Daniel Bertram
Kerstin Götze

ASS Flex Workshop Wintersemester 2009

für die FH Köln, Campus Gummersbach

25 Februar 2009

Syntax und Schlüsselwörter

DEFINITIONSSCHLÜSSELWÖRTER

var	Deklariert eine Variable `var myVar:String;`
const	Deklariert eine Konstante, also eine Variable, der nur ein einziges Mal ein Wert zugewiesen werden kann `const myConst:int = 1;`

package	Definiert ein Paket, in dem die Klasse organisiert ist.
	Dient der Strukturierung der Klassen.
	`package myPackage {`
	`  public class MyClass {}`
	`}`

class	Definiert eine Klasse von der Instanzen erstellt werden können, die die von der Klasse definierten Methoden und Attribute besitzen. `public class MyClass {}`
interface	Definiert ein Interface. `public interface MyInterface {}`
function	Deklariert eine Methode. `public function myFunction {}:String;`
extends	Definiert, dass eine Klasse von einer anderen Klasse ableitet/erbt. `public class MyClass extends HBox {}`
implements	Definiert, dass eine Klasse ein oder mehrere Interfaces implementiert. `public class MyClass implements MyFirstInterface, MySecondInterface {}`

get	Deklariert einen Getter, d.h. eine Methode, die wie ein Attribut gelesen werden kann.
	`public function get myVariable():String {`
	`  return _myVariable;`
	`}`

set	Deklariert einen Setter, d.h. eine Methode, die wie ein Attribut geschrieben werden kann.
	`public function set myVariable(value:String):void {`
	`  _myVariable = value;`

	`}`

ATTRIBUTE

public	Legt fest, dass eine Klasse, Methode, Konstante oder Variable für alle Aufrufer verfügbar ist. `public class MyClass {}`
protected	Legt fest, dass eine Methode, Konstante oder Variable nur für die Instanzen der sie definierenden Klasse oder von dieser abgeleitete Klassen verfügbar ist `protected function myFunction():void {}`
private	Legt fest, dass eine Methode, Konstante oder Variable nur für die Instanzen der sie definierenden Klasse verfügbar ist `private var myVar:String;`

static	Legt fest, dass eine Methode, Konstante oder Variable zu einer Klasse gehört, statt zu den Instanzen der Klasse
	`public class MyClass {`
	`public static function myFunction():void {}`
	`}`
	Usage: `MyClass.myFunction();`

final	Legt fest, dass eine Klasse nicht abgeleitet oder eine Methode nicht überschrieben werden kann. `final class MyClass {}`

dynamic	Legt fest, dass den Instanzen einer Klasse zur Laufzeit Attribute hinzugefügt werden können.
	`dynamic class RuntimeExtendableClass() {}`
	Usage:
	`var extClass:RuntimeExtendableClass = new RuntimeExtendableClass();`
	`extClass.newProperty = "new";`

override	Legt fest, dass eine Methode, die gleichnamige von der Elternklasse geerbte Methode ersetzt. `override public function myFunction():void {}`

SPEZIELLE DATENTYPEN UND KONSTANTEN

void	Legt fest, dass eine Methode keinen Wert zurückgibt `public function myFunction():void {}`
null	Ein spezielle Wert, der Veriablen zugewiesen oder von Methoden zurückgegeben werden kann, wenn keine Daten vorhanden sind `var value:String = null;`

this	Eine Referenz in einer Klasse auf die konkrete Instanz dieser Klasse
	`public class MyClass {`
	`  private var property1:String;`
	`  public static function myFunction():void {`
	`  this.property1 = "test";`
	`  }`
	`}`

NaN	Kann Variablen vom Typ Number zugewiesen werden, und sagt aus, dass der aktuelle Wert der Variablen "not a number", also keine Zahl ist. Dies kann zum Beispiel vorkommen, wenn einer Variablen vom Typ Number das Ergebnis einer Division durch Null zugewiesen wird.
	`var result:Number = 100/0;`
	`trace (result); //Ausgabe: NaN`

OPERATOREN

new	Erzeugt eine neue Instanz einer Klasse `var instance:MyClass = new MyClass();`

is	Prüft, ob eine Objekt von einem bestimmten Datentyp ist bzw. eine Instanz einer bestimmten Klasse oder eines Interfaces ist. Gibt auch dann true zurück, wenn das zu testende Objekt in seiner nicht direkt vom gesuchten Typ ist, jedoch direkt oder indirekt davon ableitet.
	`var box:Hbox = new HBox();`
	`trace(box is HBox);  //Ausgabe: true`
	`trace(box is DisplayObject);  //Ausgabe: true`
	`trace(box is VBox);  //Ausgabe: false`

DIREKTIVEN

import	Macht in anderen Paketen definierte Klassen in einer Klasse verfügbar
	`package myPackage {`
	`  import mx.controls.Button;`
	`}`

include	Fügt den Inhalt einer externen Datei in den Code ein. `include "CodeSnippet.as"`

