

Entwicklung Interaktiver Anwendungen

Lektionseinführung

Prof. Dr. Gabriel Rausch

WWW HTTP

W3C

Attribute

Tags

HTML

DOM

UML

Schrift

Selektoren

Farbe

CSS

kaskadieren

Formatierung

Hex

Gestaltungsraster

Adaptive Design

Grids

Responsive Design

Browserkompatibilität

CSS Preprozessoren

Transpiler

Compiler

Frameworks

EcmaScript

TypeScript

Libraries

JavaScript

SCRIPT

Variablen

Text

Datentypen

Operatoren

Interfaces

Arrays

Bedingungen

Schleifen

DOM Manipulation

Events

Laufzeitumgebungen

VM

Funktionen

Debugging

Methoden

WWW HTTP

W3C

Attribute

Tags

HTML

DOM

UML

Schrift

Selektoren

Farbe

CSS

kaskadieren

Formatierung

Hex

Gestaltungs raster

Adaptive Design

Grids

Responsive Design

Browserkompatibilität

CSS Preprozessoren

Transpiler

Compiler

Frameworks

TypeScript

EcmaScript

JavaScript

SCRIPT

Variablen

Text

Datentypen

Operatoren

Interfaces

Arrays

Bedingungen

Schleifen

DOM Manipulation

Events

Laufzeitumgebungen

VM

Funktionen

Debugging

