

www.agilityio.com

JavaScript Notes

[**Docs: React**](https://reactjs.org/docs/getting-started.html)

# Chapter 1: Introducing JSX

SPA hay Single-page application là gì?

-là ứng dụng trang đơn hay còn gọi là ứng dụng 1 trang

+được cho là cách tiếp cận hiện đại hơn

+không yêu cầu tải lại trang trong quá trình sử dụng

MPA hay Multi-page application là gì?

-là ứng dụng nhiều trang

+là cách tiếp cận cổ điển hơn

+tải lại trang trong quá trình sử dụng (click vào đường link, chuyển trang,..)

**React** is a JavaScript library for building user interfaces.

(React là một thư viện JavaScript để xây dựng giao diện người dùng.)

It is called JSX, and it is a syntax extension to JavaScript.

(Nó được gọi là JSX, và nó là một phần mở rộng cú pháp cho JavaScript.)

JSX may remind you of a template language, but it comes with the full power of JavaScript.

(JSX có thể nhắc bạn về một ngôn ngữ mẫu ,nhưng nó mang toàn bộ sức mạnh của javascript)

JSX produces React “elements”.

(JSX tạo ra các “phần tử” React)

## Example



## Why JSX ?

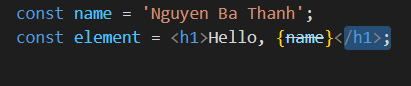
React doesn’t require using JSX, but most people find it helpful as a visual aid when working with UI inside the JavaScript code. It also allows React to show more useful error and warning messages.

(React không yêu cầu sử dụng JSX, nhưng hầu hết mọi người đều thấy nó hữu ích như một công cụ hỗ trợ trực quan khi làm việc với giao diện người dùng bên trong mã JavaScript. Nó cũng cho phép React hiển thị các thông báo lỗi và cảnh báo hữu ích hơn.)

## Embedding Expressions in JSX

declare a variable called name and then use it inside JSX by wrapping it in curly braces:

( khai báo một biến được gọi là tên và sau đó sử dụng nó bên trong JSX bằng cách đặt nó trong dấu ngoặc nhọn )



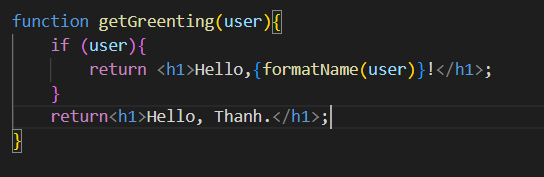
## JSX is an Expression Too

After compilation, JSX expressions become regular JavaScript function calls and evaluate to JavaScript objects.

(Sau khi biên dịch, các biểu thức JSX trở thành các lệnh gọi hàm JavaScript thông thường và đánh giá thành các đối tượng JavaScript.)

This means that you can use JSX inside of if statements and for loops, assign it to variables, accept it as arguments, and return it from functions:

(Điều này có nghĩa là bạn có thể sử dụng JSX bên trong các câu lệnh if và vòng lặp for, gán nó cho các biến, chấp nhận nó dưới dạng đối số và trả về nó từ các hàm: )



## Specifying Attributes with JSX

You may use quotes to specify string literals as attributes:

(Bạn có thể sử dụng dấu ngoặc kép để chỉ định các ký tự chuỗi làm thuộc tính: )



You may also use curly braces to embed a JavaScript expression in an attribute:

(Bạn cũng có thể sử dụng dấu ngoặc nhọn để nhúng biểu thức JavaScript vào một thuộc tính: )



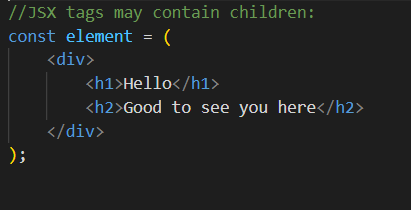
## Specifying Children with JSX

If a tag is empty, you may close it immediately with />, like XML:

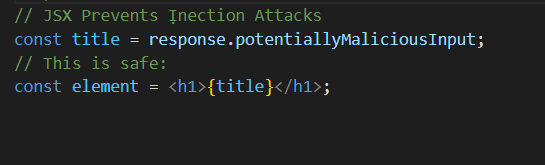
(Nếu một thẻ trống, bạn có thể đóng nó ngay lập tức bằng />XML: )

