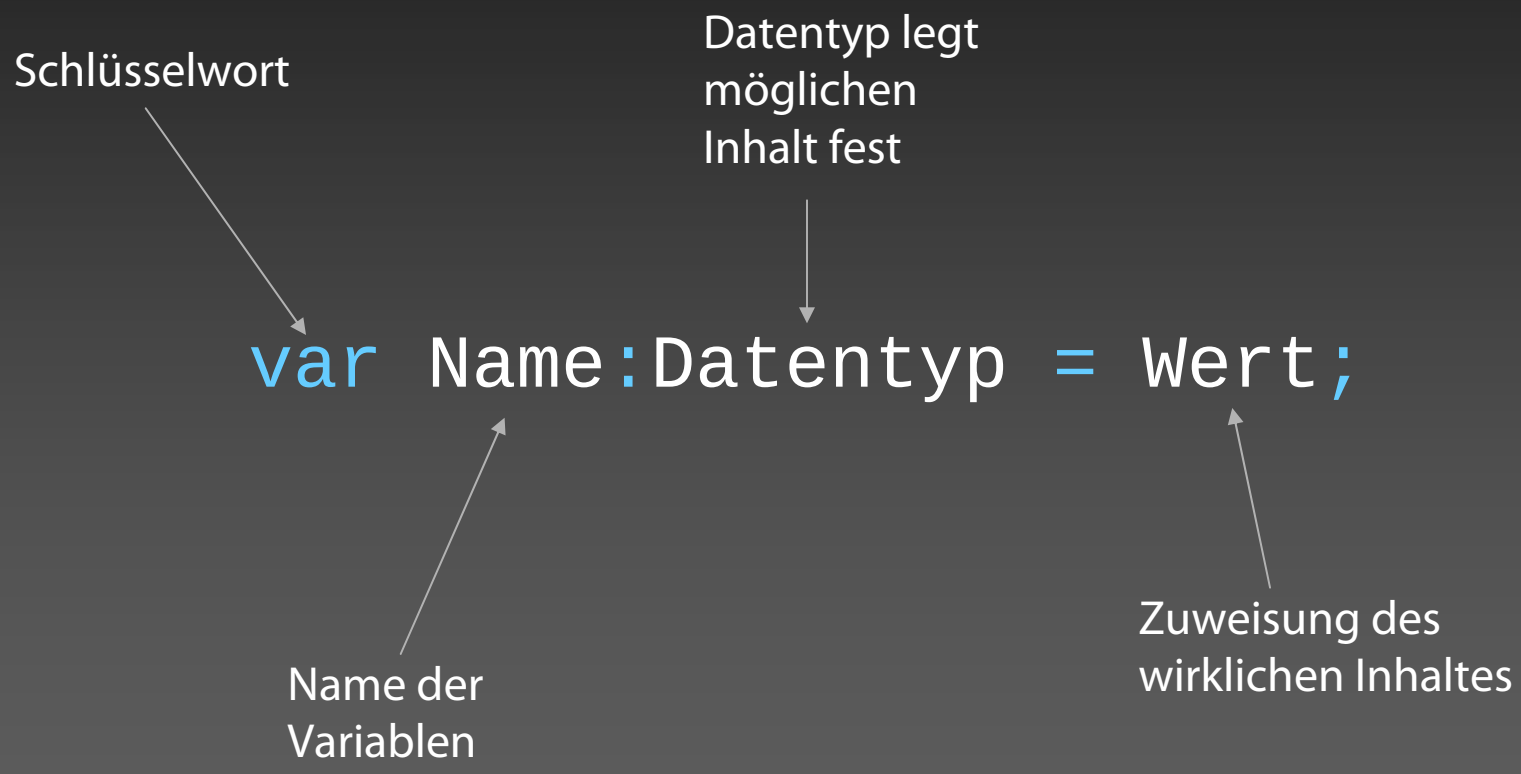
Basics der objektorientierten Programmierung in Flash AS3

