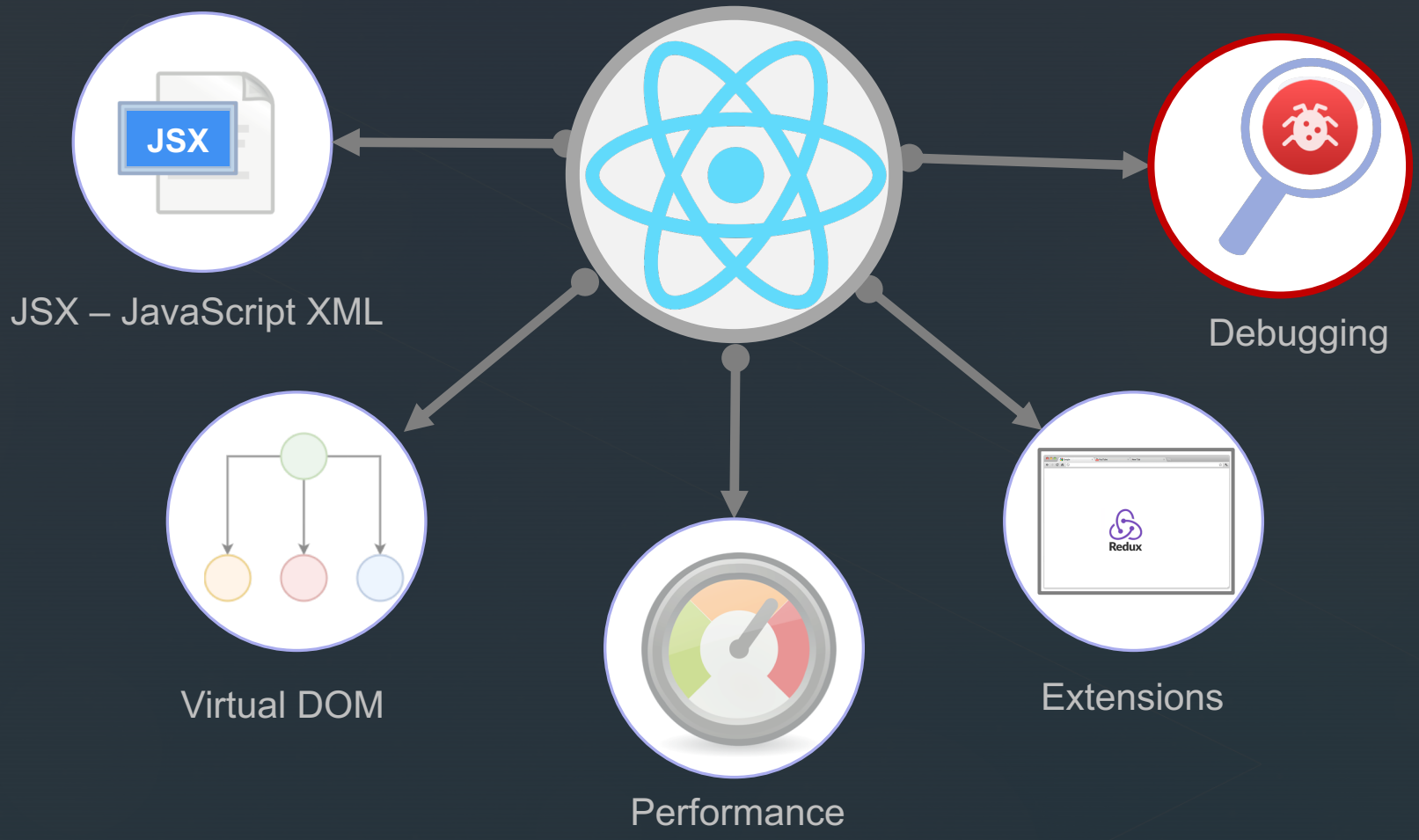


Extensions

- React goes beyond simple UI and has many extensions for complete application architecture support.
- It provides server-side rendering
- Supports mobile app development
- Extended with Redux, among othes

