**Tìm hiểu Props Validation trong ReactJS**

Trong bài viết này chúng ta sẽ cùng nhau đi **tìm hiểu về props validation**. Trước khi đi qua tìm hiểu về phần này chúng ta cần xem bài viết trước về [props trong ReactJS](https://freetuts.net/props-trong-reactjs-2354.html) để hiểu về nó đã nhé.

**Mục lục**

* [Props validation là gì ?](https://freetuts.net/props-validation-trong-reactjs-2376.html#goto-h2-0)
* [Validating props](https://freetuts.net/props-validation-trong-reactjs-2376.html#goto-h2-1)
  + [Thuộc tính propTypes](https://freetuts.net/props-validation-trong-reactjs-2376.html#goto-h3-0)
  + [Thư viện PropTypes](https://freetuts.net/props-validation-trong-reactjs-2376.html#goto-h3-1)
  + [Ví dụ thực tế](https://freetuts.net/props-validation-trong-reactjs-2376.html#goto-h3-2)

## Props validation là gì ?

**Props**là một phần rất quan trọng được truyền vào React components, các props này thường phải có một kiểu dữ liệu nhất định. Nếu props được truyền đến component có một kiểu mà không mong muốn, component đó sẽ rất khó để kiểm soát và từ đó gây ra bugs.

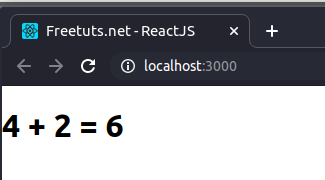
**Props validation** là cách để kiểm soát các vấn đề về props, nó cho phép bạn kiểm tra dữ liệu đầu vào của props trước khi components được render. **Props validation** giúp chúng ta giảm thiểu các lỗi không mong muốn trong quá trình xây dựng ứng dụng React.

Giả sử mình đang xây dựng một component cho phép chúng ta cộng 2 số với nhau.

*Bài viết này được đăng tại [free tuts .net]*

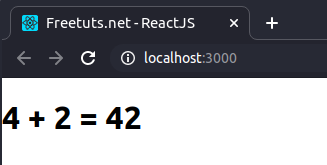
**Trường hợp 1:**Chúng ta sẽ truyền vào đó 2 giá trị là kiểu int, và bạn sẽ nhận được kết quả đúng:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | import React from "react";    class App extends React.Component {    render() {      const {number1, number2} = this.props      return (        <div>          <h1>{number1} + {number2} = {number1 + number2}</h1>        </div>      );    }  }  //Chỉ định props mặc định.  //Ở đây mình cho props number1, number2 là kiểu int  App.defaultProps = {    number1: 4,    number2: 2  }  export default App; |



**Trường hợp 2:** Nếu bạn để props là kiểu string thì component sẽ hiển thị sai kết quả:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | import React from "react";    class App extends React.Component {    render() {      const {number1, number2} = this.props      return (        <div>          <h1>{number1} + {number2} = {number1 + number2}</h1>        </div>      );    }  }  // Chỉ định props mặc định.  // Ở đây mình chỉ định props number1, number 2 là kiểu string.  App.defaultProps = {    number1: '4',    number2: '2'  }  export default App; |



Đây là lỗi mà bạn không mong muốn, khi mà kiểu dữ liệu của props được truyền sai. Trước khi chạy component bạn cần phải kiểm tra kiểu dữ liệu trước, điều này rất quan trọng trong các dự án lớn. Bởi vậy chúng ta sẽ đi tìm hiểu về **Props validation.**

## Validating props

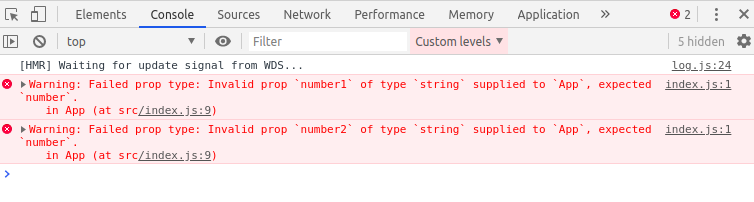
Trong phần trước chúng ta đã biết tại sao cần phải kiểm tra props trước khi khởi chạy, đây là bước quan trọng để chúng ta có thể kiểm tra kiểu và các giá trị của props trong React app. Trong phần này chúng ta sẽ đi tìm hiểu về cách để **kiểm tra props trong một components**.

