長居AS3勉強会　第一回

今日の目標

- AS3に慣れよう
- クラスを書こう
- イベントリスナーを理解しよう
- クラスを使おう
- エラーに慣れよう

今日の目標

AS3を覚える（使う）理由

とにかく最新技術が使えてカッコイイ!
やっとかないと何かヤバいのでは？という
不安感

理由は様々ですが

今日の目標

AS3にとって大事な考え方は
下準備と段取り

今日の目標

スクリプトが複雑になる。
スクリプトは簡単でも後日に
修正や手直しが発生する。
納期直前に変更がある。
↓

今日の目標

どれだけ複雑でも流れを理解しやすい。
修正や変更が容易になる。
最終的な手直しにも強くなる
↓

今日の目標

仕事が楽になるという事は
細部まで突き詰められるという事。

今日の目標

下準備や段取りは恐ろしく地味だし
わからないことはつまらないし面倒。
でも「わからないところで止まる」より
「わからないところをやる」方法で
みんなで勉強すれば楽しいんじゃない？

ということでやってみましょう。

AS3に慣れよう

そもそも自分にとってAS3の
何が難しいのかを知る
↓
わからないところがわからないではダメ

AS3に慣れよう

とは言っても「AS3は簡単！」とよく聞くけど、
とっかかり始めでは本当にわけが分からない。

→自分の場合

AS3に慣れよう

ステージにムービークリップを作成。
インスタンス名に「my_mc」と入力。
↓
my_mcをクリックしたら回転し始めるスクリプトをAS3で書きたい。

→やってみよう

AS3に慣れよう

```
my_mc.addEventListener(MouseEvent.CLICK, onMouseClickHandler);

function onMouseClickHandler(event:MouseEvent):void{
	my_mc.addEventListener(Event.ENTER_FRAME, onEnterFrameHandler);
}

function onEnterFrameHandler(event:Event):void{
	my_mc.rotation ++;
}
```

AS3に慣れよう

addEventListenerでイベント登録

↓

かろうじてわかる

↓

Functionの引数

event:MouseEventって？

event:Eventってのもあるし

なんかややこしい！

↓

AS3やる気なくすわ～

AS3に慣れよう

実はAS3の世の中を支配しているのは全てクラス。
クラスのお約束に慣れることがAS3に慣れる近道。
↓
クラスに慣れよう

AS3に慣れよう

その前に

AS3に慣れよう

変数の型について

→やってみよう

AS3に慣れよう

```
a = “1”;
b = 3;
c = a + b;
trace(c);
```

AS3に慣れよう

```
a = “マサムーネ”;
b = 3;
c = a + b;
trace(c);
```

AS3に慣れよう

プログラムが複雑になればなるほど変数の中身が
曖昧だと大変！！

AS3に慣れよう

```
var a:String = “1”;
var b:String = 3;
var c:String = a + b;
trace(c);
```

AS3に慣れよう

変数の「型」について
ここで覚える要素は
３つ

AS3に慣れよう

1. 変数を最初に使う場合
2. 変数の頭に　var を付ける
3. 変数のお尻に　: を付けて型を指定する

AS3に慣れよう

1. 変数を最初に使う場合

っちゅうのが現時点ではイマイチ曖昧ですが、それはクラスを書く時に明確になります。

AS3に慣れよう

とりあえず変数の「型」については
さっきの３つの条件だけ覚えておけば
あとはクラス書けばなんとかなります。

クラスを書こう

クラスを書こう

クラスを書くときの基本ルール
大きく 3 つ

クラスを書こう

デスノートのルールよりは
簡単だし少ない。

クラスを書こう

とりあえずFlaファイルと同じディレクトリに
.as
ファイルを作る。
ファイル名の最初は大文字のアルファベットにしておく。
このファイル名が今から作るクラスの
クラス名
になる。

クラスを書こう

.as

ファイルの中身はとりあえず

```
package{

}
```

で囲む

クラスを書こう

packageの中はとりあえず

```
public class  < クラス名 > {

}
```

で囲む

クラスを書こう

最終的に次の形になる

```
package{
    public class ＜クラス名＞{

    }
}
```

クラスを書こう

基本ルールはたったこれだけです。

クラスを書こう

次にそれを踏まえて実際に
「動く」クラスを書くためのルール
を覚えます。

クラスを書こう

手っ取り早く「動く」クラスを作るには
「ドキュメントクラス」
を書くといいです。

クラスを書こう

Flaファイルの
「シーンプロパティ」の
「ドキュメントクラス」
にクラス名を入力します。

クラスを書こう

基本ルールその 1
Flaファイルと同じフォルダにクラスファイルを作る。
クラスファイル名とクラス名は同じ。
ファイル名 (クラス名) の最初は大文字で。

を思い出して…。

→やってみよう

クラスを書こう

```
package{
    public class Main{
        //この中にスクリプトを書いていきます！
        trace(“ほげほげ〜”);
    }
}
```

