Programmierungsgrundlagen Teil 2

Hans Christian Feßl

Operatoren

Logische Operatoren

Es werden boolsche Werte miteinander in Verbindung gebracht.

AND &&

OR ||

NOT

Logische Operatoren

UND

Der UND Operator

&&

MagKaffee	&&	HabeGeld	HoleKaffee
TRUE	&&	TRUE	
TRUE	&&	FALSE	
FALSE	&&	TRUE	
FALSE	&&	FALSE	

Logische Operatoren

UND

Der UND Operator

&&

MagKaffee	&&	HabeGeld	HoleKaffee
TRUE	&&	TRUE	TRUE
TRUE	&&	FALSE	
FALSE	&&	TRUE	
FALSE	&&	FALSE	

Logische Operatoren

UND

Der UND Operator

&&

MagKaffee	&&	HabeGeld	HoleKaffee
TRUE	&&	TRUE	TRUE
TRUE	&&	FALSE	FALSE
FALSE	&&	TRUE	
FALSE	&&	FALSE	

Logische Operatoren

UND

Der UND Operator

&&

MagKaffee	&&	HabeGeld	HoleKaffee
TRUE	&&	TRUE	TRUE
TRUE	&&	FALSE	FALSE
FALSE	&&	TRUE	FALSE
FALSE	&&	FALSE	

Logische Operatoren

UND

Der UND Operator

&&

MagKaffee	&&	HabeGeld	HoleKaffee
TRUE	&&	TRUE	TRUE
TRUE	&&	FALSE	FALSE
FALSE	&&	TRUE	FALSE
FALSE	&&	FALSE	FALSE

Logische Operatoren

ODER

Der ODER Operator

||

MagKaffee	||	HabeGeld	HoleKaffee
TRUE	||	TRUE	
TRUE	||	FALSE	
FALSE	||	TRUE	
FALSE	||	FALSE	

Logische Operatoren

ODER

Der ODER Operator

||

MagKaffee	||	HabeGeld	HoleKaffee
TRUE	||	TRUE	TRUE
TRUE	||	FALSE	
FALSE	||	TRUE	
FALSE	||	FALSE	

Logische Operatoren

ODER

Der ODER Operator

||

MagKaffee	||	HabeGeld	HoleKaffee
TRUE	||	TRUE	TRUE
TRUE	||	FALSE	TRUE
FALSE	||	TRUE	
FALSE	||	FALSE	

Logische Operatoren

ODER

Der ODER Operator

||

MagKaffee	||	HabeGeld	HoleKaffee
TRUE	||	TRUE	TRUE
TRUE	||	FALSE	TRUE
FALSE	||	TRUE	TRUE
FALSE	||	FALSE	

Logische Operatoren

ODER

Der ODER Operator

||

MagKaffee	||	HabeGeld	HoleKaffee
TRUE	||	TRUE	TRUE
TRUE	||	FALSE	TRUE
FALSE	||	TRUE	TRUE
FALSE	||	FALSE	FALSE

Logische Operatoren

NICHT

Der NICHT Operator

!

MagKaffee	&&	!HabeGeld	HoleKaffee
TRUE	&&	!TRUE	
TRUE	&&	!FALSE	
FALSE	&&	!TRUE	
FALSE	&&	!FALSE	

Logische Operatoren

NICHT

Der NICHT Operator

!

MagKaffee	&&	!HabeGeld	HoleKaffee
TRUE	&&	!TRUE	FALSE
TRUE	&&	!FALSE	
FALSE	&&	!TRUE	
FALSE	&&	!FALSE	

Logische Operatoren

NICHT

Der NICHT Operator

!

MagKaffee	&&	!HabeGeld	HoleKaffee
TRUE	&&	!TRUE	FALSE
TRUE	&&	!FALSE	TRUE
FALSE	&&	!TRUE	
FALSE	&&	!FALSE	

Logische Operatoren

NICHT

Der NICHT Operator

!

MagKaffee	&&	!HabeGeld	HoleKaffee
TRUE	&&	!TRUE	FALSE
TRUE	&&	!FALSE	TRUE
FALSE	&&	!TRUE	FALSE
FALSE	&&	!FALSE	

Logische Operatoren

NICHT

Der NICHT Operator

!

