■教育実施記録帳

管理No.	FND27 Day5 Page1					
実施内容	条件分岐入門					
場所	ライン	設備・工程				
実施日	2024/5/8	実施時間	9:00 ～ 12:00			
実施者	Teradaさん	受講者				
	toraさん					
作成者	本田		作成日(開始)	2024/5/8	作成日(完了)	2024/5/8

<Time> <Contents>

★テンプレートリテラル リテラル

・「`**(バッククォート)**」で挟んだ部分が文字列型になる

☆入力：半角英数字入力モードで「shift ＋ @」

・文字列を連結するための「＋」演算子が不要

・改行文字「/ n 」も不要で「Enter」でOK

・空白を表す「” ”」も不要で「Space」でOK

・文字列内で変数を謳う場合は変数名を「 **${** 」と「 **}** 」で挟む

```
// 「"」の場合
console.log("Hello, my name is" + " " + lastName + " " + firstName + "." + "I live in" + " " + city + "." );
// テンプレートリテラルの場合
console.log(`Hello, my name is ${ lastName } ${ firstName } . I live in ${ city }.` );
```

☆コードがシンプルになり、可読性が上がる

★アロー関数

・function命令と同様、関数の書き方の一種

```
// function命令の場合
function greeting (name) {
return "Hello" + "," + name + "!";
}
// アロー関数+テンプレートリテラルの場合
const greeting = (name) => {
return `Hello, ${name} !`;
}
```

☆コードのシンプル化

☆実行する命令によっては意図しない返り値が発生するから注意らしい…その場合、

アロー関数ならではの実行命令の書き方が多々ある為その都度参照…(長くなるから書きたくない！)

★テスト駆動開発(TDD)とは

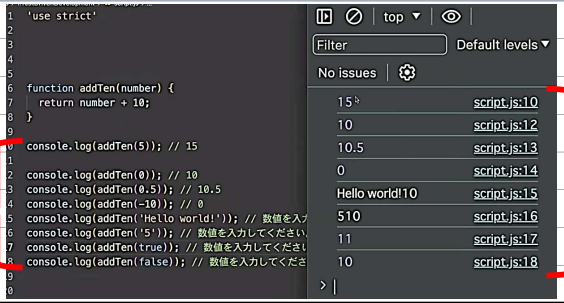
・「test-driven-development」の略

・ソフトウェア開発のプロセス プロセス

☆プロダクションコードを書く前に「**機能要件**」と「**期待値**」が書かれる

```
// プロダクションコード(関数自身)
function addTen(number) {
// 実行命令
}
// テストコード(機能要件と期待値)
console.log(addTen(5));  // 15 を出力するはず
```

・テストコードが増加するにつれて、どのテストをパスしたのか確認困難となる



テストコードの増加 確認困難

■教育実施記録帳

管理No.	FND27 Day5 Page2					
実施内容	条件分岐入門					
場所	ライン	設備・工程				
実施日	2024/5/8	実施時間	9:00 ～ 12:00			
実施者	Teradaさん	受講者				
	toraさん					
作成者	本田		作成日(開始)	2024/5/8	作成日(完了)	2024/5/8

<Time>

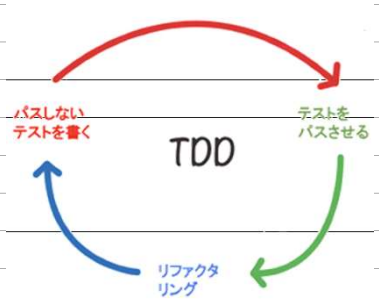
<Contents>

★TDD テストの自動化とは

```
// プロダクションコード(関数自身)
function addTen(number) {
// 実行命令
}
// テストコード(機能要件と期待値)
console.log(addTen(5)); // 15 を出力するはず
// テストコードの自動化(「実際の結果」と「期待する結果」の指定)
let actual = addTen(5);                                    actual
let expected =15;                                          expected
// テストコードの自動化(上記を比較し成功時と失敗時の表示)
if (actual === expected) {                    // 比較
  console.log("Yay! Test Passed.");           // 成功時のコメント表示
} else {
  consol.log("Test Failed. Keep trying!");    // 失敗時のコメント表示
  consol.log("    actual : ", actual);        // 失敗時の実際値の表示
  consol.log("expected : ", expected);        // 失敗時の期待値の表示
}
```

★TDDの基本的なプロセス

1. プロダクションコードを書く前にテストコードを書く
2. テストコードを実行する
3. 期待値を返すだけの最小限のコードを書く
4. 再度テスト実行
5. 必要に応じてプロダクションコードを修正
6. 十分な数のテストコードが書けるまで1～5を繰返す
7. リファクタリングして最終テスト実行



☆ポイント

・テストではエッジケースも考量する　エッジケース

・コツが掴めるまで 1 個のテストコードを書き、その都度テストする

・テストコードがプロダクションコードよりも多くなるがそういうもの

★TDDの必要性

・コードを書く前にコードの設計について考える助けになる（視野を広げる的な？…）

・ミスコードがあってもテスト実行で問題の後追いが可能

・テストをパスするたびに達成感があり、クセになり、ハイになる＼('ω')／ｲﾔｯﾌｰ！

★コードスタイルとは

・人間にとって読み易く、理解し易いコード

★コーディング規約(スタイルガイドライン)とは

・コードの品質を高めるための基準

☆インデントやスペース、括弧の始終位置、ファイルやクラス・関数・変数の名前、プロジェクトの構成

☆使用プログラミング言語、活用技術、チームや会社の好みで異なる

■教育実施記録帳

管理№	FND27　　Day5　　Page3					
実施内容	条件分岐入門					
場所	ライン	設備・工程				
実施日	2024/5/8	実施時間	9:00　～　12:00			
実施者	Teradaさん	受講者				
	toraさん					
作成者	本田		作成日(開始)	2024/5/8	作成日(完了)	2024/5/8

<Time>	<Contents>
	★変数名のコードスタイル
	・意味のある、見分けが容易な名前を付ける
	・名前の付け方も一貫させる
	★関数名のコードスタイル
	・動詞をつける
	・何をする関数なのか解るようにする
	★インデントは大事
	https://beautifier.io/
	☆js-beautify：JavaScript,CSS,HTMLそれぞれ可能
	・コードを貼り付けるとインデントを揃えてくれるサイト
	★コードスタイルの参考書
	https://github.com/airbnb/javascript
	☆Airbnbの公開コードスタイル
	・コードスタイルを真似て学ぶ、自分の引き出しを増やせるサイト
	★JSDocコメント「/ ○○ */」**
	・ルールに沿ったコメントを書くことで可読性を高める
	/**
	* @param {number} num - 引数として与えられる数値
	* @returns {number} 引数の数字に 10 を足したもの
	*/
	・汎用性のある例
	/**
	* @param {string} param1 - 1 番目の引数
	* @param {boolean} param2 - 2 番目の引数
	* @returns {number} 返り値
	*/
	function funcName(param1, param2) {
	return 42;
	}
	☆JavaScriptのコーディング規約「Google JavaScript スタイルガイド」にも、
	「すべてのファイル、クラス、メソッド、プロパティにJSDocコメントが、適切なタグとデータ型を伴って記されるべきです。 また名前から明白に判断できる場合を除き、プロパティ、メソッド、メソッドの引数、メソッドの戻り値を説明する文章が含まれているべきです。」
	とJSDocコメントの活用を推奨されている。