1. 変数の宣言(定義)

大文字小文字を区別する(textKey と textkey は別の変数として認識される)

コード	説明	再代入	再宣言	スコープ	テンポラルデッドゾーン
const 変数名 = 値;	スコープも狭いconstは最も制約が多いからこそ、意図しない実装を防ぐ = 安全なキーワードといえる。(*1)	×	×	ブロックスコープ(*2)	○ 代入前に読み込みあるいは書き込みをしようとするとエラーになる。
let 変数名 = 値;	変数の中身を更新する必要がある場合には、letを使う。(*1)	○	×		
var 変数名 = 値;	いまは使わないが、古いソースコードにはあるので理解しておく必要あり。 新しく作るプログラムの場合はconstかletを使う。	○	○	関数スコープ (*3)	×

*1: ECMAScript2015(エクマスクリプト…JavaScriptの標準仕様)にて採用された宣言方法
*2: ブロックスコープ…{}の中で定義した変数は、その{}の中でのみ有効
*3: 関数スコープ... 同じ関数内であれば、{}の中で定義した変数を{}の外で使用可能

参考文献:TechAcademy Magagine 「JavaScriptで書く「var,let,const」の違いと使い分け方法」 2018/3/9
https://magazine.techacademy.jp/magazine/14872#ta-toc-2

2. getElement 要素の取得

DOM操作による要素へのアクセス

コード	引数	使用例
getElementById("引数")	id	HTMLのidが"btn1"の要素を取得し、変数btn1にセット。 `const btn1 = document.getElementById("btn1");`

getElementsByTagName("引数")	tag	HTMLのタグが"body"の要素を取得し、変数fireworksにセット。そして背景画像を変える。
		`const fireworks =document.getElementsByTagName("body");`
		`fireworks[0].style.backgroundImage =`
		`"url('img/fireworks.gif')";`

getElementsByName("引数")	name属性	HTMLのnameが"radio1"の要素をすべて取得し、変数radio1にセット。 `const radio1 = document.getElementsByName("radio1");`
getElementsByClassName("引数")	class	HTMLのclassが"square"の要素をすべて取得し、変数squaresにセット。 `const squares = document.getElementsByClassName("square");`

ほかにも色々あるので、自分でしらべてみましょう。

3. EventListener イベントリスナー

イベントリスナー	コード

基本	`要素.addEventListener("イベントの種類",`
	`  function() {`
	`  // イベントが発生した時に動かす処理をかく`
	`  }, false(*1)`
	`);`

使用例
ページ本体が読み込まれたタイミングで実行する
	`window.addEventListener("DOMContentLoaded",`
	`  function() {`
	`  // ページ本体が読み込まれたタイミングで実行するコードをかく`
	`  }, false`
	`);`

使用例
ターゲットの要素がクリックされたタイミングで実行する
	`const btn1 = document.getElementById("btn1");`
	`btn1.addEventListener("click",`
	`  function() {`
	`  // ターゲットの要素がクリックされたタイミングで実行するコードをかく`
	`  }, false`
	`);`

省略(アロー関数)	`要素.addEventListener("イベントの種類", () => {`
	`  // イベントが発生した時に動かす処理をかく`
	`  });`

使用例
ページ本体が読み込まれたタイミングで実行する
	`window.addEventListener("DOMContentLoaded", () => {`
	`  // ページ本体が読み込まれたタイミングで実行するコードをかく`
	`});`

使用例
ターゲットの要素がクリックされたタイミングで実行する
	`const btn1 = document.getElementById("btn1");`
	`btn1.addEventListener("click", () => {`
	`  function() {`
	`  // ターゲットの要素がクリックされたタイミングで実行するコードをかく`
	`});`

*1: イベントリスナーの第3引数 useCapture の詳細は、
Memoアプリ 10. version-up5(Event Delegation イベントの委任)にて勉強します。