Methoden

Entwicklung Interaktiver Anwendungen

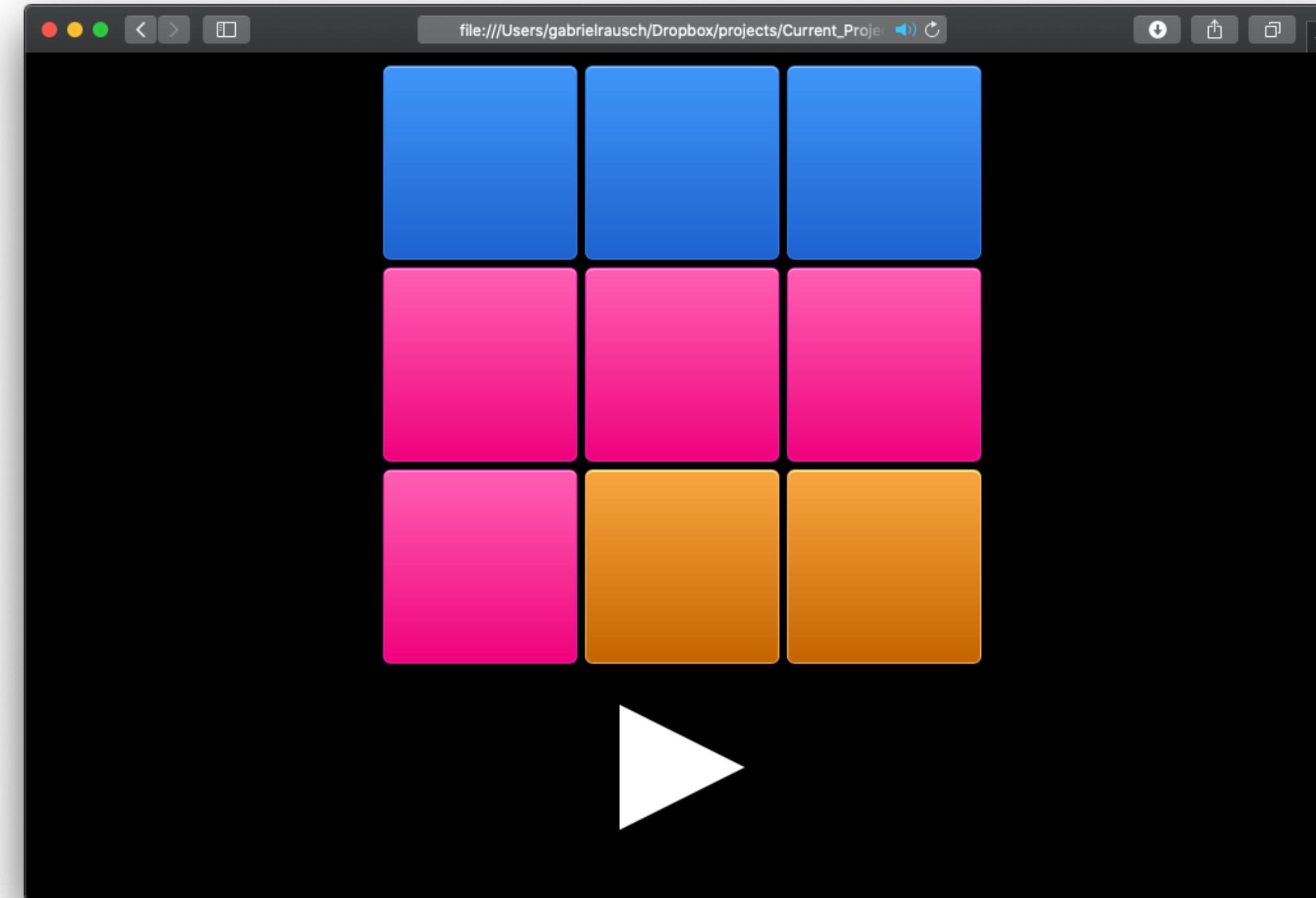
Bedingte Anweisung / If-Anweisung

Prof. Dr. Gabriel Rausch

Mögliche Einsätze einer Bedingung:

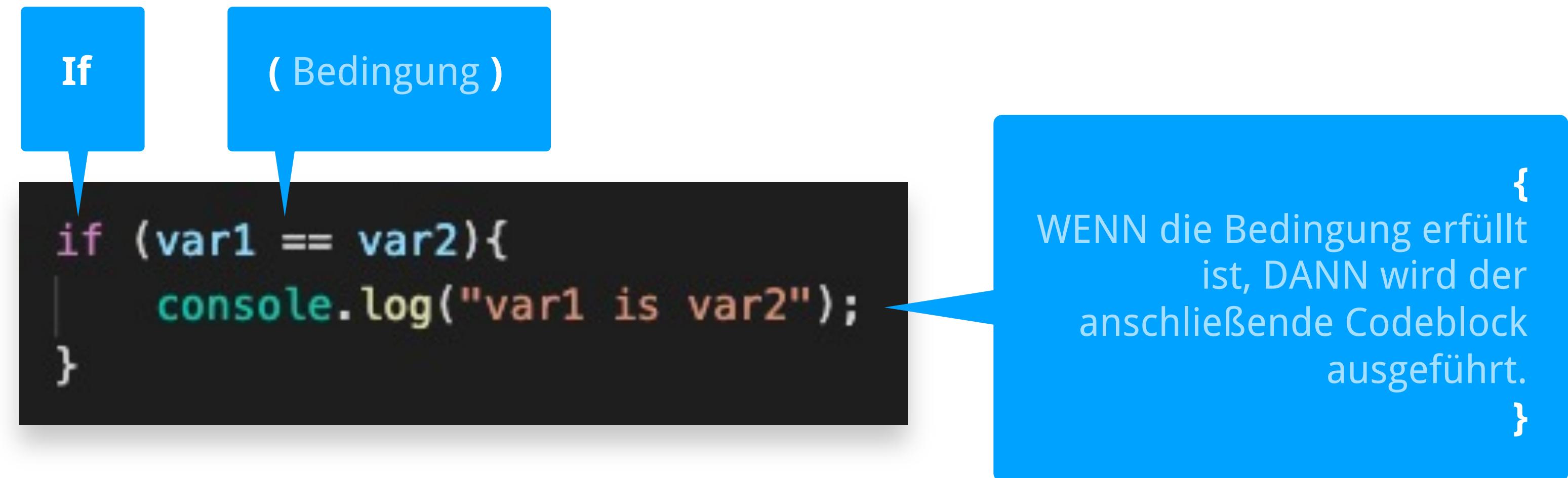
“Wenn Play-Button geklickt wurde, dann soll Play-Button als Stop-Button fungieren”.

“Wenn Ende des Drum-Arrays erreicht wurde, dann soll wieder vom Anfang des Arrays gelesen werden”



Bedingte Anweisung If-Anweisung

Bedingte Anweisung



Bedingte Anweisung Examples

```
if (var1 == 10) {  
    console.log("var1 is 10");  
}  
  
if (var1 > 10) {  
    console.log("var1 is greater than 10");  
}  
  
if (var1 != 10) {  
    console.log("var1 is not 10");  
}  
  
if (var1 == (var2 + 8)) {  
    console.log("var1 is calculated result of var2 + 8");  
}
```

Entwicklung Interaktiver Anwendungen

Vergleichsoperatoren

Prof. Dr. Gabriel Rausch

Vergleichsoperatoren

(Wertevergleich mit Vergleichsoperatoren)

```
if (var1 == var2){  
    console.log("var1 is var2");  
}
```