### Thuộc tính propTypes

React cung cấp cho chúng ta một phương thức để kiểm tra props trong mỗi component. Trong tất cả các React component đều có một thuộc tính có tên propTypes:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | /\*\*   \* FUNCTIONAL COMPONENTS   \*/  function ReactComponent(props) {    // ...render ()...  }    ReactComponent.propTypes = {    // ...chúng ta sẽ viết props type ở đây  }      /\*\*   \* CLASS COMPONENTS: METHOD 1   \*/  class ReactComponent extends React.Component {    // ...render()...  }    ReactComponent.propTypes = {    // ...chúng ta sẽ viết props type ở đây  }      /\*\*   \* CLASS COMPONENTS: METHOD 2   \* Sử dụng class statics...   \*/  class ReactComponent extends React.Component {    // ...component class body here      static propTypes = {       // ...chúng ta sẽ viết props type ở đây    }  } |

Khi xuất hiện lỗi, chúng ta có thể thấy được lỗi được hiển thị trong console, nó chỉ hiển thị trong môi trường phát triển, khi deloy dự án lên thì lỗi này sẽ không được hiển thị.



### Thư viện PropTypes

Trong các phiên bản cũ, người ta thường dùng React.PropTypes làm cách giá trị truyền vào trong propTypes, ví dụ như :

|  |  |
| --- | --- |
| 1  2  3  4 | App.propTypes = {    number1: React.PropTypes.number,    number2: React.PropTypes.number  } |

Kể từ phiên bản React v15.5 trở đi React.PropTypes đã bị xóa bỏ và thay vào đó là thư viện **PropsTypes**, cho phép chúng ta chỉ định kiểu dữ liệu mà bạn mong muốn cho props. Vì bị tách ra thành thư viện nên bạn cần phải cài thêm vào React bằng npm:

|  |  |
| --- | --- |
| 1 | npm i prop-types |

Để sử dụng bạn chỉ cần import nó vào:

|  |  |
| --- | --- |
| 1 | import PropTypes from 'prop-types'; |

Đây là hầu hết các thuộc tính PropsType có trong thư viện:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97 | import PropTypes from "prop-types";    MyComponent.propTypes = {    // Kiểu dữ liệu    optionalArray: PropTypes.array,    optionalBool: PropTypes.bool,    optionalFunc: PropTypes.func,    optionalNumber: PropTypes.number,    optionalObject: PropTypes.object,    optionalString: PropTypes.string,    optionalSymbol: PropTypes.symbol,      // Bất cứ thứ gì có thể render như : numbers, strings, elements hoặc array    optionalNode: PropTypes.node,      //  React Element    optionalElement: PropTypes.element,      // React Element Type    optionalElementType: PropTypes.elementType,      // Instanace    optionalMessage: PropTypes.instanceOf(Message),      // Gía trị của props bao gồm    optionalEnum: PropTypes.oneOf(["News", "Photos"]),      // Một object có thể bao gồm nhiều kiểu    optionalUnion: PropTypes.oneOfType([      PropTypes.string,      PropTypes.number,      PropTypes.instanceOf(Message)    ]),      // Một mảng chứa giá trị là các kiểu    optionalArrayOf: PropTypes.arrayOf(PropTypes.number),      // Một object có thuộc tính là kiểu    optionalObjectOf: PropTypes.objectOf(PropTypes.number),      // Một object mà mỗi phần có kiểu dữ liệu riêng    optionalObjectWithShape: PropTypes.shape({      color: PropTypes.string,      fontSize: PropTypes.number    }),      // Một object có cảnh báo về các thuộc tính bổ sung    optionalObjectWithStrictShape: PropTypes.exact({      name: PropTypes.string,      quantity: PropTypes.number    }),      // Bạn có thể xâu chuỗi bất kỳ câu hỏi nào ở trên với `isRequired` để đảm bảo cảnh báo    // được hiển thị nếu prop không được cung cấp.    requiredFunc: PropTypes.func.isRequired,      // Bất kì kiểu dữ liệu nào nhưng nó phải tồn tại..    requiredAny: PropTypes.any.isRequired,      //Chỉ định props validation tùy chỉnh    customProp: function(props, propName, componentName) {      if (!/matchme/.test(props[propName])) {        return new Error(          "Invalid prop `" +            propName +            "` supplied to" +            " `" +            componentName +            "`. Validation failed."        );      }    },      // Bạn cũng có thể cung cấp trình xác nhận tùy chỉnh cho `ArrayOf` và` objectOf`.    // Nó sẽ trả về một đối tượng Error nếu xác nhận thất bại. Trình xác nhận    // sẽ được gọi cho mỗi khóa trong mảng hoặc đối tượng. Hai cái đầu tiên    // đối số của trình xác nhận là chính mảng hoặc đối tượng và    // khóa hiện tại của mục.    customArrayProp: PropTypes.arrayOf(function(      propValue,      key,      componentName,      location,      propFullName    ) {      if (!/matchme/.test(propValue[key])) {        return new Error(          "Invalid prop `" +            propFullName +            "` supplied to" +            " `" +            componentName +            "`. Validation failed."        );      }    })  }; |

