

## Lesbrief 2: AS3, Inheritance, MovieClips en meer

### De verschillen tussen AS2 en AS3

In AS3 zijn er een aantal dingen veranderd. Zo is er nu ondersteuning voor packages, is het koppellen van een zogenaamde document klasse aan een flash applicatie nu mogelijk, zijn er een aantal datatypes bij gekomen en moeten we imports aan gaan geven. De belangrijkste veranderingen behandelen we in het eerste gedeelte van deze lesbrief.

### Packages

In ActionScript 3 (AS3) moeten de klasse definities nu binnen een package staan. Met behulp van een package kunnen we onze klassen indelen in mappen. Dit is erg handig als je applicatie wat uitgebreider gaat worden, bijvoorbeeld met de doorlopende opdracht. Als je alle klassen bij elkaar wilt zetten dan laat je de package leeg, maar je definieert hem wel:

```
package {  
    class Spookje {  
        ...  
    }  
}
```

Als je je klassen wel in verschillende packages wilt onderverdelen, dan kun je een mappenstructuur maken. Iedere package krijgt zijn eigen map.

Bijvoorbeeld:

Een package met de naam:

```
package nl.rocvantwente.bm.web.lesbrief2 {  
    class Voertuig {  
    }  
}
```

Dit betekent dat je .as bestanden met de klassen voor die package in de volgende een mappenstructuur moeten staan:

```
nl\rocvantwente\bm\web\lesbrief2
```

### Imports

Om een klasse uit een andere package te gebruiken, bijvoorbeeld een standaard klasse van Flash of een klasse die je zelf hebt gedefinieerd in een andere package, dan zul je Flash moeten vertellen waar die klasse staat. Dit kun je doen met behulp van het import keyword.

Bijvoorbeeld een klasse Spookjesbos die erft van de ingebouwde Flash klasse MovieClip:

```
package {  
    import flash.display.MovieClip;
```

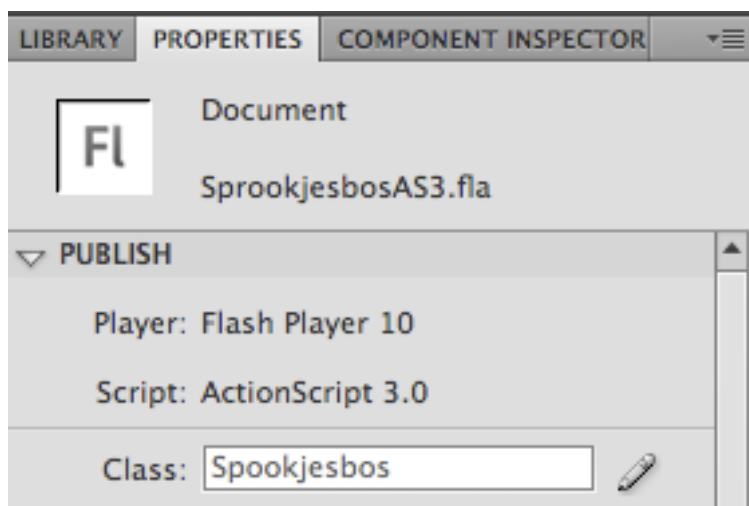
```
public class Spookjesbos extends MovieClip {  
}  
}
```

Import statements moeten binnen de package worden gedeclareerd maar buiten de klasse definitie;

### Document klasse

In AS2 moesten we steeds vanuit het actions scherm van Flash onze ActionScript objecten aanmaken om. Dat hoeft met AS3 niet meer. We kunnen nu bij onze Flash applicatie (.fla bestand) aangeven welke klasse er gekoppeld moet worden aan de applicatie, de zogenaamde "Document" klasse.

Dit moet een klasse zijn die erft van MovieClip of een andere Display klasse. De document klasse kunnen we opgeven bij de eigenschappen van de .fla file, bij de publish settings vullen we dan de klasse in die we hiervoor willen gebruiken, in het screenshot hieronder is dat de klasse "Spookjesbos"



Deze klasse wordt automatisch aangemaakt bij het starten van de applicatie. Hierbij wordt de constructor van deze klasse aangeroepen. Vanuit de constructor kunnen we dan onze applicatie op gaan bouwen.