STATEMENTS

super	Führt den Konstruktor der Elternklasse oder eine Methode der Elternklasse aus. Bei Methoden ist dieser Aufruf nur nötig, wenn diese Methode in der aktuellen Klasse überschrieben wurde, jedoch explizit die Methode aus der Elternklasse aufgerufen werden soll.
	`// Aufruf des Elternkonstruktors`
	`public function MyClass():void{`
	`  super();`
	`}`
	
	`// Aufruf der Elternmethode`
	`super.myMethod();`

for each ... in	Iteriert durch alle Elemente eines Arrays
	`var myArray:Array = {"Hello", "Flex."};`
	`for each (var item:String in myArray) {`
	`  trace(item);`
	`}`
	`/* Ausgabe:`
	`Hello`
	`Flex.`
	`*/`

return	Beendet die Ausführung eine Methode und kann einen Wert aus der ausführenden Methode zurückgeben.
	`private var myFunction_1():void{`
	`  return;`
	`}`
	
	`private var myFunction_2():int{`
	`  return 0;`
	`}`

METADATEN

[Bindable]	Sorgt dafür, dass alle Methoden und Variablen einer Klasse bzw. eine einzelne Methode oder Variable als Quelle für ein DataBinding verwendet werden kann `[Bindable] public var myProperty:int;`
[Embed]	Bettet ein Bild, SWF, Sound,... ein `[Embed(source="logo.png")] public var imgCls:Class;`
[Event]	Definiert, dass eine Klasse ein bestimmtes Event wirft bzw. werfen kann. Dieses Tag ist nicht nötig, um Events zu werfen und per AS3 abzufangen. Soll das Event allerdings auch bei Verwendung der Klasse in einem MXML-Block direkt mit einem Handler belegt werden können, muss das Tag gesetzt sein. `[Event(name="myClickEvent", type="flash.events.Event")]`
[Style]	Definiert, dass eine Klasse ein bestimmtes Style Attribut besitzt und sorgt dafür, dass dieser Style auch bei Verwendung der Klasse in einem MXML-Block gesetzt werden kann. `[Style(name="color",type="uint",inherit="yes")]`
[ArrayElementType]	Definiert, dass ein Array nur Objekte eines bestimmten Typs aufnehmen darf. Grundsätzlich sind Arrays ungetypt und können gleichzeitig Objekte verschiedener Typen aufnehmen. `[ArrayElementType("String")] public var myArray:Array;`

KOMMENTARSYNTAX

//...	Kommentiert eine Zeile aus `// einzeiliger Kommentar`

/*...*/	Kommentiert beliebig viele Zeilen aus
	
	`/* mehrzeiliger`
	`Kommentar */`

GLOBALE FUNKTIONEN

trace()	Gibt eine Nachricht in der Konsole aus `trace("test"); //Ausgabe: test`

CASTS

Class(instanceToCast)	Cast von einem Objekt in einen bestimmten Datentyp. Wirft eine Fehler, wenn das Objekt nicht in diesen Typ umgewandelt werden kann.
instanceToCast as Class	Cast von einem Objekt in einen bestimmten Datentyp. Gibt `null` zurück, wenn das Objekt nicht in diesen Typ umgewandelt werden kann.

Daniel Bertram
Kerstin Götze

ASS Flex Workshop Wintersemester 2009
für die FH Köln, Campus Gummersbach

25 Februar 2009

Syntax und Shortcuts

Shortcuts:

Alt + Pfeil hoch Pfeil runter	Zeile verschieben
Ctrl + Leertaste	Kodevervollständigung
Ctrl + .	Wort Vervollständigung
Tab	Einrücken
Shift + Tab	Einrückung reduzieren
cmd + D	Löschen der aktuellen Zeile
Shift + cmd + D	Erstellt einen CDATA Block

Syntax:

Klassen in MXML

```
<HBox
    xmlns:mx="http://www.adobe.com/2006/mxml"
    height="100"
/>
</HBox>

<Elternklasse
    namespace:namespaceName="Pfad des Namespaces"
    weiteresAttribut="Wert"
/>
```

</Elternklasse>

Klassen in AS3

```
package flex.workshop {
	public class MyClass  {}
}
```

```
Schlüsselwort paketpfad {
	Sichtbarkeit Schlüsselwort KlassenName {}
}
```

Methoden in AS3

```
private function myMethod (parameter:String):Boolean {return true;}
```

```
Sichtbarkeit Schlüsselwort methodenName (parameter:Datentyp):DatentypDesRückgabewertes
```

Variablen in AS3

```
protected var myVariable:Number = 0;
```

```
Sichtbarkeit Schlüsselwort variablenName:Datentyp = Wert;
```

Kommentare in MXML

```
<!-- Ein Kommentar -->
```

Kommentare in AS3

```
// Ein einzeiliger Kommentar
/* Ein
mehrzeiliger
Kommentar */
```