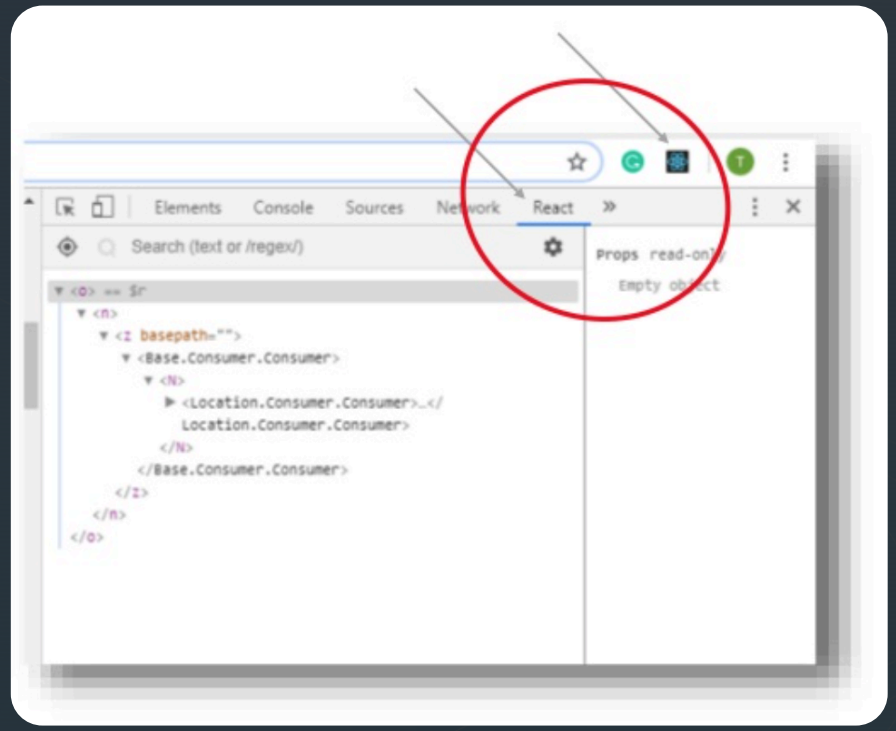


Features of React



Debugging

- React applications are extremely easy to test due to a large developer community
- Facebook even provides a small browser extension that makes React debugging faster and easier (React Developer Extension)



Components, State & Props

Reac Components

- Components are the building blocks of any React application, and a single app usually consist of multiple components
- A component is sessentially a piece of the User Interface. It is like a function/class that return HTML elements.
- It splits the User Interface into independent reusable pieces that can be processed seperately

Reac Components cont…

```
import React, { Component } from 'react';

export default class Simple extends Component {
    render() {
        return (
            <div>
                <h1>This is a Component</h1>
            </div>
        );
    }
}
```