クラスを書こう

ここで「動く」クラスのルール
その 1

ドキュメントクラスはMovieClipクラスかSpriteクラスを継承しよう。

クラスを書こう

ちなみにMovieClipクラスかSpriteクラスのどちらを継承するかに関しては、どっちでもいいですが、慣れないうちはMovieClipクラスを継承しておいた方がいいです。
(理由は口頭で説明)

クラスを書こう

継承の仕方は、
class Main の後に
extends MovieClip
を付け足すだけ。

クラスを書こう

```
package{
    public class Main extends MovieClip{
        //この中にスクリプトを書いていきます！
        trace(“ほげほげ〜”);
    }
}
```

クラスを書こう

ここで「動く」クラスのルール
その２

クラスに必要なクラスはインポートしなければならない。

クラスを書こう

ドキュメントクラス「Main」はMovieClipクラス
を継承
↓
MovieClipクラスが必要
↓
packageの中に必要なクラスを
インポートします。

クラスを書こう

クラスって独立国家みたいなもの。
自分のクラスに無いものは
あらかじめインポート
(輸入) しておく必要があるんですね。

クラスを書こう

どこにどうやって書く
(インポートする) の？

クラスを書こう

```
<body id=”package”>
    <div id=”class” name=”Main”>

    </div>
</body>
```

クラスの基本形はこれと良く似ています。

クラスを書こう

「packageの中に書きましょう」

具体的には

```
import flash.display.MovieClip;
```

と書いていきます。

クラスを書こう

```
package{
    import flash.display.MovieClip;
    public class Main extends MovieClip{
        //この中にスクリプトを書いていきます！
        trace(“ほげほげ〜”);
    }
}
```

クラスを書こう

```
<body id=”package”>
    <h1 class=”import”>MovieClip</h1>
    <div id=”class” name=”Main”>

    </div>
</body>
```

正確にはちょっと違うけど、htmlだとこんなイメージ

クラスを書こう

ここで「動く」クラスのルール
その 3

classと同じ名前の関数（function）が一番最初に実行される。
この関数を「コンストラクタ」と呼ぶ。

クラスを書こう

どこにどうやって書くの？

クラスを書こう

書く場所はclassに囲まれた中。

クラスを書こう

書き方は

```
function Main():void{

}
```

クラスを書こう

```
package{
    import flash.display.MovieClip;
    public class Main extends MovieClip{
        function Main():void{
            trace(“ほげほげ〜”);
        }
    }
}
```

クラスを書こう

もうね、完璧。

クラスを書こう

とにかく基本さえ覚えれば
これから複雑になってきても
対応できるので
がんばって！

変数を使おう

変数を使おう

クラスの中で変数を使うために、
最初の方に「変数の型」でやったルール
1. 変数を最初に使う場合
の解説に入ります。

変数を使おう

クラスで使う変数のルール

クラスの中に直接作った変数は、
そのクラス内でどこでも自由に使えます。
関数の中で作った変数は
その関数の中だけで使えます。

変数を使おう

```
package{
    import flash.display.MovieClip;
    public class Main extends MovieClip{
        var a:String = “マサムーネ”;
        function Main():void{
            var b:String = “ほげほ～げ”;
        }
    }
}
```

変数を使おう

```
package{
    import flash.display.MovieClip;
    public class Main extends MovieClip{
        var masamune:String = “マサムーネ”;
        var kotoba:String = “かっこいい！(^-^)v”;
        function Main():void{
            trace(masamune + kotoba);
            mouikkai();
        }
        function mouikkai():void{
            var kotoba:String = “かっこわるい(><;)”;
            trace(masamune + kotoba);
        }
    }
}
```

変数を使おう

慣れてないと実に不思議というかややこしい動きをしているように見えますが、
慣れてくると変数を調べるときに
最初は関数内でvar で宣言されていないか調べて、なければクラス内でvarされてないか調べればいいので、
とても合理的に作業が出来るようになります。

