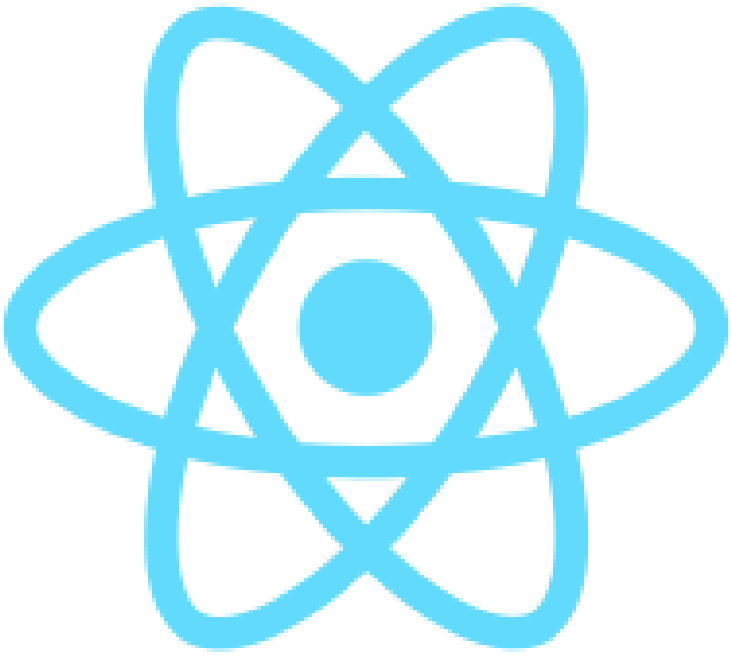


React Report

Giới thiệu React

Trần Huy Công - RikkeiSoft

React là gì ?

React.js là 1 thư viện JavaScript tạo ra bởi Facebook.

http://facebook.github.io/react/

Như khái niệm trên trang web chính thức "A JavaScript library for building user interface", React.js là một thư viện sinh ra để xây dựng giao diện người dùng (UI). Nó không phải là Framework mà chỉ là thư viện.

2 đặc điểm chính của React, đó là virtual DOM và one-way data binding. Nếu như virtual DOM cung cấp 1 cách tiếp cận hoàn toàn mới để làm việc với HTML DOM thì one-way data binding lại là 1 phương thức quản lý luồng dữ liệu rất dễ hiểu và dễ quản lý.

Virtual DOM

DOM ảo không được phát minh ra bởi React, mà React sử dụng nó.

DOM ảo là một bản sao chép trừu tượng của DOM thật (HTML DOM). Bạn có thể tưởng tượng nó giống như một bản thiết kế, chứa các chi tiết cần thiết để cấu hình lên một DOM. Ví dụ, thay vì tạo một thẻ thật chứa các thẻ <ul> bên trong, nó sẽ tạo một div object chứa ul object bên trong. Cụ thể ở trong React sẽ là các React.div và React.ul. Khi tương tác, ta có thể tương tác với các object đó rất nhanh mà không phải động tới DOM thật hoặc thông qua DOM API.

Đối tượng DOM ảo có các thuộc tính tương tự như đối tượng DOM thật, nhưng không có quyền để thay đổi trực tiếp những thứ trên màn hình.

