Webデザイン実習3B

2019/5/27

Kazuma Sekiguchi
class@cieds.jp

JSの基本

- 何故か変数に関数を格納可能

```
var func = function(){
return 2+3;
}
document.write(func());//5
```

- 呼び出すときは、変数に()を付ける
- ＝の右側は名前の無い関数を定義しているため、無名関数とも呼ばれる

JSの基本

- 配列
  - 1つの変数内にインデックスという番号を付けていって複数の値を格納したもの

```
var arr = [];//配列の初期化
var arr2 = new Array();//配列の初期化方法2
arr[0] = “リンゴ”;
arr2[0] = “うさぎ”;
document.write(arr[0]);
document.write(arr2[0]);
```

- ループで処理することで複数の値を一気に処理することが可能

配列の処理

```
var goukei = 0;
var tokuten = [98,34,53,74,75,42,72,16,73,93,27,52,92,69,27,34,25,61,98,42,21,79,29];
for(var i = 0; i < tokuten.length; i++){
goukei += tokuten[i];//goukeiにtokuten[i]の値を足す
}
document.write(goukei);
```

スコープ

- スコープ＝変数が有効な範囲
- プログラム全体からアクセス可能＝グローバル変数（グローバルスコープ）
- 関数内のみからアクセス可能＝ローカル変数（ローカルスコープ）
- 関数内で宣言した変数はローカルスコープになる
  - 宣言しない変数はグローバルスコープ
- 関数外で宣言または宣言しない変数はグローバルスコープになる

スコープ

- できるだけグローバルスコープの使用を控える
  - グローバルスコープはどこからでもアクセス可能
  - ほかの関数などの影響で変化することがある
  - 名前が被ってしまう可能性も高い

違い

```
var a = 10;
function calc(){
var b = 5;
document.write(a + b);//15
}
calc();
document.write(b);
```

```
var a = 10;
var b = 15;
function calc(){
a = 5;
document.write(a + b);//20
}
calc();
document.write(a);//5;
```

こんな書き方も

```
function calc(){
        var a = 10;
        return function(){
                a += 10;
                document.write(a);
        }
}
var calc1 = calc();
calc1();
a = 100;
calc1();
var calc2 = calc();
calc2();
```

即時関数

- 普通の関数は

```
function func(){
docment.write(“Run!!”);
}
func();
```

定義して

実行

- つまり、呼び出す必要がある
- 呼び出さなくても実行できる関数＝即時関数

即時関数

- このように書くと呼び出さなくても実行される
  - 呼び出さなくても実行されるため、関数名すら要らない

```
(function(){
document.write(“Run!”);
})();
```

- 利点：即時関数であれば、全ての変数がローカル変数になる
  - 利点を感じないのであれば、使う必要は無い
  - ちなみに、jQueryの実行時に使っているのは即時関数である

グローバルオブジェクト

- JSでは全てがオブジェクトに属する
  - 関数もメソッドとして存在
  - Math.random()などは典型例
  - .lengthなどは？
    - もちろんオブジェクトとして作成されるからこそ利用可能なメソッド
- 変数を作成した時点で既にプロパティになっている
  - どのオブジェクトのプロパティか？
  - windowオブジェクト（これがグローバルオブジェクト）

ゆえに

```
var a = 1;
if(a === window.a){
document.write(“一緒”);
}
```

- 変数aはグローバル変数
- グローバル変数はグローバルオブジェクトのプロパティになる
- ゆえにwindow.aでも同じ
- グローバルオブジェクトは記述を省略できる
  - いくつかの関数はwindowオブジェクトのメソッドであるが、記述を省略できるために書かない
  - 例:alert();