Funktionen in JavaScript

Eine Funktion enthält gebündelten Code, der sich in dieser Form wiederverwenden lässt. Es können ganze Programmteile aufgenommen werden. Man muss sie nur einmal definieren und kann sie beliebig oft aufrufen, sprich benutzen. Man spart sich damit **Arbeit**, wenn man bestimmte Arbeitsschritte öfter benötigt.

- Mithilfe von Funktionen kann man denselben Code von mehreren Stellen des Programms aus aufrufen.
- Funktionen sind sehr nützlich, wenn sich eine Berechnung oder Aktion innerhalb eines Programms ständig wiederholt.
- Bestimmte Funktionen sind schon bekannt, wie z.B. „prompt“ oder „alert“. Nun werden wir eigene Funktionen erstellen.

Dies erfolgt mit

- function name()
- Danach werden in geschwungenen Klammern die Anweisungen geschrieben, die bei Funktionsaufruf ausgeführt werden sollen.
- Die Variable kann später mit name() aufgerufen werden.

Definiert wird eine Funktion meist im head des Dokuments.

- Die Definition beginnt mit dem Schlüsselwort function.
- Anschließend folgt der Name der Funktion, für den dieselben Regeln gelten wie für die Namen von Variablen.
- Danach stehen runde Klammern, hier noch ohne Inhalt. Es folgt ein Block von Anweisungen, immer mit Blockklammern.

Syntax zum Erstellen einer Funktion

```
function name () {
        document.write(„Mach was“);
}
```

Beispiel: Das ist eine Funktion, die bei jedem Aufruf genau dasselbe Ergebnis erzeugt.
Speichere unter „funktion.html“.

Sie wird hier im <head> erstellt und im <body> aufgerufen:

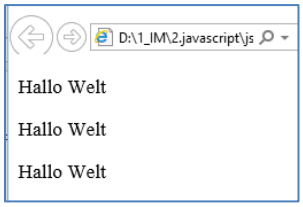
```
<script>
function hallo()
{ document.write("<p> Hallo Welt</p>"); }
</script>
</head>

<body>

<script>
hallo();
hallo();
hallo();

</script>
</body>
```

Ergebnis:



Beachte: Nach der geschweiften Klammer einer Funktion wird **grundsätzlich kein Semikolon** gesetzt.

Bei Funktionen muss zwischen **Definition und Aufruf** unterschieden werden.

- Der Browser liest die Definition einer Funktion und weiß dann, wie sie arbeiten soll. **Sie wird allerdings erst ausgeführt, wenn sie aufgerufen wird.** D.h. sie macht nach der Definition einmal gar nichts, sondern wartet nur, dass sie aufgerufen wird.

- **Aufrufen** kann man eine Funktion an jeder Stelle in einer Webseite. Einfach nur den Namen, ohne das Schlüsselwort „function“ und dahinter die leeren runden Klammern und dahinter ein Semikolon.

Der Aufruf erfolgt als Anweisung in einem Skriptbereich. Man ruft die Funktionen unmittelbar beim Laden der Webseite auf. Dazu muss man nur den Funktionsnamen samt Klammern und optionaler Werte für die Parameter (nicht die Parameterbezeichner) in einer Anweisung angeben.

Jede Funktion gibt einen Rückgabewert aus, der dann an einer anderen Stelle des Codes weiterverwendet werden kann.

Parameter (Argumente) an eine Funktion übergeben

Bei Bedarf lässt sich ein Parameter aber auch **mehrere Parameter** übergeben, getrennt durch Komma.

- Diese werden in der runden Klammer bei der Funktion angegeben.

Diese Parameter müssen angegeben werden, nämlich dort, wo die Funktion ausgeführt wird. Dort werden sie dann nicht mehr Parameter genannt, sondern „Argumente“.

Mithilfe von Argumenten kann man einer Funktion bestimmte Werte übergeben. Damit beeinflusst man, wie sich die aufgerufene Funktion verhält.

Argumente stehen immer zwischen den Klammern der Funktion, sowohl bei der Definition als auch beim Aufrufen der Funktion.