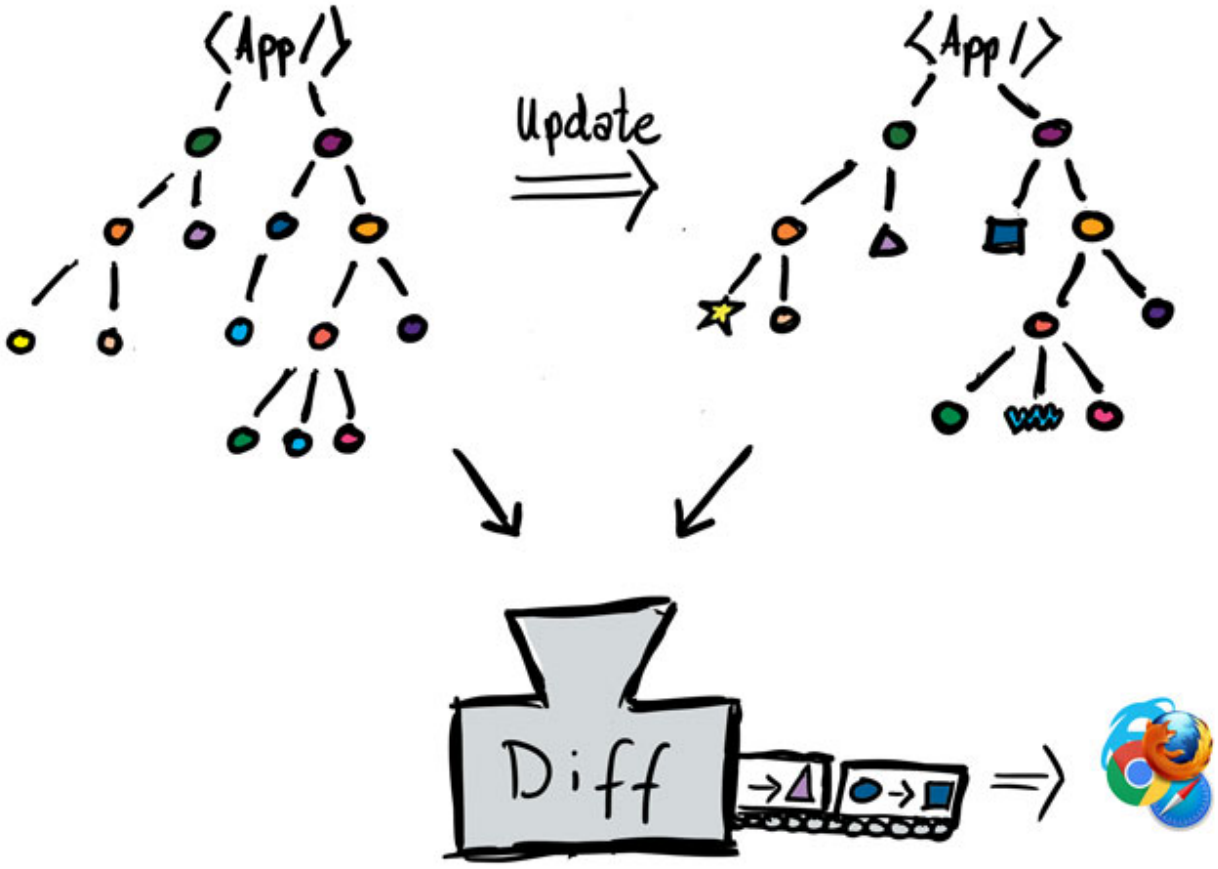
Thao tác với DOM thì chậm. Còn thao tác với DOM ảo nhanh hơn, bởi vì không có gì phải vẽ trên màn hình. Có thể hình dung thao tác với DOM ảo giống như chỉnh sửa

một bản thiết kế, trái ngược với việc di chuyển các căn phòng trong một ngôi nhà thực sự.

Khi chúng ta render một phần tử JSX, mọi đối tượng DOM ảo sẽ được cập nhật.

Điều này nghe có vẻ không hiệu quả, nhưng chi phí là không đáng kể bởi vì DOM ảo có thể cập nhật cực nhanh.

Khi DOM ảo đã cập nhật xong, React sẽ so sánh DOM ảo với một *snapshot (phiên bản)* của nó được tạo ngay trước khi cập nhật.



Bằng cách so sánh DOM ảo mới với một phiên bản của chính nó ngay trước khi cập nhật, React sẽ tìm ra *chính xác các đối tượng DOM ảo nào đã thay đổi.* Quá trình này gọi là "diffing".

Khi React biết các đối tượng DOM ảo nào được thay đổi, React sẽ cập nhật các đối tượng này, *và chỉ các đối tượng này*, trên DOM thật. Trong ví dụ ở phần trên, React đủ thông minh để chỉ rebuild lại mục đã được đánh dấu và giữ nguyên phần còn lại của danh sách.

Điều này tạo nên sự khác biệt lớn! React có thể chỉ cập nhật những phần cần thiết của DOM. Sự nổi tiếng về hiệu năng của React chủ yếu đến từ sự đổi mới này.

JSX

JSX = JS + XML

JSX là cú pháp để thay thế Javascript với cách viết gần giống XML.

