

# Getting started with React

Let's render a Hello World div using React

## WE'LL COVER THE FOLLOWING ^

- `ReactDOM.render()`
- `React.createElement()`

React consists of two libraries, `React` and `ReactDOM`. `React` allows you to create elements, which we render with `ReactDOM`. They are split because you could (theoretically) render those `ReactElements` anywhere, not only to the browser DOM.

For example, there are initial experiments out there for rendering React to Canvas, WebVR and even hardware!

Take a look at the following HTML Page.

HTML CSS JavaScript

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>React</title>
6   <link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/react/0.14.3/react.min.css">
7   <script src="//fb.me/react-with-addons-0.14.3.js"></script>
8   <script src="//fb.me/react-dom-0.14.3.js"></script>
9 </head>
10 <body>
11   <div id="container"></div>
12 </body>
13 </html>
```

```
1 ReactDOM.render(
2   React.createElement('h1', { className: 'hello' }, 'Hello World'),
3   document.getElementById('container')
4 );
```

# Hello World!

You will see an `<h1>` with the text “Hello World”. This is the source code generating that text:

```
ReactDOM.render(  
  React.createElement('h1', {className: 'heading'}, 'Hello World'),  
  document.getElementById('container')  
);
```

We use the `ReactDOM.render()` function to render a `ReactElement` created with the `React.createElement()` function.

## `ReactDOM.render()` #

The `ReactDOM.render()` function takes two arguments: The `ReactElement` to render, and the DOM node we want to render into. (the “entry point”)

```
ReactDOM.render(  
  React.createElement('h1', {className: 'heading'}, 'Hello World'),  
  document.getElementById('container')  
);
```

Now you might think creating a `ReactDOM.render()` function for every `ReactElement` you have is the way to go. That’s not a very good idea – it empties the DOM node we use as an entry point. How do we render multiple `ReactElements` then? Let’s take a look at the `React.createElement()` function to figure it out!

## `React.createElement()` #

This function takes the node (or `ReactElement`) we want to create as the first argument and some properties (like `type`) in an object as the second

argument:

```
React.createElement('input');  
// -> <input></input>  
React.createElement('input', { type: 'radio' });  
// -> <input type="radio"></input>  
React.createElement('input', { className: 'heading', type: 'radio' });  
// -> <input class="heading" type="radio"></input>
```

Notice how the HTML `class` attribute has to be set via `className` property in react. This is because `class` is a reserved keyword in JavaScript, which means we might introduce unwanted problems into our apps by passing in `class`. React mitigates this by using the `className` property instead of `class`!

We can also (optionally) pass children as the third argument! The simplest usecase here is to render a bit of text:

```
React.createElement('h1', null, 'Hello World');  
// -> <h1>Hello World</h1>
```

The children (above: `'Hello World'`) can also be another `ReactElement`! Let's say we want to add a `<div class="wrapper">` around our heading. We use `React.createElement` to render a `div` with a `className` of `'wrapper'`, and then pass our heading in as the child:

```
React.createElement('div', { className: 'wrapper' },  
  React.createElement('h1', null, 'Hello World')  
)
```

This will render this HTML:

```
<div class="wrapper">  
  <h1>Hello World</h1>  
</div>
```

Let's look at the working example.

```
1 ReactDOM.render(  
2   React.createElement('div', { className: 'wrapper' },  
3     React.createElement('h1', { className: 'heading' }, 'Hello World!')  
4   ),  
5   document.getElementById('container')  
6 );
```

javascript

# Hello World!

output

This `.wrapper` div might have a `max-width` and other styling associated with it. By reusing this element, we make sure our application is consistent since it has the same styling everywhere! This is what *components* are for.