JSX tags may contain children:

(Các thẻ JSX có thể chứa con: )



**JSX Prevents Injection Attacks**



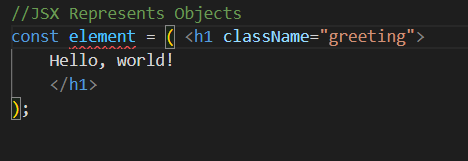
By default, React DOM [escapes](https://stackoverflow.com/questions/7381974/which-characters-need-to-be-escaped-on-html) any values embedded in JSX before rendering them. Thus it ensures that you can never inject anything that’s not explicitly written in your application. Everything is converted to a string before being rendered. This helps prevent [XSS (cross-site-scripting)](https://en.wikipedia.org/wiki/Cross-site_scripting) attacks.

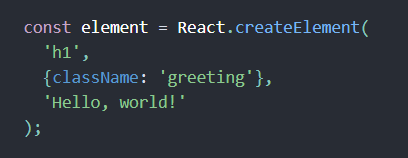
(Theo mặc định, React DOM [thoát khỏi](https://stackoverflow.com/questions/7381974/which-characters-need-to-be-escaped-on-html) bất kỳ giá trị nào được nhúng trong JSX trước khi hiển thị chúng. Vì vậy, nó đảm bảo rằng bạn không bao giờ có thể chèn bất cứ thứ gì không được viết rõ ràng trong ứng dụng của bạn. Mọi thứ được chuyển đổi thành một chuỗi trước khi được hiển thị. Điều này giúp ngăn chặn các cuộc tấn công [XSS (cross-site-scripting)](https://en.wikipedia.org/wiki/Cross-site_scripting) .)

## JSX Represents Objects

Babel compiles JSX down to React.createElement() calls.  
 (Babel biên dịch JSX thành React.createElement()các cuộc gọi.)

Example:





React.createElement() performs a few checks to help you write bug-free code but essentially it creates an object like this:

(React.createElement()thực hiện một số kiểm tra để giúp bạn viết mã không có lỗi nhưng về cơ bản nó tạo ra một đối tượng như thế này:)



These objects are called “React elements”. You can think of them as descriptions of what you want to see on the screen. React reads these objects and uses them to construct the DOM and keep it up to date.

(Những đối tượng này được gọi là “phần tử React”. Bạn có thể coi chúng như mô tả về những gì bạn muốn xem trên màn hình. React đọc các đối tượng này và sử dụng chúng để xây dựng DOM và cập nhật nó.)

# Chapter 2: Rendering Elements

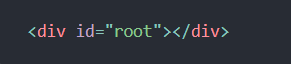
Unlike browser DOM elements, React elements are plain objects, and are cheap to create. React DOM takes care of updating the DOM to match the React elements.

(Không giống như các phần tử DOM của trình duyệt, các phần tử React là các đối tượng đơn giản và rẻ để tạo. React DOM đảm nhận việc cập nhật DOM để phù hợp với các phần tử React.)

## Rendering an Element into the DOM

Let’s say there is a <div> somewhere in your HTML file:

(Giả sử có một <div>nơi nào đó trong tệp HTML của bạn: )



We call this a “root” DOM node because everything inside it will be managed by React DOM.

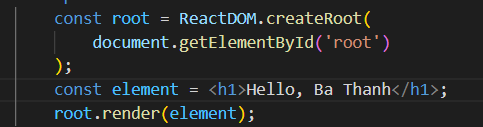
(Chúng tôi gọi đây là nút DOM “gốc” vì mọi thứ bên trong nó sẽ được quản lý bởi React DOM.)

Applications built with just React usually have a single root DOM node. If you are integrating React into an existing app, you may have as many isolated root DOM nodes as you like.

(Các ứng dụng được xây dựng chỉ với React thường có một nút DOM gốc duy nhất. Nếu bạn đang tích hợp React vào một ứng dụng hiện có, bạn có thể có nhiều nút DOM gốc riêng biệt tùy thích.)