### Inheritance (overerving)

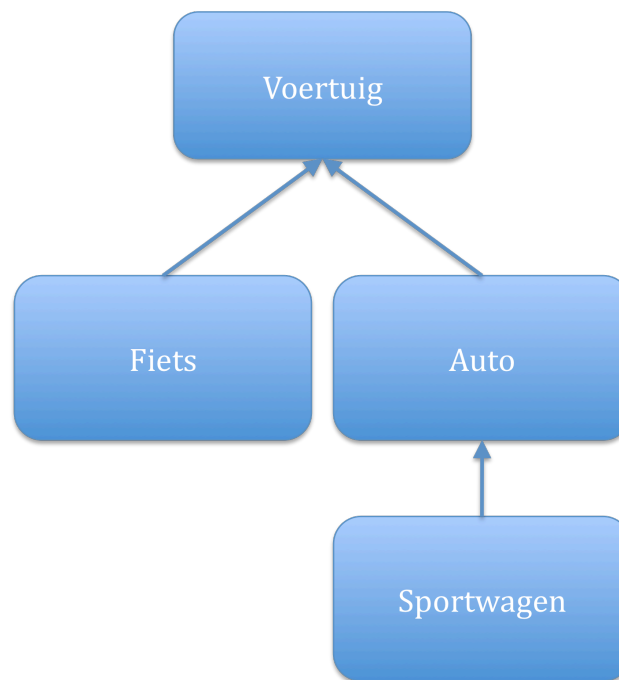
Binnen Flash kunnen we net als binnen andere OOP talen gebruik maken van overerving (inheritance). Veel klassen binnen Flash maken hier zelf ook gebruik van. Het is daarom belangrijk om te begrijpen wat inheritance is en hoe je dat kunt gebruiken.

### Overerving

Binnen OOP bestaat er de mogelijkheid om een klasse eigenschappen en verantwoordelijkheden te laten erven van een andere klasse. Hiermee kunnen we een specifiekere klasse maken van een super klasse.

Bijvoorbeeld de definitie van twee klassen, Voertuig en Auto. Een Auto is een Voertuig met een aantal specifieke kenmerken en een aantal kenmerken die een Voertuig ook heeft. Zo heeft een Auto bijvoorbeeld een stuur en wielen en een

vorm van aandrijving. Een klasse Fiets zou ook de eigenschappen van de klasse Voertuig erven, maar zou geen sub klasse zijn van Auto.



Auto erft dus net als Fiets de eigenschappen van Voertuig. Sportwagen erft de eigenschappen van Auto.

We kunnen dus zeggen dat een Auto een specifiekere klasse is dan Voertuig.

We kunnen dus zeggen dat een Auto een soort Voertuig is, en dat Sportwagen een soort Auto is en daardoor ook een soort Voertuig.

### Overerving in ActionScript

In ActionScript kun je inheritance gebruiken met het extends keyword. Daarmee geef je aan dat een klasse erft van een andere klasse. Een klasse kan maar van een andere klasse erven:

```
package {
    class Auto extends Voertuig {
    }
}
```

### Grafische opbouw van een applicatie in Flash

In de vorige lesbrief hebben we gezien dat we met behulp van het trace commando uitvoer naar de Outputview kunnen schrijven. Maar over het algemeen maken we in Flash toch applicaties met een grafische gebruikers interface, een GUI (Graphical User Interface). Flash is hiervoor speciaal voor ontworpen en is daar dus bijzonder geschikt voor. Het onderstaande overzicht geeft schematisch weer hoe de GUI opgebouwd is binnen Flash:



De basis van je GUI is de **Stage**. De Stage bepaald de grootte van je applicatie en bijvoorbeeld de achtergrondkleur. Binnen de Stage hebben we de **Timeline** (hoofd tijdlijn), de representatie van tijd. Binnen de Timeline hebben we frames. We kunnen van het ene frame naar het andere “springen”.

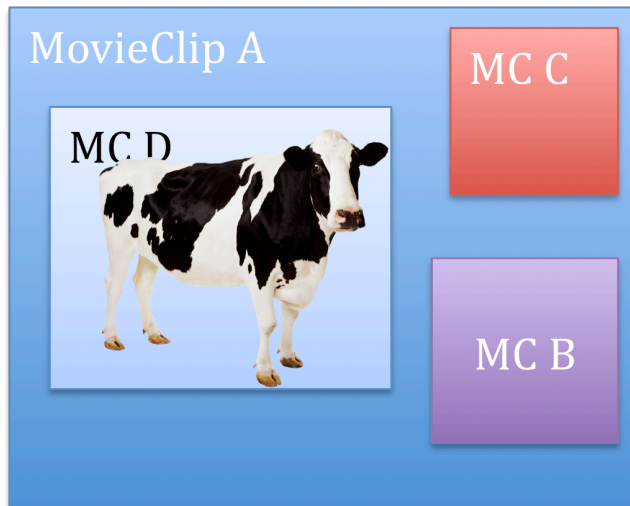
Binnen de stage komen meerdere **Symbols** voor. Op de FlashFocus wiki staat een mooie definitie van Symbols en van de MovieClip:

