

CodeSandbox で React の UI コンポーネントを作成してみよう

仕事や趣味を通して JavaScript のコードを書いている方であれば、「CodeSandbox を使って React の UI コンポーネントを作る」といってピンとくる方も多いと思いますが、実際に興味はあるけれど、どう React を書けばよいのかわからないという方も多いと思います。

基本的な問題を CodeSandbox 上で動作確認することを通して、JavaScript に触れる機会をつくっていきたいと考えています。コードを書くことに慣れるという点でも手を動かして考える機会にして頂ければ幸いです。

今回は CodeSandbox を使って、「React の UI コンポーネントを作成してみよう」を確認していきたいと思います。

○ はじめに

JavaScript の開発で、最も基本的なツールと言えるのは、「Web ブラウザ」でしょう。Web ブラウザには、開発を強力にサポートしてくれる開発者ツールが用意されていますので、開発者ツールの使い勝手も選択肢の一つになるように思います。

Windows では、F12 キー

macOS では、Command+Option+I キー を押すと表示できます。

HTML / JavaScript の開発では、適宜テキストエディタやオンラインのコードエディタを使うことで仕事も早く進めることができます。ここでは、Visual Studio Code と CodeSandbox の紹介をします。

・テキストエディタ

Visual Studio Code

Microsoft によって提供されているクロスプラットフォームのテキストエディタで、Windows だけでなく MacOS / Linux に対応しています。Git によるソースコード管理や、IntelliSense、デバッガなどの機能を搭載し、拡張機能をインストールすることで、開発のさまざまな用途で 사용할 ことができます。

- ・オンラインのコードエディタ

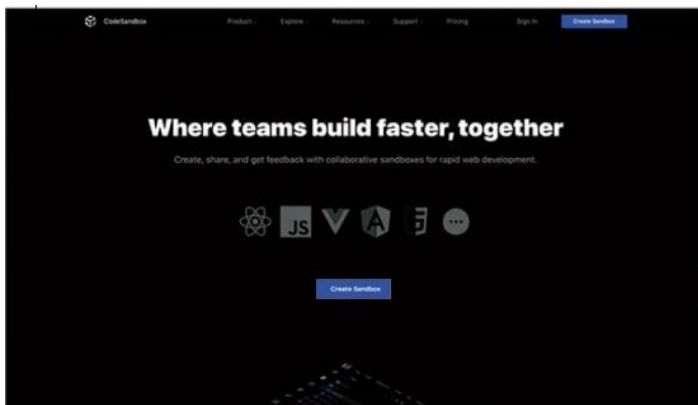
CodeSandbox

Web サイト : <https://codesandbox.io/>

今回使用するオンラインのコードエディタです。Web エディタ上で簡単に JavaScript の作成ができて、コーディングや、URL での共有、GitHub との連携も行うことができるサービスです。

開発の現場で使われるというより、個人の勉強やコードの共有の為に使われるサービスで、環境構築自体に時間を割かずにコードを書く勉強に取り組むことができます。

▶ CodeSandbox の TOP ページ



CodeSandbox メニュー画面で表示されたテンプレートの中にある[React]をクリックして、React のプロジェクトを開始します。



○ React ～ モダンな Web 開発のためのライブラリ について

JavaScript といえば、Web アプリケーションを作成する際、HTML ソースに組み込んで動的な処理を書くといったイメージをお持ちの方も多いのではないでしょうか。HTML5 ではさまざまな機能を JavaScript から使うことが前提とされ、サーバー側で JavaScript の処理を書くということも一般的になりつつあります。

JavaScript は、いまでは多くの端末やプラットフォームで動くプログラミング言語といっても良いでしょう。

開発で使われることも多い「TypeScript」などは、プログラムを書き、コンパイラを通して従来の JavaScript として出力することができます。

Web 開発やスマホアプリを作る際に、JavaScript に変換する機能があるといっても、他のプログラミング言語を使うというより JavaScript 自体のモダンな言語仕様を使ったほうが開発の効率はあがるでしょう。

そこで、ECMA International は、2015 年に大幅に機能の追加された ECMAScript の第 6 版を公開しました。この仕様は、とても画期的で、クラスやモジュール、イテレータ、ジェネレータ、アロー演算子、テンプレート文字列、型付き配列など、さまざまなモダンな要素が盛り込まれました。

ES2015 の登場により、言語仕様も非常にモダンになりました。それは、言語仕様の話だけではなく、JavaScript 開発を支援するライブラリ・フレームワークも日々登場しています。その中で注目されているのが、React.js, Vue.js, Angular.js などのライブラリです。

ここでは、UI コンポーネントの作り方を確認することで、「React の基礎」から確認していきたいと思います。

Reactとは？

- ・ Meta社とコミュニティが開発している「**UIライブラリ**」
～ HTMLをコンポーネントとして定義し、組み合わせて使うことができる

> 特徴

Webページ内の各パーツを
コンポーネントとして扱える

JSXが使える (JavaScriptの中に
HTMLタグを記述できる)