MagKaffee	&&	!HabeGeld	HoleKaffee
TRUE	&&	!TRUE	FALSE
TRUE	&&	!FALSE	TRUE
FALSE	&&	!TRUE	FALSE
FALSE	&&	!FALSE	FALSE

Logische Operatoren

NICHT

(!myVar) = ich frage ab ob die Variable false ist anstatt true

Logische Operatoren

Vorrangregel

NICHT (!)

vor

UND (&&)

vor

ODER (||)

Logische Operatoren

Vorrangregel

NICHT (!)

vor

UND (&&)

vor

ODER (||)

Logische Operatoren

Vorrangregel

MagKaffee	&&	HabeGeld	||	IstGratis	HoleKaffee
JA	&&	JA	||	JA	
JA	&&	JA	||	NEIN	
JA	&&	NEIN	||	JA	
JA	&&	NEIN	||	NEIN	
NEIN	&&	NEIN	||	NEIN	
NEIN	&&	JA	||	NEIN	
NEIN	&&	NEIN	||	JA	
NEIN	&&	JA	||	JA	

Logische Operatoren

Vorrangregel

MagKaffee	&&	HabeGeld	||	IstGratis	HoleKaffee
JA	&&	JA	||	JA	JA
JA	&&	JA	||	NEIN	JA
JA	&&	NEIN	||	JA	JA
JA	&&	NEIN	||	NEIN	NEIN
NEIN	&&	NEIN	||	NEIN	NEIN
NEIN	&&	JA	||	NEIN	NEIN
NEIN	&&	NEIN	||	JA	JA
NEIN	&&	JA	||	JA	JA

Logische Operatoren

Vorrangregel

MagKaffee	&&	HabeGeld	||	IstGratis	HoleKaffee
JA	&&	JA	||	JA	JA
JA	&&	JA	||	NEIN	JA
JA	&&	NEIN	||	JA	JA
JA	&&	NEIN	||	NEIN	NEIN
NEIN	&&	NEIN	||	NEIN	NEIN
NEIN	&&	JA	||	NEIN	NEIN
NEIN	&&	NEIN	||	JA	JA
NEIN	&&	JA	||	JA	JA

Beispiel für fehlerhafte Semantik – Syntax ist trotzdem richtig

Logische Operatoren

Vorrangregel

MagKaffee && (HabeGeld || IstGratis)!

Erweitern um Eigenschaft ob Kaffeeautomat in Betrieb ist

InBetrieb!

Logische Operatoren

Vorrangregel

InBetrieb && MagKaffee && (HabeGeld || IstGratis)

MagKaffee && InBetrieb && (HabeGeld || IstGratis)

InBetrieb || MagKaffee && (HabeGeld || IstGratis)

MagKaffee || InBetrieb && (HabeGeld || IstGratis)

MagKaffee && (HabeGeld || IstGratis) && InBetrieb

MagKaffee && (HabeGeld || IstGratis) || InBetrieb

MagKaffee && (HabeGeld || (IstGratis && InBetrieb))

MagKaffee && (HabeGeld || (IstGratis || InBetrieb))

Logische Operatoren

Vorrangregel

InBetrieb && MagKaffee && (HabeGeld || IstGratis)

MagKaffee && InBetrieb && (HabeGeld || IstGratis)

InBetrieb || MagKaffee && (HabeGeld || IstGratis)

MagKaffee || InBetrieb && (HabeGeld || IstGratis)

MagKaffee && (HabeGeld || IstGratis) && InBetrieb

MagKaffee && (HabeGeld || IstGratis) || InBetrieb

MagKaffee && (HabeGeld || (IstGratis && InBetrieb))

MagKaffee && (HabeGeld || (IstGratis || InBetrieb))

Operatoren

Verkettungsoperatoren

```
var Vorname = 'Anna';

var Nachname = 'Musterfrau';


var Name = Vorname + ' ' + Nachname;

alert(Name);
```

Ausgabe: Anna Musterfrau

Achtung: In JS wird anhand der Datentypen entschieden, ob addiert oder zusammengehängt wird!

Operatoren

Verkettungsoperatoren

```
var x = 42;
var y = 21;
alert(x + y);
```

Ergebniss:

```
var x = '42';
var y = '21';
alert(x + y);
```

Ergebniss:

Operatoren

Verkettungsoperatoren

```
var x = 42;
var y = 21;
alert(x + y);
```

Ergebniss: 63

```
var x = '42';
var y = '21';
alert(x + y);
```

Ergebniss: '4221'

Operatoren

Verkettungsoperatoren

```
var x = 42;
var y = 21;
alert(x + y);
```

Ergebniss: 63

```
var x = '42';
var y = '21';
alert(x + y);
```

Ergebniss: '4221'

Kontrollstrukturen

if-else

```
if(Bedingung) {
        Anweisungen...
}else{
        Anweisungen...
}
```

