



G's ACADEMY
UNIT_SAPPORO

JavaScript 2

~Memo Pad~



アジェンダ

目標とゴール

アジェンダ

❖ MemoPad

配列

繰り返し処理

配列と繰り返し処理の応用

localStorage（ブラウザの保管庫）

演習： MemoPad

❖ 課題

MemoPadの発展型（LocalStorageを使った何か）

目標とゴール

- ❖ 配列が何かわかる
- ❖ 繰り返し処理がわかる
- ❖ localStorage（ブラウザの保管庫）の使い方がわかる

配列

- Array -

JavaScriptの配列

【 配列 (array) 】

複数の値に順番をつけてまとめて扱う方法です。

配列の順番を識別する番号を「**インデックス**」と呼びます。

インデックスは「**0**」から始まります。

```
<script>
  const list1 = ['大吉','中吉','小吉','吉','凶'];

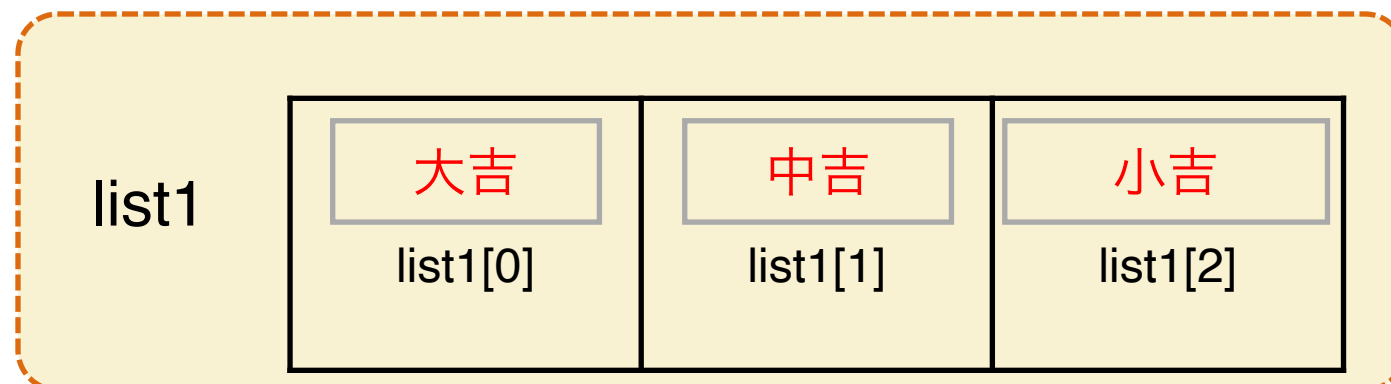
  const len = list1.length;    //lengthは配列の数を取得できる

</script>
```

【 配列のアクセスイメージ (array) 】

インデックス「0」から値が**格納**されます。

list[1]では「css」が取得可能



おみくじも配列で可能

配列使えば「3行」で作れる

配列を扱う関数

<https://qiita.com/takeharu/items/d75f96f81ff83680013f>

反復処理

JavaScriptの基礎

【反復処理（Iterate）】

プログラム中で、ある条件が満たされているかどうかによって、次に実行するコードを切り替える命令

- **for文**：条件が真の間だけ処理を続ける”繰り返し処理”
- **while文**：条件が真の間だけ処理を続ける”繰り返し処理”
- **for in文, foreach文** :配列/オブジェクトを繰り返す際に使用
(あとで覚えましょう！)

【インクリメント・デクリメント (increment)】

演算子の短縮したようなものです。

i++	i += 1	i = i + 1	インクリメント
i--	i -= 1	i = i - 1	デクリメント

JavaScriptの基礎

【 for 処理 】

for文を使用することで、反復処理をおこなうことができる

```
for( 初期値; 条件式; 再初期値 ){  
    条件式の結果がtrueの場合実行されるスクリプト  
}
```

例) for文

```
for( let i=0; i<10; i++ ){ //条件：iより10が大きい場合=trueで繰り返す  
    console.log( i );      // 変数iに代入されてる値を表示  
}
```