To render a React element, first pass the DOM element to [ReactDOM.createRoot()](https://reactjs.org/docs/react-dom-client.html#createroot), then pass the React element to root.render():

(Để hiển thị một phần tử React, trước tiên hãy chuyển phần tử DOM tới [ReactDOM.createRoot()](https://reactjs.org/docs/react-dom-client.html#createroot), sau đó chuyển phần tử React vào root.render(): )



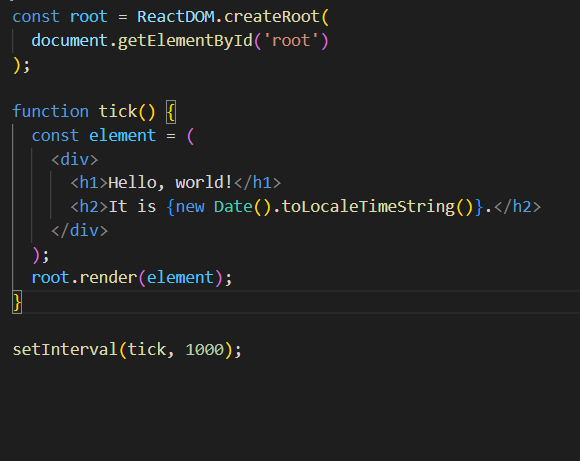
## Updating the Rendered Element

React elements are [immutable](https://en.wikipedia.org/wiki/Immutable_object). Once you create an element, you can’t change its children or attributes. An element is like a single frame in a movie: it represents the UI at a certain point in time.

(Các phần tử React là [bất biến](https://en.wikipedia.org/wiki/Immutable_object) . Sau khi tạo một phần tử, bạn không thể thay đổi phần tử hoặc thuộc tính của phần tử đó. Một phần tử giống như một khung hình duy nhất trong phim: nó đại diện cho giao diện người dùng tại một thời điểm nhất định.)

With our knowledge so far, the only way to update the UI is to create a new element, and pass it to root.render().

(Với kiến ​​thức của chúng tôi cho đến nay, cách duy nhất để cập nhật giao diện người dùng là tạo một phần tử mới và chuyển nó cho root.render().    )



## React Only Updates What’s Necessary

React DOM compares the element and its children to the previous one, and only applies the DOM updates necessary to bring the DOM to the desired state.

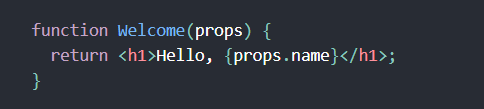
(React DOM so sánh phần tử và phần tử con của nó với phần tử trước đó và chỉ áp dụng các bản cập nhật DOM cần thiết để đưa DOM về trạng thái mong muốn.)

# Chapter 3: Components and Props

## Function and Class Components

The simplest way to define a component is to write a JavaScript function:

(Cách đơn giản nhất để xác định một thành phần là viết một hàm JavaScript)

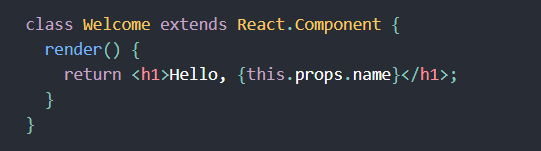


This function is a valid React component because it accepts a single “props” (which stands for properties) object argument with data and returns a React element. We call such components “function components” because they are literally JavaScript functions.

(Hàm này là một thành phần React hợp lệ vì nó chấp nhận một đối số đối tượng “props” (viết tắt của thuộc tính) với dữ liệu và trả về một phần tử React. Chúng tôi gọi các thành phần như vậy là “các thành phần chức năng” bởi vì chúng thực sự là các hàm JavaScript.)

You can also use an [ES6 class](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Classes) to define a component:

(Bạn cũng có thể sử dụng một [lớp ES6](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Classes) để xác định một thành phần)



The above two components are equivalent from React’s point of view.

(Hai thành phần trên là tương đương nhau theo quan điểm của React.)

## Rendering a Component

Previously, we only encountered React elements that represent DOM tags

## Composing Components

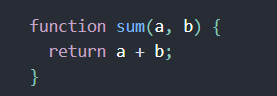
Components can refer to other components in their output. This lets us use the same component abstraction for any level of detail. A button, a form, a dialog, a screen: in React apps, all those are commonly expressed as components.