Kontrollstrukturen

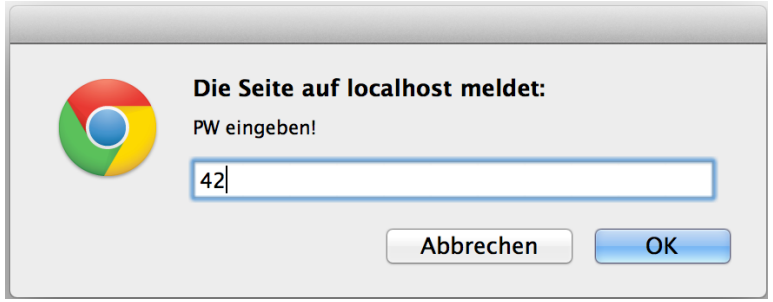
if-else

```
var meineZahl = 52;


if(meineZahl == 52){
        alert(“Die Zahl ist richtig“);
}else{
        alert(“Die Zahl ist falsch“);
}
```

Kontrollstrukturen

if-else

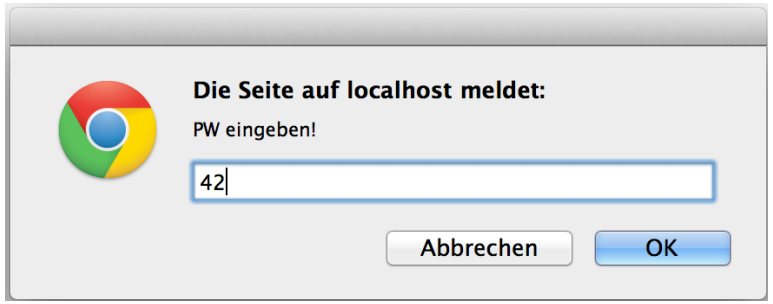
```
var passwort = 42;

var eingabe = window.prompt("PW eingeben!");


if(eingabe == passwort){
    alert("richtig");
}else{
    alert("falsch");
}
```

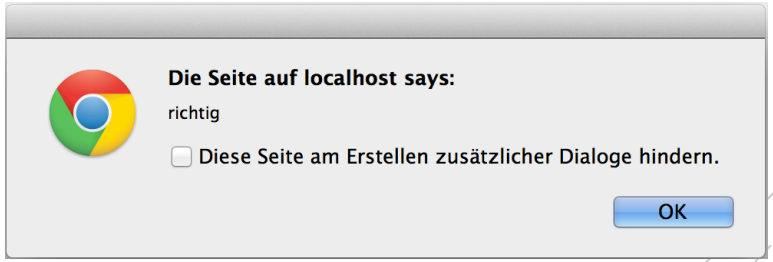


Kontrollstrukturen

if-else

```
var passwort = 42;

var eingabe = window.prompt("PW eingeben!");


if(eingabe == passwort){

alert("richtig");

}else{

alert("falsch");

}
```





Kontrollstrukturen

if-else

```
var meineZahl = 52;


if(meineZahl == 52){
        alert(“Die Zahl ist richtig“);
}else{
        alert(“Die Zahl ist falsch“);
}
```

Kontrollstrukturen

if-else

```
var x = 52;

var y = 30;


if(x == 52 && y == 30){
        alert(“???????“);
}else{
        alert(“???????“);
}
```

Kontrollstrukturen

if-else

```
var x = 52;

var y = 30;


if(x == 52 && y == 30){

      alert(“Beides ist richtig“);

}else{

      alert(“???????“);

}
```

Kontrollstrukturen

if-else

```
var x = 52;

var y = 30;


if(x == 52 && y == 30){

      alert(“Beides ist richtig“);

}else{

      alert(“Mindestens eines falsch“);

}
```

Kontrollstrukturen

if-else

```
var jetzt = new Date();

var stunde = jetzt.getHours();


if(stunde <= 7){

alert('Schnarch...viel zu Früh!');

}
```

Kontrollstrukturen

if-else

```
var jetzt = new Date();

var stunde = jetzt.getHours();


if(stunde <= 7){

        alert('Schnarch...viel zu Früh!');

} else if(stunde <= 12){

         alert('Guten Morgen!');

}
```

Kontrollstrukturen

if-else

```
var jetzt = new Date();
var stunde = jetzt.getHours();


if(stunde <= 7){
        alert('Schnarch...viel zu Früh!');
} else if(stunde <= 12){
        alert('Guten Morgen!');
} else if(stunde <= 22){
        alert('Ich hau mich hin!');
}else{
        alert('Irgendwas zwischen 22 und 0');
}
```

Schleifen

while

```
while(Bedingung){

Anweisungen...

}
```

Beispiel:

```
var zehnmal = 1;

while(zehnmal <= 10){
      alert(zehnmal);
       zehnmal++;
}
```

Schleifen

INFO

Endlosschleifen laufen endlos!

Bedingung MUSS erfüllt werden

Schleifen

do-while

```
do{
        Anweisungen...
}while(Bedingung)
```

do-while wird immer mindestens einmal ausgeführt!

do-while wird Fußgesteuert gennant.

while wird Kopfgesteuert genannt.