Vergleichsoperatoren

Gleichheit der Werte	<code>==</code>	<code>5 == 5</code>	<code>true</code>
		<code>6 == 5</code>	<code>false</code>
Ungleichheit der Werte	<code>!=</code>	<code>6 != 5</code>	<code>true</code>
		<code>5 != 5</code>	<code>false</code>
Größer als	<code>></code>	<code>5 > 2</code>	<code>true</code>
Kleiner als	<code><</code>	<code>5 < 8</code>	<code>true</code>
Größer oder gleich	<code>>=</code>	<code>6 >= 5</code>	<code>true</code>
Kleiner oder gleich	<code><=</code>	<code>5 <= 5</code>	<code>true</code>

Entwicklung Interaktiver Anwendungen

Logische Operatoren

Prof. Dr. Gabriel Rausch

Logische Operatoren

```
if (var1 == var2){  
    console.log("var1 is var2");  
}
```

Logische Operatoren

```
if (var1 == var2 && var1 > 10) {  
    console.log("var1 is var2 and var1 is greater than 10");  
}
```

Logische Operatoren

Logisches „UND“	<code>&&</code>	<code>x = 5; y = 2;</code>	<code>x > 3 && y < 5</code>	<code>true</code>
			<code>x > 5 && y < 5</code>	<code>false</code>
Logisches „ODER“	<code> </code>	<code>x = 5; y = 2;</code>	<code>x > 3 y < 5</code>	<code>true</code>
			<code>x > 5 y < 5</code>	<code>true</code>
Logisches „NOT“	<code>!</code>	<code>x = 5; y = 2;</code>	<code>!(x == y)</code>	<code>true</code>

Entwicklung Interaktiver Anwendungen

Einfache Verzweigung / If-Else-Anweisung

Prof. Dr. Gabriel Rausch

Einfache Verzweigung

```
if (Bedingung) {  
}
```

```
else {  
}
```

Wird ausgeführt, wenn
Bedingung nicht erfüllt wurde

```
if (var1 == 10) {  
    console.log("var1 is 10");  
} else {  
    console.log("var1 is not 10");  
}
```

Entwicklung Interaktiver Anwendungen

Mehrfache Verzweigung (If - Else If - Else)

Prof. Dr. Gabriel Rausch

Mehrfache Verzweigung

If - Else If - Else - Anweisung

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



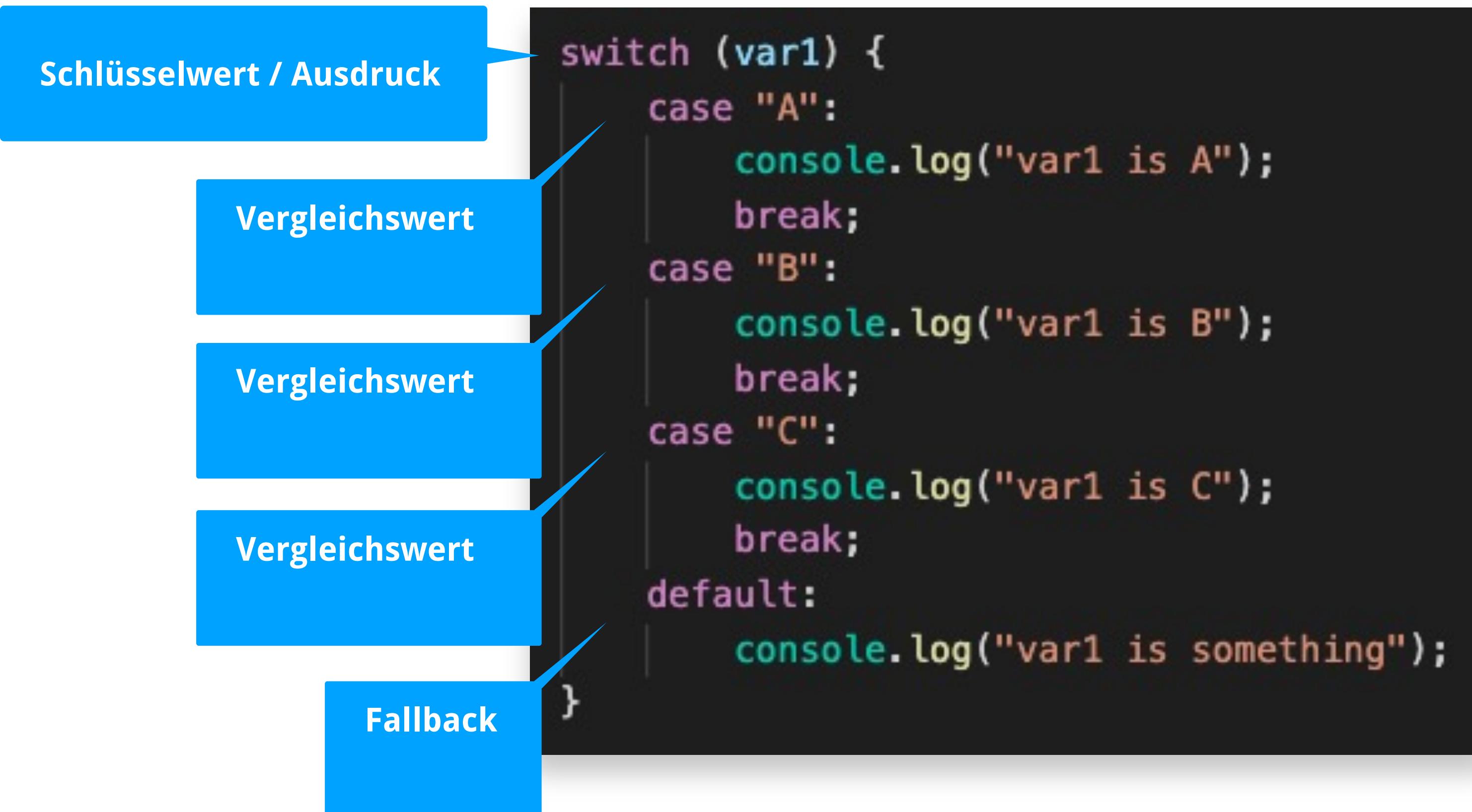
```
if (var1 == 10) {  
    console.log("var1 is 10");  
} else if(var1 == 20) {  
    console.log("var1 is 20 (and not 10)");  
} else {  
    console.log("var1 is neither 10 nor 20");  
}
```

