TypeScriptとReactで開発をしていると、必ずと言ってもいいほど使うのがReact.FCです。

最初に見たときは「なんぞや？」と思ったので、基本的な点を説明していきます。

React.FCとは？

略さずに書くと、React：FunctionComponent。型の名前です。

Reactには、関数（ファンクション）コンポーネントと、クラスコンポーネントがあるのは有名ですが、その関数コンポーネントを表します。

以下のような形で使用可能で、「MainはReactの関数コンポーネントですよ。」と定義されているわけです。

```
import React from 'react';

const Main: React.FC = () => {
     return(
         <div>
           <h1>Main</h1>
         </div>
     );
}
```

ちなみに、'react' から FCをインポートしておけば、以下のように書くことも可能です。

```
import { FC } from 'react'; // ReactではなくFCをインポート

const Main: FC = () => {
     return(
         <div>
           <h1>Main</h1>
         </div>
     );
}
```

React.FCの特徴

childrenを定義しなくても使用できます

引数には記載する必要がありますが、定義なしで使用可能です。これはかなり楽です。

以下のような形です。

```
import React from 'react';

const Main: React.FC = ({children}) => {
     return(
         <div>
           <h1>Main</h1>

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

ちなみに、childrenの使用例を1つあげます。

上記のchildrenを渡したMainコンポーネントの中にSubコンポーネントが入っていますね。これは、Mainコンポーネントの{children}が入っている部分にSubコンポーネントが入ることを意味しています。

```
import React from 'react';
import Main from 'Main';
import Sub from 'Sub';

const App: React.FC = () => {
    return (
        <Main>
            <Sub/>
        </Main>
    );
}
```

Propsを使用できます

当たり前といえば当たり前ですが。。。

一旦書き方を見てみましょう。

```
import React from 'react';

// Propsの型指定
type InfoProps = ({
    id: number
    name?: string //TypeScriptの場合「?」をつければ、そのプロパティはオプションになる。
    infoList: Array<string>
});
```

```
const Main: React.FC<InfoProps> = ({id, name, infoList}) => {
    return(
        <div>
            {infoList.map((el)=>{
                return(
                    <div>
                        // 省略
                    </div>
                )
            })}
        </div>
    );
}
```

注意ポイントは以下2つです。

- PropsをFCの直後に**<**指定したProps名**>**で指定する
- Props内のプロパティを引数として**{}**に入れて設置