変数を使おう

ちなみに関数の戻り値に使う「型」はこんな感じ

変数を使おう

```
function Main():void{
    traceMasamune();
    var modori:String = modosuMasamune();
}
function traceMasamune():void{
    trace(“マサムーネムネムーネ”);
}
function modosuMasamune():String{
    var masamune:String = “正宗の双子の弟です。”;
    return masamune;
}
```

変数を使おう

引数を使う場合はこう

変数を使おう

```
function Main():void{
    traceMasamune(“は３３歳”);
    traceMasamune(“の好きな忍法帖はくノ一忍法帖”)
}
function traceMasamune(kotoba:String):void{
    trace(“マサムーネ” + kotoba);
}
```

変数を使おう

クラスで使う変数のルール

クラスの中に直接作った変数は、
そのクラス内でどこでも自由に使えます。
関数の中で作った変数は
その関数の中だけで使えます。

を覚えておけば、
スクリプトが複雑になっても
やっぱり基本的なルールの組み合わせだけで
何とかなるようになります。

イベントリスナーを理解しよう

イベントリスナー

なんでもかんでも
addEventListener
でイベントを登録！！

イベントリスナー

my_mcをクリックしたら
メッセージを表示する
スクリプトを書いてみよう。

イベントリスナー

なにかムービークリップを作って、
インスタンス名

「my_mc」
と入力しておきます。

イベントリスナー

```
package{
    import flash.display.MovieClip;
    public class Main extends MovieClip{
        function Main():void{
            my_mc.addEventListener("click", onClickHandler);
        }
        function onClickHandler(event:MouseEvent):void{
            trace("あなた、クリックしましたね！？");
        }
    }
}
```

イベントリスナー

```
package{
    import flash.display.MovieClip;
    import flash.events.MouseEvent;
    public class Main extends MovieClip{
        function Main():void{
            my_mc.addEventListener("click", onClickHandler);
        }
        function onClickHandler(event:MouseEvent):void{
            trace("あなた、クリックしましたね！？");
        }
    }
}
```

イベントリスナー

flaファイルのタイムラインに直接スクリプトを書いたときは、インポートしなくてもよかったものが、クラスの場合はいろいろインポートしないといけないわけですね。

イベントリスナー

addEventListener(< イベント > , < 処理 >);

↓

(< 何が起こったら？ > , < 何をする？ >);

イベントリスナー

＜イベント＞（ ＜何が起こったら？＞ ）
の部分は、
例えば”click”とかでもいいんですが、
例えば
“clik”とか書き間違えたとき
エラーも出ないし動かないし、
困ります。

イベントリスナー

さてここで一旦イベントリスナー
から離れて、

イベントリスナー

flaファイルやMain.asファイルと同じフォルダに
Test.as
というクラスファイルを作って下さい。
(今まで覚えたルールをよ〜く思い出して)

イベントリスナー

```
package {
    public class Test {
        function Test():void {

        }
    }
}
```

イベントリスナー

Testクラスはドキュメントクラスではないので
MovieClipの継承は行わなくていいんです。
継承する必要がないのでインポートも今のところ
必要ないです。

引っかけじゃないんですが慣れてないと
意地の悪いパズルみたいですね。

イベントリスナー

ふつう変数は（さっきやったように）
そのクラス内だけで使いますが、
ここでTestクラスで作った変数を
Mainクラスで使ってみましょう。

イベントリスナー

```
package {
    public class Test {
        var CLICK:String = “click”;
        function Test():void {

        }
    }
}
```

イベントリスナー

変数もいろいろな使い方があって、
「スクリプトが実行されてるときに変数の中身が
変わる変数」は普通なんですが、

「変数の中身が絶対に変わらない変数」
という使い方もあります。

イベントリスナー

前者（中身が変わる）は普通にvar で宣言するんですが、後者（中身が変わらない）は
「定数」と言って
var のかわりに
const
で宣言します。

イベントリスナー

```
package {
    public class Test {
        const CLICK:String = “click”;
        function Test():void {

        }
    }
}
```

イベントリスナー

くどいようですが、ふつう変数（や定数）は
そのクラス内だけで使いますが、
今回みたいにMainという外部からTestクラスの
中の変数（や定数）を使いたい場合、
var（やconst）の前に
public static
を付けます。