Entwicklung Interaktiver Anwendungen

Mehrfache Verzweigung (Switch Case)

Prof. Dr. Gabriel Rausch

Mehrfache Verzweigung Switch-Case-Anweisung



Entwicklung Interaktiver Anwendungen

Bedingungen – Beispiele und Schreibweisen

Prof. Dr. Gabriel Rausch

If-Else-Anweisung Beispiel

Button CSS-Klasse wechseln (toggle)

```
<body>
|   <button id="myBtn">Button</button>
|</body>

button {
|   background: red;
}

button.active {
|   background: blue
}
```

```
var btn:HTMLElement = document.querySelector('button');

if (btn.getAttribute('class') == 'active') {
|   btn.setAttribute('class', '');
} else {
|   btn.setAttribute('class', 'active');
}

if (btn.classList.contains('active')) {
|   btn.classList.remove('active');
} else {
|   btn.classList.add('active');
}
```

Bedingungen Beispiel Alternative-Schreibweisen

```
function mySound(){
    var MyMelody:HTMLAudioElement = new Audio(mybeat [index]);
    MyMelody.play();
    index += 1;
    if (index>11) index=0;
    console.log(mybeat [index]);
}
```

Bedingungen Beispiel Kurz-Schreibweisen

```
if(var1 == 10) console.log('bedingung erfüllt anweisung 1');
```

Nur mit einer Anweisung möglich

```
if(var1 == 12) console.log('bedingung erfüllt anweisung 1'); console.log('eine andere Anweisung');
```

```
(var1 == 10) ? console.log('bedingung erfüllt') : console.log('bedingung nicht erfüllt');
```

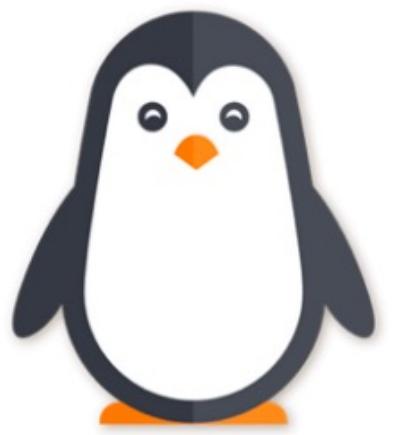
Entwicklung Interaktiver Anwendungen

Schleifen – Einführung

Prof. Dr. Gabriel Rausch

Viele Elemente nacheinander manipulieren?