Daniel Frank

Sommersemester 2011

Variablen


Schlüsselwort
Datentyp legt möglichen Inhalt fest
var Name:Datentyp = Wert;
Name der Variablen
Zuweisung des wirklichen Inhaltes

Variablen

Beispiel

```
var myValue:Number = 20.0;

trace(myValue);


var myValue2:Number;
myValue2 = 22.5;

trace(myValue2);
```

Variablen

int	ganze Zahlen	-5, -10, 44
uint	positive ganze Zahlen	1, 10, 200
Number	Gleitkommazahlen	-1.5, 22.5, 1.48995
Boolean	„true“ oder „false“	true, false
String	Text	„hallo welt!“

Variablen

Beispiel

```
var myValue:int = 12;
var myValue2:int = 8;

var myValue3:int = myValue + myValue2;

trace(myValue3);

myValue3 = myValue * myValue2;

trace(myValue3);
```

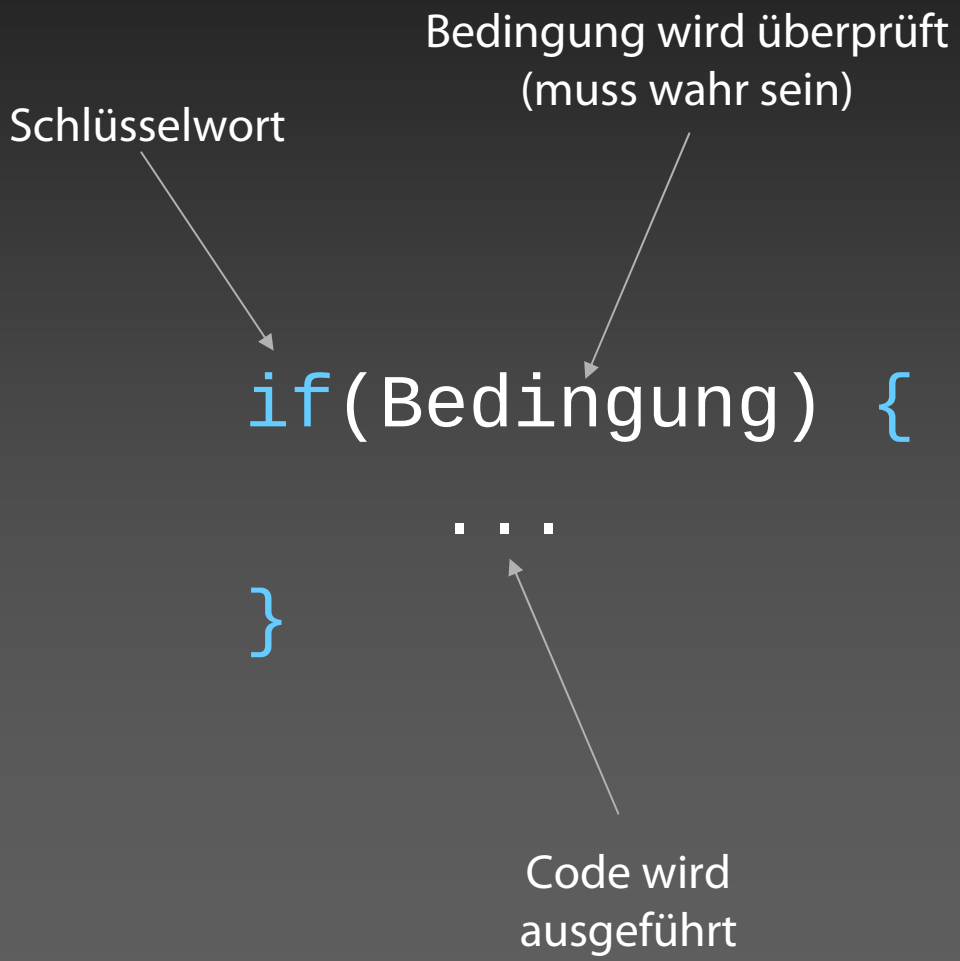
Variablen

Rechenoperatoren

```
trace(2 + 3);     //5
trace(10 – 2);    //8
trace(2 * 5);     //10
trace(10 / 2);    //5
trace(5 % 3);     //2
trace(9 % 3);     //0
```

```
var i = 0;
i++;
trace(i);         //1
i--;
trace(i);         //0
```

