Credits: https://www.w3schools.com/REACT/react_components.asp

React Components

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 return HTML.

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

In older React code bases, you may find Class components primarily used. It is now suggested to use Function components along with Hooks, which were added in React 16.8. There is an optional section on Class components for your reference.

Create Your First Component

When creating a React component, the component's name *MUST* start with an upper case letter.

Class Component

A class component must 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>;

  }
```

```
}
```

Function Component

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

A Function component also returns HTML, and behaves much the same way as a Class component, but Function components can be written using much less code, are easier to understand, and will be preferred in this tutorial.

Example

Create a Function component called `Car`

```
function Car() {

  return <h2>Hi, I am a Car!</h2>;

}
```

Rendering a Component

Now your React application has a component called Car, which returns an `<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:

```
const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Car />);
```

Props

Components can be passed as `props`, which stands for properties.

Props are like function arguments, and you send them into the component as attributes.

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:

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

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

Components in Components

We can refer to components inside other components:

Example

Use the Car component inside the Garage component:

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

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

```
const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Garage />);
```

Run Example »

Components in Files

React is all about re-using code, and it is recommended to split your components into separate files.

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

Note that the filename must start with an uppercase character.

Example

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

```
function Car() {

  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 "Car.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/client';

import Car from './Car.js';


const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Car />);
```