Pingu and Friends



Pingu 0

Pingu 1

Pingu 2

Pingu 3

Pingu 4

Pingu 5

Pingu 6

Pingu 7



Pingu 8

Pingu 9

Pingu 10

Pingu 11

Pingu 12

Pingu 13

Pingu 14

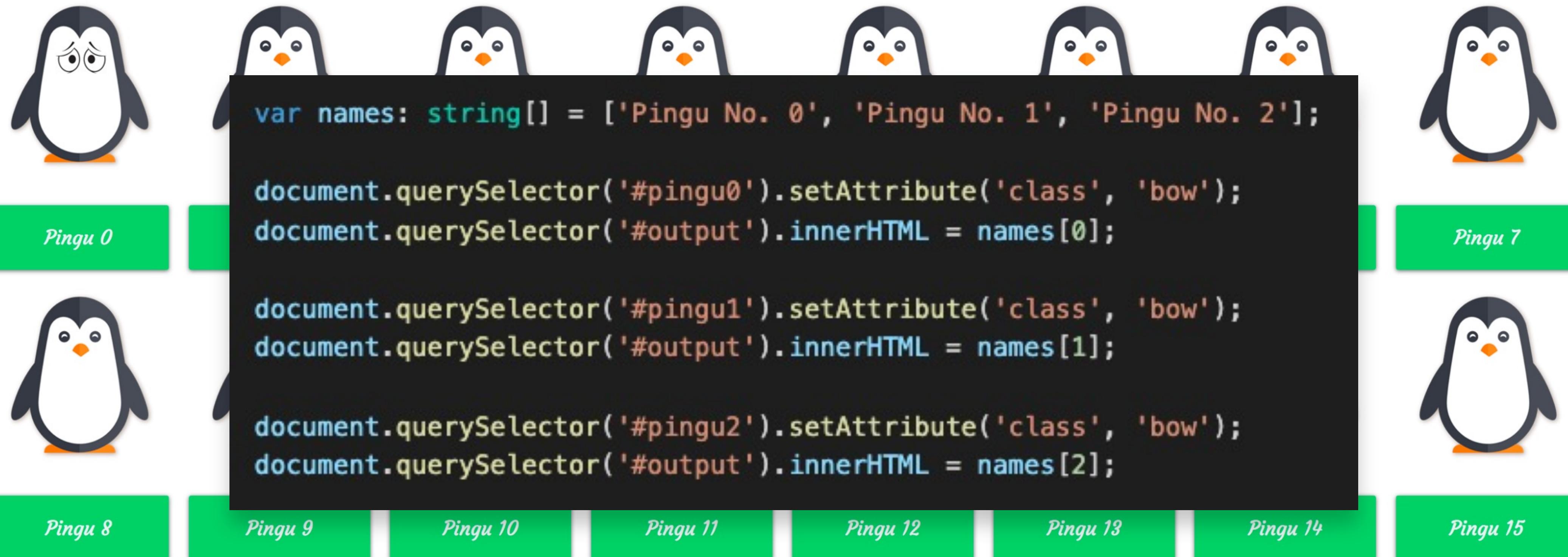
Pingu 15

Hallo ich bin **Pingu No. 0**

Hallo ich bin **Pingu No. 1**

Hallo ich bin **Pingu No. 2**

Viele Elemente nacheinander manipulieren? Bad practice



Entwicklung Interaktiver Anwendungen

While-Schleifen

Prof. Dr. Gabriel Rausch

Kopfgesteuerte Schleifen

while-Schleife

```
while ( Bedingung ) {  
}
```

```
var i: number = 0;  
var penguinsTotal: number = 15;  
  
while (i <= penguinsTotal) {  
    console.log("Hallo, ich bin Pingu Numero " + i);  
    i++;  
}
```

So lange die Bedingung erfüllt ist
werden die Anweisungen im Schleifen-Body wiederholend ausgeführt

Fußgesteuerte Schleifen

do-while-Schleife

Eine do..while-Schleife
durchläuft mindestens einmal
den Loop.

```
var i: number = 0;
var penguinsTotal: number = 15;

do {
    console.log("Hallo, ich bin Pingu Numero " + i);
    i++;
} while (i <= penguinsTotal)
```

