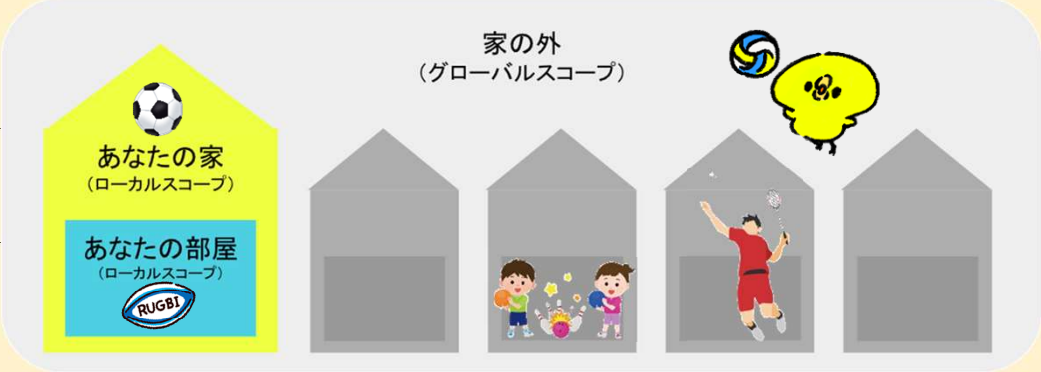


■教育実施記録帳

管理No.	FND27		Day9		Page1	
実施内容	オブジェクト入門、スコープ					
場所	5F	図書・工室				
実施日	2024/5/10	実施時間	9:00 ~ 12:00			
実施者	DJ さん	受講者				
	Ai さん					
作成者	本田		作成日(開始)	2024/5/14	作成日(完了)	2024/5/14
<Time>	<Contents>					
	★オブジェクト (object)とは					
	[.js] const 変数名 = {					
	キー : 値 ,					
	キー : 値 ,					
	};					
	・どのようなデータ型の値でも格納できる					
	・オブジェクトの中は「キー」と値が対になっている					
	・「key-valueペア」(キーと値のペア) の順番は保証されない					
	[.js] 例 : オブジェクト					
	const info = {					
	name : "Hana",					
	dog : true,					
	age : 12,					
	} ;					
	★オブジェクト参照 (ブラケット記法)					
	[.js] 変数名 [" キー "]					
	・オブジェクトの中の値を参照するときは「キー」を指定することで値を参照できる					
	・「キー」は文字列で書く					
	[.js] 例 : ブラケット記法で参照					
	console.log(info ["name"]);				// ⇒ Hana	
	console.log(info ["dog"]);				// ⇒ true	
	console.log(info ["age"]);				// ⇒ 12	
	★オブジェクト参照 (ドット記法)					
	[.js] 変数名 . キー					
	・オブジェクトの中の値を参照するときは「キー」を指定することで値を参照できる					
	・「キー」はありのまま書いてok					
	[.js] 例 : ブラケット記法で参照					
	console.log(info.name);				// ⇒ Hana	
	console.log(info.dog);				// ⇒ true	
	console.log(info.age);				// ⇒ 12	
	★オブジェクトの値を変更・追加					
	[.js] 変数名 [" キー "] = "変更後の値" ;					
	・追加と変更の方法は同じ					
	☆キーが既にあれば値が更新され、キーが元から存在しなければ「key-valueペア」が追加される					
	[.js] 例 : オブジェクト					
	const lesson = {				//オブジェクトの宣言	
	instructor : "Eriko",					
	coding : true,					
	} ;					
	lesson.instructor = "Tmaroh";				//値の変更 (ドット記法)	
	lesson["place"] = "Zoom";				//値の追加 (ブラケット記法)	
	lesson.school = "DIG";				//値の追加 (ドット記法)	
	console.log(lesson);				// ⇒ {instructor:"Tamaroh", coding:true, place:"Zoom", school:"DIG"}	

■教育実施記録帳

管理No.	FND27	Day9	Page2
実施内容	オブジェクト入門、スコープ		
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	DJ さん	受講者	
	Ai さん		
作成者	本田	作成日(開始)	2024/5/14
		作成日(完了)	2024/5/14
<Time>	<Contents>		
	★スコープとは ・ある区切られた範囲を指し、スコープの中で宣言された変数はその中でしかアクセスできない		
	★スコープの概念 		
	☆あなたには 自分の部屋の中 にあるラグビーボールが見えますが、それはあなたしか使えません。 ☆あなたには 自分の家の中 にあるサッカーボールも見えますし、それは家族で使えます。 ☆ 家の外 にあるバレーボールも見えますが、それは地域の皆さんで使えます。 ☆でも他の人の家にあるシャトルや、部屋にあるボーリングの球は見えませんし、使えません。 ☆他の人も あなたの家の中 にあるサッカーボールや、 部屋にある ラグビーボールは見えませんし、使えません。		
	・ボールは「変数」を表しており、使える範囲を「スコープ」という ・すべての変数が、コードの全ての場所で使えるわけではない		
	★関数とスコープ <pre> let a = 5; let b = 3; function addX(num) { let x = 5; return num + x; } addX(2); </pre> ↓これがスコープ let num = 2; x = 5; return num + x; ↑		
	//関数呼出し ⇒ JavaScriptがコンピュータの中に呼出す区画を作る ・関数の処理が終了すると、スコープもその中の変数もコンピュータ上から消える ☆ 関数を定義する ということは 処理をまとめる というだけでなく、 変数が有効な範囲を決める 新しいスコープを作っている と言えます（抜粋：JavaScript Primer）		
	★スコープの内と外 <pre> let a = 5; let b = 3; function addX(num) { let x = 5; return num + x; } addX(2); console.log(x); </pre> スコープの内からは変数 x にアクセス可能 let num = 2; x = 5; return num + x; ↑ //関数呼出し ⇒ JavaScriptがコンピュータの中に呼出す区画を作る //Uncaught ReferenceError: x is not defined ↑スコープの外からは変数 x にアクセス出来ない		