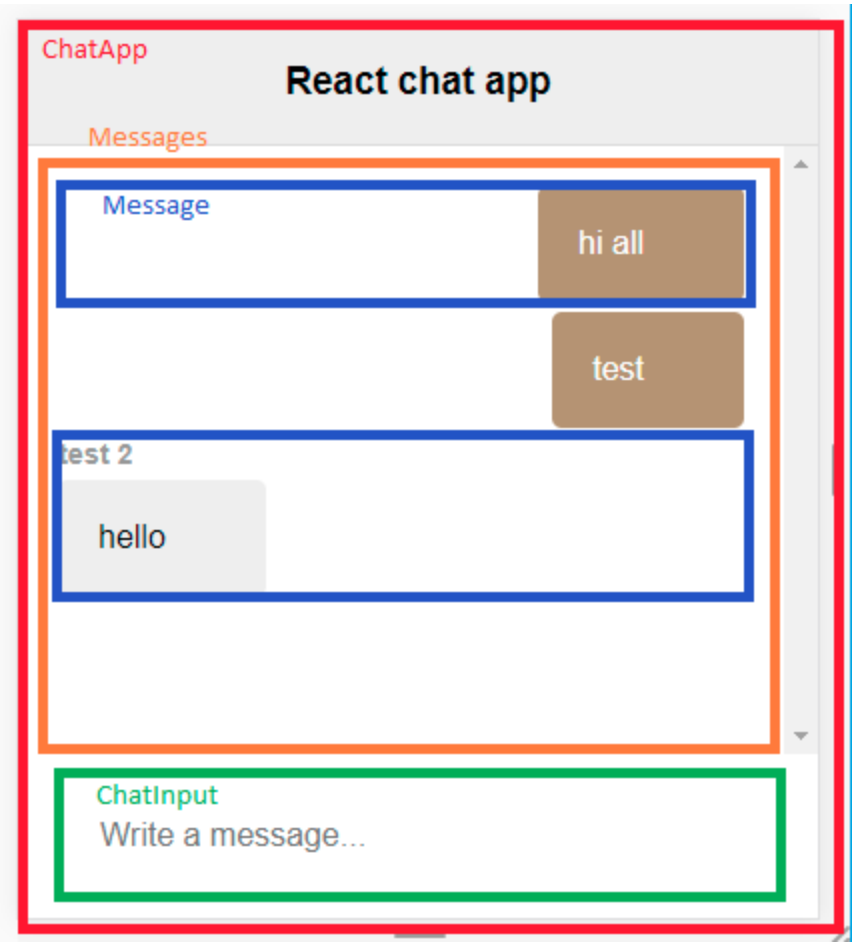
```
const element = (
  <div>
    <h1>Hello!</h1>
    <h2>Good to see you here.</h2>
  </div>
);
```

Lợi thế khi sử dụng JSX trong react:

1. Cú pháp cây gần giống XML, giúp dễ dàng định nghĩa và quản lí các component. Ví dụ thay vì viết: `React.DOM.div(null, 'hello')` ta có thể viết `<div>hello</div>`.
2. Không làm thay đổi ngữ nghĩa của Javascript