Beispiel: addieren

```
function calculate(number1, number2)        // hier stehen die Parameter
{
        document.write(number1 + number2);  // hier wird addiert und ausgegeben
}
calculate(5, 10);                           // hier stehen Werte = Argumente. Die Anzahl
                                            muss der Anzahl der Parameter
                                            entsprechen. Der erste Wert wird für den
```

ersten Parameter verwendet,…

Erstelle eine neue HTML-Datei in **Boilerplate-Code - das** bezeichnet **Standardcode**, der in vielen Programmen oder Dateien gleich oder sehr ähnlich ist. Er dient oft als **Grundgerüst**.
In VS-Code erstelle die Datei „funktion.html“ und drücke das Rufzeichen und danach die Enter-Taste.

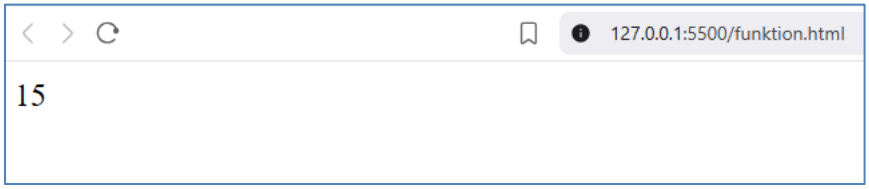
In den <head> die Funktion:

```
6       <title>Document</title>
7       <script>
8           function calculate(number1, number2)
9           {
10              document.write(number1 + number2);
11          }
12      </script>
13  </head>
```

Im <body> den Aufruf der Funktion mit 2 Argumenten:

```
13  </head>
14  <body>
15      <h2>Diese Funktion addiert 2 Werte zu folgendem Ergebnis:</h2>
16      <script>
17          calculate(5, 10);
18      </script>
19  </body>
20  </html>
```

Ergebnis im Browser:



Erklärung

```
function calculate(number1, number2)
{
```

Es handelt sich bei „number1 bzw. number2“ um eine neue Variable, wobei das Schlüsselwort „let“ hier nicht erlaubt ist. Der Name ist beliebig. Es handelt sich hier um eine „lokale Variable“, die ausschließlich innerhalb der Funktion vorhanden ist.

Zum Aufrufen einer Funktion, die ein Argument akzeptiert, schreibt man dessen gewünschten Wert in die Klammer hinter dem Funktionsnamen

- bei einem „string“ in Anführungszeichen
- bei Nummern, wie oben im Beispiel, ohne Anführungszeichen.

```
16      <script>
17          calculate(5, 10);
18      </script>
```

Bei den Parametern handelt es sich um Informationen, die beim Aufruf an die Funktion übermittelt werden. Die Funktion verarbeitet diese Informationen und erzeugt ein Ergebnis.

return

Das gleiche kann man auch mit „return“ erreichen. Hier ist die Ausgabe nicht in der Funktion festgelegt, sondern außerhalb.
Ohne return würde die Funktion zwar intern die Addition machen, aber nach außen gäbe es keinen Zugriff auf das Ergebnis.
Wenn keine explizite return-Anweisung vorhanden ist, gibt die Funktion **undefined** zurück.

```
6     <title>Document</title>
7     <script>
8         function calculate(number1, number2)
9         {
10            return number1 + number2;
11        }
12    </script>
13 </head>
```

```
14 <body>
15    <h2>Diese Funktion addiert 2 Werte zu folgendem Ergebnis:</h2>
16    <script>
17        document.write(calculate(5, 10));
18    </script>
19 </body>
```

Übung mit return

Rechne Euro in Dollar um. Kurs ist 1 € = 1,1 $

Ziel: Ausgabe

Wechselkurs

Ich erhalte 1100 Dollar

Hier wurde die Funktion und deren Aufruf in einen <script>-Bereich im <body> geschrieben:

```
18    </script>
19    <h3>Wechselkurs</h3>
20    <script>
21        function exchangeRate(amount) {
22            let rate = amount * 1.1; // Beispiel-Wechselkurs von Euro zu Dollar
23            return rate;
24        }
25        document.write(" Ich erhalte " + exchangeRate(1000) + " Dollar");
26    </script>
27 </body>
```