Reac Components cont…

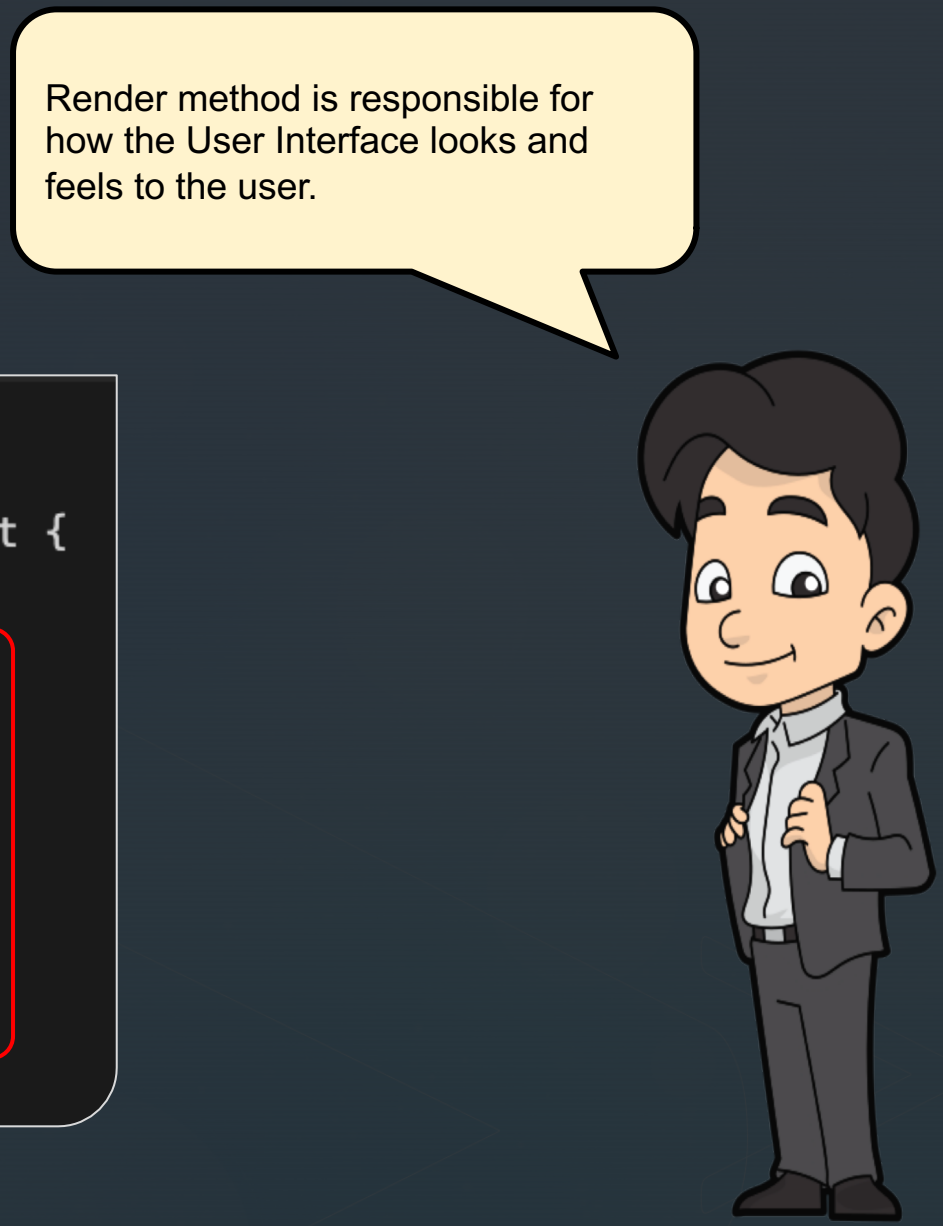
```
import React, { Component } from 'react';

export default class Simple extends Component {
    constructor(){
        this.state={ }
    }
    render() {
        return (
            <div>
                <h1>This is a Component</h1>
            </div>
        );
    }
}
```

```
import React, { Component } from 'react';

export default class Simple extends Component {
    state = { }
    render() {
        return (
            <div>
                <h1>This is a Component</h1>
            </div>
        );
    }
}
```

State Object of a component is mutable. It means only methods of the same component can modify it‘s content.

Reac Components cont…

```
import React, { Component } from 'react';

export default class Simple extends Component {
    state = { }
    render() {
        return (
            <div>
                <h1>This is a Component</h1>
            </div>
        );
    }
}
```



State of a Component

- State of a component is an object that holds some data
- This data influence the output of a component
- To modify state properties:
  1. Use `this.setState({propName:value})`
  2. Use spread operator if property assigned to array or object.
     Example:

     ```
     this.setState({propName: […array, newEelm]})
     ```
  3. You can use push, pop, shift, unshift, splice to manipulate the state object
     Example:

     ```
     this.state.users.push(newUserObject);
     ```

State of a Component

- This is how we can access State‘s properties:

```
import React, { Component } from 'react'

export default class App extends Component {
    state = {
        name: 'fahim'
    }

    render() {
        return (
            <div>
                Hi, {this.state.name}. Welcome here!
            </div>
        )
    }
}
```

Props object of a component

- Props is short for properties, that allows us to pass arguments or data to components
- Props are passed to components in the way similar to that HTML tag attributes