## Extracting Components

## Props are Read-Only

Whether you declare a component [as a function or a class](https://reactjs.org/docs/components-and-props.html#function-and-class-components), it must never modify its own props. Consider this sum function:

(Cho dù bạn khai báo một thành phần [là một hàm hay một lớp](https://reactjs.org/docs/components-and-props.html#function-and-class-components) , nó không bao giờ được sửa đổi các đạo cụ của chính nó. Hãy xem xét sum chức năng này:)

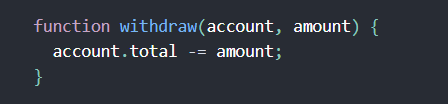


Such functions are called [“pure”](https://en.wikipedia.org/wiki/Pure_function) because they do not attempt to change their inputs, and always return the same result for the same inputs.

(Các hàm như vậy được gọi là [“thuần túy”](https://en.wikipedia.org/wiki/Pure_function) bởi vì chúng không cố gắng thay đổi đầu vào của mình và luôn trả về cùng một kết quả cho các đầu vào giống nhau.)

In contrast, this function is impure because it changes its own input:

(Ngược lại, hàm này không tinh khiết vì nó thay đổi đầu vào của chính nó:)



# Chapter 4: State and Lifecycle

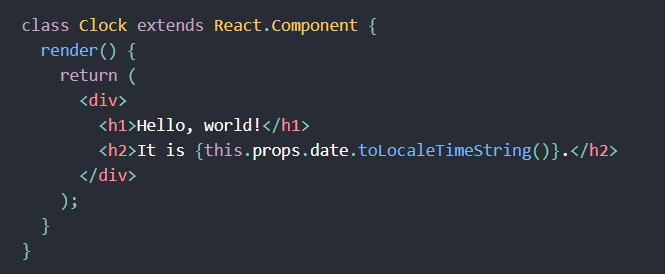
## Converting a Function to a Class

You can convert a function component like Clock to a class in five steps:

1. Create an [ES6 class](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Classes), with the same name, that extends React.Component.
2. Add a single empty method to it called render().
3. Move the body of the function into the render() method.
4. Replace props with this.props in the render() body.
5. Delete the remaining empty function declaration.

(Bạn có thể chuyển đổi một thành phần hàm như Clockthành một lớp trong năm bước:

1. Tạo một [lớp ES6](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Classes) , có cùng tên, mở rộng React.Component.
2. Thêm một phương thức trống duy nhất vào nó được gọi render().
3. Di chuyển phần thân của hàm vào render()phương thức.
4. Thay thế propsbằng this.propstrong render()cơ thể.
5. Xóa phần khai báo chức năng trống còn lại.)



## Adding Local State to a Class

We will move the date from props to state in three steps:

Chúng tôi sẽ chuyển date từ đạo cụ sang trạng thái trong ba bước:

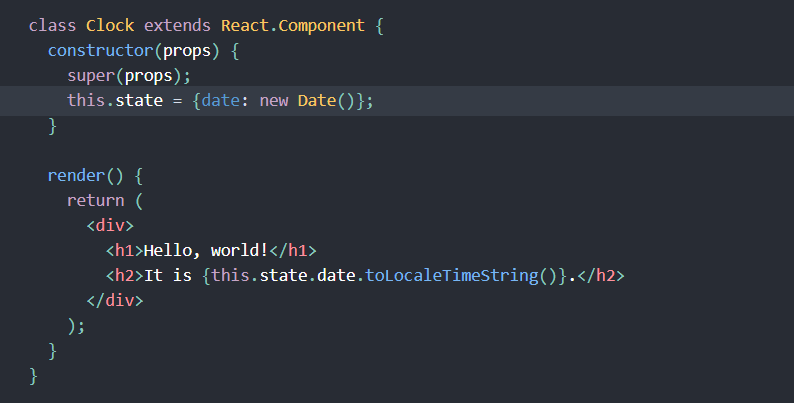
1. Replace this.props.date with this.state.date in the render() method:

.Thay thế this.props.date bằng this.state.date trong render()phương thức:



1. Add a [class constructor](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Classes#Constructor) that assigns the initial this.state

Thêm một phương thức [khởi tạo lớp](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Classes#Constructor) gán giá trị ban đầu this.state



Note how we pass props to the base constructor:

Lưu ý cách chúng tôi truyền propscho hàm tạo cơ sở:

## Adding Lifecycle Methods to a Class

## Using State Correctly

## The Data Flows Down