■教育実施記録帳

■ 羽月天川		Davic	Da == 1		٦				
管理No.	FND27	Day5 Page1			4				
実施内容	条件分岐入門	i 股份·工程			4				
場所		<u> </u>			4				
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	<u> </u>	作成日(開始)	2024/5/8	作成日(完了)	2024/5/8			
<time></time>	<contents></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								
		角英数字入力モー							
	・文字列を連結するための「+」演算子が不要								
		'n 」も不要で「Ent							
	・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("H	console.log("Hello, my name is" + " " + <i>lastName</i> + " " + <i>firstName</i> + "." + "I live in" + " " + <i>city</i> + ".");							
	// テンプレートリ	テラルの場合							
	console.log(`H	Iello, my name is \$	{ lastName } \${ firs	stName } . I live in	\${ city } .`);				
	☆コードがシン	プルになり、可読性	生が上がる						
	★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								
	☆ ま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。 								
	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・・・(長なるから書きたくない!)								
	、ロールメン・いっくにはい人にはは、ロッロにハルグ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・								
	★テスト駆動開発(TDD)とは								
	★テスト能動用発(TDD)とは •「test-driven-development」の略								
	・ソフトウェア開		101-21-E			プロセス			
			機能要件ル「期待	寺値 が書かれる		7.1.0%			
	☆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)								
	// プロダクションコート(関数目身) function addTen(number) {								
		// 実行命令	Chi(hamber) \						
	// - -7 -								
	// 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								
	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かる								
	・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								
		2	Filter	Default level	s▼				
		4 5 6 function addTen(number)	{	sues 🕏					
	-	7 return number + 10;	15						
		console.log(addTen(5));	// 15	.5 script.js:1	3				
	1.	2 console.log(addTen(0)); 3 console.log(addTen(0.5))); // 10.5 He	script.js:1- llo world!10 script.js:1-	5				
	テストコードの増加 -	4 console.log(addTen(-10)) 5 console.log(addTen('Hell 6 console.log(addTen('5'))); // 0 lo world!')); // 数値を入す 51 b; // 数値を入力してください 11	0 script.js:10					
		7 console.log(addTen(true) 8 console.log(addTen(false))); // 数値を入力してください						
		0	>						

■教育実施記録帳

		Davis	D= == 2							
管理No.	FND27	Day5	Page2		_					
実施内容	条件分岐入門	i 設備・工程			_					
場所		!			_					
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	<u> </u>	作成日(開始)	2024/5/8	作成日(完了)	2024/5/8				
<time></time>	<contents></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(// 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	// 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	d = 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.ld	og("Yay! Test Pas	ssed.");	// 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									
		}	<u> </u>	, , , , , , , , , , , , , , , , , , , ,	77 - 33 - 3 - 3 - 3 - 3 - 3 - 3 - 3 - 3					
		,								
	★TDDの基	★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									
		ど返すだけの最小	限のコードを書く	テスト	を書く	デストを パスさせる				
	4. 再度テス			1	TDD	1				
		こ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									
		ノングして最終テス								
	7. リファクタリングして最終テスト実行 ☆ポイント									
		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									
		・コッカ・担めるまで 1 個のテストコートを書き、その印度テスト 9 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								
	7,7,13,17	・ フトロコードがフロフノンコンコードをフロ多くはるかでしていていい								
	★ TDDΦ'⋈	★TDDの必要性								
		★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?…)								
					אינאס :)					
		・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									
	★ 」 ートスタイルとは ・ 人間にとって読み易く、理解し易いコード									
	・八川ICCノし訳の分へ、上呼し勿いコート									
	→ コーデッンが担約/フカノ# ゼノビニノンント/t									
	★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

	施記録帳				_					
管理No.	FND27	Day5	Page3							
実施内容	条件分岐入門	i 設備·工程			_					
場所										
実施日	2024/5/8	実施時間	9:00	~ 12:00						
実施者	Teradaさん	受講者		¦ ∔		ļ -				
	toraさん									
作成者	本田	<u> </u>	作成日(開始)	2024/5/8	作成日(完了)	2024/5/8				
<time></time>	<contents></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								
	・名前の付けた	・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	・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								
	・コードを貼りた	・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								
	43 K364									
		★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	・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								
	+15Doc74	+1CDoc747.b[/** 00 */								
		★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									
	*/									
	/**									
	* @param	* @param {string} param1 - 1 番目の引数								
		* @param {boolean} param2 - 2 番目の引数								
		{number}								
	*/									
	•	function funcName(param1, param2) {								
		return 42;								
	}	·								
	☆JavaScrip	tのコーディング規	見約「Google JavaS	Script スタイルガ	イド」にも、					
	「すべてのファ	「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。」								
	∠اSDoc	とJSDocコメントの活用を推奨されている。								