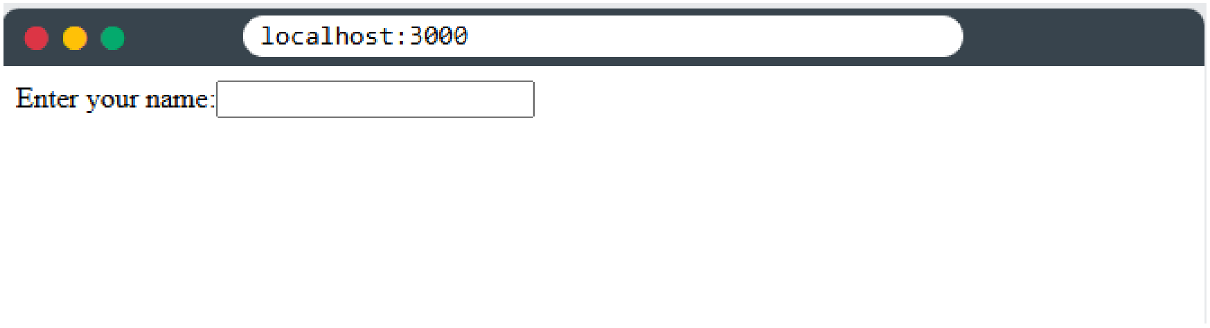
React Forms

Just like in HTML, React uses forms to allow users to interact with the web page.

Adding Forms in React

You add a form with React like any other element:

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

function MyForm() {
  return (
    <form>
      <label>Enter your name:
        <input type="text" />
      </label>
    </form>
  )
}

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



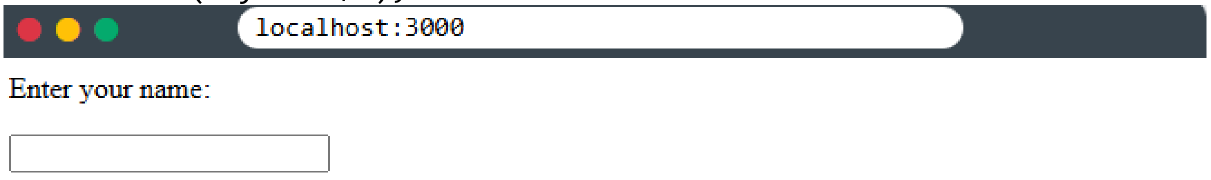
Enter your name:

This will work as normal, the form will submit and the page will refresh. But this is generally not what we want to happen in React. To prevent this default behavior and let React control the form.

Handling Forms

Handling forms is about how you handle the data when it changes value or gets submitted.In HTML, form data is usually handled by the DOM. In React, form data is usually handled by the components. When the data is handled by the components, all the data is stored in the component state.
You can control changes by adding event handlers in the `onChange` attribute.
We can use the `useState` Hook to keep track of each inputs value and provide a "single source of truth" for the entire application.

```
import { useState } from "react";
import ReactDOM from 'react-dom/client';

function MyForm() {
  const [name, setName] = useState("");

  return (
    <form>
      <label>Enter your name:
        <input
          type="text"
          value={name}
          onChange={(e) => setName(e.target.value)}
        />
      </label>
    </form>
  )
}

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

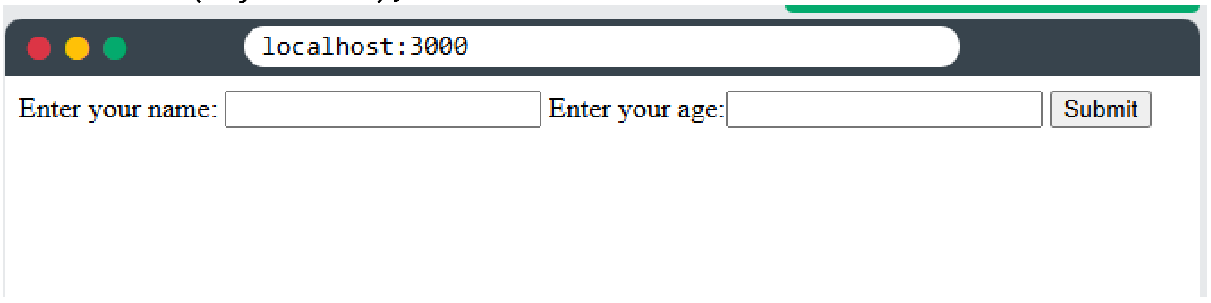


Enter your name:

Multiple Input Fields

You can control the values of more than one input field by adding a `name` attribute to each element.We will initialize our state with an empty object. To access the fields in the event handler use the `event.target.name` and `event.target.value` syntax.
To update the state, use square brackets [bracket notation] around the property name.