*“in Flash bestaan drie soorten symbols: MovieClips, Buttons en Graphics. Een MovieClip symbol kan beschouwd worden als een 'container' die content zoals een animatie, afbeelding, video, button, tekst of code kan bevatten, die je op de TimeLine (of een andere tijdlijn) kunt plaatsen. Een MovieClip kan ook leeg zijn. MovieClips kunnen "genest" worden: Dat wil zeggen dat je een MovieClip in een andere MovieClip kunt plaatsen die zelf ook weer in een MovieClip staat.*

*MovieClips kunnen voorzien worden van een instancename waardoor je ze met ActionScript kunt aanspreken. Het grootste verschil tussen een MovieClip en een Graphic is dat wanneer de hoofdtijdlijn wordt stilgezet een MovieClip door blijft lopen, en een Graphic niet (tenzij je de MovieClip aanspreekt met ActionScript). MovieClips spelen dus onafhankelijk van de tijdlijn waar ze op geplaatst zijn.”*

Referentie: <http://www.flashfocus.nl/wiki/index.php/MovieClip>

Een MovieClip die andere MovieClips bevat zou je je als volgt voor kunnen stellen:



MovieClip A bevat drie andere MovieClips, B,C en D. MovieClip D bevat een Bitmap van een koe.

### Een MovieClip maken en gebruiken in ActionScript

We kunnen zelf MovieClips maken door klassen te laten erven van een MovieClip. Op die manier kunnen we grafische objecten maken. We kunnen bijvoorbeeld een klasse definiëren die een afbeelding toont. Vervolgens kunnen we meerdere instanties maken van die klasse en deze objecten aan onze stage of aan een andere MovieClip toevoegen. Een klasse definiëren die erft van MovieClip gaat als volgt (vergeet de import niet):

```
package {
    import flash.display.MovieClip;

    public class Spookje extends MovieClip {
    }
}
```

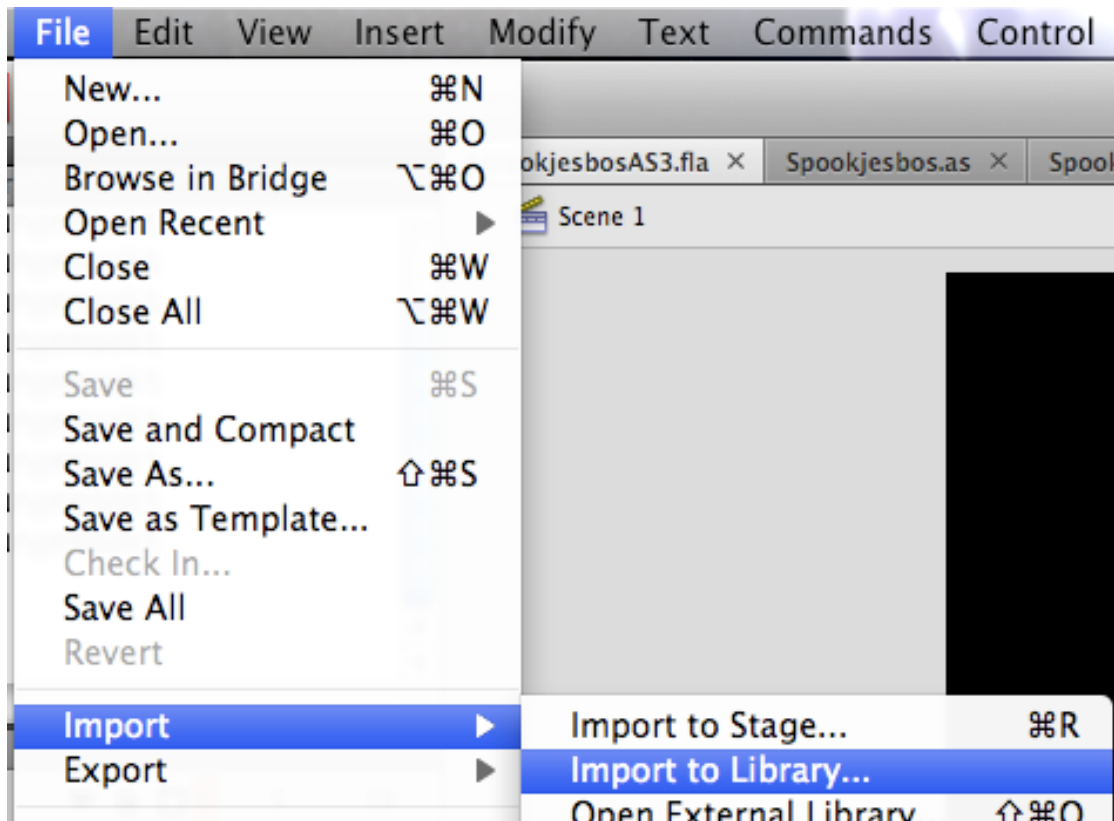
Hier wordt een klasse met de naam Spookje gedefinieerd. Een Spookje zou bijvoorbeeld een afbeelding kunnen tonen. Een Spookje erft dus van MovieClip en heeft daardoor alle eigenschappen van een MovieClip, aangevuld met eigenschappen die we zelf toevoegen aan de klasse (bijvoorbeeld een naam of een beschrijving van het spookje).