Bedingungen



Bedingungen

Beispiel

```
var myVariable:Boolean = true;

if(myVariable) {
   trace("true");
}

myVariable = false;

if(myVariable) {
   trace("false");
}
```

Bedingungen

if... else

```
var myVariable:Boolean = false;

if(myVariable) {

   trace("true");

} else {

   trace("false");

}
```

Bedingungen

if... else if

```
var myVariable:int = 10;

if(myVariable < 10) {

   trace("small");

} else if(myVariable == 10){

   trace("equal");

} else { trace("other"); }
```

Vergleiche – kleiner oder größer

```
trace(2 < 3);      //true

trace(2 > 3);      //false

trace(2 <= 2);     //true

trace(2 >= 3);     //false
```

Bedingungen

Vergleiche – gleich oder ungleich

```
trace(2 == 2);    //true

trace(2 == 3);    //false

trace(2 != 2);    //false

trace(2 != 3);    //true
```

Bedingungen

UND

```
trace( true &&  true );        //true

trace( true && false );        //false

trace( false && true );        //false

trace( false && false);        //false
```

Bedingungen

ODER

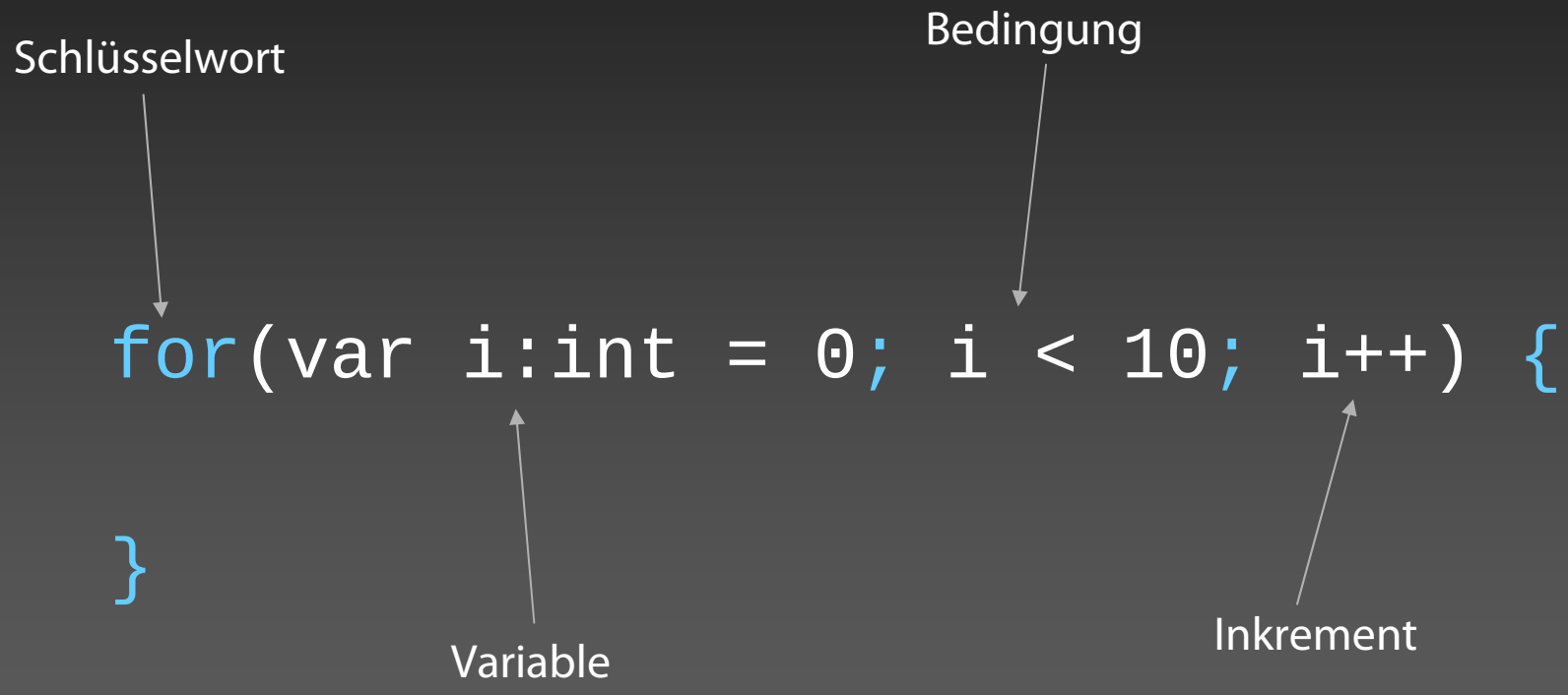
```
trace( true ||  true );       //true

trace( true || false );       //true

trace( false || true );       //true

trace( false || false );      //false
```

