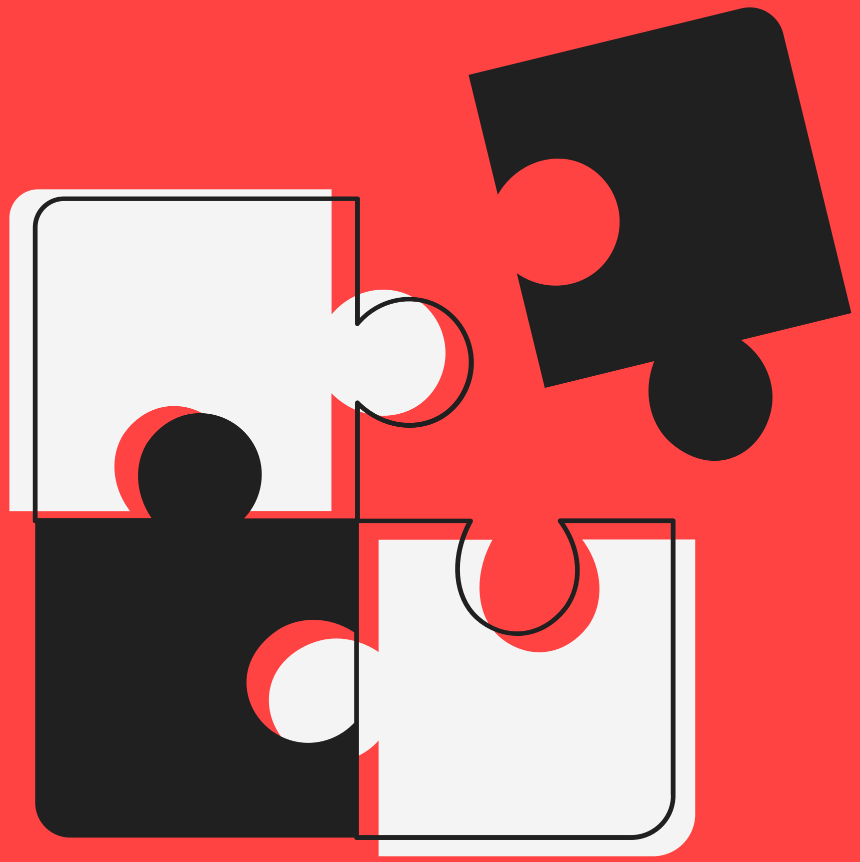


JSX

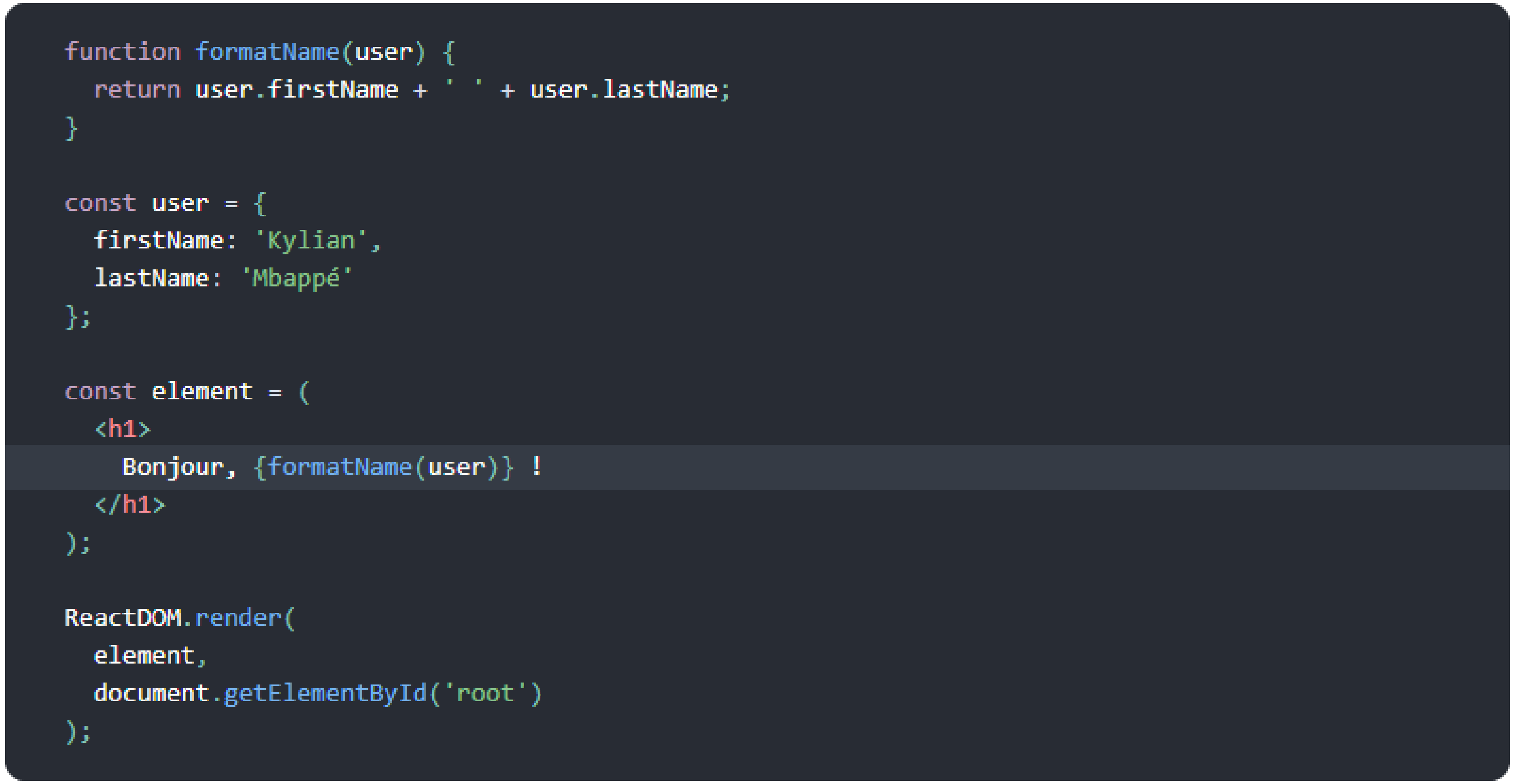
JAVASCRIPT XML
JAVASCRIPT SYNTAX EXTENSION

JSX (JavaScript + XML) is an extension of JavaScript that allows you to write HTML directly within JavaScript, which has a few benefits of making your code more readable and exercising the full power of JavaScript within HTML.

Since JSX is not a valid JS code, it needs to be compiled into JS with a tool like Babel or similar.

JSX produces React “elements”. We will explore rendering them to the DOM in the next section. Below, you can find the basics of JSX necessary to get you started.

VOUS POUVEZ UTILISER N'IMPORTE QUELLE EXPRESSION JAVASCRIPT VALIDE DANS DES ACCOLADES EN JSX.



```jsx
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Kylian',
  lastName: 'Mbappé'
};

const element = (
  <h1>
    Bonjour, {formatName(user)} !
  </h1>
);

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

How can I write comments in JSX?