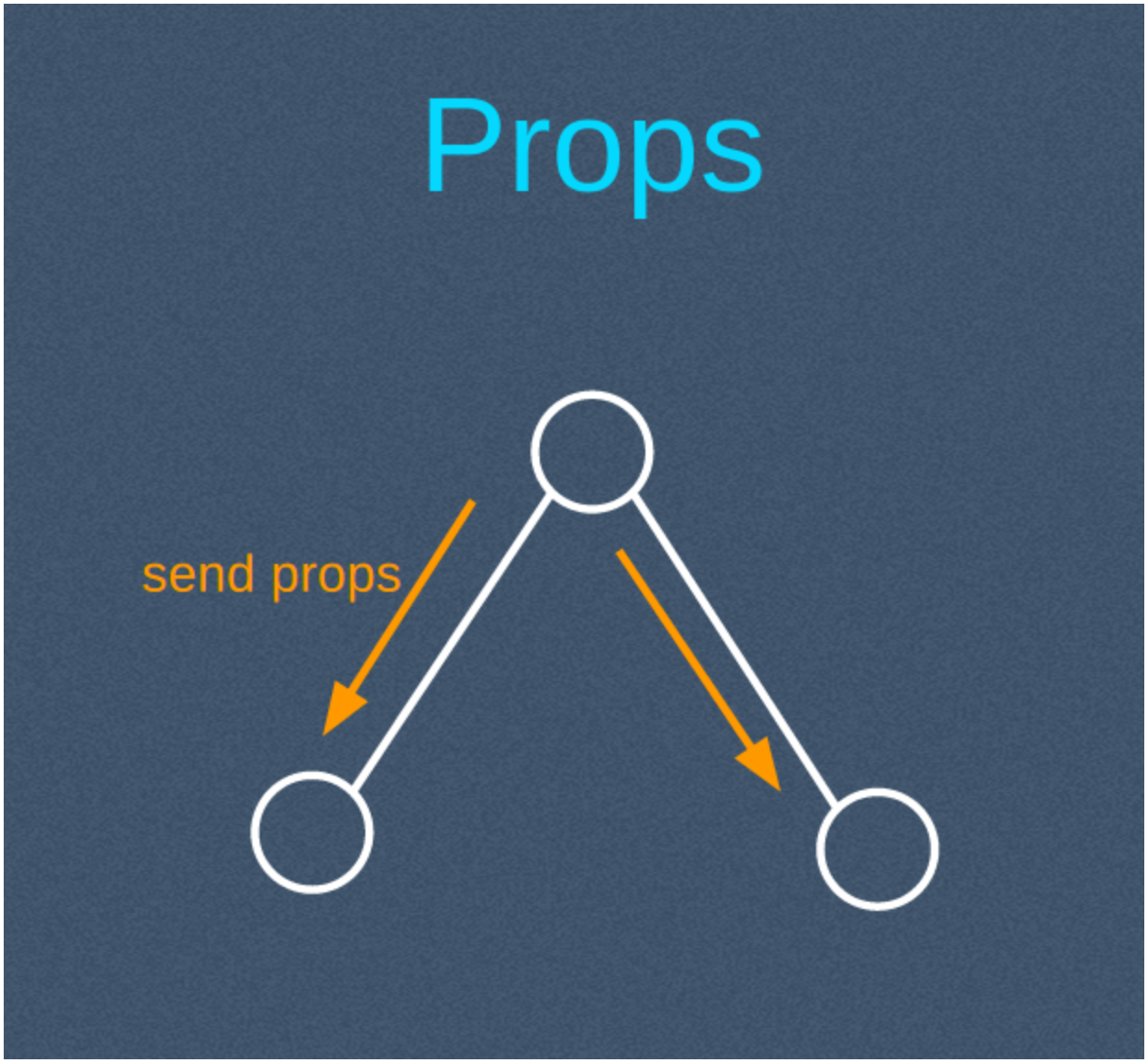
Components

Components là các thành phần tạo nên giao diện. Trong React, components được tạo nên với tư tưởng chia nhỏ nhất có thể, để đảm bảo tính độc lập, khả năng tái sử dụng và hiệu năng tốt.



Props

Props là các thuộc tính của 1 component, được truyền từ component cha. Khác với state, props không thể bị thay đổi bởi component sử dụng nó.



Ví dụ:

Messgages component:

```
<Message
       key={i}
       username={message.username}
       message={message.message}
       fromMe={message.fromMe}
  />
```

Message conponent:

