

Javascript Grundlagen

Repetition 1



Wieviel wisst ihr noch?



Wo nachschlagen?



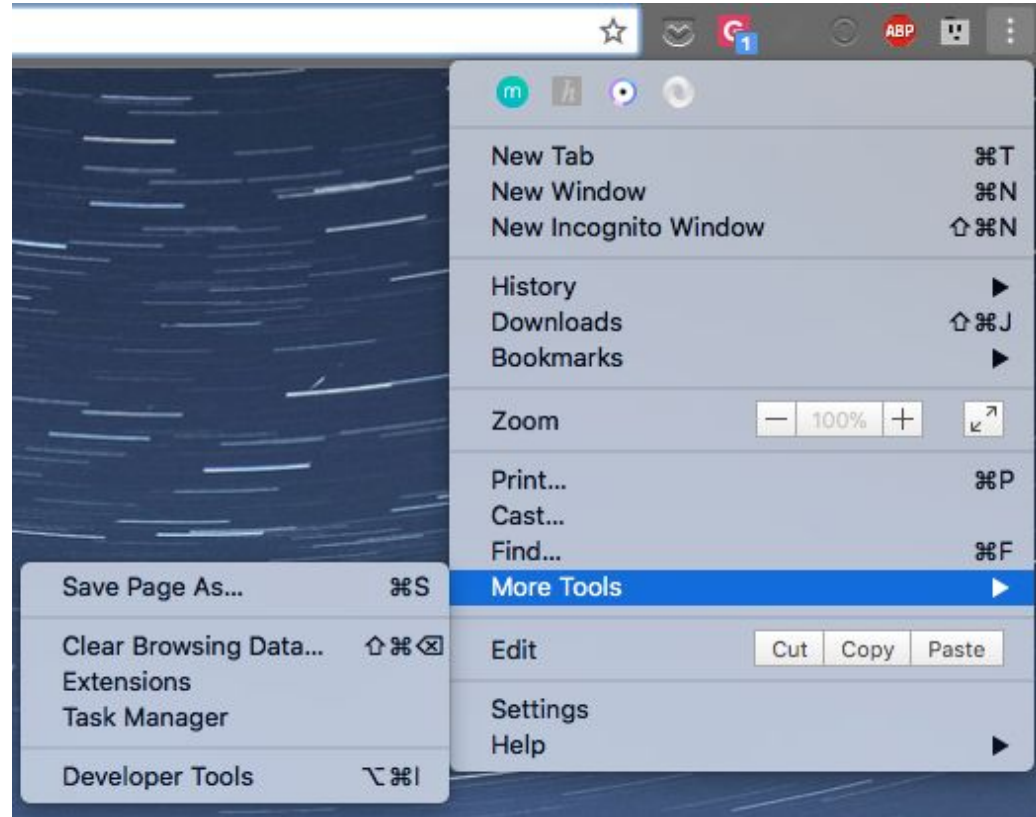
MDN

<https://developer.mozilla.org>

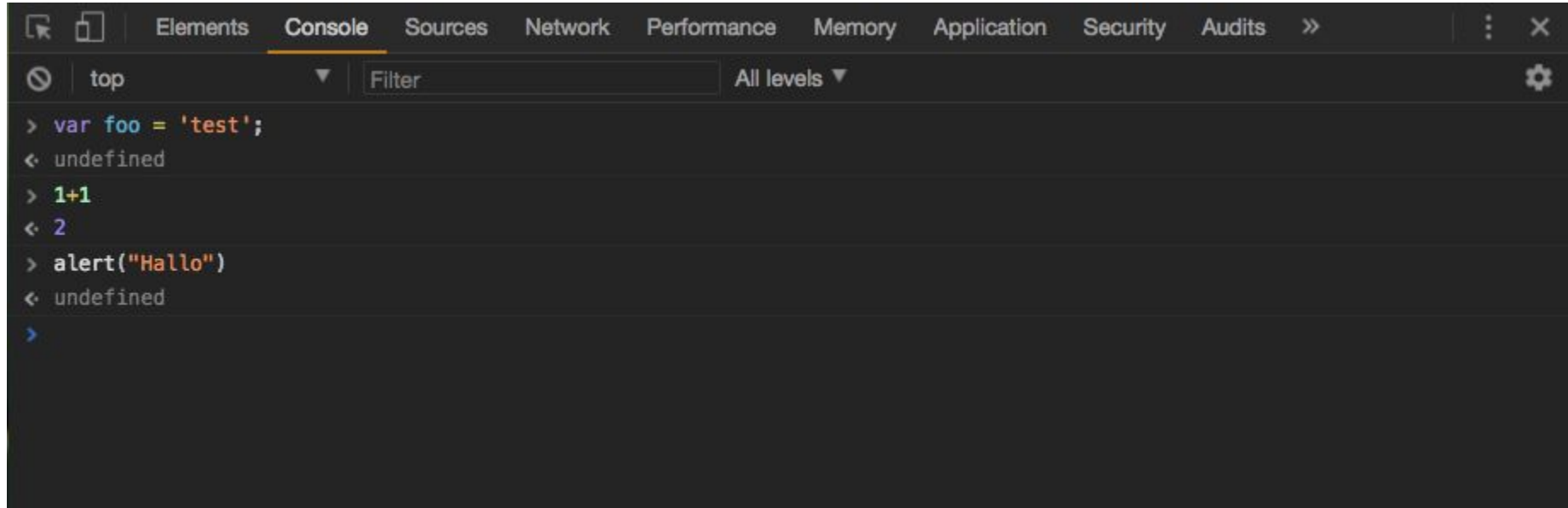


Die Konsole

- Der beste Freund des Javascript Entwicklers
- Gibts in schwarz und weiss
- Zeigt Javascript Fehler an
- Interaktive Javascript Eingabe
- Gut für kleine Tests
- Debugging



Die Konsole



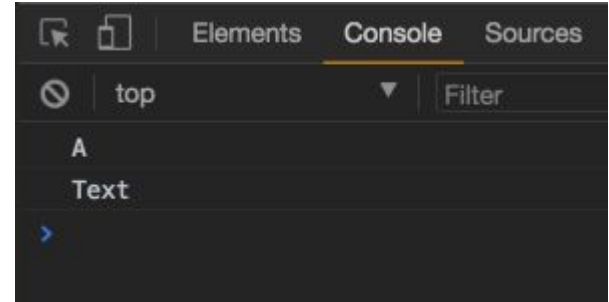
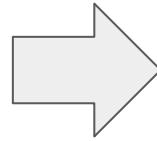
The screenshot shows the Chrome DevTools Console with the 'Console' tab selected. The interface includes a toolbar with icons for opening the console, copying, and a search icon. Below the toolbar, there's a filter dropdown set to 'top', a 'Filter' input field, and a log level selector set to 'All levels'. The console log contains the following entries:

```
> var foo = 'test';  
← undefined  
  
> 1+1  
← 2  
  
> alert("Hallo")  
← undefined  
  
>
```



Ausgabe auf der Konsole

```
var variable = "A";  
console.log(variable);  
console.log("Text");
```



Variablen

Variablen in Javascript müssen vorher mit **var** definiert werden

```
var VARIABLENNAME = WERT;
```



Variablen-Namen

- Können Buchstaben, Zahlen, Underscores und Dollar-Zeichen enthalten
- Müssen mit einem Buchstaben beginnen
- Können aber in spezifischen Fällen auch mit einem \$ oder _ beginnen
- Sind Case Sensitive (x ist nicht gleich X)
- Reservierte Wörter können nicht als Variablen-Namen verwendet werden

```
var 123 = "Das geht nicht";      // Funktioniert nicht
var b123 = "Das geht";          // Funktioniert
var $variable = "Das geht auch"; // Funktioniert
var if = "Das geht aber nicht";  // Funktioniert nicht
```



Datentypen

- Javascript ist eine dynamische Programmiersprache
- Variablen Typ wird automatisch bestimmt

```
var foo = 42;    // foo is now a Number
```

```
var foo = "bar"; // foo is now a String
```

```
var foo = true;  // foo is now a Boolean
```



Datentypen

Boolean	true / false
Null	null
Undefined	undefined
Number	12, 1.437,
String	"das ist ein String"
Array	[1, 2, 3, 4]
Object	{ key: value }

Welcher Typ ist meine Variable?

