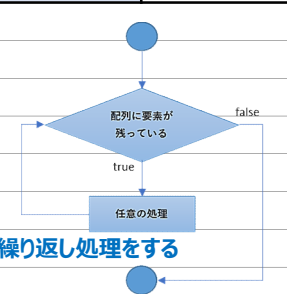


## ■教育実施記録帳

管理No.	FND27	Day10	Page1
実施内容	for...inループ、 値渡しと参照渡し		
場所	ライオン	設備・工数	
実施日	2024/5/10	実施時間	9:00 ~ 12:00
実施者	toraさん Masaさん	受講者	
作成者	本田	作成日(開始)	2024/5/15
		作成日(完了)	2024/5/15
<Time>	<Contents>		
	・オブジェクト入門 ウォーミングアップ		
	[. js] JSON-block		
	/**		
	* @param {Array<object>} arrMultipleObjects 複数のオブジェクトが入った配列		
	* @param {string} キー		
	* @returns {Array<any>} pluckArrayObjs 配列の中のオブジェクトから、		
	* 第 2 引数と同じキーに対応する値だけを拾って (= pluck して)、配列に入れたもの		
	*/		
	・第1引数[参考]arrayOfObjects //関連性を持たせる (どこから、何から)		
	⇒配列の中の3つのオブジェクト:「複数形」にして解りやすくする		
	・第2引数[参考]key //どこに書くのか、どう書くのか迷った箇所		
	⇒key & valueペアのキーが変数(key)の場合、アクセスはブロック記法で書く		
	・問題の逆読み(本来は必要に応じて自分が書くもの): @paraが二つ⇒引数が二つ		
	・返回值[参考]resultArray		
	[. js] test-code		
	test(pluck(arrayOfObjects, "a"), [1, 4, 7]);		
	☆関数シグネチャ		
	関数pluckは、変数arrayOfObjects内の3つのオブジェクトが入った配列を1個と、		
	test(pluck(arrayOfObjects, "a"), [1, 4, 7]);		
	<pre>const arrayOfObjects = [   { a: 1, b: 2, c: 3 },   { a: 4, b: 5, c: 6 },   { a: 7, b: 8, c: 9 }, ];</pre>		
	変数arrayOfObjects内のオブジェクトにアクセスするための文字列型のキーを1個とり、		
	test(pluck(arrayOfObjects, "a"), [1, 4, 7]);		
	第2引数と同じキーに対応する値だけを拾って入れた配列を返す		
	[. js] 関数宣言		
	Function pluck(arrMultipleObjects, key) {		
	const pluckArrayObjs = [];		
	for (const arrayOfObject of arrMultipleObjects) {		
	//キーに変数 (key) を使う時はブラケット記法を使う ドット記法ではうまくいかない		
	pluckArrayObjs.push(arrayOfObject[key]);		
	console.log("pluckArrayObjs:", pluckArrayObjs);		
	}		
	return pluckArrayObjs;		
	}		
	<pre>pluckArrayObjs: ▶ [1] pluckArrayObjs: ▶ (2) [1, 4] pluckArrayObjs: ▶ (3) [1, 4, 7] Yay! Test PASSED. actual: ▶ (3) [1, 4, 7] expected: ▶ (3) [1, 4, 7]</pre>		

