Webプログラミング実習1

2017/4/19

Kazuma Sekiguchi
class@cieds.jp

JavaScriptを書いてみる

- <script></script>の間に記述する
  - 別ファイルに記述してもOK
  - 別ファイルの場合は、拡張子をjsとして<script src=“〇〇〇.js”></script>として読み込ませる
- 実行されるタイミングは基本的にブラウザーでHTMLが表示された後

プログラミング言語

- JSでは変数、メソッド、イベントなどが用意されている
  - 変数＝一時的に保存しておくための場所。名前を付けることでいろいろな内容を保持可能
  - メソッド＝JavaScriptで利用することができる機能のこと。1つ1つでは大したことはできない
    - 組み合わせることで処理をしていくことが可能
  - イベント＝ユーザーのインタラクション
    - 後日説明

JSの基本的なもの

- 変数
  - 何かの値やオブジェクトを格納しておくもの
  - JSの場合、いろいろなものを格納可能
    - オブジェクトも格納可能
  - 先頭にvarを付ける（変数宣言）
    - スペースを開けて変数名を指定（通常1文字目は英小文字）
  - 数値と文字が違うので注意（” ” で括れば文字）

```
var a = “JavaScript”;//aに”JavaScript”という文字列を格納
var b = 123;//bに123という数値を格納
```

変数

- 定義関数を格納できるところはJS独特
- 通常は変数宣言を行って利用する（宣言しなくてもOKだが、最近は警告が出ることが多い）
  - 先頭にvarを付ける（変数宣言）

```
var a;//aという変数を利用する（宣言のみ）
var b = 1;//bという変数を利用すると宣言し、1という数値を格納
var c = “js”//cという変数を宣言し、jsという文字列を格納

a = 4;//宣言済みの変数aに4を格納
```

変数宣言

- プログラムの一番最初にまとめて記述することが多い
  - ほかの人が見たときにどういう変数を利用しているか把握しやすい
  - 変数に設定値（リンク先のURLや何らかのパラメータ）などを格納した場合、変更しやすい

JSの基本的なもの

- 何らかの動作をまとめたもの＝関数
- 引数は複数の値を指定可能。返り値は1つだけ

```
function 関数名（引数）{
何らかの処理処理
return 返り値
}
```

```
function keisan（a,b）{
        var c = a + b;
        return c;
}
```

```
var kotae = keisan(2,3);
```

kotaeには5が入ってくる

ユーザ定義関数

- 何らかの処理をまとめたもの
- 同じような処理を繰り返す際に利用する
  - イチイチ同じ処理を何度も書くのは面倒。間違えも増える
- ()内に引数を指定可能
  - 引数はカンマで区切ることで複数指定可能
  - 引数に値を与えて関数を呼び出せば、その引数を利用して処理してくれる
- ユーザ定義関数から更にほかのユーザ定義関数を呼び出すことも可能
- 記述場所は自由
  - 通常まとめて書いておく
  - 関数だけのJSファイルを作ることも多い

ユーザ定義関数

```
function 関数名(引数1,引数2・・・){
    処理の内容
    return 処理の結果
}
```

基本的な記述方法

```
function Calc(a,b){
    var c = a + b;
    return c;
}
```

作成（Calcという関数を作成）

```
var g = Calc(4,6);//gには10が格納される
function Calc(a,b){
    var c = a + b;
    return c;
}
```

Calcを利用

```
function caution(){
    alert(“注意！ ！”);
}
```

returnの無いものも可能

変数宣言の場所による違い

- ユーザ定義関数内で宣言した変数
  - その関数内でのみ利用可能＝ローカル変数
- 関数外で宣言した変数
  - ユーザ定義関数内でも利用可能＝グローバル変数
- グローバル変数の場合、意図しないところで上書きすることがあるため、できるだけ避ける
  - 無理なことも多い

```
var a = 1;
function ShowAlert(){
    alert(a);//1が表示される
}
```

```
function SetA(){
    var a = 1;
}
function ShowAlert(){
    alert(a);//何も表示されない
}
```

変数宣言の場所による違い

```
var a = 1;
function ShowAlert(){
    alert(a);//1が表示される
}
```

グローバル変数としてaを利用

```
function SetA(){
    var a = 1;
}
function ShowAlert(){
    alert(a);//何も表示されない
}
```

aはローカル変数のため、ShowAlertでは利用できない

```
var a;//グローバル変数としてaを宣言
SetA();//関数呼び出し
ShowAlert();//関数呼び出し
function SetA(){
    a = 4;
}
function ShowAlert(){
    alert(a);//4が表示される
}
```

aはグローバル変数。ほかの関数で変更されるとそのまま変数の中身も変化する

条件

- 条件に応じて、処理を分けるケースは多い
  - クリックされてもこれ以上右に動かせない→左に動かす
  - いわゆる判断をさせる場合に利用
  - 条件式を記述して判断させる
  - ifのみ必須
    - 他は任意
  - else ifは複数回利用可能

```
if(条件式1){
        条件式1に当てはまるならここが動作
}
else if(条件式2){
        条件式2に当てはまるならここが動作
}
else{
        条件式に当てはまらない場合に動作
}
```

条件式

- 0以外の数、true ，何らかの文字列は全て条件式で当てはまる
  - 0,falseだと条件式に当てはまらない

- 比較して判断することも
  - if(a > 3)→aが3よりも大きければtrueになる
  - if(a == 3)→aが3のときだけtrueになる
  - if((a >= 3) && (a < 100))→aが3以上で、100未満のときだけtrueになる
  - if((a >= 100) || (a < 10))→aが100以上または10未満のときだけtrueになる

条件式の成立

- 成立する場合
  - if(true)
  - if(1以上の数値)
  - if(‘何らかの文字’)
  - 条件式が成立しない場合
- 不成立の場合
  - if(false)
  - if(null)
  - if(undefined)
  - if(0)
  - 条件式が成立しない場合

ループ

- 同じ処理を繰り返す際に利用
  - いくつか方法がある
  - プログラムは楽するために利用する＝同じ処理はできるだけループを使って処理する
  - 条件が成立している限りループする
    - 初期値に0を入れて1回回るごとに1を足していくと100回回った段階で条件を満たさなくなる

```
for(var i =0; i< 100; i++){
    var a = i;
    document.write(a);//aを表示。0〜99まで表示される
}
```

```
for(初期値; 条件; カウントアップ){
    繰り返す内容
}
```