ReactでDOMを更新する際、
必要最低限の部分のみ更新
(Virtual DOM)

UIライブラリ

コンポーネント、JSX、
Virtual DOM 他

ここからは始める「React」～基礎事項の確認をしていきます

まずは、CodeSandbox で React のプロジェクトを作成し、Hello World を表示させてみましょう。

CodeSandbox のメニュー画面で表示されたテンプレートの中にある [React] をクリック
一度 src フォルダ内にあるファイルを全て削除し、index.js を新規に作成します。

以下のコードを記載して、「Hello World」を表示してみましょう。

JSX記法

- ・ コンポーネントを作るには、「**JSX**」で**コンテンツを返す関数を定義**する

⇒ 関数の返却値として **HTMLのタグを記述し、コンポーネントとして画面を構成**

file: src/index.js

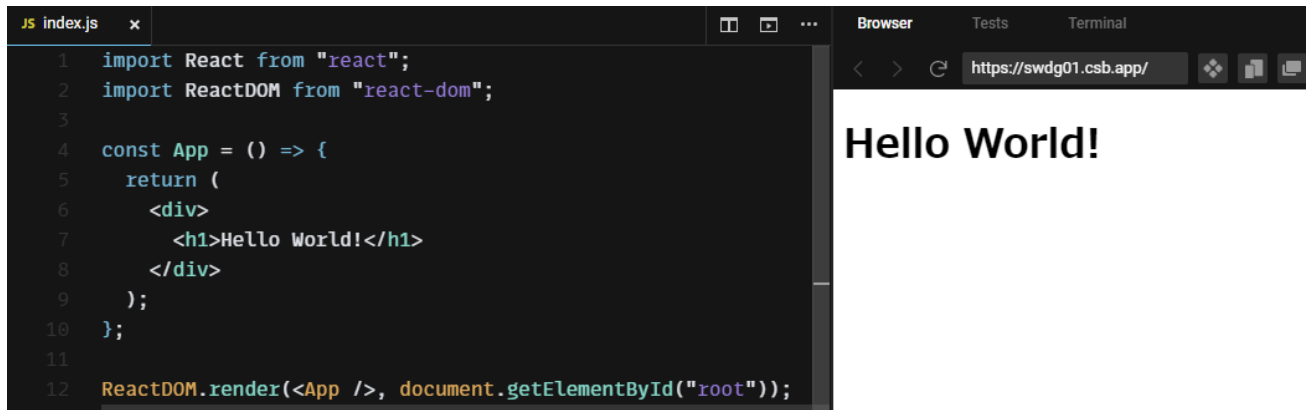
```
import React from "react";
import ReactDOM from "react-dom";

const App = () => {
  return (
    <div>
      <h1>Hello World !</h1>
    </div>
  );
};
```

ReactDOMのrender関数内
第1引数：render対象
第2引数：render箇所

関数名をHTMLのようにタグで囲う
ことでコンポーネントとして扱える

```
ReactDOM.render(<App />,
  document.getElementById("root") );
```



○ React の基礎 ～問題演習～

それでは、実際に資料をダウンロードして、React 基礎の演習問題を解いていきましょう。

React 基礎演習 ～ReactのUIコンポーネントを作成～

＊該当のシートに移動するには、問題番号のハイパーリンクを押下

問題番号	シート名	内容
No1	JSX記法	CodeSandboxでJSX記法を用いて、簡単なHTMLを出力する
No2	コンポーネント 分割	Appという名前の関数コンポーネントを新規作成し、読み込む
No3	関数でコンポーネント 作成	Greetingという関数コンポーネントを定義して、表示する
No4	スタイルの 扱い方	style属性を記述してスタイルを適用する
No5	Propsの利用	色とテキストをPropsとして受け取り、文字列を表示する
No6	Stateの利用	数値のStateを定義し画面に表示し、ボタン押下時にカウントアップする

まずは、関数でコンポーネントを作ってみることから考えたいと思います。アロー関数と JSX 記法について確認していきましょう。

まずは関数でコンポーネントを作ってみる

▶ 押さえておきたいTips

・アロー関数とは？

アロー演算子 (`=>`) を用いて、function の定義をより短く記述できる記法

下の例を見てみましょう

変数に関数を値として代入

引数を () で記載

```
// 従来の関数
function sample(a, b) {
  return a + b;
}
```

```
// アロー関数
const sample = (a, b) => {
  return a + b;
}
sample(4,5); // 9
```

JSX記法

・コンポーネントを作るには、「**JSX**」でコンテンツを返す関数を定義 する

⇒ 関数の返却値として **HTMLのタグを記述し、コンポーネントとして画面を構成**

file: src/index.js

```
import React from "react";
import ReactDOM from "react-dom";
```

```
const App = () => {
```