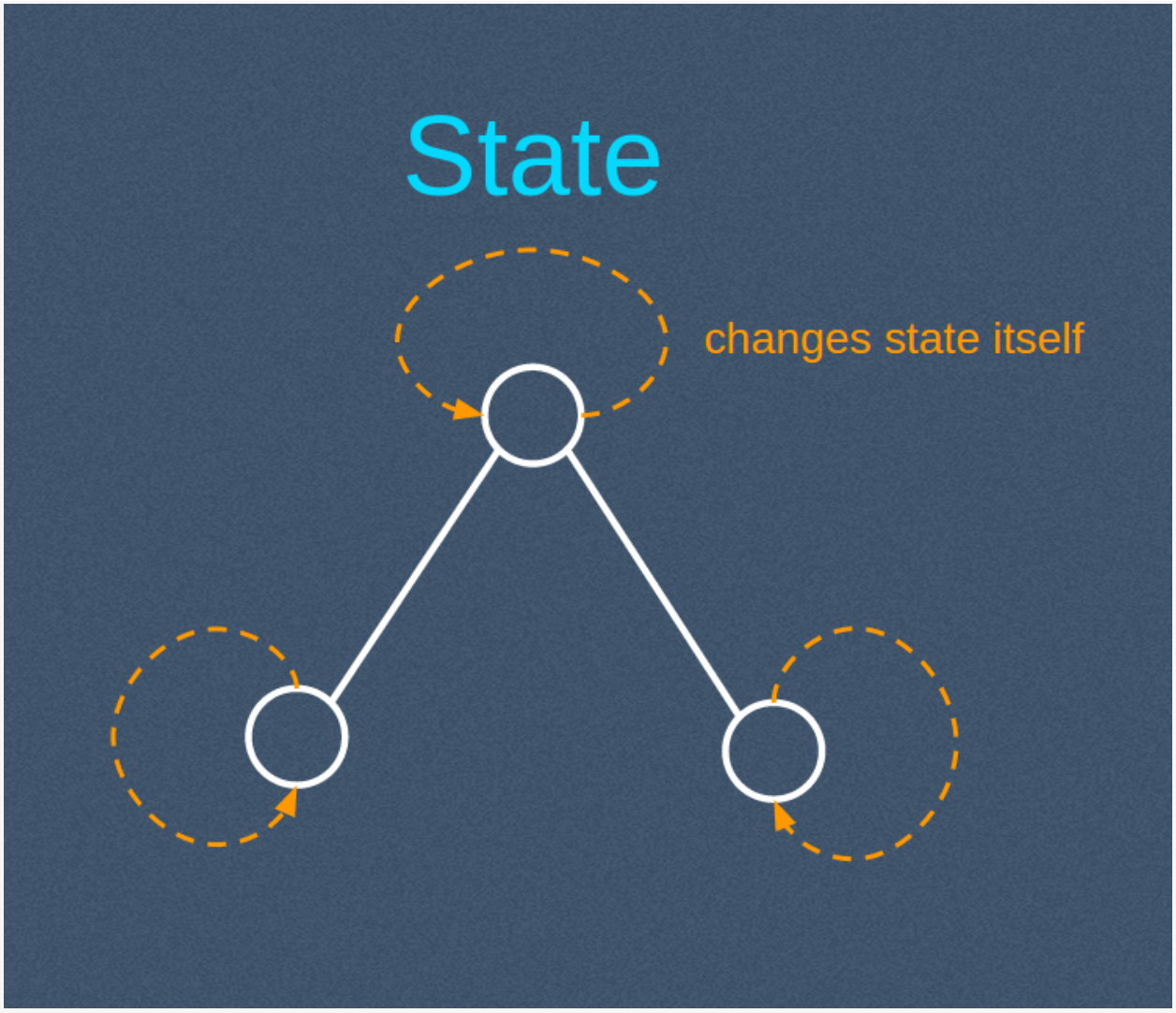
```
class Message extends React.Component{
  render(){
    const fromMe = this.props.fromMe ? 'from-me' : '';
    return(
      <div className={`message ${fromMe}`}>
```

```
        <div className="username">
          {this.props.username}
        </div>
        <div className="message-body">
          {this.props.message}
        </div>
      </div>
    )
  }
}
```

Nguyên tắc trong React, props là bất biến. Trong trường hợp props là hàm hoặc class thì nó cũng không được thay đổi khi gọi ra trong components. Khi muốn sử dụng và thay đổi các giá trị ta dùng đến state.

State

State là trạng thái của 1 component, nó có thể được khởi tạo bên trong component hoặc được truyền từ component cha. State có thể được thay đổi bên trong component.



Thông thường các components không có state và được gọi là stateless. Nếu nó sử dụng state thì component đó được gọi là stateful.

```
class ChatApp extends Component{

  constructor(props){
    super(props);
    this.state= {
      messages: []
    };
    …
  }

  sendHandler = (message)=>{
```

```
    const messageObject = {
      username: this.props.username,
      message
    }

    this.socket.emit('client:message', messageObject);
    messageObject.fromMe = true;
    this.addMessage(messageObject);
  };

  addMessage = (message)=>{
    const messages = this.state.messages;
    messages.push(message);
    this.setState({ message });
  };


  render(){
    return(
      <div className="container">
        <h3>React chat app</h3>
        <Messages messages={this.state.messages} />
        <ChatInput onSend={this.sendHandler} />
      </div>
    )
  }
}
```

Như vậy, Props và State làm những điều tương tự nhưng được sử dụng theo những cách khác nhau. Phần lớn các components của bạn sẽ stateless. Các props được sử dụng để chuyển dữ liệu từ components cha sang components con hoặc bằng chính nó. Chúng không thay đổi và do đó sẽ không bị thay đổi. State được sử dụng cho các dữ liệu có thể thay đổi hoặc dữ liệu sẽ thay đổi. Điều này đặc biệt hữu ích cho đầu vào của người dùng. Như các thanh tìm kiếm. Người dùng sẽ gõ dữ liệu và điều này sẽ cập nhật những gì họ nhìn thấy.

