■教育実施記録帳

管理No.	FND27	Day8	Page1						
実施内容									
場所	forof ループ、TDD レビュー、HTML + CSS &レイアウト								
実施日	2024/5/13	実施時間	9:00	~ 12:00					
実施者	Toraさん	受講者		İ		İ			
	Masaきん			.+ 		-} 			
作成者	本田		作成日(開始)	2024/5/13	作成日(完了)	2024/5/13			
<time></time>	<contents></contents>				,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,				
	★.indexOf()メソッド								
		[構文] arr.indexOf(検索する要素,検索開始位置)							
	[利用] 配列の要素の位置を 先頭から 検索								
	[戻り値] 見つ	かった要素位置(インデックス)、要素が見つからない場合は「-1」を返す							
	•検索開始位	・検索開始位置は省略可能							
	// 配列の作成								
	const arrTreats = ["apple", "banana", "chocolate", "doughnut"];								
	// 例1:要素	の位置を先頭から	検索						
	console.log	(arrTreats .inde	exOf("doughnu	ıt")); // ⇒ 3					
	// 要素がない	場合							
	console.log	(arrTreats .inde	exOf("ramen")); // ⇒ -1					
	// ウォーミング	アップ復習:末尾	に追加した要素を	取得する					
	arrTreats.p	ush("takoyaki"));						
	console.log	console.log(arrTreats[arrTreats.length -1]); // ⇒ " takoyaki"							
	//分解すると								
	_	(arrTreats.leng	* :		//23	//ここまでだと要素番号			
		(arrTreats[5 - 1		yaki	//23	こでインデックス値に			
		立置を指定した場合							
	console.log(arrTreats.indexOf("chocolate", 0)); $// \Rightarrow 2$								
	console.log(arrTreats. indexOf("chocolate", 1)); $//\Rightarrow 2$								
	console.log(arrTreats.indexOf("chocolate",2)); $// \Rightarrow 2$								
	console.log(arrTreats.indexOf("chocolate",3)); $// \Rightarrow -1$								
	const arrTreats = ["apple", "banana", "chocolate" , "doughnut", "takoyaki"];								
	1 フテック.	$\lambda \Rightarrow [0,$	1 ,	<u> </u>	3 , 4 1				
						>			
	ここから探し始めてここから後にはないよ								
		インデックス「2」(: xOf()メソッド :		 立署を 士尾から 検	·····································	_			
	₩.lastilide	:XUI()メンット .	凹がりの安米の仏	立	(术				
	★for…of命	 숙							
		or(変数 of 配列	ii) {			c列に要素が 残っている			
		…繰り返しの処理	true						
	}	1X 7 2 0 1 7 2 - 1							
	任意の処理								
	[例] 配列の値を1つずつ変数へ代入して繰り返し処理をする								
	[戻り値] 繰り返し処理が終わるまで、処理毎の値を変数へ都度代入								
	// 配列の作成								
	const colors = ["red", "green", "blue"];								
	// 例								
	for (const color of colors) //colors配列に取出していない内容が有ったらcolor変数に内容を一つ取出し								
	console.log(color); //コンソールに表示。								
	} //colors配列に取出していない内容が無くなったら処理終了								
	// 戻り値								
	red	// 1回目の処理	の結果 ・・・ 配列	列に要素残ってい	3				
	green // 2回目の処理の結果・・・・配列に要素残っている								
	blue	// 3回目の処理	の結果 ・・・ 配列	別に要素残っている	ないから処理終了				

■教育実施記録帳

	他記録帳	<u> </u>	D 1		1					
管理No.	FND27	Day8	Page1	<i>1</i> →±1	4					
実施内容	forof ループ、7	DD レビュー、HTI	ML + CSS &レ	1						
場所			·TE							
実施日	2024/5/13	実施時間	9:00	~ 12:00	ļ					
実施者	Toraさん	受講者		¦ ∤	 	 				
	Masaきん									
作成者	本田	<u> </u>	作成日(開始)	2024/5/13	作成日(完了)	2024/5/13				
<time></time>	<contents></contents>									
		★テストコード、プロダクションコードの中で配列の繰り返し処理を行うには?								
		// 関数にしたテストを書く								
	function test(actual, expected) {									
	` `	if (JSON.stringify(actual) === (JSON.stringify(expected))) {								
		g("Yay! Test P	PASSED.");							
	} else {									
		console.error("Test FAILED. Keep trying");								
		console.log("expected:",expected);								
		.log(" actual:",	,actual);							
	}									
	}									
	// JSONJ'םי	りを書く								
	/**	/**								
	•	* @param {Array <number>} arrNums //数値型の要素を持つ配列</number>								
		* @returns {number} sumResalt //与えられた配列のすべての数字の合計								
	* 関数sumArrayは数値型の要素を持つ配列を1個取り、									
	*	* 与えられた配列のすべての数字の合計を数値型データで返す								
	*/	*/								
	// テストコード	// テストコードを書く								
	test(sumArr	test(sumArray([1, 2, 3, 4]),10);								
	・求める期待値	・求める期待値に使用さる配列の内容と、処理結果の実際値が指定される								
	// プロダクション	// プロダクションコードを書く								
	function sum	function sumArray(arrNums) {								
	•const sumArray = [1, 2, 3, 4] //[]部分を再利用するために関数にしてい									
	// 処理結果の初期化									
	let sumRes	et sumResalt = 0;			//合計値用変数の初期値決定					
	// 何かする									
	for (const a	<i>rrNum</i> of <i>arrl</i>	Nums) {	//[:	1 <mark>, 2, 3, 4]</mark> から1つ	ずつarrNumに代入				
		salt = sumRes			//合計値用変数に	rrNumの値を加算				
		.log("sumResal	tl:", <mark>sumResa</mark>	lt)	//現段階でのsun	Resaltの値を表示				
	}				//上記3点を配	列末尾まで繰り返す				
	// 処理結果を返す									
	return sumResalt;									
	•test(sumA	・test(sumArray([1, 2, 3, 4])の返り値がsumResaltになり、,実際値10);との比較となる								
	・実行時のコンソールの表示									
	sumResalt : 1									
	sumResalt : 3									
	sumResalt : 6									
	sumResalt : 10									
	Yay! Test	Yay! Test PASSED.								
	☆「結果の初期化、何かする、結果を返す」をパターンとして覚える									

■教育実施記録帳

	FND27 Day8 Page1				_				
	FND27	Day8							
実施内容	forof ループ、T	DD レビュー、HTI							
場所	512	数据-工程							
実施日	2024/5/13	実施時間	9:00	~ 12:00	7				
実施者	Toraさん	受講者		i !	!	İ			
	Masa さん			 		} !			
 作成者	本田		作成日(開始)	2024/5/13	作成日(完了)	2024/5/13			
<time></time>	<contents></contents>	!	(ווינויו) בויינויו	202 1/ 3/ 13	1113211(203)	202 1/ 3/ 13			
VIIIIC>	★テスト用の関数を書く目的								
	★ナスト用の関数を書く目的・同じ(しかも)長いコードを何度も書かないで済むようにするため								
			ではいるいではの	14 71C 9 31C0)					
	★コードを関数にする前に考えるべき2つのこと								
	・人刀個/引数	と 出力値/retu	urn又						
	★コードを関数にする際の最後のステップ								
	・再利用したい	コードを関数の中身	身に入れる						