イベントリスナー

```
package {
    public class Test {
        public static const CLICK:String = “click”;
        function Test():void {

        }
    }
}
```

イベントリスナー

さて、
ここまで意味の分からない事をさせて
すみません。

Mainクラスのイベントリスナーに戻りましょう。

イベントリスナー

```
package{
    import flash.display.MovieClip;
    import flash.events.MouseEvent;
    public class Main extends MovieClip{
        function Main():void{
            my_mc.addEventListener(Test.CLICK,
onClickHandler);
        }
        function onClickHandler(event:MouseEvent):void{
            trace("あなた、クリックしましたね！？");
        }
    }
}
```

イベントリスナー

また新しいルール。

同じフォルダにあるクラス同士は
import
しなくても使える！
(ただ何のクラスを使ってるのか
あとあとわからなくなるので
importしてもいい。
つまりしてもしなくてもいい)

イベントリスナー

こうしておくと、”click”をスペルミスしてエラーも出ないし動かない、
なんてことがなくなります。

イベントリスナー

でも毎回Testクラスみたいなの作るの面倒。
Testクラスの中の”click”そのものがスペルミスしてたら意味ないし。
↓

イベントリスナー

実は今作った、Testクラスの
public const CLICK:String = “click”;
っての、
わざわざ作らなくても
もうあるんです。

イベントリスナー

それが
MouseEvent
クラス

イベントリスナー

```
package{
    import flash.display.MovieClip;
    import flash.events.MouseEvent;
    public class Main extends MovieClip{
        function Main():void{
            my_mc.addEventListener(MouseEvent.CLICK,
onClickHandler);
        }
        function onClickHandler(event:MouseEvent):void{
            trace("あなた、クリックしましたね！？");
        }
    }
}
```

イベントリスナー

一番最初に見た

```
my_mc.addEventListener(MouseEvent.CLICK, onMouseClickHandler);

function onMouseClickHandler(event:MouseEvent):void{
	my_mc.addEventListener(Event.ENTER_FRAME, onEnterFrameHandler);
}

function onEnterFrameHandler(event:Event):void{
	my_mc.rotation ++;
}
```

イベントリスナー

こんなふうに、
いろんな簡単なルールの組み合わせで
合理的に作業が出来るようになっているのが
AS3
ということになります。

クラスを使おう

クラスを使おう

独学できるようになるまであとちょっと。

クラスの使い方

クラスを使おう

また、Main.asと同じフォルダに
TraceTest
というクラスを作ってみましょう。
(Testクラスをコピペして
クラス名とコンストラクタを
かえてもOK)

クラスを使おう

```
package{
    public class TraceTest{
        function TraceTest():void{
            trace("ほげほげ〜");
        }
    }
}
```

クラスを使おう

Mainクラスではこういうふうに使います。

クラスを使おう

```
package{
    import flash.display.MovieClip;
    public class Main extends MovieClip{
        function Main():void{
            var t:TraceTest = new TraceTest();
        }
    }
}
```

クラスを使おう

クラスを使うときの基本ルールは 2 つ

new でクラスをインスタンス化する。
そのインスタンスの型は基本的に
クラス名と同じ型になる。

クラスを使おう

ちなみにクラスのコンストラクタに
引数を渡すこともできます。

クラスを使おう

```
package{
    public class TraceTest{
        function TraceTest(moji:String):void{
            trace(moji);
        }
    }
}
```

クラスを使おう

```
package{
    import flash.display.MovieClip;
    public class Main extends MovieClip{
        function Main():void{
            var t1:TraceTest = new TraceTest(“ほねほね”);
            var t2:TraceTest = new TraceTest(“ロック”);
        }
    }
}
```

クラスを使おう

クラスって
ライブラリで言うシンボルみたいなもの。

使うときはnew でインスタンス化する、
って覚えときましょう。

まとめ

まとめ

今回やったことは
基本の基本のところでもあるし、
逆にいきなり難しいところもしています。
また、とりあえず動かせるようになるため
説明が正確でないところもあります。

まとめ

ただ、どんな複雑なスクリプトも、
ひとつひとつは簡単なルールの組み合わせなので
先ずは大枠がつかめるよう、
とにかくルールに慣れる事から
初めてみましょう。

まとめ

難しい、面倒くさい段取りは
あとあと楽になるための下準備です。

是非AS3に慣れて、
仕事のクオリティアップと
飲みの時間を増やしましょう。

まとめ

おつかれさまでした。