```
<div>
  {/* Comment goes here */}
  Hello, {name}!
</div>
```

```
<div>
  {/* It also works
  for multi-line comments. */}
  Hello, {name}!
</div>
```

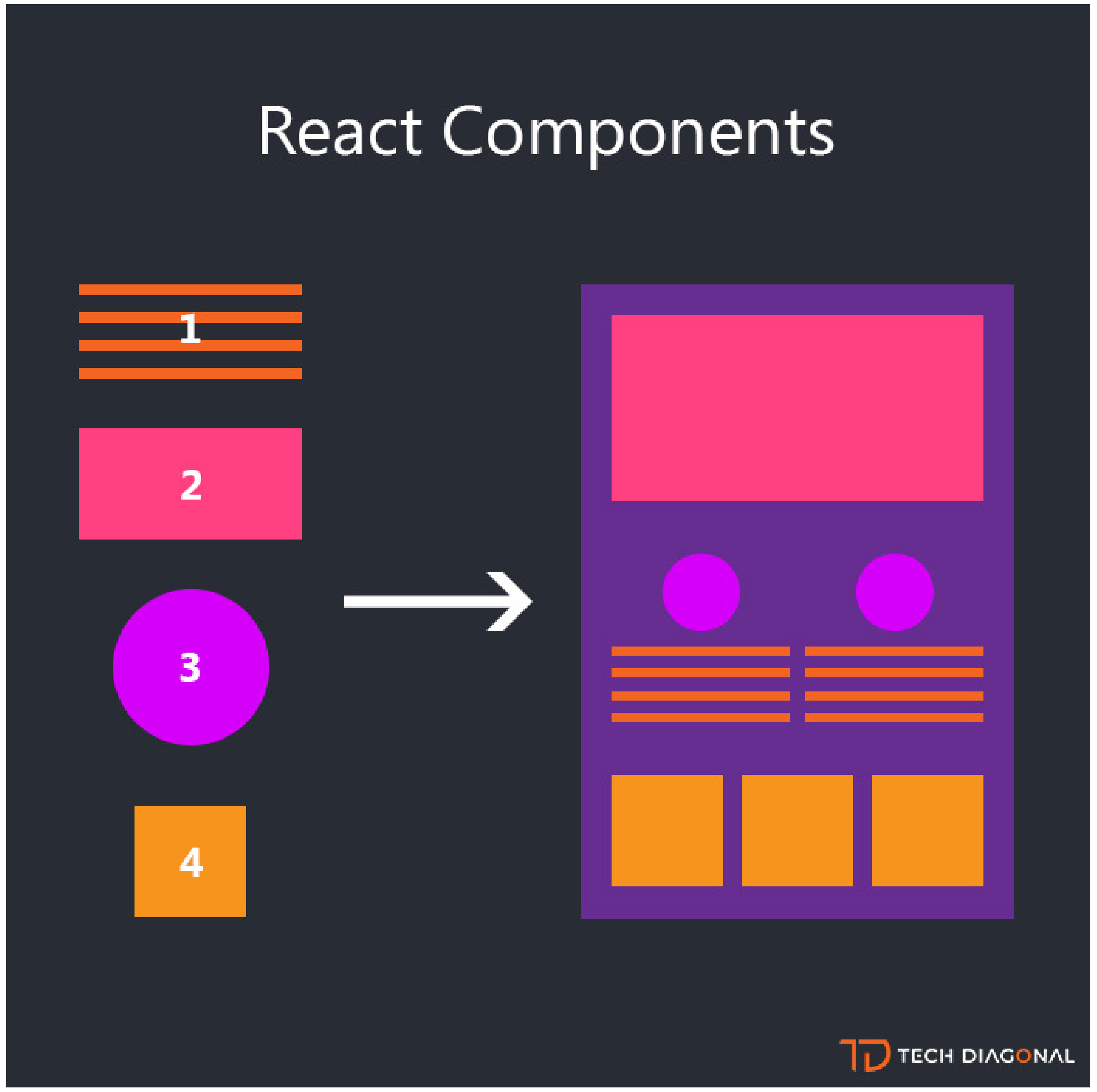
JSX vs HTML

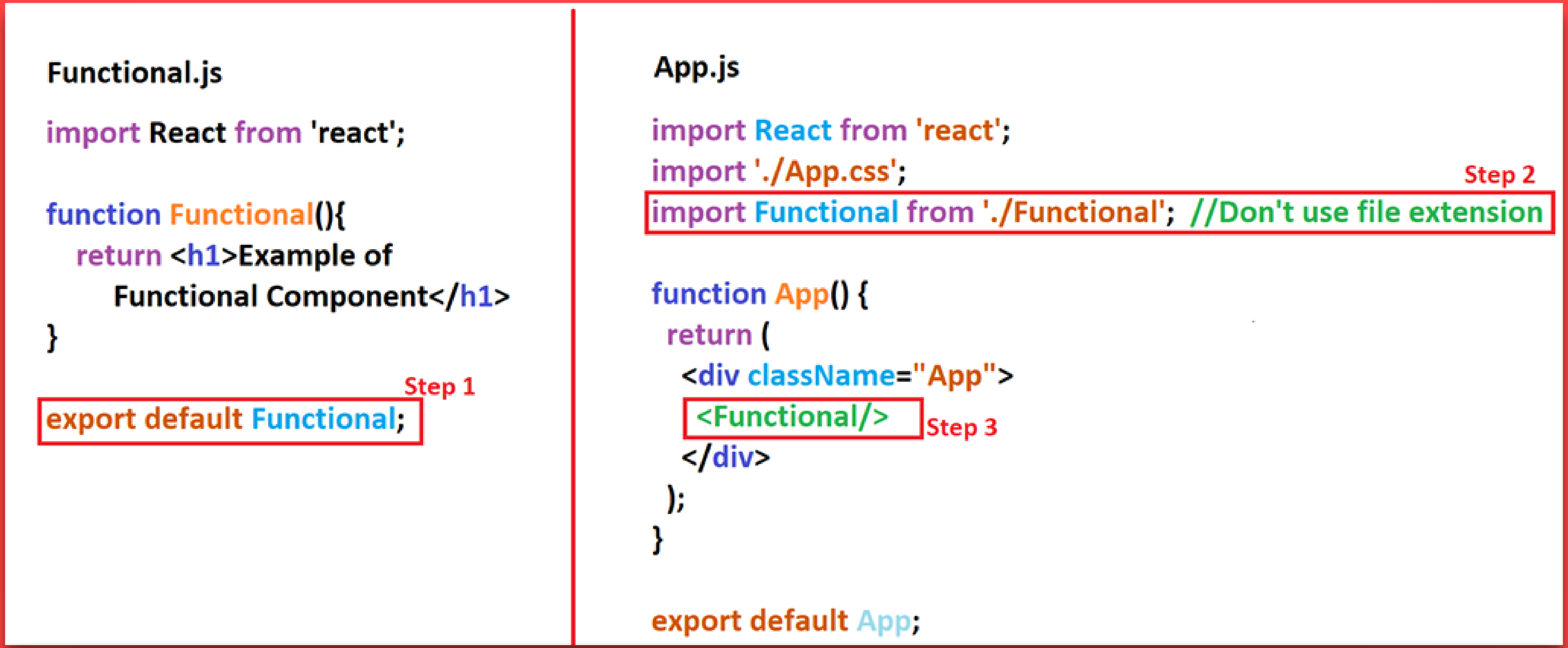
Another important difference is that in JSX you can't use the world class to define HTLM classes,
since **class** is a reserved word in JavaScript, instead, use — **className**.

In React, we can render JSX directly into HTML DOM using React rendering API, aka ReactDOM. The formula for rendering React elements looks like this:
ReactDOM.render(componentToRender, targetNode)