`typeof VARNAME`

=> "object", "number", string, "boolean", "undefined"

```
var foo = 42;  
typeof foo;  
=> "number"
```



Arithmetische Operatoren

- + Addition
- Subtraktion
- * Multiplikation
- / Division
- % Modulo
- ++ Inkrement (um 1 erhöhen)
- Dekrement (um 1 senken)

Arithmetische Operatoren

$1 + 1$	2	$5 * 5$	25
$1 - 1$	0	$10 / 2$	5
'1' + 1	"11"	$5 \% 4$	1
1 + '1'	"11"		
$'1' - 1$	0		
$'1' - '1'$	0		



Das Math Objekt

Bietet viele mathematische Funktionen und Konstanten wie Pi, Sinus, Quadrieren, Wurzel und und und...

Math.PI

Pi

Math.pow(BASIS, EXPONENT)

Potenzieren

Math.sqrt(Zahl)

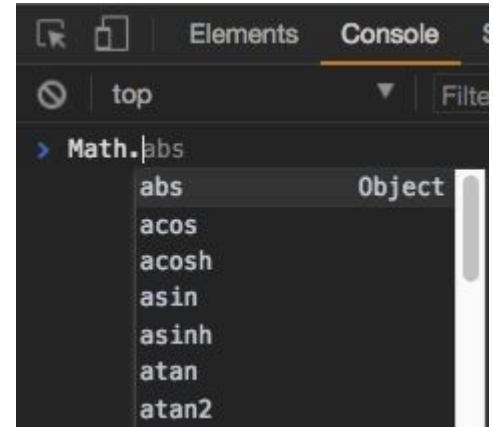
Quadratwurzel

Math.random();

Zufallszahl

Math.floor(Zahl)

Abrunden



Vergleiche

> grösser >= grösser oder gleich

< kleiner <= kleiner oder gleich

== gleich (Datentyp wird angepasst)

=== gleich (Datentyp wird nicht angepasst)

!= ungleich (Datentyp wird angepasst)

!== ungleich (Datentyp wird nicht angepasst)





Was ist **gleich**? (a == b)

1 == "1"

1 == true

false == []

null == undefined

0 == ""

"0" == false

true

true

true

true

true

true

[illegible]

(a === b)

1 == "1"

false

1 == true

false

false == []

false

null == undefined

false

0 == ""

false

"0" == false

false

[illegible]

If-Bedingungen

Codeabschnitte nur unter gewissen Umständen ausführen

```
if (VERGLEICH) {  
    // VERGLEICH wahr  
}
```

```
if (VERGLEICH) {  
    // VERGLEICH wahr  
} else {  
    // nicht wahr  
}
```

```
if (VERGLEICH) {  
    // Wenn VERGLEICH_1 wahr  
} else if (VERGLEICH) {  
    // Wenn VERGLEICH_2 wahr  
} else if (VERGLEICH) {  
    // Wenn VERGLEICH_3 wahr  
} else {  
    // Wenn alle nicht wahr  
}
```



Konditionen mit AND / OR

```
if (VERGLEICH || VERGLEICH2 || VERGLEICH2) {  
    // VERGLEICH ODER VERGLEICH2 ODER VERGLEICH3  
}
```

```
if (VERGLEICH && VERGLEICH2 && VERGLEICH2) {  
    // VERGLEICH UND VERGLEICH2 UND VERGLEICH3  
}
```

```
if (VERGLEICH && (VERGLEICH2 || VERGLEICH2)) {  
    // VERGLEICH UND VERGLEICH2 ODER VERGLEICH3  
}
```



Was ist wahr (true)?

true	<input checked="" type="checkbox"/>	if (true) { /* executes */ }
false	<input type="checkbox"/>	if (false) { /* does not execute */ }
1	<input checked="" type="checkbox"/>	if (1) { /* executes */ }
0	<input type="checkbox"/>	if (0) { /* does not execute */ }
-1	<input checked="" type="checkbox"/>	if (-1) { /* executes */ }
"true"	<input checked="" type="checkbox"/>	if ("true") { /* executes */ }
"false"	<input checked="" type="checkbox"/>	if ("false") { /* executes */ }
"1"	<input checked="" type="checkbox"/>	if ("1") { /* executes */ }
"0"	<input checked="" type="checkbox"/>	if ("0") { /* executes */ }
"-1"	<input checked="" type="checkbox"/>	if ("-1") { /* executes */ }
""	<input type="checkbox"/>	if ("") { /* does not execute */ }

null	<input type="checkbox"/>	if (null) { /* does not execute */ }
undefined	<input type="checkbox"/>	if (undefined) { /* does not execute */ }
Infinity	<input checked="" type="checkbox"/>	if (Infinity) { /* executes */ }
-Infinity	<input checked="" type="checkbox"/>	if (-Infinity) { /* executes */ }
[]	<input checked="" type="checkbox"/>	if ([]) { /* executes */ }
{}	<input checked="" type="checkbox"/>	if ({}) { /* executes */ }
[[]]	<input checked="" type="checkbox"/>	if ([[]]) { /* executes */ }
[0]	<input checked="" type="checkbox"/>	if ([0]) { /* executes */ }
[1]	<input checked="" type="checkbox"/>	if ([1]) { /* executes */ }
NaN	<input type="checkbox"/>	if (NaN) { /* does not execute */ }