```
import { useState } from "react";
import ReactDOM from "react-dom/client";

function MyForm() {
  const [inputs, setInputs] = useState({});

  const handleChange = (event) => {
    const name = event.target.name;
    const value = event.target.value;
    setInputs(values => ({...values, [name]: value}))
  }

  const handleSubmit = (event) => {
    event.preventDefault();
    console.log(inputs);
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>Enter your name:
      <input
        type="text"
        name="username"
        value={inputs.username || ""}
        onChange={handleChange}
      />
      </label>
      <label>Enter your age:
        <input
          type="number"
          name="age"
          value={inputs.age || ""}
          onChange={handleChange}
        />
        </label>
        <input type="submit" />
    </form>
  )
}

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



Textarea

The textarea element in React is slightly different from ordinary HTML. In HTML the value of a textarea was the text between the start tag `<textarea>` and the end tag `</textarea>`.

```
<textarea>
  Content of the textarea.
</textarea>
```

In React the value of a textarea is placed in a value attribute. We'll use the `useState` Hook to manage the value of the textarea:

Example:

A simple textarea with some content:

```
import { useState } from 'react';
import ReactDOM from 'react-dom/client';
function MyForm() {
  const [textarea, setTextarea] = useState(
    "The content of a textarea goes in the value attribute"
  );
  const handleChange = (event) => {
    setTextarea(event.target.value)
  }
  return (
    <form>
      <textarea value={textarea} onChange={handleChange} />
    </form>
  )
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<MyForm />);
```

Select

A drop down list, or a select box, in React is also a bit different from HTML.
In HTML, the selected value in the drop down list was defined with the `selected` attribute:

HTML:

```
<select>
 <option value="Ford">Ford</option>
  <option value="Volvo" selected>Volvo</option>
  <option value="Fiat">Fiat</option>
</select>
```

In React, the selected value is defined with a `value` attribute on the `select` tag:

Example:

A simple select box, where the selected value "Volvo" is initialized in the constructor:

```
function MyForm() {
  const [myCar, setMyCar] = useState("Volvo");
  const handleChange = (event) => {
    setMyCar(event.target.value)
  }
  return (
    <form>
      <select value={myCar} onChange={handleChange}>
        <option value="Ford">Ford</option>
        <option value="Volvo">Volvo</option>
        <option value="Fiat">Fiat</option>
      </select>
    </form>
  )
}
```

By making these slight changes to `<textarea>` and `<select>`, React is able to handle all input elements in the same way.

Styling React Using CSS

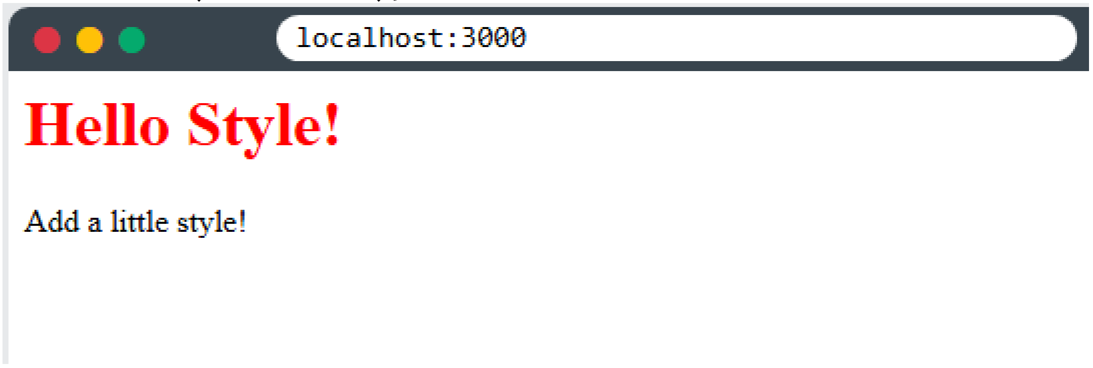
There are many ways to style React with CSS

1) **inline styling**, and 2) **CSS stylesheet**.

Inline Styling

To style an element with the inline style attribute, the value must be a JavaScript object:

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

const Header = () => {
  return (
    <>
      <h1 style={{color: "red"}}>Hello Style!</h1>
      <p>Add a little style!</p>
    </>
  );
}

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



```
Hello Style!

Add a little style!
```

Note: In JSX, JavaScript expressions are written inside curly braces, and since JavaScript objects also use curly braces, the styling in the example above is written inside two sets of curly braces {{}}.