アロー関数

```
  return (
    <>
      <h1>こんにちは！</h1>
      <p>お元気ですか？</p>
    </>
  );
}
```

ReactDOMのrender関数内
第1引数：render対象
第2引数：render箇所

```
);
};
```

関数名をHTMLのようにタグで囲う
ことでコンポーネントとして扱える

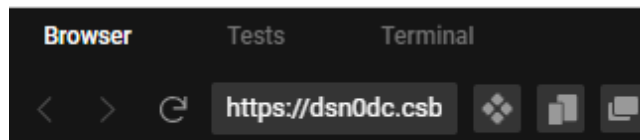
```
ReactDOM.render(<App />,
  document.getElementById("root") );
```

問題 1 JSX 記法 ~ CodeSandbox で JSX 記法を用いて、簡単な HTML を出力する

CodeSandbox を使って、index.js に JSX 記法を用いてコードを記載し、次の HTML が表示されることを確認してください。

▶ 表示する HTML

```
<h1>こんにちは！</h1>
<p>お元気ですか？</p>
```



こんにちは！

お元気ですか？

問題 2 コンポーネント分割 ～ App という名前の関数コンポーネントを新規作成し、読み込む

問題 1 で記載したコードからコンポーネントを分割する為、関数の内容を「App.js」を新規作成してコードを記載してください。

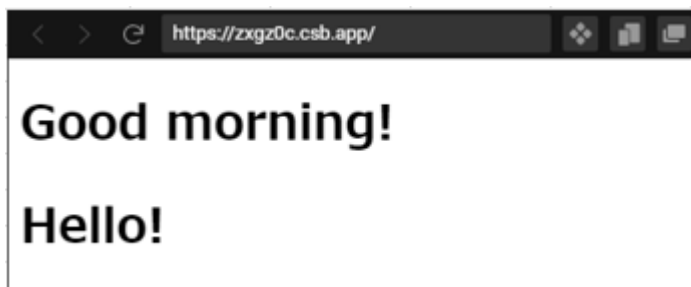


index.js で App という関数コンポーネントを以下のように読み込みましょう。

```
import { App } from “./App”;
```

問題3 関数でコンポーネント作成 ～ Greeting という関数コンポーネントを定義して、表示する

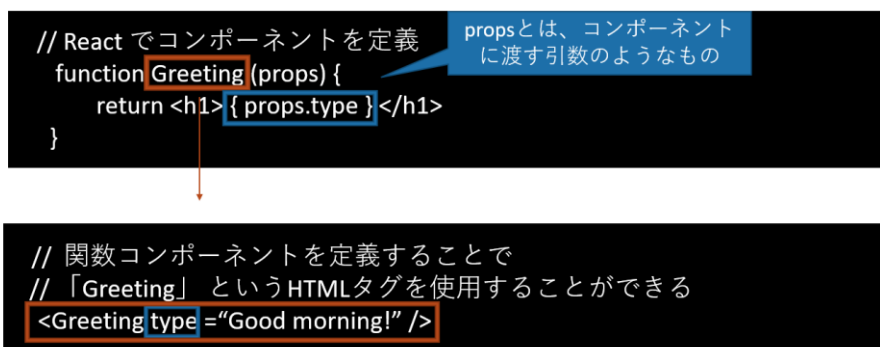
以下のように index.js に関数を定義してコンポーネントを作成し、Greeting というタグを使用して、画面に「Good morning! / Hello!」と表示させてください。