Parent Comp

```
import React, { Component } from 'react'
import Child from 'Child.js'

export default class App extends Component {
    state = {
        name: 'fahim'
    }

    render() {
        return (
            <div>
                <h3>Hi, {this.state.name}. Welcome here!</h3>
1               <Child name={this.state.name} />
            </div>
        )
    }
}
```

Child Comp

```
import React, { Component } from 'react'

export default class Child extends Component {
    render() {
        return (
            <div>
2               <h5>Hey {this.props.name}.
                    This is child-component!</h5>
            </div>
        )
    }
}
```

Bracket Notation

- Since React let us to write HTML and JavaScript in same file so we are using JSX.
- JSX code needed to be written inside ( )
- If you want to write JavaScript between JSX use following syntax

```
( <JSX> { JavaScript } </JSX> )
```

Note: Sometime we need to use array.map(()=>{})

- In such cases if you need to write JSX in the callback, do it such as:

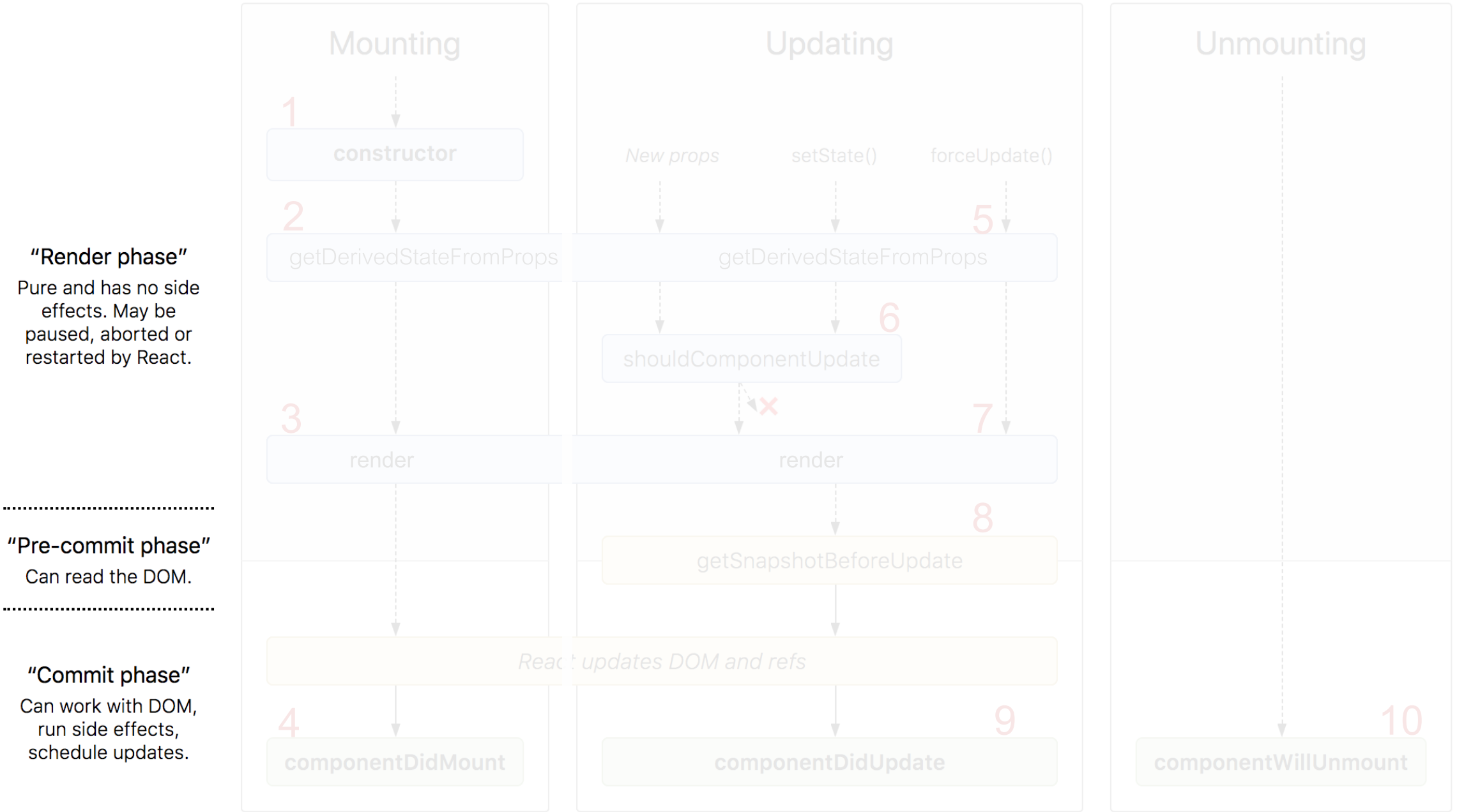
```
{array.map(()=>{ return (<JSX> </JSX>) }) }
{array.map(()=>(<JSX></JSX>))}
```