Vòng đời của component - hàm thường dùng

1. componentWillMount – được gọi một lần trước khi render diễn ra.

2. componentDidMount – được gọi một lần sau khi render.
3. shouldComponentUpdate – trả lại giá trị quyết định xem component có được update không.
4. componentWillUnmount – được gọi trước khi tách component.

Events

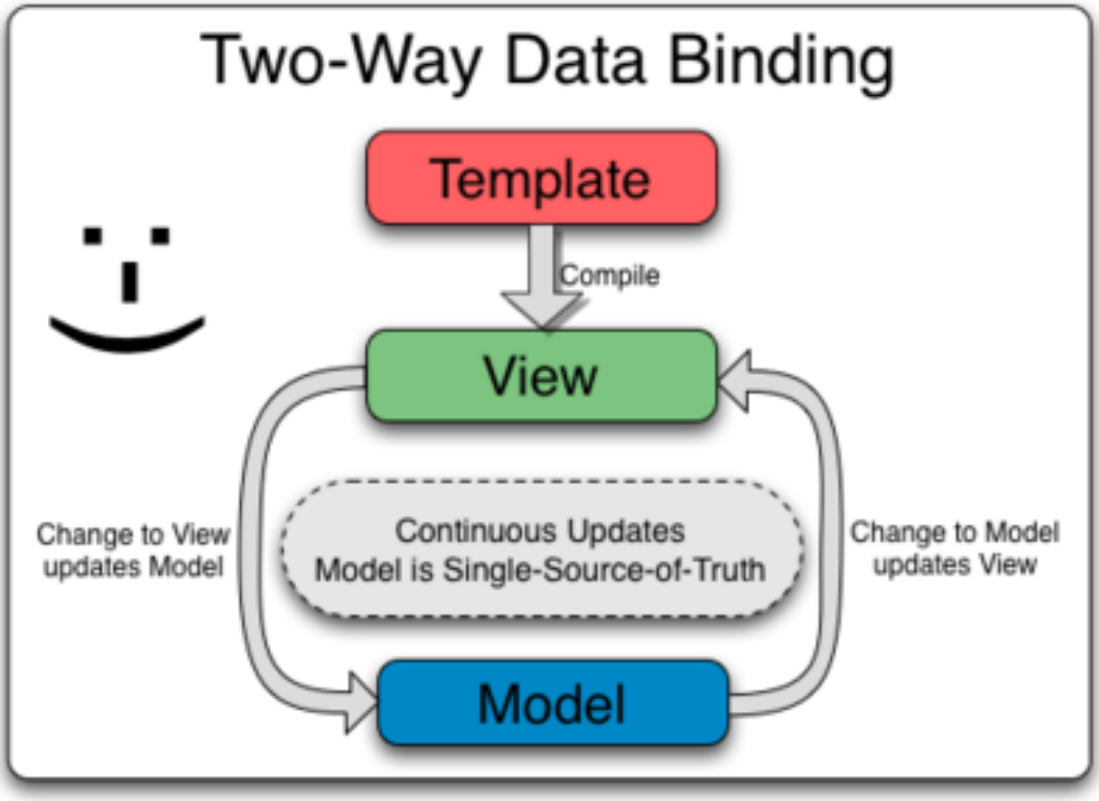
React cũng có một hệ thống events sử dụng được ở các browser khác nhau. Những events này được gắn liền như thành phần của components và có thể kích hoạt các hàm.

```
<button onClick={doAction}>
  Activate Lasers
</button>
```

One-way data binding

Khái niệm data binding

Data binding là quá trình thiết lập kết nối giữa giao diện ứng dụng (application UI) và các chức năng ở tầng logic của ứng dụng (business logic). Sự kết nối này đảm bảo rằng mỗi thay đổi bên phía giao diện (application UI) sẽ tạo ra thay đổi đối với các dữ liệu, dựa theo logic của hệ thống (business logic). Khái niệm này thường được nói đến khi làm việc với các dịch vụ web (WPS - Web Processing Service). 1 ví dụ đơn giản là nếu chúng ta thay đổi nội dung nhập vào cho 1 khung tìm kiếm, các kết quả trả về sẽ được thay đổi dựa theo giá trị mà chúng ta nhập vào.



