COMPONENTS

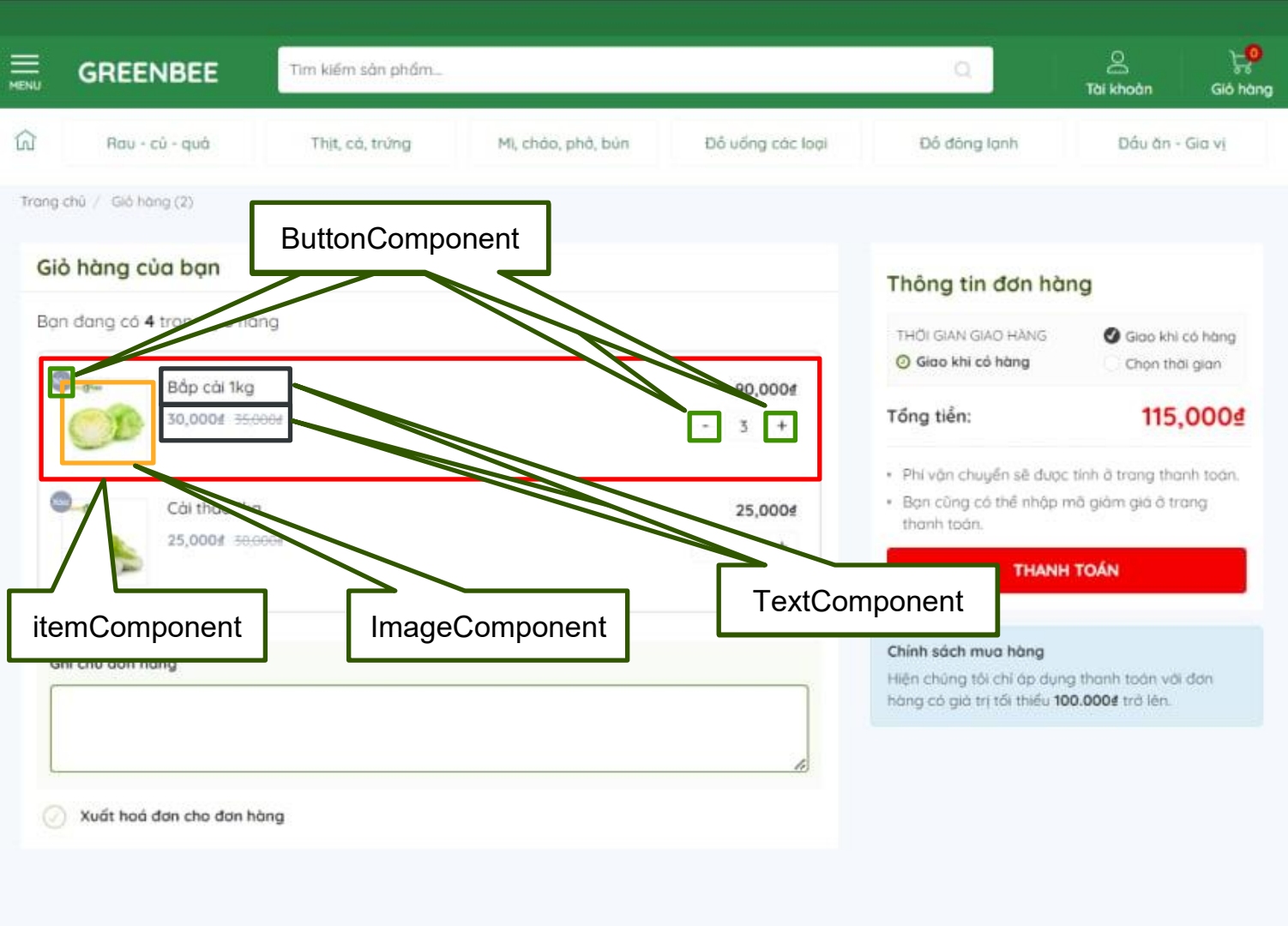
1. Giới thiệu
2. Tạo component
3. Function component
4. Class component
5. Props
6. Component trong component
7. Component trong file

Giới thiệu

- Component là các đoạn mã độc lập và có thể tái sử dụng.
- Component dùng để xây dựng giao diện người dùng.
- Có thể chia nhỏ giao diện thành các thành phần nhỏ hơn.

