■教育実施記録帳

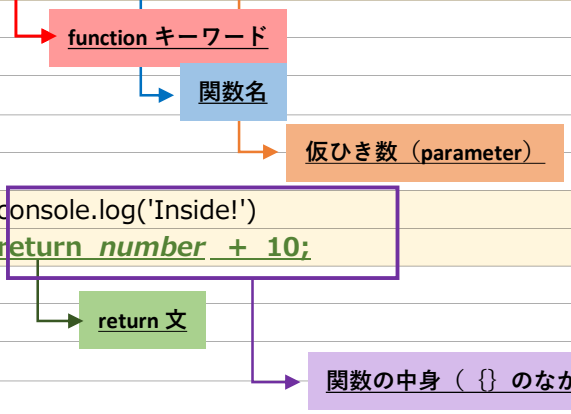
管理No.	FND27 Day12 Page1					
実施内容	値としての関数					
場所	ライン	設備・工程				
実施日	2024/5/17	実施時間	9:00 ～ 12:00			
実施者	Ururaさん Aiさん	受講者				
作成者	本田		作成日(開始)	2024/5/17	作成日(完了)	2024/5/17

<Time>

<Contents>

★関数の構成要素

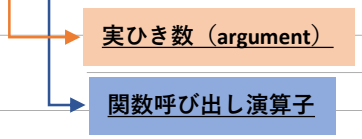
〔JS〕 function addTen(number) {



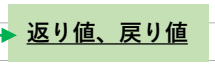

```
    console.log('Inside!')
    return number + 10;
```



```
}
addTen( 5 );
```




```
> 15
```



★関数が行う3つのこと

1. 値を返す
2. 副作用を作り出す
3. 値を返し、副作用も作り出す

関数は何かしらの値を返す

「今更ですが、やっとこさVSCodeの文字色と
ブロックが意味を伴って見えてきた今日この頃」

[. js] 1. 実行すると関数は返り値を返す

```
function average(array) {
  let result = 0;
  for (const number of array) {
    result = result + number;
  }
  result = result / array.length;
  return result;                          //関数avrageは「result」という値を返す
}
```

[. js] 2. 「副作用」を作り出す

```
function sayhello(friend) {
  const hellos = ["Hello", "こんにちは", "Hola", "你好(nǐ hǎo)"];
  const randomIndex = Math.round(Matn.random() * (hellos.length - 1));
  const randomGreeting = hellos[randomIndex];
  console.log(randomGreeting + " " + friend + "!");        //関数sayHelloは副作用を作り出している
}
```

・副作用：返り値を返す(本来の作用)以外の働き

☆スコープ外のなんらかの状態を変えること

■教育実施記録帳

管理No.	FND27 Day12 Page2					
実施内容	値としての関数					
場所	ライン	教室・工場				
実施日	2024/5/17	実施時間	9:00 ～ 12:00			
実施者	Ururaさん	受講者				
	Aiさん					
作成者	本田		作成日(開始)	2024/5/17	作成日(完了)	2024/5/17

<Time> <Contents>

[. js] 3. 返り値を返し、「副作用」を作り出す

```
let cache = [];

  function average(array) {
    let result = 0;
    for (const number of array) {
      result = result + number;
    }
    result = result / array.length;
                                                    //関数averageの
    cache.push(result);                             //副作用と
    return result;                                  //返り値
}
```

★**関数宣言と関数式**

[. js] 関数宣言

```
function addTen( number ) {
  return number + 10;
}
```

[. js] **関数式**（関数を値として変数に代入している式）

```
const addTen = function( number ) {
  return number + 10;
}
```

データ	例
数値	1, -5, 1.0001
文字列	"Hello", "World"
ブーリアン	true, false
undefined	undefined
オブジェクト	{ a: 1 }
配列	[1, 234.05, true, '#hello', [1, 2, 3]]
関数	function nameOfFunction(input1) {}

思い出してみると、**関数はデータ型の一種**。
だから、数値や文字列同様に**関数も変数に代入**したり、**引数として渡したり**、**返り値として関数を返す**ことができて然るべき。

☆引数として渡される関数は**コールバック関数**と呼ばれる

[. js] コールバック関数

```
const double = function(x) {          //変数doubleに関数を値として代入(関数宣言しているわけではない)
  return 2 * x;                       //↑は「function double(x) { return 2*x;};」**ということではない**
};                                                  //変数doSomethingに関数を代入
                                                    //引数は(数値型のnumber,**function型のaction**)
const doSomething = function( number, action) {
  return action (number) ;
};
doSomething(5, double);                             引数に関数の入った変数doubleを渡している
```

☆コールバックする際、一旦変数に代入してからでないと渡せないわけではない

[. js] コールバック関数と無名関数

```
doSomething(5, function(x) {          //関数も値であるため、そのまま引数として渡すことが可能
  return 2 * x;                                     //このとき関数名を入れないスタイルを使う
});                                                 //**無名関数**と呼ばれる
```

☆無名関数は使いまわし不可な、１回しか使わない関数

■教育実施記録帳

管理No.	FND27　Day12　Page3					
実施内容	値としての関数					
場所	ライン	教室・工場				
実施日	2024/5/17	実施時間	9:00 ～ 12:00			
実施者	Ururaさん	受講者				
	Aiさん					
作成者	本田		作成日(開始)	2024/5/17	作成日(完了)	2024/5/17

<Time>

<Contents>

★関数宣言と関数式の違いと関数の巻き上げ

・関数宣言は、スコープの先頭で定義されたと見なされる

・関数宣言で宣言された関数は、スコープの先頭に巻き上げられる

[.js]　関数宣言(functionn命令) と関数式を同義と捉えた場合

```
//関数を定義を格納した変数myNameはまだ宣言していないため両方エラーになるはずだが...
//関数宣言
myName("Yan", "Fan");

function myName( first, last) {
   window.alert( first + " " + last);
}
```

関数宣言は
何事もなく表示される


```
//関数式
myName("Yan", "Fan");

const myName = function( first, last) {
   window.alert( first + " " + last);
}
```

関数式は
エラーになる

「myNameが初期化される前にアクセスすることは出来ません」

関数宣言がスコープの先頭にあるかのように「他のコードが実行されるより先に
コンピュータに読み込まれ処理される」ためエラーにならない

[.js]　関数宣言(functionn命令) は関数の巻き上げが起きる

```
//関数宣言
function myName( first, last) {
   window.alert( first + " " + last);
}

myName("Yan", "Fan");

function myName( first, last) {
   window.alert( first + " " + last);
}
```

```
//関数式

myName("Yan", "Fan");

const myName = function( first, last) {
   window.alert( first + " " + last);
}
```

関数式以降に書くべき

☆関数式はコンピュータがその行に達して初めて読み込まれる。関数読み込めるのはそれ以降。