

LEARN MORE

We can then make an index.html file with the following:

We'll be using Parcel as our bundler, but you can elect to use webpack or another bundler if you wish. Or, check this section if you prefer using Create React App. Let's add Parcel to our project:

```
# Install Parcel to our DevDependencies
$ npm i parcel-bundler -D

# Install TypeScript
$ npm i typescript -D

# Install types for React and ReactDOM
$ npm i -D @types/react @types/react-dom
```

We can update our package.json with a new task that will start our development server:

```
"name": "react-typescript",
   "version": "1.0.0",
   "description": "An example of how to use React and TypeScript w
   "scripts": {
       "dev": "parcel src/index.html"
    },
    "keywords": [],
   "author": "Paul Halliday",
   "license": "MIT"
}
```

We can now populate a Counter.tsx file with a simple counter:

```
import * as React from 'react';
export default class Counter extends React.Component {
  state = {
    count: 0
  };
  increment = () => {
    this.setState({
      count: (this.state.count + 1)
    });
  };
  decrement = () => {
    this.setState({
     count: (this.state.count - 1)
  };
  render () {
     <div>
       <h1>{this.state.count}</h1>
        <button onClick={this.increment}>Increment
        <button onClick={this.decrement}>Decrement/button>
     </div>
}
```

Then, inside of App.tsx, we can load the Counter:

```
import * as React from 'react';
import { render } from 'react-dom';
```

```
import Counter from './Counter';
render(<Counter />, document.getElementById('main'));
```

Our project can be ran with \$ npm run dev and accessed at http://localhost:1234.

## Create React App and TypeScript

If you'd rather use <u>Create React App</u> to initiate your project, you'll be pleased to know that CRA now supports TypeScript <u>out of the box</u>.

Just use the --typescript flag when invoking the create-react-app command:

```
$ create-react-app my-new-app --typescript
```

## Functional Components

Stateless or <u>functional components</u> can be defined in TypeScript like so:

```
import * as React from 'react';

const Count: React.FunctionComponent<{
   count: number;
}> = (props) => {
   return <h1>{props.count}</h1>;
};

export default Count;
```

We're using React.FunctionComponent and defining the object structure of our expected props. In this scenario we're expecting to be passed in a single prop named count and we're defining it in-line. We can also define this in other ways, by creating an interface such as Props:

```
interface Props {
   count: number;
}

const Count: React.FunctionComponent<Props> = (props) => {
   return <h1>{props.count}</h1>;
};
```

## Class Components.

Class components can similarly be defined in TypeScript as such:

```
import * as React from 'react';
import Count from './Count';
interface Props {}
interface State {
 count: number;
};
export default class Counter extends React.Component<Props, State
 state: State = {
   count: 0
 };
  increment = () => {
   this.setState({
     count: (this.state.count + 1)
   });
 };
  decrement = () => {
   this.setState({
     count: (this.state.count - 1)
   });
 };
  render () {
   return (
     <div>
       <Count count={this.state.count} />
       <button onClick={this.increment}>Increment/button>
       <button onClick={this.decrement}>Decrement
```

```
</div>
);
}
```

## Default Props.

We can also define **defaultProps** in scenarios where we may want to set default props. We can update our Count example to show this:

```
import * as React from 'react';
interface Props {
  count?: number;
}

export default class Count extends React.Component<Props> {
  static defaultProps: Props = {
    count: 10
  };

render () {
    return <h1>{this.props.count}</h1>;
  }
}
```

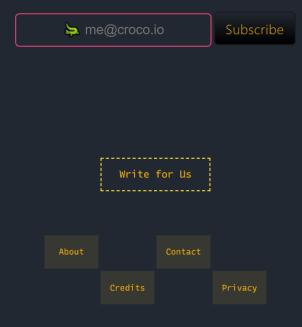
We'll need to stop passing this.state.count in to the Counter component too, as this will overwrite our default prop:

You should now have a project that's set up to use TypeScript and React, as well as the tools to create your own functional and class-based components!

Published: February 11, 2019



Catch up with the latest in front-end web dev with our monthly newsletter:



 $\underline{\textbf{Support us on Patreon}} \; \bullet \; \underline{\textbf{Advertise}}$