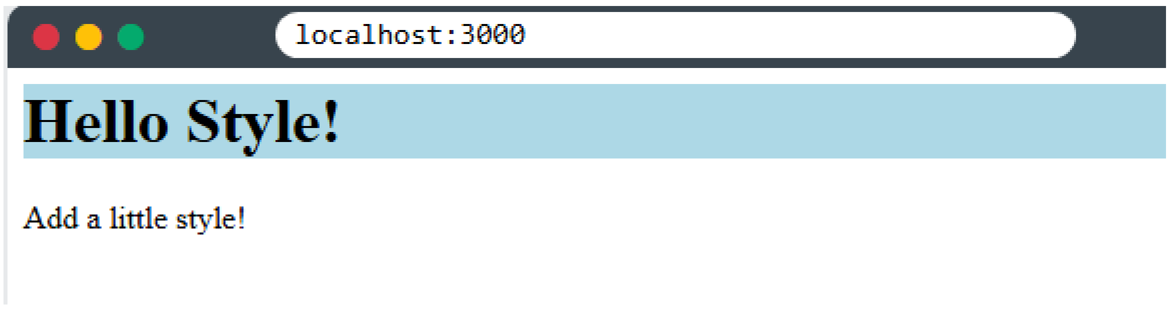
camelCased Property Names

Since the inline CSS is written in a JavaScript object, properties with two names, like `background-color`, must be written with camel case syntax:

Example:

Use `backgroundColor` instead of `background-color`:

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

const Header = () => {
  return (
    <>
      <h1 style={{backgroundColor: "lightblue"}}>Hello Style!</h1>
      <p>Add a little style!</p>
    </>
  );
}

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



Hello Style!

Add a little style!

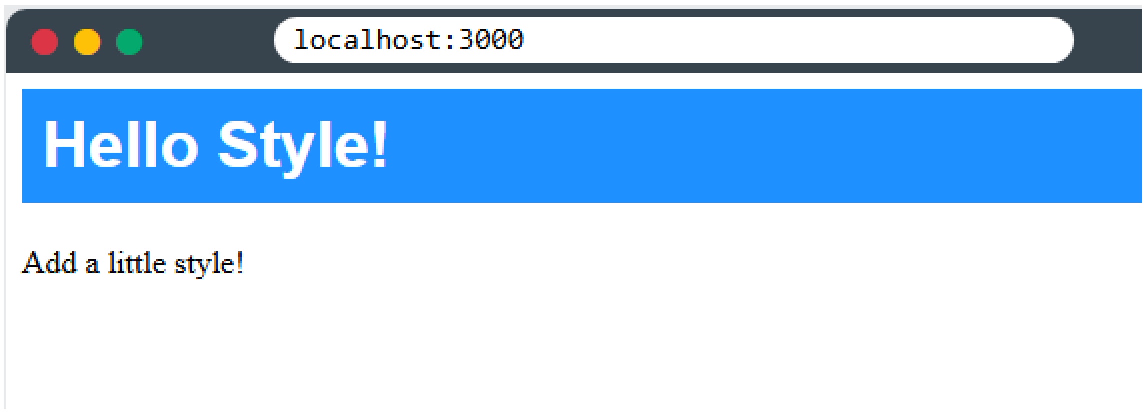
JavaScript Object

You can also create an object with styling information, and refer to it in the style attribute:

Example:

Create a style object named `mystyle`:

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

const Header = () => {
  const myStyle = {
    color: "white",
    backgroundColor: "DodgerBlue",
    padding: "10px",
    fontFamily: "Sans-Serif"
  };
  return (
    <>
      <h1 style={myStyle}>Hello Style!</h1>
      <p>Add a little style!</p>
    </>
  );
}

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


localhost:3000

Hello Style!

Add a little style!

CSS Stylesheet

write the CSS styling in a separate file, just save the file with the `.css` file extension, and import it in your application.