4. HTML elements 要素の中身を変更

htmlで使われているタグの中身を取得・変更

コード	説明	使用例

要素名.innerHTML	参考文献:Qiita 「【JavaScript】innerHTMLの使い方」2021年01月10日 https://qiita.com/mzmz__02/items/7bcbce347bc3c5d64b93	const elementResult = document.getElementById("result");
		elementResult.innerHTML = 0;
		---------- 	
		const td1 = document.createElement("td");
		td1.innerHTML = "<input name='radio1' type='radio'>";

要素名.textContent	参考文献:ITSakura Blog for business and development「JavaScript textContentとinnerHTMLの違いのサンプル」2021/10/12 https://itsakura.com/js-textcontent-innerhtml	const btn1 = document.getElementById("btn1"); btn1.textContent = "Happy!!";

5. HTML createElement / appendChild 要素の生成

htmlのタグを作成

コード	説明	使用例
document.createElement(tagName);	tagName で指定された HTML 要素を生成する。	**変数trに、HTMLの「trタグ」を生成してセット** const tr = document.createElement("tr");

親要素.appendChild(child);	特定の親要素の中に要素を追加する。	**HTMLの「id="list"」が付与されているタグが親要素となり、その中に、trタグを生成。**
		さらにtrタグの中に、tdタグを生成。
		const list = document.getElementById("list");
		const tr = document.createElement("tr");
		list. appendChild(tr);
		tr.appendChild(td);

6. change style スタイルの変更

DOM操作によるスタイルの変更

コード	使用例
要素名.style.color	const btn1 = document.getElementById("btn1"); btn1.style.color = "#ff0000";
要素名.style.fontSize	const btn1 = document.getElementById("btn1"); btn1.style.fontSize = "55px";

要素名.style.backgroundImage	const body = document.body;
	body.style.backgroundImage = "url('img/fireworks.gif')";
	---------- 	
	const fireworks = document.getElementsByTagName("body");
	fireworks[0].style.backgroundImage = "url('img/fireworks.gif')";

ほかにも色々あるので、自分でしらべてみましょう。

7. classList スタイルの変更

要素にクラス名を追加・削除、参照することが出来る。

コード	使用例
要素名.classList.add("クラス名")	const start = document.getElementById("start"); start.classList.add("js-inactive");
要素名.classList.remove("クラス名")	const start = document.getElementById("start"); start.classList.remove("js-inactive");
要素名.classList.contains("クラス名")	const start = document.getElementById("start"); start.classList.contains("js-inactive");

ほかにもあるので、自分でしらべてみましょう。

使用例

html

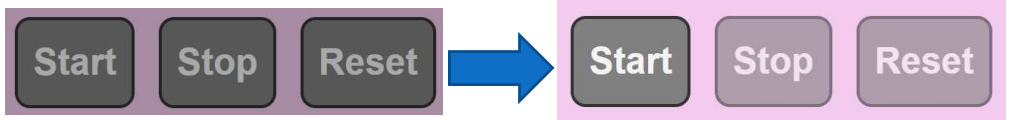
```
<div class="btn" id="start" type="button">Start</div>
<div class="btn" id="stop" type="button">Stop</div>
<div class="btn" id="reset" type="button">Reset</div>
```

css

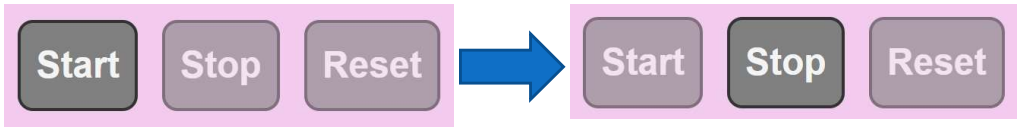
```
.js-inactive {
  opacity: 0.6;
}
```

js

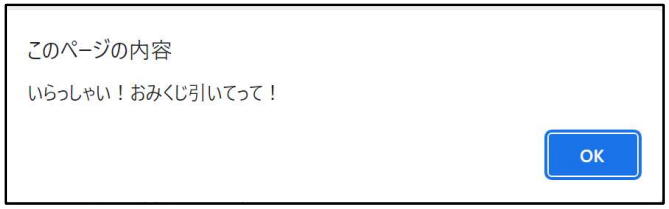
```
const start = document.getElementById("start");
const stop = document.getElementById("stop");
const reset = document.getElementById("reset");

start.classList.remove("js-inactive");
stop.classList.add("js-inactive");
reset.classList.add("js-inactive");
```




```
start.classList.add("js-inactive");
stop.classList.remove("js-inactive");
reset.classList.add("js-inactive");
```



