HTML CSS JAVASCRIPT MORE

# React Components

❮ Previous Next ❯

Components are like functions that return HTML elements.

## React Components

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and returns HTML via a render function.

Components come in two types, Class components and Function components, in this tutorial we will concentrate on Class components.

## Create a Class Component

The component has to include the `extends React.Component` statement, this statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.

The component also requires a `render()` method, this method returns HTML.

## Example

Create a Class component called `Car`

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

Now your React application has a component called Car, which returns a `<h2>` element.

To use this component in your application, use similar syntax as normal HTML: `<Car />`

## Example

Display the `Car` component in the "root" element:

```
ReactDOM.render(<Car />, document.getElementById('root'));
```

# Create a Function Component

Here is the same example as above, but created using a Function component instead.

A Function component also returns HTML, and behaves pretty much the same way as a Class component, but Class components have some additions, and will be preferred in this tutorial.

## Example

Create a Function component called `Car`

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

Once again your React application has a Car component.

Refer to the Car component as normal HTML (except in React, components *must* start with an upper case letter):

## Example

Display the `Car` component in the "root" element:

Run Example »

# Component Constructor

If there is a `constructor()` function in your component, this function will be called when the component gets initiated.

The constructor function is where you initiate the component's properties.

In React, component properties should be kept in an object called `state`.

You will learn more about `state` later in this tutorial.

The constructor function is also where you honor the inheritance of the parent component by including the `super()` statement, which executes the parent component's constructor function, and your component has access to all the functions of the parent component (`React.Component`).

## Example

Create a constructor function in the Car component, and add a color property:

```
class Car extends React.Component {
  constructor() {
    super();
    this.state = {color: "red"};
  }
  }
}
```

Use the color property in the render() function:

### Example

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

Run Example »

# Props

Another way of handling component properties is by using `props`.

You will learn more about `props` in the next chapter.

### Example

Use an attribute to pass a color to the Car component, and use it in the render() function:

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

ReactDOM.render(<Car color="red"/>, document.getElementById('root'));
```

Run Example »

## Components in Components

We can refer to components inside other components:

### Example

Use the Car component inside the Garage component:

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

class Garage extends React.Component {
  render() {
    return (
      <div>
      <h1>Who lives in my Garage?</h1>
      <Car />
      </div>
    );
  }
}

ReactDOM.render(<Garage />, document.getElementById('root'));
```

Run Example »

## Components in Files

React is all about re-using code, and it can be smart to insert some of your components in separate files.

To do that, create a new file with a `.js` file extension and put the code inside it:

## Example

This is the new file, we named it "App.js":

```
import React from 'react';
import ReactDOM from 'react-dom';

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

export default Car;
```

To be able to use the Car component, you have to import the file in your application.

## Example

Now we import the "App.js" file in the application, and we can use the Car component as if it was created here.

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

Run Example »

❮ Previous Next ❯

## HOW TO

Tabs
Dropdowns
Accordions
Side Navigation
Top Navigation
Modal Boxes
Progress Bars
Parallax
Login Form
HTML Includes
Google Maps
Range Sliders
Tooltips
Slideshow
Filter List
Sort List

## SHARE

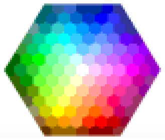
## CERTIFICATES

JavaScript
SQL
Python
PHP
jQuery
Bootstrap
XML

Read More »

HTML CSS JAVASCRIPT MORE

## COLOR PICKER



SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
jQuery Tutorial
Java Tutorial
C++ Tutorial

## Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
jQuery Examples
Java Examples
XML Examples

Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
jQuery Reference
Java Reference
Angular Reference

## Web Certificates

HTML Certificate
CSS Certificate
JavaScript Certificate
SQL Certificate
Python Certificate
jQuery Certificate
PHP Certificate
Bootstrap Certificate
XML Certificate

Get Certified »

REPORT ERROR FORUM ABOUT SHOP

Top Tutorials

HTML Tutorial
CSS Tutorial

Top References

HTML Reference
CSS Reference