App.css:

Create a new file called "App.css" and insert some CSS code in it:

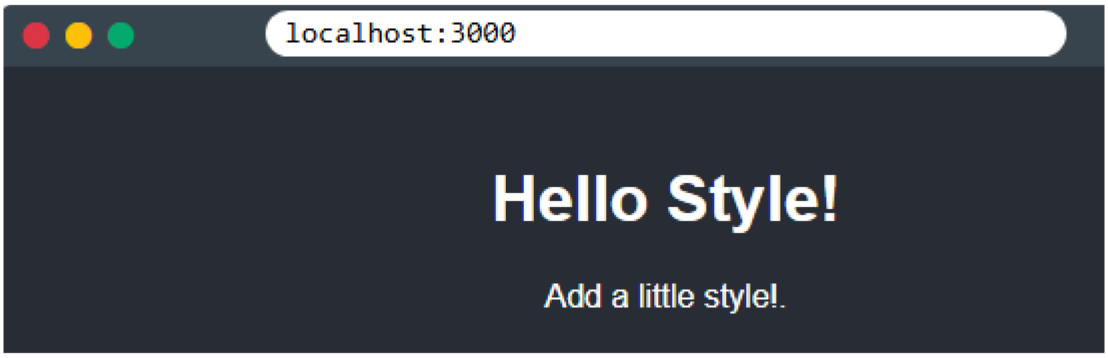
```
body {
  background-color: #282c34;
  color: white;
  padding: 40px;
  font-family: Arial;
  text-align: center;
}
```

Note: we can call the file whatever you like, just remember the correct file extension.
Import the stylesheet in your application:

index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './App.css';

class MyHeader extends React.Component {
  render() {
    return (
      <div>
      <h1>Hello Style!</h1>
      <p>Add a little style!.</p>
      </div>
    );
  }
}

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

/*
Notice that you now have three files in your project:
```

```
'index.js', 'index.html', and 'App.css'.
*/
```



Hello Style!

Add a little style!.

CSS Modules

Another way of adding styles to your application is to use CSS Modules.
CSS Modules are convenient for components that are placed in separate files.
The CSS inside a module is available only for the component that imported it, and you do not have to worry about name conflicts.
Create the CSS module with the `.module.css` extension, example: `mystyle.module.css`.

mystyle.module.css:

Create a new file called "mystyle.module.css" and insert some CSS code in it:

```
.bigblue {
  color: DodgerBlue;
  padding: 40px;
  font-family: Arial;
  text-align: center;
}
```

Import the stylesheet in your component:

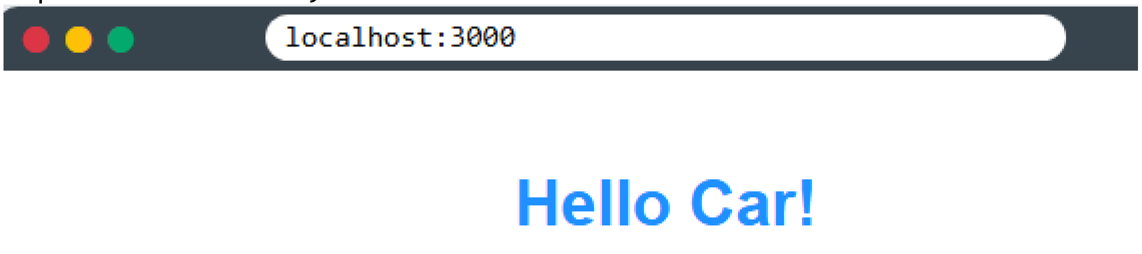
App.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import styles from './mystyle.module.css';
class Car extends React.Component {
  render() {
    return <h1 className={styles.bigblue}>Hello Car!</h1>;
  }
}
export default Car;
```

Import the component in your application:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import styles from './mystyle.module.css';
```

```
class Car extends React.Component {
  render() {
    return <h1 className={styles.bigblue}>Hello Car!</h1>;
  }
}

export default Car;
```



Hello Car!

Styling React Using Sass

Sass is a CSS pre-processor.Sass files are executed on the server and sends CSS to the browser.

Can I use Sass?

If you use the `create-react-app` in your project, you can easily install and use Sass in your React projects.
Install Sass by running this command in your terminal:

```
npm i sass
```

Now you are ready to include Sass files in your project!

Create a Sass file

Create a Sass file the same way as you create CSS files, but Sass files have the file extension `.scss`
In Sass files you can use variables and other Sass functions:

ExampleGet your own React.js Server

my-sass.scss:
Create a variable to define the color of the text:

```
$myColor: red;
h1 {
  color: $myColor;
}
```

Import the Sass file the same way as you imported a CSS file:

Example

index.js:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './my-sass.scss';
const Header = () => {
  return (
    <>
      <h1>Hello Style!</h1>
      <p>Add a little style!.</p>
    </>
  );
}
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Header />);
```