練習：for文

//1.変数の入れ物を作成

```
let str="";
```

//2.繰り返し処理で、文字列を作成

```
for ( let i=0; i<10; i++ ) {  
    str += "ループ：" + i + "回目<br>";  
}
```

//3.変数「str」に入ってる文字列を pタグid="view"に表示

```
$("#view").html( str );
```

配列と反復処理の応用

JavaScriptの基礎

【例）配列と反復処理】

配列と一緒に使用することが多い。

//1.配列を作成（必要な分だけ。。。今回は適当数に記述）

```
const week = ["日","月","火","水","木","金","土"];
```

//2.変数の入れ物を作成

```
let str = "";
```

//3.繰返し処理で、文字列と配列を組み合わせ作成

```
for ( let i=0; i<week.length; i++ ) {  
    str += week[i]+"<br>"; //配列:ar[i]  
}
```

//4.変数「str」に入ってる文字列を pタグid="view"に表示

```
$("#view").html(str);
```

POINT: length を使って配列の長さを取得

JavaScriptの基礎

Select Boxをループを使って作成

<!-- ここにセレクトボックスの値が生成されます -->

<select id="date"></select>

<script>

//1.変数strを作成:<select開始タグ>

let str = "";

//2. <option>タグを〇〇個作成

```
for( let i=1900; i<2022; i++ ) {  
    str += "<option>" + i + "</option>";  
}
```

//4.変数「str」に入ってる文字列を pタグid="view"に表示

\$("#date").html(str);

</script>

+ a

- 今風の技術力をつけたい方は
ES6 (ES2015) を学ぼう

localStorage

WebStorage

シンプルに使える localStorage

ブラウザ内に永続的にデータを保存するストレージ。

保存は「ドメイン名:ポート番号」の組み合わせ「オリジン」単位で保存されます。（例：<http://www.localhost:80>）

「オリジン」が同じであればブラウザを閉じた後も再度データにアクセス可能。

※ :80はブラウザが自動で付与してるの人間は入力していません。

保存量は「オリジン単位：10M」 保存期間は特になし。

自身の意志で削除しない限りデータは残ります。

シンプルに使える ～ localStorage ～

属性n	説明
DATA取得	localStorage.getItem(KEYネーム);
DATA登録or更新	localStorage.setItem(KEYネーム, 値);
DATAを全削除	localStorage.clear();
1レコード削除	localStorage.removeItem(KEYネーム);
DATA数:データ個数	localStorage.length
DATA取得:0~n	localStorage.key(インデックス)



HTML

```
<main>
  <input type="text" id="key">          . . . タイトル(key)
  <textarea id="memo"></textarea> . . . メモ(value)
  <ul>
    <li id="save">Save</li>          . . . データ保存
    <li id="clear">Clear</li>        . . . データ削除
  </ul>
</main>

<table id="list"> . . . データ一覧表示
  <!-- ここにappendで追加データが挿入される -->
</table>
```

JS

//1.Save クリックイベント

```
$("#save").on("click",function(){
  const key    = $("#key").val();
  const value = $("#memo").val();
  localStorage.setItem(key,value);
  //一覧表示に追加

  const html = '<tr><th>'+key+'</th><td>'+value+'</td></tr>';
  $("#list").append(html);
});
```

JS

//2.Clear(全削除) クリックイベント

```
$("#clear").on("click",function(){  
    localStorage.clear();  
    $("#list").empty();  
});
```

JS

```
for(let i=0; i<localStorage.length; i++){
```

```
  //key(何番)でkey名を取得
```

```
    const key = localStorage.key(i);
```

```
    const value = localStorage.getItem(key);
```

```
    //一覧表示
```

```
    const html = '<tr><th>'+key+'</th><td>'+value+'</td></tr>';
```

```
    $("#list").append(html);
```

```
  }
```

課題

【課題】 MemoPadアプリを作成

◇ 課題仕様

他なんでもあり！！localStorage使ってれば。

- 1 データ削除（授業では全て削除しか作ってない）
- 1 データ変更（登録内容を変更）
- 付箋アプリ。EverNote...とか
- Todoアプリとか
- じゃんけんの点数を記憶させる（履歴を残す）
-簡単！？好きなものを作ってしまいましょう！！