Ví dụ về data binding là mô hình two-way data binding trong Angular, đầu tiên AngularJS compile Template ra ngoài View cho người dùng thấy được (chẳng hạn như là 1 ô textbox). Khi có bất kì thày đổi trên View này, chẳng hạn như trong ô textbox mình sẽ nhập là abc. Thì tự động dữ liệu abc này sẽ ngay lập tự được đồng bộ vào bên trong Model. Và tiếp theo bên trong Model nếu có bất kỳ sự thay đổi nào sẽ ngay lập tức được đồng bộ ngược lại ra View. Vì thế dữ liệu được trung chuyển từ Model sang View hay từ View sang Model là một thể thống nhất.

Ưu điểm của cách tiếp cận này là giúp cho lập trình viên dễ dàng hơn khi xây dựng giao diện, bởi khi cần cập nhật chức năng, họ chỉ cần thêm vào các hàm quản lý là xong. Họ không phải suy nghĩ về tổ chức các thành phần trên trang web ngay từ đầu. Bởi chúng vốn dĩ không cần có tổ chức rõ ràng.

Tuy nhiên, nhược điểm của cách làm này đến từ sự thiếu tổ chức của nó. Các lập trình viên có thể thêm bớt các chức năng thoải mái mà không cần có tổ chức cụ thể. Điều

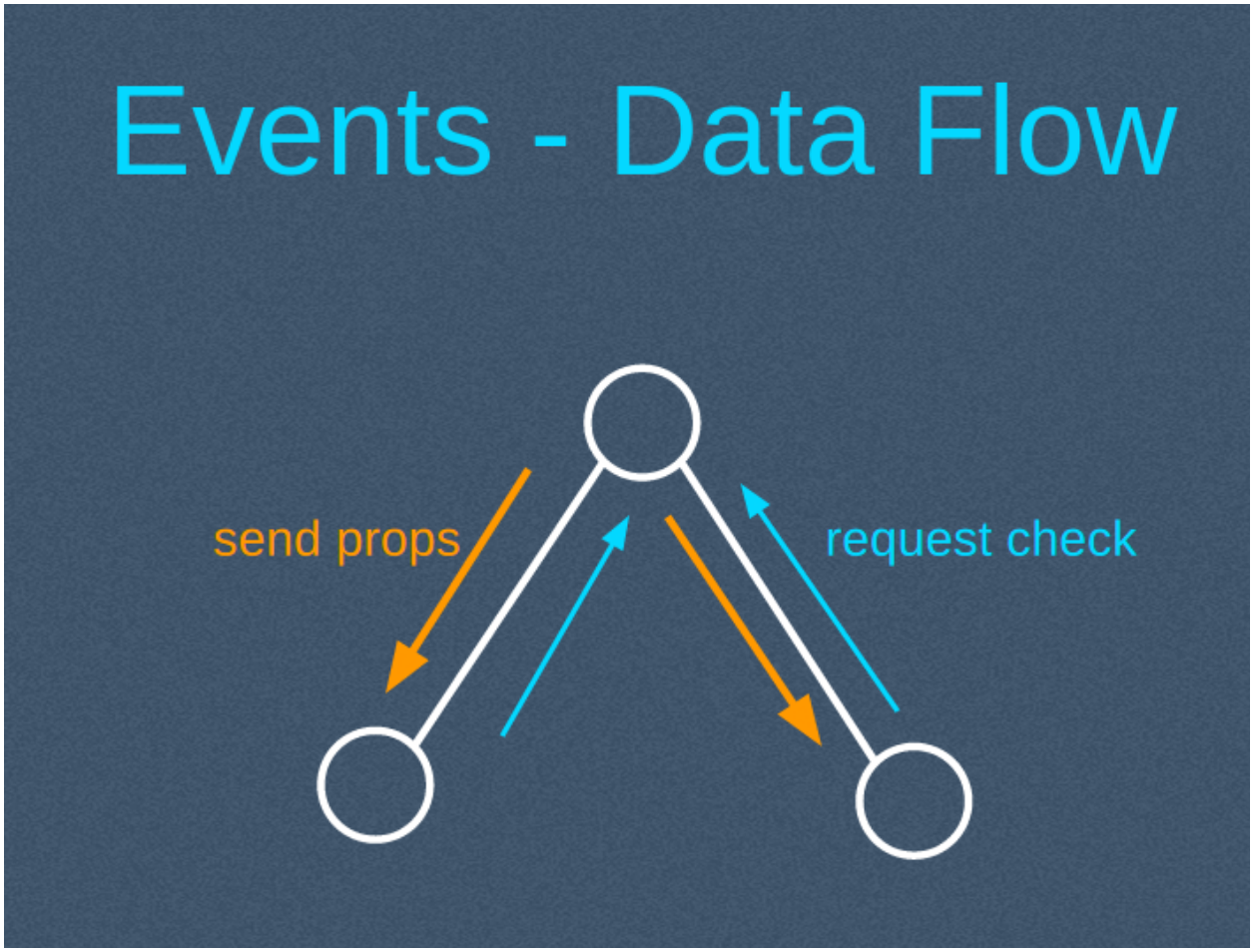
này gây ra khó khăn khi hệ thống lớn lên, khó quản lý được luồng dữ liệu cũng như các thành phần tác động lên 1 đối tượng trên website. Khi trang web lớn lên thì việc kiểm tra lỗi và sửa lỗi trở nên rất khó khăn.