### Afbeeldingen importeren in de library

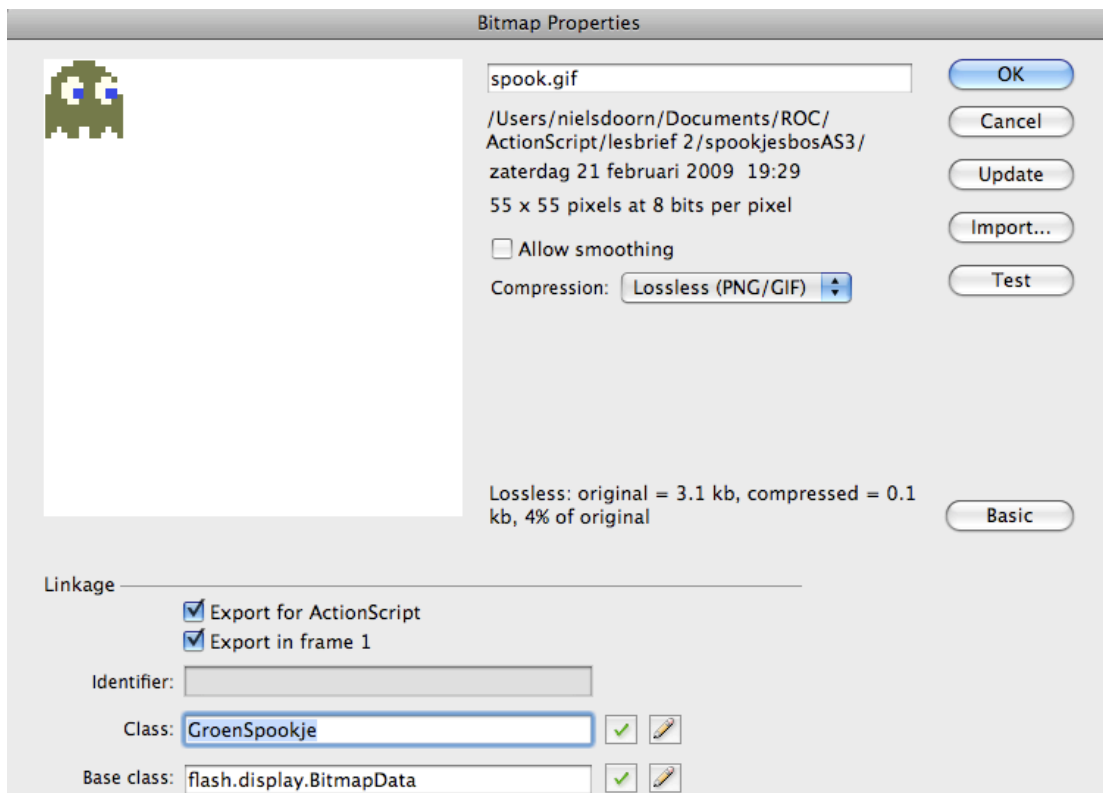
Binnen iedere Flash applicatie beschikken we over een 'Library'. In deze library kunnen we externe bestanden opslaan die we binnen onze applicatie willen gebruiken, bijvoorbeeld afbeeldingen en geluidsfragmenten.

Afbeeldingen in de library kunnen we als BitmapData, de ruwe data van de afbeelding, exporteren naar ActionScript zodat we deze kunnen gebruiken in ActionScript.