ReactDOM.render() must be called after the JSX elements declarations.

.Moreover, all HTML attributes and event references in JSX become camelCase, this way, **onclick** event becomes **onClick** and **onchange** — > **onChange**.

Component splits UI into independent and reusable pieces which optionally accepts input called props and returns react element



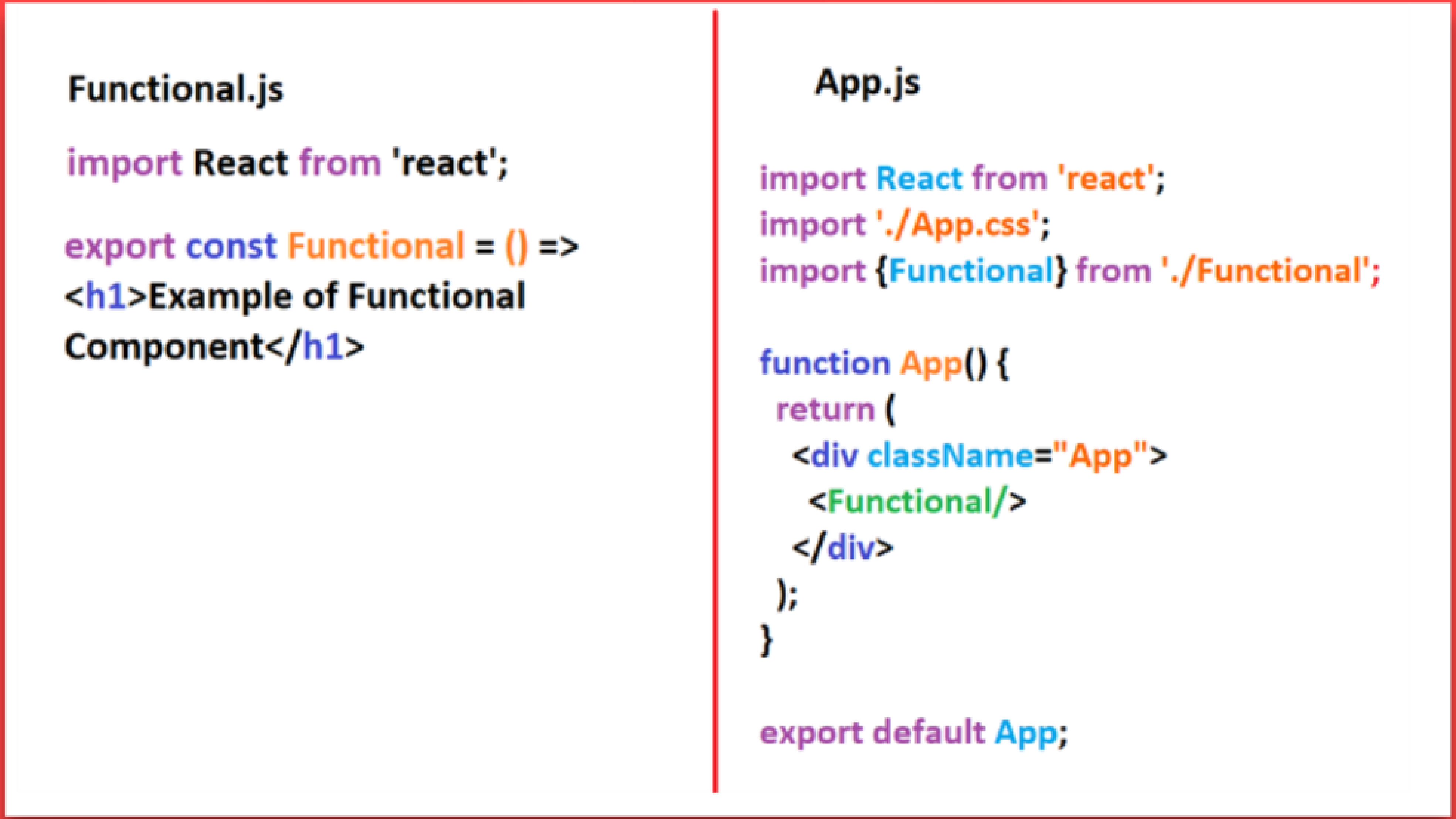


Functional.js

```
import React from 'react';

function Functional(){
   return <h1>Example of
        Functional Component</h1>
}

export default Functional;
```

App.js

```
import React from 'react';
import './App.css';
import Functional from './Functional';  //Don't use file extension

function App() {
 return (
  <div className="App">
   <Functional/>
  </div>
 );
}

export default App;
```



Functional.js

```
import React from 'react';

export const Functional = () =>
<h1>Example of Functional
Component</h1>
```

App.js

```
import React from 'react';
import './App.css';
import {Functional} from './Functional';

function App() {
 return (
  <div className="App">
   <Functional/>
  </div>
 );
}

export default App;
```

RESSOURCES GRATUITES

Utilisez ces icônes et ces illustrations gratuites dont vous pouvez changer la couleur dans votre design Canva.