## ■教育実施記録帳

管理No.	FND27	Day10	Page2
実施内容	for…inループ、 値渡しと参照渡し		
場所	ライオン	設備・工数	
実施日	2024/5/10	実施時間	9:00 ~ 12:00
実施者	toraさん Masaさん	受講者	
作成者	本田	作成日(開始)	2024/5/15
		作成日(完了)	2024/5/15
<Time>	<Contents>		
	<b>★for…in命令</b> [構文] <b>for ( キー(変数) in オブジェクト ) {</b> …繰り返しの処理… <b>}</b>		
	[例] <b>キーで指定されたオブジェクトの要素を1つずつ取出して変数に入れ繰り返し処理をする</b> [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 <pre>const infoObject = {   name: "Hana",   dog: true,   age: 12 };</pre>		
	// 例1 <pre>for (const key in infoObject) {                //infoObjectkからkey要素を一つずつ取出して   console.log(key);                             //オブジェクト内のkey&amp;valueペアのkey(要素)を表示する }</pre> // 戻り値 name dog age		
	// 例2 <pre>for (const key in infoObject) {                //infoObjectkからkey要素を一つずつ取出して   console.log(infoObject[key]);                //オブジェクト内のkey&amp;valueペアを表示する }</pre> // 戻り値 {name: "Hana", dog: true, age: 12} {name: "Hana", dog: true, age: 12} {name: "Hana", dog: true, age: 12}		
	// 例3 <pre>for (const key in infoObject) {                //infoObjectkからkey要素を一つずつ取出して   console.log(infoObject[key]);                //オブジェクト内のkey&amp;valueペアのvalue(値)を表示する }</pre> // 戻り値 Hana true 12		
	☆["key"]と文字列にすると"key"という名前のキー(要素)を探しに行くので注意！！ // 戻り値 "key"という要素はないため… undefined undefined undefined		

■教育実施記録帳

管理No.	FND27	Day10	Page3																			
実施内容	for...inループ、 値渡しと参照渡し																					
場所	ライオン	設備・工数																				
実施日	2024/5/10	実施時間	9:00	～	12:00																	
実施者	toraさん	受講者																				
	Masaさん																					
作成者	本田		作成日(開始)	2024/5/15	作成日(完了)	2024/5/15																
<Time>	<Contents>																					
	★プリミティブ型データ																					
	・数値型(number)、文字列型(string)、論理型(boolean)																					
	★オブジェクト型データ																					
	・配列(array)、オブジェクト(object)、関数(function)																					
	★プリミティブ型データとオブジェクト型データの違い																					
	!「値を変数に格納する方法」が異なる																					
	☆プリミティブ型：値そのものが直接メモリに格納（値渡し）																					
	・ある変数から別の変数へ値を渡すと、変数は個別に値を持つことになる																					
	<div><pre>let x = 6; let y = x; x = 7;  console.log(x); // 7 console.log(y); // 6</pre></div> <table><tr><td>x</td><td>7</td></tr><tr><td>y</td><td>6</td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table>						x	7	y	6												
x	7																					
y	6																					
	☆オブジェクト型：参照値(値を実際に格納しているメモリ上のアドレスのようなもの)を格納(参照渡し)																					
	・ある変数から別の変数へ参照値を渡すと、両変数ともに同じ一つの参照元の値を持つことになる																					
	・参照元の値（データ）を変化させたいのか、させたくないのか意識する必要あり																					
	<div><pre>let x = [1, 2, 3]; let y = x;  x.push(4);  console.log(x); // [1, 2, 3, 4] console.log(y); // [1, 2, 3, 4]</pre></div> <table><tr><td>x</td><td>a0001</td></tr><tr><td>y</td><td>a0001</td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr><tr><td></td><td></td></tr></table> <div><p>a0001 [1, 2, 3, 4]</p><p>.push(4)</p></div>						x	a0001	y	a0001												
x	a0001																					
y	a0001																					
	★メモリといえば																					
	・RAM（Random Access Memory）																					
	⇒コンピュータが一時的にデータを保管（格納）するするためのパーツ。																					
	⇒セカンダリメモリ（ストレージ）より高速でアクセスできる																					
	☆ここにコード中のデータが格納される																					
	★コンピュータの中で起きていること																					
	・コードを書く																					
	・書いたコードを保存する																					
	・メプログラムを実行する																					
	・保存先からコード中のデータをメインメモリに格納する																					
	・CPUがメインメモリ中のデータにアクセスして演算する																					
	<div><div>コード</div><div><pre>const a = "Hello "; const b = "world"; console.log(a+b);</pre></div><div>HDD / SSD (セカンダリメモリ)</div><div>(メインメモリ。単に 「メモリ」とも呼ばれる。)</div><div>CPU</div><div>Intel platinum</div></div>																					