Identisch ist besser als gleich

```
if (VARIABLE) {  
    // Wenn wahr  
}
```

entspricht

```
if (VARIABLE == true) {  
    // Wenn wahr  
}
```

besser

```
if (VARIABLE === true) {  
    // Wenn wahr  
}
```



Übungs Session #1

Kopiert aus dem Google Drive die Datei **lektion1.zip** (Module/Javascript/) in euer **htdocs** Verzeichnis in einen eigenen Ordner. Öffnet die Dateien in PHPStorm und macht die Aufgaben. Zum testen ob es funktioniert, müsst ihr die Datei im Browser öffnen.

- uebung_1_0.html (direkt im Browser öffnen)
- uebung_1_1.html (in PHPStorm öffnen)
- uebung_1_2.html ...
- uebung_1_3.html ...



Schleifen



For-Schleife

Code N-mal ausführen bis Bedingung erfüllt

```
for (VARIABLE; BEDINGUNG; NACH_SCHLAUFE) {  
    // Wird ausgeführt solange BEDINGUNG wahr  
}
```



For-Schleife

Code N-mal ausführen bis Bedingung erfüllt

```
for (var i = 0; i <= 10; i++) {  
    console.log('Durchgang nummer ' + i);  
}
```

Durchgang nummer 0
Durchgang nummer 1
Durchgang nummer 2
Durchgang nummer 3
...
Durchgang nummer 8
Durchgang nummer 9
Durchgang nummer 10



While-Schleife

Code solange ausführen bis Bedingung erfüllt

Laufbedingung muss in Schleife geändert werden, sonst infinite Loop

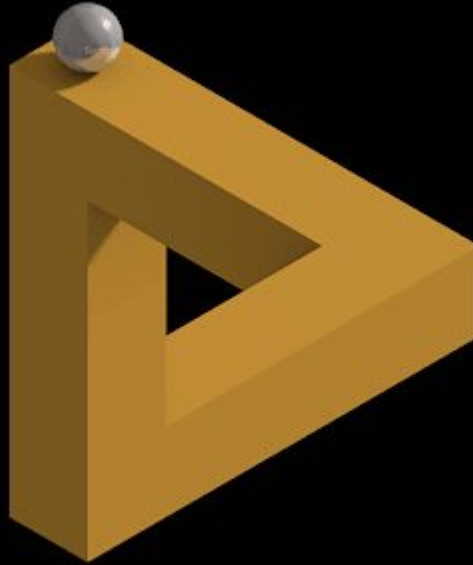
```
var weiterlaufen = true;
while (weiterlaufen) {
    console.log('In der Schleife');
    weiterlaufen = false;
}
```

In der Schleife

```
var weiterlaufen = true;
while (weiterlaufen) {
    console.log('In der Schleife');
}
```

In der Schleife
In der Schleife
In der Schleife
In der Schleife
In der Schleife
In der Schleife

Objekte



Objekte

Attribut DOPPELPUNKT Wert

```
var objekt = {  
  name: "Müller",  
  vorname: "Hans",  
  alter: 23  
};
```

```
objekt.name      => Müller  
objekt.vorname   => Hans  
objekt["name"]   => Müller
```



Arrays



Arrays

Arrays können alle Datentypen von Javascript enthalten

```
var einArrayMitZahlen = [1, 2, 3, 4];  
var arrayMitStrings = ["a", "b", "c", "d", "e"];  
var arrayMitObjekten = [{a: 1}, {a: 2}, {a: 3}];  
var arrayMitBooleans = [true, true, false, true];  
var arrayMitArrays = [  
    [1, 2, 3],  
    [4, 5, 6],  
    [7, 8, 9]  
];
```