■教育実施記録帳

管理No.	FND27	Day9	Page3
実施内容	オブジェクト入門、スコープ		
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	DJ さん	受講者	
	Ai さん		
作成者	本田	作成日(開始)	2024/5/14
		作成日(完了)	2024/5/14
<Time>	<Contents>		
	★関数スコープ		
	let a = 5;		
	let b = 3;		
	function addX(num) {		
	let x = 5;		
	return num + x;		
	}		
	addX(2);		
	//関数呼出し ⇒ JavaScriptがコンピュータの中に呼出す区画を作る		
	console.log(x);		
	//Uncaught RefarenceError: x is noto defined		
	↑スコープの外からは変数 x にアクセス出来ない		
	・関数スコープは関数呼出し演算子で呼び出されると作成される		
	・関数内で宣言された「変数・仮引数」は、関数スコープ内でのみアクセス可能		
	☆仮引数は、関数スコープ内で宣言され、引数によって初期化される		
	☆仮引数は、関数に引数を取り込むのに使用されるローカル変数(特定のスコープ内だけの変数)		
	★ブロックスコープ		
	・ { } で囲んだ範囲をブロックと呼び、ブロックもスコープを作成し、それをブロックスコープと呼ぶ		
	・ブロック内で定義した変数は、スコープ内でのみ参照可能		
	{		
	const x = 1;		
	console.log(x);		
	// ⇒ 1		
	}		
	console.log(x);		
	// ⇒ RefarenceError: x is not defined.		
	↑スコープの外から x を参照できないためエラー		
	★ブロックスコープを使うもの		
	・if文やwhile文		
	let num = 20;		
	if (num > 0) {		
	let resalt = "正の数だ！"		
	console.log(resalt);		
	// ⇒ "正の数だ！"		
	}		
	console.log(resalt);		
	// ⇒ RefarenceError: x is not defined.		
	↑スコープの外から x を参照できないためエラー		
	★for文のブロックスコープ		
	・ループごとに新しいブロックスコープを作成し、処理が完了すると破棄される		
	☆ループするたびに新しいブロックスコープ内で変数が宣言され、初期化されているからconstで		
	変数を定義しても重複エラーにならない		
	★ローカルスコープ		
	・関数スコープとブロックスコープを合わせてローカルスコープと呼ぶ		
	・ローカルスコープの外側をグローバルスコープと呼ぶ		

■教育実施記録帳

管理No.	FND27	Day9	Page4
実施内容	オブジェクト入門、スコープ		
場所	5F	設備・工室	
実施日	2024/5/10	実施時間	9:00 ~ 12:00
実施者	DJ さん	受講者	
	Ai さん		
作成者	本田	作成日(開始)	2024/5/14
		作成日(完了)	2024/5/14
<Time>	<Contents>		
	★グローバルスコープ		
	・ローカルスコープで宣言された変数は、グローバルスコープからアクセスできない		
	・グローバルスコープで宣言された変数は、コードの内のどこからでもアクセス出来る		
	const greeting = "Hello";		グローバルスコープ
	function greet(person) {		
	return greeting + " " + person;		ローカルスコープ
	}		
	console.log(greet("Urara"));		
	// ⇒ "Hello Urara"		
	★コンピュータの挙動		
	・スコープ内で変数の値が見つからない場合、一致する名前の変数が見つかるまで、		
	内側のスコープから外側のスコープへと順に探す		
	☆変数の値を外側に探しに行く		
	const greeting = "Hello";		グローバルスコープ
	function greet(person) {		
	return greeting + " " + person;		ローカルスコープ
	}		
	console.log(greet("Urara"));		
	// ⇒ "Hello Urara"		
	★スコープ中のスコープ（スコープがネストされている場合）		
	・外部スコープの変数は、内部スコープからアクセス可能		
	・内部スコープ内の変数は、外部スコープからアクセスできない		
	function sum(arrOfNums) {		
	let total = 0;		
	for (const number of arrOfNums) {		
	total += number ;		
	}		
	console.log(number);		
	return total;		
	}		