関数定義してコンポーネント作成

- ・コンポーネントを作るには、「JSX」でコンテンツを返す関数を定義する

例えば、「Greeting」というコンポーネントを定義する場合



問題4 スタイルの扱い方 ～ style 属性を記述してスタイルを適用する

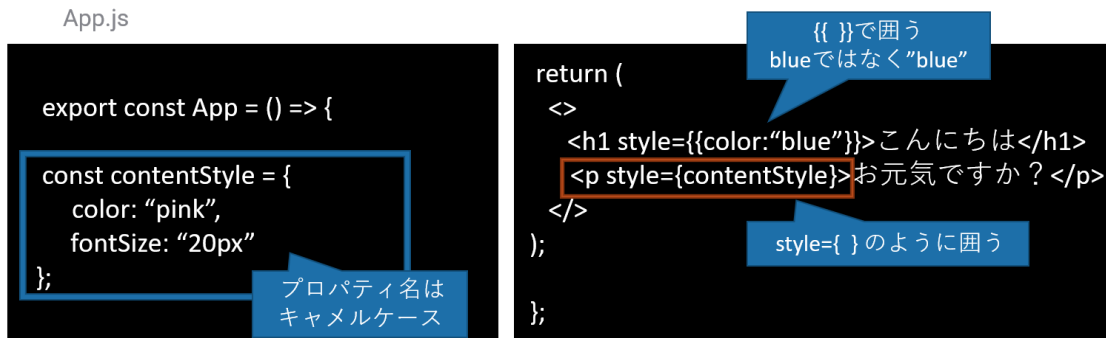
以下のように index.js に作成したコンポーネントに

style 属性を記述してスタイルを適用し、画面に「こんにちは！ / お元気ですか？」と表示させてください。



Reactでスタイルを扱う

- Reactでもstyle属性を記述することでスタイルを適用することができる
- JavaScriptのオブジェクトでもCSSを指定できる



問題 5 Props の利用 ～ 色とテキストを Props として受け取り、文字列を表示する

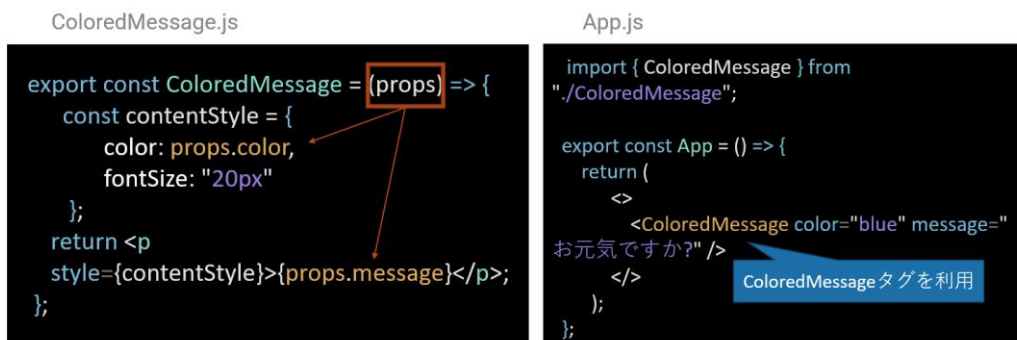
以下のように ColoredMessage.js に Props を利用してコンポーネントを作成し、色とテキストを Props として受け取って色付きの文字を返し、画面に「 お元気ですか? 」と表示させてください。



お元気ですか?

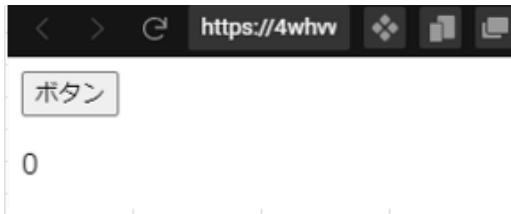
Propsとは

- Propsは、コンポーネントに渡す引数のようなもの
- 「色とテキストをPropsとして受け取って色付きの文字を返す」場合を考えてみましょう



問題 6 State の利用 ～ 数値の State を定義し画面に表示し、ボタン押下時にカウントアップする

以下のように App.js に数値の State を定義し画面に表示し、ボタン押下時にカウントアップする機能を実装してください。



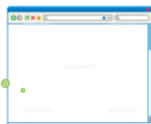
State とは

- State は、コンポーネントの状態を表す値

「状態」は全てStateとして管理し、更新処理を行うことで動的な表現をする

🔗 画面に表示するデータや、可変の状態をStateとして管理

状態管理



cf) エラーはあるか、ボタンを押せるか
テキストボックスに何を入力したか 等

- ▶ useState (ReactHooksと総称される機能群にある関数)

```
const [ num, setNum ] = useState();
```

1つ目：State変数

2つ目：Stateを更新するための関数

▶ useStateの import
import { useState } from "react"

State とは 続き

- State は、コンポーネントの状態を表す値

「数値のStateを定義し画面に表示し、ボタン押下時にカウントアップ」する場合を考えてみましょう

App.js

```
export const App = () => {  
  const [num, setNum] = useState(0);  
  
  // ボタンを押したときに、Stateをカウントアップ  
  const onClickButton = () => {  
    setNum(num + 1);  
  };  
};
```

初期値「0」

```
return (  
  <>  
    <button onClick={onClickButton}>  
      ボタン</button>  
    <p>{num}</p>  
  </>  
);  
}
```

=====
今回の内容はいかがだったでしょうか。スクリプトを実行しながら動作確認できると面白いなと感じて頂けた方もいらっしゃるかもしれません。自分にできる範囲のものから少しずつ **JavaScript** にも挑戦してみようかなと思っていただければ幸いです。

以上となります。

参考文献：

- ・クジラ飛行機『いまどきの JS プログラマーのための Node.js と React アプリケーション開発テクニック』(第 5 版) ソシム株式会社 (2020 年)
- ・岡田 拓巳『モダン JavaScript の基本から始める **React** 実践の教科書』Informatics&IDEA (2021 年)