JavaScript基礎-第1回資料

今回は軽めにJavaScriptの簡単なルールとif/else/else if構文について
演算子、関数に関しては簡単にしか書いていないが、それについてはあとの講義で解説する
解説を聞く前に一度読んでおき、できれば実行してみるといいかも

◇データ型/代入

例1

```
1  var box;//boxという名前の変数
2  box=[0,1,2,3];//配列を格納
3  box=0;//数値を格納
4  box="1+1="+2;//文字列+数値の格納
5  print(str);//内容の確認
```

変数とはプログラム上でのデータを保持しておく箱のようなもので、 たいていの言語では変数の宣言によって、格納できるデータの種類が限定されてしまう
しかし、JavaScriptには変数にデータの種類の限定は無く、数値・文字・配列など種類を問わずに格納できる
よって2-4行目はエラーにならず、同じ型同士の結合だけでなく、4行目のような異なる型どうしの結合も可能である
ここでの"左辺=右辺"は=の左辺に右辺を代入、"文字列+データ"もしくは"データ+文字列"は文字列として+の左右を結合するという意味である

◆プログラムの実行順序

例2

```
1  var str;//変数strの宣言
2  str="Hello world!";//strに文字列を格納
3  print(str);//内容の表示
```

JavaScriptも他の言語と同じように上から下へとコードは解析される<例2>

例3

```
1  var str;
2  str=hello();
3  print(str);
4  function hello(){
5          return "Hello world!";
6  }
```

ここで出てきた"function 関数名(){～}"は括弧内の処理を一まとめにする書き方で、 "関数名()"と書くことで括弧内の処理を実行させることができる
<例2>の考え方だと2行目の時点では関数helloは存在しないが、 ただし、一度すべてのコードはコンパイル（機械語への翻訳）されてから実行されるイメージを持ってほしい
<例3>で示したコードも後に定義された関数がエラーになることなく実行できるのである
つまり、すべてのコードをコンパイル後にトップレベル（インデント[字下げ]0の階層）が実行される

◇グローバル・ローカル変数

例4

```
1  var A='a1';
2  B='b1';
3  function test1(){
4          A='a2';
5          return A;
6  }
7  function test2(){
8          var B='b2';
9          return B;
10 }
11 print(test1());//'a2'
12 print(A);//'a2'
13 print(test2());//'b2'
14 print(B);//'b1'
```

JavaScriptでは変数にvarをつけて"var 変数名=値;"とするか、varなしで"変数名=値;"と定義する（"=値"は値の代入であり、省略可）
トップレベルでvarをつけて定義するか、どこかでvarをつけずに変数を定義した場合、その変数はグローバル変数となる
また、ローカル変数はすべてvarをつけて定義され、その際の参照可能範囲は定義したレベル以下である
<例4>では関数test1のAはグローバル、関数test2のBはローカル変数にアクセスしているため、各関数実行後のAとBの値が異なる
このあたりもvarつきの変数はCと同様の考え方である
なお、変数名は_か$か英字（大文字・小文字）から始まり2文字目以降は数字も使用可能 （2文字目以降は他の文字も使えるがエラーの元となるので普通使わない）

◆ステートメントの解釈

例5

```
1  var sw=3;
2  if(sw==0){
3          print("a");
4  }
5  else if(sw==1){
6          print("b");
7  }
8  else{
9          print("c");
10 }
```

例6

```
1  var sw=3;
2  if(sw==0)print("a");
3  else if(sw==1)print("b");
4  else print("c");
```

1命令（≒ステートメント）は基本的に行頭からセミコロン（;）までである
何かしらの制御構文の後にコードをひとまとめにするときはたいてい{と}でくくる
これを1単位として1ブロックとする

JavaScriptにおいては他の言語と同じように、1命令であればブロック（{}）を省略できる
つまり、<例5>と<例6>は意味的には同等のコードである
ここでのifやelse if、elseについては次を参照

◇if/else/else if構文

例7

```
1  var num=0;
2  if(num==0){
3  //num==0が真（true）の場合実行
4  }
5  else{
6  //num==0が偽（falae）の場合実行
7  }
```

if構文はifの後に続くカッコ内の条件が真になった（正しい）ときに後ろに続くブロックが実行され、 偽の（正しくない）場合elseのブロックが実行される<例7>
if構文はそれ単体で完結でき、else構文は省略可能である(ifのみで使える)
ここでの"左辺==右辺"は左辺が右辺と等しいか否か、という意味である

例8

```
1  var sw=3;
2  if(sw==0){
3          //num==0が真の場合実行
4  }
5  else if(sw==1){
6          //num==0が偽の場合、かつnum==1が真の場合実行
7  }
8  else if(sw==2){
9          //num==0が偽の場合、かつnum==2が真の場合実行
10 }
11 else{
12         //ここまでのifとelse ifがすべて偽の場合実行
13 }
```

例9

```
1  var sw=3;
2  if(sw==0){
3          //num==0が真の場合実行
4  }
5  else{
6          if(sw==1){
7                  //num==0が偽の場合、かつnum==1が真の場合実行
8          }
9          else{
10                 if(sw==2){
11                         //num==0,num==1が偽の場合、かつnum==2が真の場合実行
12                 }
13                 else{
```

```
14              //ここまでのifとelse ifがすべて偽の場合実行
15                  }
16          }
17  }
```

if/else if/else構文はifとelseの間にelse ifをはさむことでできている<例8>
else ifは直前の構文が偽であるときに実行され、else ifの後の条件が真なら次のブロックが実行される
なお、else ifはいくらでもifとelseの間にはさむことができる
もちろん、if/elseと同様に、最後のelse文はなくてもよい（else ifで終わってもよい）
簡単に言うと、else ifは<例8>を<例7>のように書くことができると言うことである
基本的に、<例9>のような書き方は読みにくいため、else ifを用いて<例8>のように書くことがほとんどである