React Life Cycle

React Life Cycle

- Seriese of pre-defined events that triggers from initating to death of a react component.
- React life cycle consist of three phases:
  - **Mounting**: Component will render for the first time (Birth phase).
  - **Updating**: By state change or forceUpdate needs to render the components again and again. (Growth phase)
  - **Unmounting:** When no need to display the component anymore and remove it from user interface. (Death phase)

React version 16.4 Language en-US



Mounting Phase – constructor()

- Constructur is the first method of a component that will execute in mounting phase.
- This method is a good place to initialize properties and methods of the class component.
- The method is not mandatory for component‘s definition.

```
constructor() {
  super();
  this.state = {
    name: "fahim",
    email: "fahim@gmail.com",
    users: [],
  };
  console.log("Constructor called");
}
```

Mounting Phase – getDerivedStateFromProps()

- This method helps the programmer to set properties of state object from props object or statically.
- This method calls immediately before render() method.
- This method is **static** method. Means no need for instance of the class to invoke it.
- It returns an object to update the state, but returns null when there is no changes.

```
static getDerivedStateFromProps(props, state) {
  return {
      //name is a property of state object
      name: props.name
  }
}
```

Mounting Phase – render()

- This method is the only required method in class components and it is responsible to handle rendering of component to UI.
- Return value of this method is a single JSX element that can contains childs.
- React requires that your render() be pure.
  - Pure functions are those that do not have any side-effects and will always return the same output when the same inputs are passed
  - This means setState() can not be called within render().

Mounting Phase – componentDidMount()

- This method calls when the component mounted successfully and being visible for the end-user.
- This method is a good place to sends requests to APIs and update the state object.

```
componentDidMount() {
  console.log("ComponentDidMount");
  axios({
    method: "GET",
    url: "https://randomuser.me/api/?results=5",
    headers: { "Content-type": "application/json" },
  }).then((res) => {
    this.setState({ users: [...res.data.results] });
    this.displayUsers(this.state.users);
  });
}
```

Updating Phase – shouldComponentUpdate()

- This method calls before render method and let the programmer to cancel rendering on state/props changes manually.
- This method helps to increase performance by preventing from un-necessary rendering.
- The method returns a boolean value that specify needs to render or not.

```
shouldComponentUpdate(nextProps, nextState) {
 return this.props.title !== nextProps.title ||
  this.state.input !== nextState.input }
```

Updating Phase – componentDidUpdate()

- This method calls in the update phase.
- The common use-case for it is updating DOM in response to state/prop changes.
- setState() is allowed here but needs to be wrapped into an if condition.

```
componentDidUpdate(prevProps) {
 //Typical usage, don't forget to compare the props
 if (this.props.userName !== prevProps.userName) {
   this.fetchData(this.props.userName);
 }
}
```

Unmounting Phase – componentwillUnmount()

- This method calls immediatly before unmounting the component from user interface.
- It‘s a good place to do the cleanup actions such as clearing timers, cancelling api calls or clearing caches in storage.
- setState() is not allowed here. Because there is no more render for the component, when this method called.

```
componentWillUnmount() {
 window.removeEventListener('resize', this.resizeListener)
}
```