- Können beliebig gross sein
- Zugriff erfolgt per Index
- Index beginnt bei 0
- Grösse kann mit
ARRAYNAME.length
abgefragt werden



1-Dimensionale Arrays



```
var eierKarton = ["A", "B", "C", "D"];
```

```
console.log(eierKarton[0]);
```

=> A

```
console.log(eierKarton[1]);
```

=> B

```
console.log(eierKarton[2]);
```

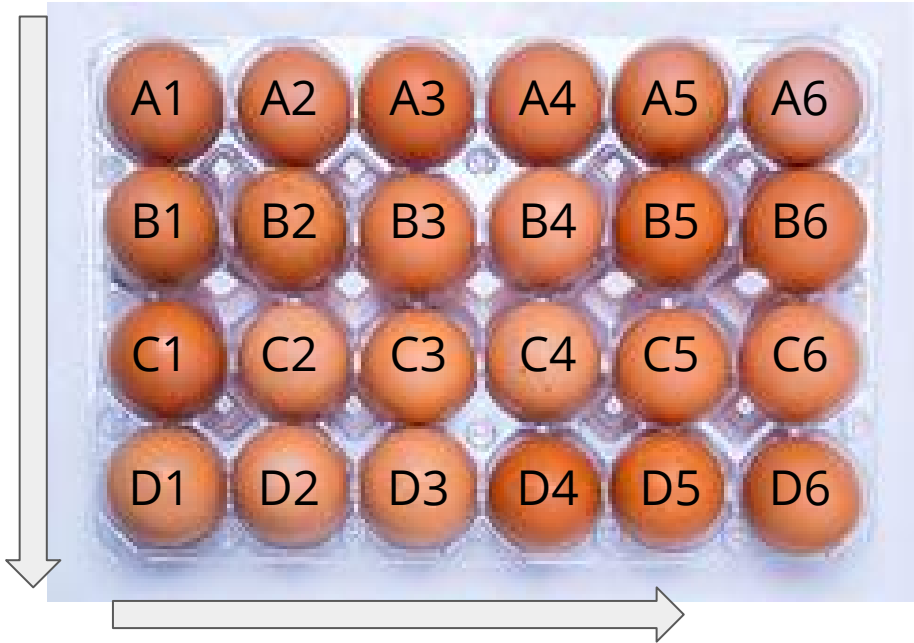
=> C

```
console.log(eierKarton[3]);
```

=> D



2-Dimensionale Arrays



```
var grosserKarton = [ // Haupt Array
  ["A1", "A2", "A3", "A4", "A5", "A6"], // Array A
  ["B1", "B2", "B3", "B4", "B5", "B6"], // Array B
  ["C1", "C2", "C3", "C4", "C5", "C6"], // Array C
  ["D1", "D2", "D3", "D4", "D5", "D6"], // Array D
]
```

```
console.log(grosserKarton[0][0]);    => A1
console.log(grosserKarton[1][5]);    => B6
console.log(grosserKarton[2][2]);    => C3
console.log(grosserKarton[3][5]);    => D6
```



3-Dimensionale Arrays



```
var matrixKarton = [  
  [ "AA1", "AA2"], [ "AB1", "AB2"], [ "AC1", "AC2"] ],  
  [ [ "BA1", "BA2"], [ "BB1", "BB2"], [ "BC1", "BC2"] ],  
  [ [ "CA1", "CA2"], [ "CB1", "CB2"], [ "CC1", "CC2"] ],  
  [ [ "DA1", "DA2"], [ "DB1", "DB2"], [ "DC1", "DC2"] ],  
]
```

```
console.log(matrixKarton[0][0][0]);  
console.log(matrixKarton[0][0][1]);  
console.log(matrixKarton[2][1][1]);  
console.log(matrixKarton[3][2][0]);
```

=> AA1

=> AA2

=> CB2

=> DC1





Arrays mit Schleifen durchgehen

Arrays mit Schleifen durchgehen

```
var testArray = ["A", "B", "C", "D", "E"];

for (var i = 0; i < testArray.length; i++) {
    console.log(testArray[i]);
}
```

- Schleife läuft so lange wie Array lang ist
- i wird als Index verwendet
- i wird erhöht
- Danach wird wiederholt



Übungs Session #2

Macht die Übungen mithilfe von PhpStorm und eurem Browser

- uebung_2_0.html
- uebung_2_1.html
- uebung_2_2.html
- uebung_2_3.html