Afbeeldingen kun je importeren in de library door de bestanden er in te slepen met de muis of via de import to library functie in het menu:



Kies daarna de properties van een bestand om deze te exporteren naar ActionScript:



- Vink 'Export for ActionScript' en 'Export in frame 1' aan.

- Kies een Klasse naam (bijvoorbeeld GroenSpookje o.i.d.). Dit mag een bestaande klasse zijn (maar die moet dan wel erven van de Base class) of een niet bestaande klasse, flash genereert in dat laatste geval dan deze klasse automatisch.

### In ActionScript de afbeelding gebruiken

Hierna kun je in je ActionScript code een nieuwe instantie van de klasse 'GroenSpookje' instantieren. Dit is dan een object van het type BitmapData, de data om te gebruiken voor een Bitmap, een afbeelding.

Hier is een klasse Spookje, waar de BitmapData wordt geïnitieerd en wordt gebruikt voor een Bitmap:

```
package {
    import flash.display.MovieClip;
    import flash.display.Bitmap;
    class Spookje extends MovieClip {
        var groenSpookje = new GroenSpookje(55,55);
        var bitmap:Bitmap;
        function Spookje() {
            bmap = new Bitmap(groenSpookje);
            this.addChild(bmap);
        }
    }
}
```

GroenSpookje is een klasse die erft van BitmapData (een standaard Flash klasse). Bij het aanmaken van BitmapData moet je de breedte en hoogte van de afbeelding meegeven (in dit geval 55 bij 55 pixels).

Spookje erft van MovieClip en kan daardoor worden toegevoegd aan een andere MovieClip of aan de stage. De bitmap, bmap, kan ook worden toegevoegd aan een MovieClip. Op de laatste regel code van de constructor, wordt de bitmap toegevoegd aan de instantie van Spookje (this).

Op het moment dat er nu een nieuw spookje wordt aangemaakt en toe wordt gevoegd aan een MovieClip of aan de Stage, verschijnt er een nieuw afbeelding in de Flash applicatie.

Door de x en y coördinaten van een MovieClip aan te passen, kan de positie van een MovieClip worden aangepast.

Bijvoorbeeld het volgende stukje code dat een Spookje een willekeurige plek geeft:

```
// geef het spookje een plek om te wonen in het bos
spookje.x = Math.ceil(Math.random() * stage.stageWidth);
spookje.y = Math.ceil(Math.random() * stage.stageHeight);
```

Spookje is een object van de klasse "Spookje" die erft van MovieClip. Het object "stage" is de stage van je Flash applicatie, daar kun je altijd bij (een globale variabele). Om de hoogte en breedte van de stage op te vragen, kun je de

eigenschappen `stageWidth` en `stageHeight` gebruiken (2<sup>e</sup> en 3<sup>e</sup> regel van het voorbeeld);

`Math.random()` levert een willekeurig getal tussen 0 en 1 op (bijvoorbeeld 0,121323). `Math.ceil()` rond een getal af naar boven, `Math.floor` rond een getal af naar beneden en `Math.round()` rond een getal naar de dichtstbijzijnde gehele waarde af (9,89 wordt 10).

### Enter Frame event

Een flash applicatie roept een aantal keer per seconde (instelbaar in je .fla bestand) een frame aan. Hierbij wordt er een event gegenereerd, het enter frame event. Dit kun je gebruiken in je ActionScript code om animaties en dergelijke te programmeren. We gaan in deze lesbrief niet behandelen wat voor events er allemaal zijn in Flash, we houden het voor nu even bij het enter frame event.

We kunnen aangeven dat we een bepaalde functie iedere keer als het event plaatsvindt aan willen roepen. Dit doen door een zogenaamde eventlistener toe te voegen aan de stage. Hierbij geven we aan in welk event we geïnteresseerd zijn. We kunnen dit bijvoorbeeld in de constructor van onze Document klasse definiëren:

```
package {
    import flash.display.MovieClip;
    import flash.events.Event;

    public class Spookjesbos extends MovieClip {
        function Spookjesbos() {
            ...
            stage.addEventListener(Event.ENTER_FRAME, mainloop);
            ...
        }

        function mainloop(evt:Event) {
            ...
        }
    }
}
```

Het eerste argument van de functie `addEventListener` is een verwijzing naar het event dat we willen registreren (dit kan bijvoorbeeld ook een event m.b.t. een mouse click zijn). Het tweede argument is de naam van de functie die aangeroepen moet worden op het moment dat het event plaatsvindt, in dit geval de functie “mainloop”.

De functie `mainloop` krijgt het `Event` mee als argument.