Beispiel mit einem String:

```
3   <head>
4   <meta charset="utf-8">
5   <title>Funktion mit Argument</title>
6   <script>
7   function sagHallo(argument) {
8           document.write("Hallo " + argument);
9           }
10  </script>
11  </head>
12  <body>
13  <script>
14  sagHallo("Josef");
15  </script>
16  </body>
```

https://www.youtube.com/watch?v=cAiSbkWvBQU ca. bei 33:10 Minuten

Beispiel mit Variablen UND direkt – überlege dir dieses Beispiel mit dem/der NachbarIn – nicht selbst nachbauen

Man sieht die Funktion „verbinden“ und die Ausgabe mittels „document.write“.
Hier werden wieder Variablen in der Funktion genutzt.
Innerhalb der runden Klammern werden zwei Variablennamen angegeben, hier „land“ und „stadt“.
Das ist eine eigene Funktion, an die zwei Zeichenketten übergeben werden. Die Funktion erstellt daraus einen Satz und gibt ihn aus.
Speichern unter „funktion_parameter.html“

```
<!doctype html>
<html>
<head>
<title>Funktion Parameter</title>
<script>
    function verbinden(land, stadt)
    {
    var satz = "Die Hauptstadt von " + land + " ist " + stadt + "<br>";
    document.write(satz);
    }
</script>
</head>

<body>
<script>
    //mit Variablen zuordnen (Variante 1)
    var a = "Frankreich", b = "Paris";
    verbinden(a, b);
    //direkt zuordnen (Variante 2)
    verbinden("Spanien", "Madrid");
</script>
</body>
</html>
```

Es wird eine Funktion mit dem Namen verbinden() definiert. Sie besitzt zwei Parameter mit den Bezeichnungen land und stadt, durch Komma voneinander getrennt. Die jeweils aktuellen Werte der beiden Parameter werden genutzt, um einen Satz zusammenzustellen und auszugeben.

Wird die Funktion unten aufgerufen, erhalten die oberen Variablennamen die Werte, die in der Klammer angegeben sind. **Es müssen genau so viel sein, aber natürlich nicht die gleichen**. Hier sind es zwei, nämlich einmal die Variablen a und b, und das andere Mal „Spanien“ und „Madrid“.

Die Funktion verbinden() wird zweimal aufgerufen:

- beim ersten Mal mit den Werten der beiden Variablen a und b,
- beim zweiten Mal mit zwei Zeichenketten, jeweils wiederum durch Komma voneinander getrennt.

Es können also sowohl Variablen als auch Werte übermittelt werden. Bei jedem Aufruf springt das Programm zur Funktion, übergibt die beiden Parameter in der gegebenen Reihenfolge an die Variablen land und stadt und führt den Inhalt der Funktion aus.

Ergebnis:

Einige Hinweise zu den Parametern: Dabei kann es sich um Zeichenketten, Zahlen oder Wahrheitswerte handeln. Es gibt aber auch Objekte oder Felder als Parameter.

Die Anzahl der Parameter einer Funktion sollte beim Aufruf mit der Anzahl der in der Definition erwarteten Parameter übereinstimmen. JavaScript bietet aber auch die Möglichkeit, eine beliebige Anzahl von Parametern zu übermitteln.

Gültigkeitsbereich von Variablen – global vs. lokal

Die Variablen, die außerhalb von Funktionen deklariert werden, sind im gesamten Programm gültig und bekannt, auch innerhalb von Funktionen. Sie haben einen globalen Gültigkeitsbereich. Man nennt sie auch globale Variable.
Die Variablen, die **innerhalb einer Funktion** deklariert werden, sind **nur innerhalb dieser Funktion gültig und bekannt**. Man nennt sie auch lokale Variable. Sie können mehrere Variable mit demselben Namen in verschiedenen Funktionen deklarieren, da jede dieser Variablen ihren eigenen lokalen Gültigkeitsbereich hat.