Bedingungen

if... else

```
var wert:int = 15;

if( (wert > 10) && (wert < 20) ) {

   trace("im Bereich");

} else {

   trace("nicht im Bereich");
}
```

Schleifen


Schlüsselwort
Bedingung
for(var i:int = 0; i < 10; i++) {
}
Variable
Inkrement

Schleifen

Beispiel

```
for(var i:int = 0; i < 10; i++) {
   trace(i);
}
```

Ausgabe: 0

1

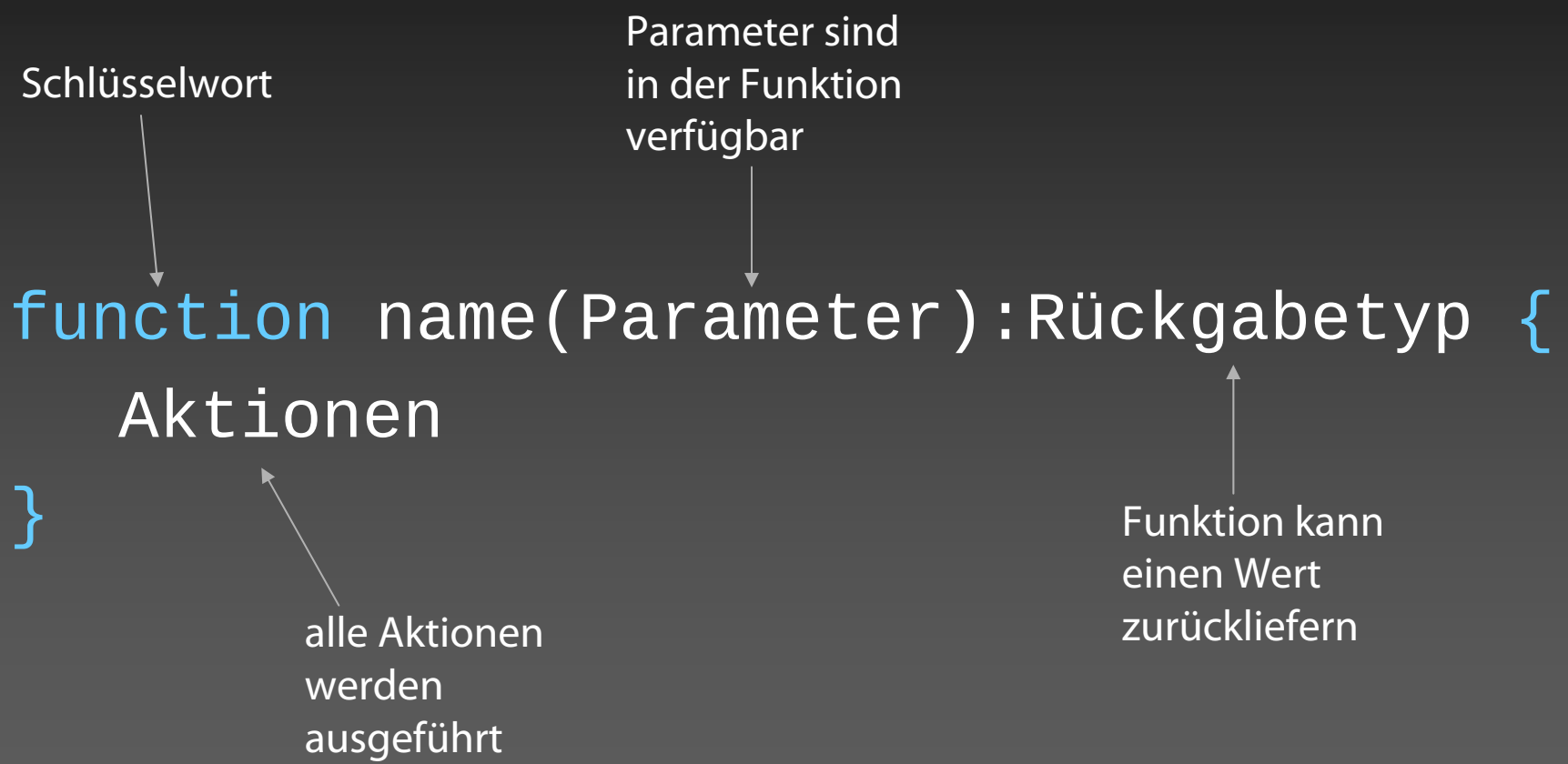
2

...

Beispiel

```
for(var i:int = 0; i < 25; i++) {

   if(i % 5 == 0) {
       trace(i + " ist teilbar durch fünf");
   } else {
       trace("rest bleibt übrig");
   }
}
```

Funktionen


Schlüsselwort
Parameter sind in der Funktion verfügbar
function name(Parameter):Rückgabetyp {
Aktionen
}
alle Aktionen werden ausgeführt
Funktion kann einen Wert zurückliefern

Funktionen

Beispiel

```
function ausgabe():void {
   trace("ausgabe");
}
```

Funktionen

Beispiel

```
function ausgabe(input:String):void {
   trace(input);
}
```

Funktionen

Beispiel

```
function berechne(input:int, input2:int):void {
   var temp:int = input + input2;

   trace(temp);
}
```

Beispiel

```