Data binding trong React - One-way data binding

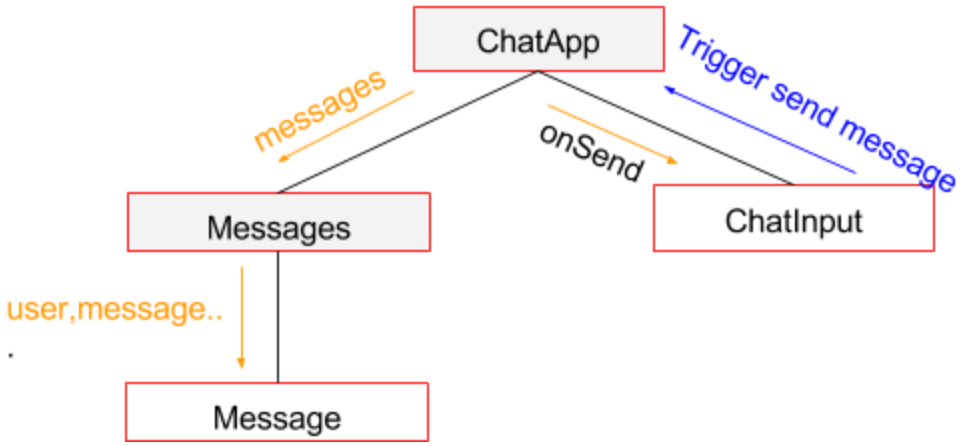
Rất đơn giản và rõ ràng, dữ liệu trong React sẽ chỉ được truyền theo 1 chiều duy nhất, đó là từ component cha đến component con thông qua props. Không có chiều ngược lại (thực ra là bạn có thể làm ngược lại nhưng như vậy là trái với quan điểm của React).

Câu hỏi đặt ra, liệu dữ liệu có thực sự được truyền theo 1 chiều duy nhất? Thường là không! Nhưng trong React, chúng ta coi việc truyền "thông tin" từ component con đến component cha là truyền "sự kiện". Trong sự kiện đó component con có thể đính kèm các thông tin của sự kiện (có thể là dữ liệu). Quá trình đó được hiểu là truyền sự kiện, không phải truyền dữ liệu.

Tóm lại, trong React, dữ liệu sẽ được truyền từ trên xuống, và sự kiện được truyền từ dưới lên.



Ví dụ về luồng dữ liệu trong ứng dụng demo :



Kết luận:

React là một thư viện rất thú vị và được phát triển dựa trên rất nhiều cấu trúc phức tạp. Tuy nhiên thư viện này lại rất dễ sử dụng và thêm vào trong nhiều application khác nhau, phù hợp cho xây dựng các ứng dụng quy mô tư nhỏ đến lớn bởi nguyên tắc xây dựng components có thể giúp lập trình viên dễ dàng tiếp cân, quản lí, tái sử dụng... các thành phần của dự án.