Phần dưới mình sẽ đi vào ví dụ cụ thể để hình dung rõ hơn chứ ngồi học lý thuyết khá buồn ngủ :'>

### Ví dụ thực tế

**Ví dụ 1:**Tiếp tục với ví dụ ở trên đầu bài viết là cộng 2 số lại với nhau chúng ta sẽ thực hiện kiểm tra bằng cách sử dụng:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | import React from "react";  import PropTypes from 'prop-types';    class App extends React.Component {    render() {      const {number1, number2} = this.props      return (        <div>          <h1>{number1} + {number2} = {number1 + number2}</h1>        </div>      );    }  }  App.defaultProps = {    number1: 4,    number2: 2  }  //Kiểm tra dữ liệu sử dụng PropTypes.  App.propTypes = {    number1: PropTypes.number,    number2: PropTypes.number  }  export default App; |

Các bạn lưu ý phải cài đặt thư viện **PropTypes** trước nhé, lúc này dữ liệu đã được kiểm tra và sẽ hiển thị lỗi ra console như bên trên mình đã đề cập.

**Ví dụ 2: T**rong ví dụ này mình sẽ hiển thị ra thông tin của những chiếc điện thoại của Apple bao gồm name, type, public\_year, storage,...

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54 | import React from "react";  import PropTypes from 'prop-types';    class App extends React.Component {    render() {        return (        <div>          <h1>{this.props.name}</h1>          <ul>            <li>{this.props.type}</li>            <li>{this.props.public\_year}</li>            <li>{this.props.storage}</li>          </ul>        </div>      );    }  }   // Đúng kiểu dữ liệu  App.defaultProps = {    name: 'iPhone Xs Max',    type: 'iPhone',    public\_year: 2018,    storage: '64 GB'  }     // Đúng kiểu dữ liệu  App.defaultProps = {    name: 'iPad Mini 2019',    type: 'iPad',    public\_year: 2019,    storage: 64  }        // Sai kiểu dữ liệu vì type phải là các giá  // trị như iPhone, Ipad, Mac, SmartWatch  App.defaultProps = {    name: 'Airpods 2',    type: 'Airpod',    public\_year: 2019  }    //Chỉ định validation props  App.propTypes = {    name: PropTypes.string,    type: PropTypes.oneOf(["iPhone", "iPad", "Mac", "SmartWatch"]),    public\_year: PropTypes.oneOfType([      PropTypes.string,      PropTypes.number,    ])  }  export default App; |

Trong bài này chúng ta đã cùng nhau đi tìm hiểu về Props validation trong ReactJS. Đây là kiến thức rất cơ bản về nó nhưng cũng hết sức quan trọng trong quá trình làm việc với ReactJS sau này. Mong rằng bài viết sẽ giúp ích cho bạn.