Entwicklung Interaktiver Anwendungen

Zählschleifen / For-Schleifen

Prof. Dr. Gabriel Rausch

Zählschleifen

for-Schleife

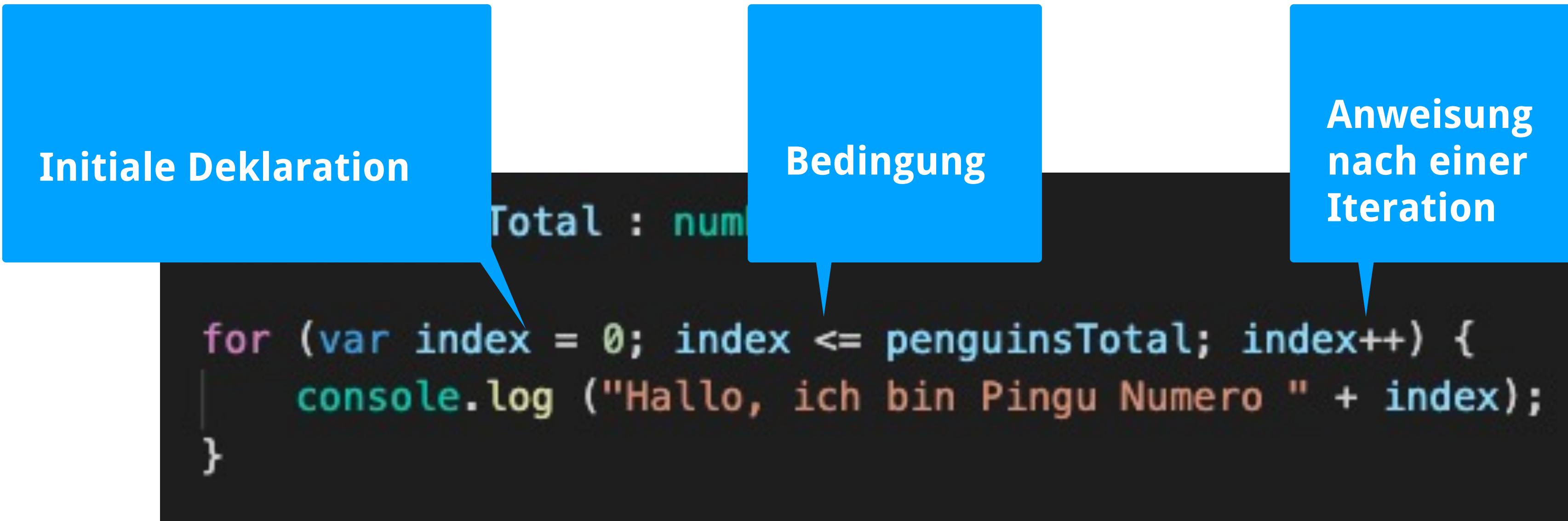
```
for ( Statements ) { }
```

```
var penguinsTotal : number = 15;

for (var index = 0; index <= penguinsTotal; index++) {
    console.log ("Hallo, ich bin Pingu Numero " + index);
}
```

Zählschleifen

for-Schleife



Statement 1 setzt eine Variable bevor die Schleife beginnt (`var index = 0`).

Statement 2 legt die Abbruchbedingung fest, wie lange die Schleife laufen soll (`index <= variable`)

Statement 3 verändert den Wert der Variable jedes mal, wenn der Code-Block in der Schleife ausgeführt wurde (`index++`).

Entwicklung Interaktiver Anwendungen

Schleifen – Zusammenfassung + Beispiele

Prof. Dr. Gabriel Rausch

Beispiel

Klassen und Text manipulieren



```
var names: string[] = ['Pingu No. 0', 'Pingu No. 1', 'Pingu No. 2'];

document.querySelector('#pingu0').setAttribute('class', 'bow');
document.querySelector('#output').innerHTML = names[0];
```

```
document.querySelector('#pingu1').setAttribute('class', 'bow');
document.querySelector('#output').innerHTML = names[1];

document.querySelector('#pingu2').setAttribute('class', 'bow');
document.querySelector('#output').innerHTML = names[2];
```

Pingu 8

Pingu 9

Pingu 10

Pingu 11

Pingu 12

Pingu 13

Pingu 14

Pingu 15

```
var penguinsTotal : number = 15;

for (var i = 0; i <= penguinsTotal; i++) {
    document.querySelector('#pingu' + i).setAttribute('class', 'bow');
    document.querySelector('#output').innerHTML = names[i];
}
```

```
var penguinsTotal : number = 15;
var outputElement: HTMLElement = document.querySelector('#output');

for (var i = 0; i <= penguinsTotal; i++) {
    document.querySelector('#pingu' + i).setAttribute('class', 'bow');
    outputElement.innerHTML = names[i];
}
```



Vergleich Schleifen

```
// While Schleife  
  
var i: number = 0;  
  
while (i<30) {  
    // do sth  
    i++;  
}
```

```
// Do-While Schleife  
  
var i: number = 0;  
  
do {  
    // do sth  
    i++;  
} while(i<30)
```

```
// for Schleife  
  
var i: number;  
  
for (i=0; i<30; i++){  
    // do sth  
}
```