Schleifen

do-while

```
var zehnmal = 11;


do {
        alert(zehnmal);
        zehnmal++;
}while(zehnmal <= 10)
```

Schleifen

do-while

```
var passwort = 42;


do {
      var ein = window.prompt("Passwort");
}while(ein != passwort)
```

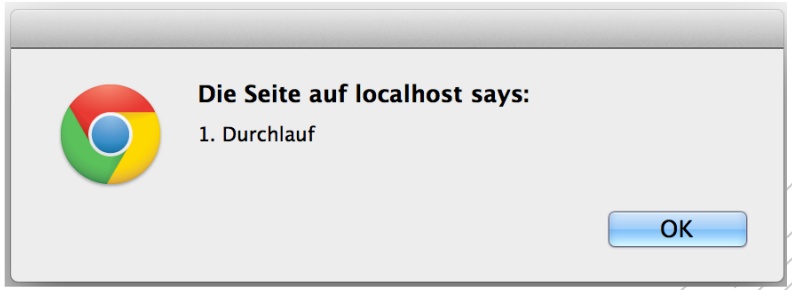
Führt die Schleife solange durch bis das Eingetragene gleich 42 ist!

Schleifen

for

```
for(Start; Bedingung; Veränderung){
        Anweisungen…
}
```

```
for(i=1; i<=10; i++){
        alert(i + ". Durchlauf");
}
```



Funktionen
(aka Unterprogramme)

Code strukturieren

Codeteile wiederverwendbar

Etliche „vordefinierte“ Funktionen

alert(), toString(), Date(), etc.

Funktionen

Schreiben

```
function myFunct(){
        alert("Meine Funktion");
}

myFunct();
```

Funktionen

Schreiben

```
function myFunct(){
        alert("Meine Funktion");
}

myFunct();
```

Funktionen

Schreiben

Egal wo sich die Funktion im Code befindet,

wir können sie von überall aus aufrufen.

```
myFunct();

function myFunct(){
alert("Meine Funktion");
}

myFunct();
```

Funktionen

Parameter

```
myFunct('Bla');

function myFunct(txt){
        alert(txt);
}

myFunct('Fasel');
```

Funktionen

Parameter

```
function myFunct(a, b){
        alert(a + b);
}

myFunct(20, 10);
```

Funktionen

Gültigkeitsbereich

Eine Variable hat einen Gültigkeitsbereich Globale und Lokale Variablen Lokale Variablen sind nur in ihrem Scope gültig.

Scope kann man sich mit den Klammern der Funktionen deklaration gleichsetzen

Funktionen

Gültigkeitsbereich

Eine Variable hat einen Gültigkeitsbereich Globale und Lokale Variablen Lokale Variablen sind nur in ihrem Scope gültig.

Scope kann man sich mit den Klammern der Funktionen deklaration gleichsetzen

Funktionen

Gültigkeitsbereich

```
var a = 42;

machwas();

alert(a + b);


function myFunct(a, b){

var b = 21;

a = a / b;

}
```

Funktionen

Gültigkeitsbereich

```
var a = 42;

machwas();

alert(a + b);


function myFunct(a, b){

var b = 21;

a = a / b;

}
```

Funktionen

Gültigkeitsbereich

```
var a = 42;

machwas();

alert(a + b);


function myFunct(a, b){

var b = 21;

a = a / b;

}


alert(b);
```

Funktionen

Gültigkeitsbereich

```
var a = 1;

bla();


function bla(){
        alert(a);
        var b = 2;
        alert(b);
}


alert(b);
```

Funktionen

Gültigkeitsbereich

```
var a = 1;

bla();


function bla(){
        alert(a);
        var b = 2;
        alert(b);
}


alert(b);   //hier nicht gültig
```

Funktionen

Gültigkeitsbereich

```
var a = 1;


function bla(){
        var a = 3;
        var b = 2;
}


bla();
alert(a);
```

Funktionen

Gültigkeitsbereich

```
var a = 1;


function bla(){
        var a = 3;
        var b = 2;
}


bla();
alert(a);
```

Funktionen

Gültigkeitsbereich

```
var a = 1;


function bla(){
        a = 3;
        var b = 2;
}


bla();
alert(a);
```

Funktionen

Rückgabewert

```
var a = bla();

alert(a);


function bla(){

    return(42);

}
```

Die Variable a bekommt somit den Wert 42.

Funktioniert ähnlich wie prompt()!

Kommentare

```
var a = bla();

alert(a);

// Kommentar

function bla(){ //Auch einer!

      return(42);

}


/*

Mehrzeiliger

Kommentar

*/
```