function berechne(input:int, input2:int):int {
   if(input < 0)
       input = input * -1;

   if(input2 < 0)
       input2 = input2 * -1;

   return(input + input2);
}
```

Events


Schlüsselwort
welches Event?
addEventListener(Event, Funktion);
Funktion wird ausgeführt

Beispiel

```
stage.addEventListener(MouseEvent.CLICK, ausgabe);

function ausgabe(e:MouseEvent):void {
   trace("geklickt!");
}
```

Events

MouseEvent	MOUSE_CLICK
	MOUSE_DOWN
	MOUSE_UP
	MOUSE_OVER
KeyboardEvent	KEY_DOWN
	KEY_UP
Event	ENTER_FRAME
	MOUSE_LEAVE

Beispiel

```
stage.addEventListener(MouseEvent.CLICK, move);

function move(e:MouseEvent):void {
   rectangle.x = e.stageX;
   rectangle.y = e.stageY;
}
```

Klassen sind Baupläne für konkrete Objekte

```
class MyClass {
   public var x:int;

   public function MyClass(){
       x = 10;
   }

   public function myTrace() {
       trace("x: " + x);
   }
}
```

Instanziierung, Zugriff auf Funktion & Attribut

```
import MyClass;

var myObject:MyClass;

myObject = new MyClass();

myObject.myTrace();

trace(myObject.x);
```

Klassen bilden Hierarchien durch Vererbung