8. alert / confirm ダイアログボックスの表示

項目	コード	説明
警告	window.alert(メッセージ) ※省略可能 alert(メッセージ)	OKボタンのみ。
使用例	let popmsg = "いらっしゃい！おみくじ引いてって！"; window.alert(popmsg);	
確認	window.confirm(メッセージ) ※省略可能 confirm(メッセージ)	OKボタンとキャンセルボタンがある。 どちらのボタンを押されたかを受け取り、処理の分岐が可能。

使用例	let w_confirm = window.confirm("LocalStorageに¥n「" + key + " " + value + "」¥nを保存(save)しますか？");
	if (w_confirm === true){
	// OKボタンがおされたときに動かす処理をかく
	}
	
	※上記のkey、valueは変数です。
		

9. Array　配列用関数（組み込み関数）

コード	説明
Array.from()	Array.from() メソッドは、反復可能オブジェクトや配列風オブジェクトからシャローコピー(*1)された、新しい Arrayインスタンスを生成する。 *1:シャローコピーとは、配列やオブジェクトなどのデータ構造を複製する際、参照のみをコピーして実体の複製は作らない方式。

使用例	// class="square" を取得（しゅとく）...squaresはHTML Collectionで、arrayではないため、Array用の組み込み関数が使えない。
	const squares = document.getElementsByClassName("square");
	
	// Array に変換（へんかん）...squaresArrayはArray用の組み込み関数が使える。
	const squaresArray = Array.from(squares);

Array.filter()	配列のfilter()メソッドは、既存の配列から指定された条件に該当する要素を持つ新しい配列を作成する。

使用例	// 配列の中から奇数だけを取り出す
	let array1 = [1, 4, 7, 12, 21];
	let result1 = array1.filter(function(e) { return e % 2 === 1; });
	console.log(result1); // [1, 7, 21]
	---------- 	
	// 配列の中から5以上の数字を抽出する
	let array2 = [1, 2, 3, 4, 5, 6, 7, 8, 9];
	let result2 = array2.filter(function(e) { return e >= 5; });
	console.log(result2); // [5, 6, 7, 8, 9]
	---------- 	
	// 文字列の中から条件に合った文字列を抽出する
	let array3 = ["item1", "item2", "item3"];
	let result3 = array3.filter(function(value) { return value === "item2"; });
	console.log(result3); // ['item2']

9. Array　配列用関数（組み込み関数）（つづき）

コード	説明
Array.some()	**someメソッドは、配列内のいずれかの要素が条件に合致する場合にtrueを返す。**

使用例	`// 33は3で割り切れるため、以下の処理はtrueを返す。`
	`let array4 = [10,20,33,40,50];`
	`let result4 = array4.some(function(value1){ return value1 % 3 ==0; });`
	`console.log(result4);  // true`
	`----------`
	`// 6で割り切れるものが、1つもないため、以下の処理はfalseを返す。`
	`let array5 = [10,20,33,40,50];`
	`let result5 = array5.some(function(value2){ return value2 % 6 ==0; });`
	`console.log(result5); // false`

Array.every()	**everyメソッドは、配列内のすべての要素に合致している場合にtrueを返す。**

使用例	`// すべて2で割り切れるため、以下の処理はtrueを返す。`
	`let array6 = [10,20,30,40,50];`
	`let result6 = array6.every(function(value3){ return value3 % 2 ==0; });`
	`console.log(result6);  // true`
	`----------`
	`// 2で割り切れないものがあるため、以下の処理はfalseを返す。`
	`let array7 = [10,20,30,40,50,61];`
	`let result7 = array7.every(function(val){ return val % 2 ==0; });`
	`console.log(result7); // false`