Ưu điểm:

- Tái sử dụng.
- Độc lập.
- Tùy chỉnh
- Phức tạp


GREENBEE
Tìm kiếm sản phẩm...
Tài khoản
Giỏ hàng
Rau - củ - quả
Thịt, cá, trứng
Mì, cháo, phở, bún
Đồ uống các loại
Đồ đông lạnh
Dầu ăn - Gia vị
Trang chủ / Giỏ hàng (2)
Giỏ hàng của bạn
Bắp cải 1kg
30,000đ
3
25,000đ
Thông tin đơn hàng
THỜI GIAN GIAO HÀNG
Giao khi có hàng
Chọn thời gian
Tổng tiền:
115,000đ
Phí vận chuyển sẽ được tính ở trang thanh toán.
Bạn cũng có thể nhập mã giảm giá ở trang thanh toán.
THANH TOÁN
Chính sách mua hàng
Hiện chúng tôi chỉ áp dụng thanh toán với đơn hàng có giá trị tối thiểu 100.000đ trở lên.
Xuất hoá đơn cho đơn hàng
ButtonComponent
itemComponent
ImageComponent
TextComponent

Tạo component

- Khi tạo component, tên của component phải bắt đầu bằng chữ in hoa.
- Có hai loại component chính trong React
  - Functional Component.
  - Class Component.
- Sử dụng component giống cú pháp html bình thường.

Tạo component

- **Functional Component**: Là component được viết dưới dạng hàm JavaScript. Functional component đơn giản hơn và dễ sử dụng hơn, nhưng không có state nội bộ.
- **Class Component**: Là component được viết dưới dạng class JavaScript. Class component phức tạp hơn functional component, nhưng có thể có state nội bộ và lifecycle methods.

Function component

- Loại component phổ biến nhất là một hàm (function) nó trả về một React element.
- Function component thì có hàm return

```
function Car() {
  return <h2>Hi, I am a Car!</h2>;
}
```

Class component

- Tạo một class component thì phải kế thừa từ thư viện React.Component
- Class component thì phải có hàm render().

```
class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}
```

Class component

- Hàm constructor dùng để khởi tạo các thuộc tính của component.
- Constructor gồm có:
  - Hàm super()
  - State: là trạng thái của thuộc tính trong component.

Class component

```
class Car extends React.Component {
  constructor() {
    super();
    this.state = {color: "red"};
  }
  render() {
    return <h2>I am a {this.state.color} Car!</h2>;
  }
}
```

Props

- Props là viết tắt của "properties" (thuộc tính).
- Props được sử dụng để truyền thông tin từ component cha (hoặc bên ngoài) vào bên trong các component con.
- Props có những đặc điểm sau:
  - Đơn hướng (dữ liệu được truyền từ cha sang con)
  - Bất biến (dữ liệu không thể thay đổi)
  - Kiểu dữ liệu đa dạng
  - Tùy chỉnh giao diện

Props

```
function Car(props) {
  return <h2>I am a {props.color} Car!</h2>;
}


root.render(<Car color="red"/>);
```

```
class Car extends React.Component {
  render() {
    return <h2>I am a {this.props.color} Car!</h2>;
  }
}

const root = ReactDOM.createRoot(document.getElement
root.render(<Car color="red"/>);
```

Component trong component

- Chúng ta có thể component trong component khác

```
function Car() {
  return <h2>I am a Car!</h2>;
}

function Garage() {
  return (
    <>
      <h1>Who lives in my Garage?</h1>
      <Car />
    </>
  );
}
```

Component trong file

- React chủ yếu là về việc sử dụng lại mã vì vậy nên chia các component thành các file riêng biệt.
- Sử dụng từ khóa: export default
- Khi sử dụng file thì component thì nhớ import file.

Component trong file

```
function Car() {
  return <h2>Hi, I am a Car!</h2>;
}

export default Car;
```

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import Car from './Car.js';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```