Entwicklung Interaktiver Anwendungen

Linter und Hinter

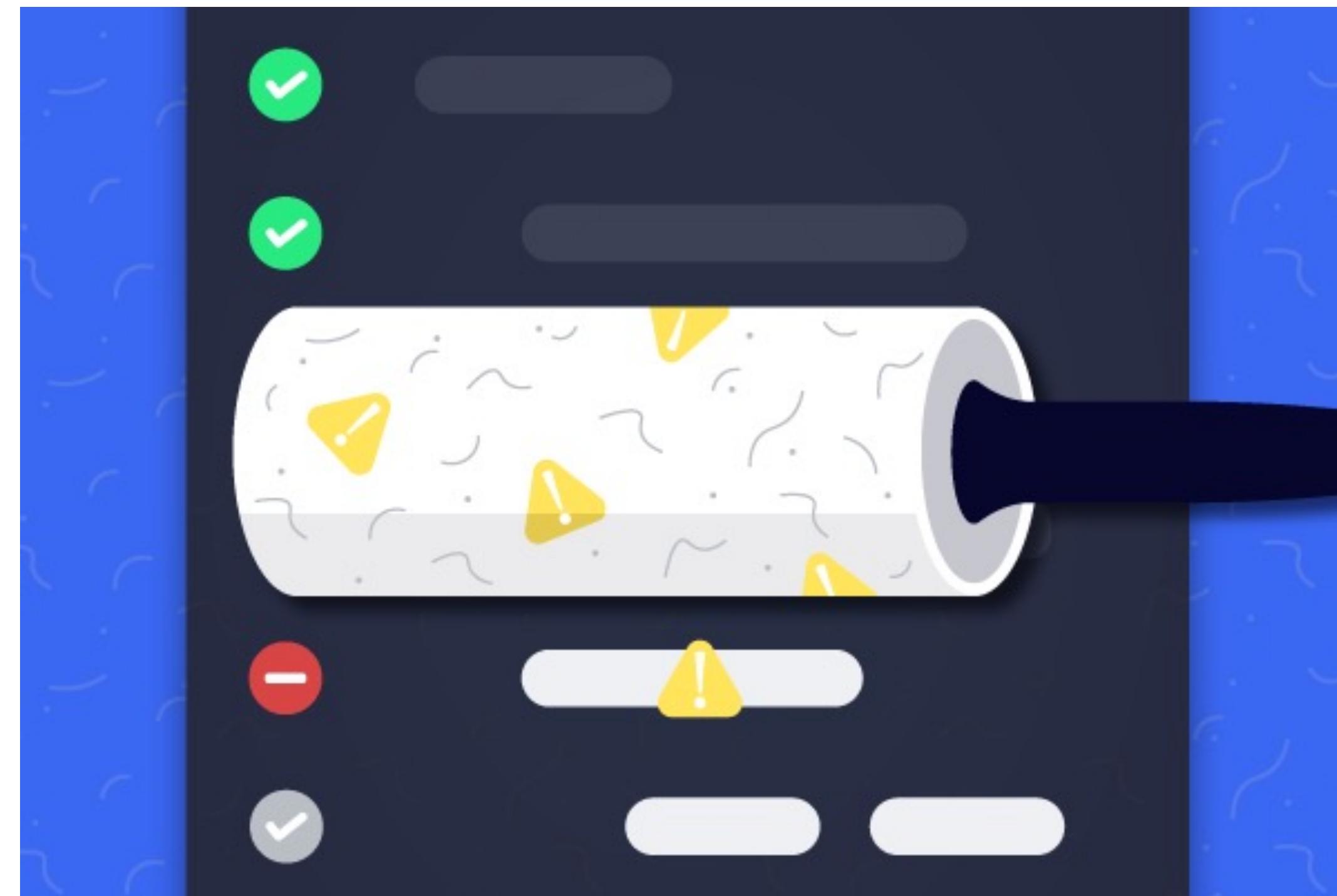
Prof. Dr. Gabriel Rausch

Linter

Ein Werkzeug für die Code-Analyse

“Linting is important to **reduce errors and improve the overall quality of your code**. Using lint tools can help you accelerate development and reduce costs by **finding errors earlier**.”

— Richard Bellairs



<https://www.perforce.com/blog/qac/why-linting-important-and-how-use-lint-tools>

Linter

Ein Werkzeug für die Code-Analyse

JSLint

JSHint

JSCS

Es existieren verschiedene Ausprägung eines Linters, hier im Kontext von JavaScript

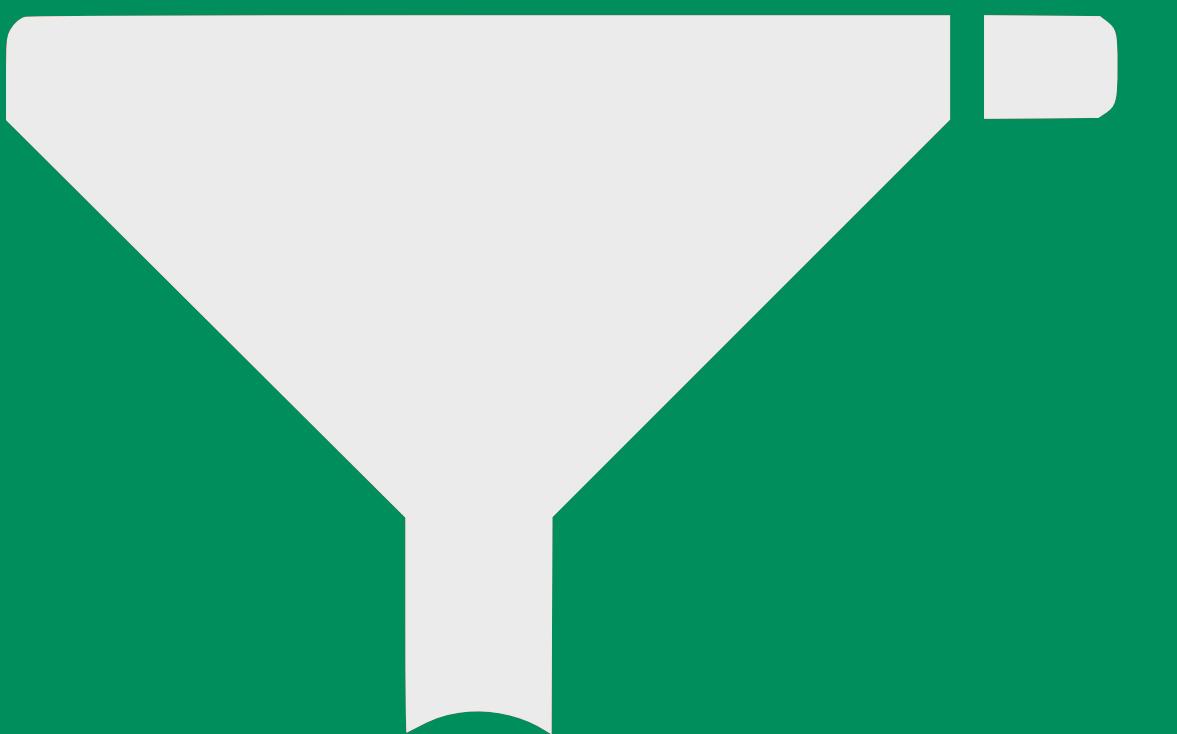
Linter für EIA

```
1  {
2      "rules": {
3          "align": [true, "parameters", "arguments", "statements"],
4          "ban": false,
5          "class-name": true,
6          "comment-format": [false, "check-space", "check-lowercase"],
7          "curly": false,
8          "eofline": false,
9          "forin": false,
10         "indent": [true, "spaces", 2],
11         "interface-name": false,
12         "jsdoc-format": false,
13         "label-position": true,
14         "label-undefined": true,
15         "max-line-length": [false, 140],
16         "member-ordering": [true, {"order": "fields-first"}],
17         "no-any": true,
18         "no-arg": true,
19         "no-bitwise": false,
20         "no-console": [false, "debug", "info", "time", "timeEnd", "trace"],
21         "no-construct": true,
22         "no-constructor-vars": true,
23         "no-debugger": false,
24         "no-duplicate-key": true,
25         "no-duplicate-variable": true,
26         "no-empty": true,
27         "no-eval": true,
28         "no-string-literal": false,
29         "no-switch-case-fall-through": false,
30         "trailing-comma": [true, {
31             "singleline": "never",
32             "multiline": "never"
33         }],
34         "no-trailing-whitespace": false,
35         "no-unreachable": true,
36         "no-unused-expression": true,
37         "no-unused-variable": true,
38         "no-use-before-declare": true,
39         "no-var-requires": false,
40         "no-conditional-assignment": true,
41         "no-shadowed-variable": false,
42         "one-line": [true, "check-open-brace", "check-catch", "check-whitespace"],
43         "quotemark": [true, "double"],
44         "radix": false,
```

Entwicklung Interaktiver Anwendungen

Take Aways

Prof. Dr. Gabriel Rausch



Bedingungen sind Kontrollstrukturen, die Anweisungsblöcke enthalten.

Die Ausführung der in den Blöcken enthaltenen Anweisungen kann durch **Regeln/Bedingungen kontrolliert** werden.

Die **Kontrollstrukturen** können in unterschiedlicher Form dargestellt werden:
bedingte Anweisung (Wenn-Dann-Bedingungen),
einfache Verzweigung (Entweder-Oder) oder
mehrfache Verzweigungen.

Eine besondere Form der mehrfachen Verzweigung bildet die Fallunterscheidung mit einer **Switch-Case** ab.

Schleifen ermöglichen das **iterative** Durchführen eines **Anweisungsblocks**.

Es gibt verschiedene **Schleifen-Typen**,
die grundlegendsten sind:
while, do-while und for

Ein **Linter** ist ein Werkzeug in der Softwareentwicklung das nach einem **Regelwerk** eine **Code-Analyse** ausführt, um frühzeitig auf Fehler hinzuweisen.