Funktionen in JavaScript

Eine Funktion enthält gebündelten Code, der sich in dieser Form wiederverwenden lässt. Es können ganze Programmteile aufgenommen werden.
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 NameFunktion (evtl. Variable).
- Danach werden in geschwungenen Klammern die Anweisungen geschrieben, die bei Funktionsaufruf ausgeführt werden sollen.
- Die Variable kann später mit NameFunktion() aufgerufen werden.

Außerdem kann man bestimmte Vorgänge, die man immer wieder benötigt, in Funktionen auslagern. 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.
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.

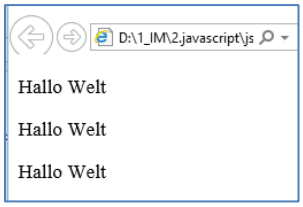
Syntax zum Erstellen einer Funktion

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

Beispiel: Das ist eine Funktion, die bei jedem Aufruf genau dasselbe Ergebnis erzeugt.
Speichere unter „2.funktion_einfach.html“

```
6  <script>
7  function hallo()
8  { document.write("<p> Hallo Welt</p>"); }
9  </script>
10 </head>
11
12 <body>
13
14 <script>
15 hallo();
16 hallo();
17 hallo();
18
19 </script>
20 </body>
21 </html>
```

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 entweder mittels sogenannter Eventhandler, mit einer Inline-Referenz oder einfach als Anweisung in einem Skriptbereich. Im letztgenannten Fall rufen wir 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.

Beispiel für das Aufrufen im <body>:

```
<body>
<script>
        function name ();
</script>
</body>
```

Jede Funktion gibt einen Rückgabewert aus, der dann an einer anderen Stelle des Codes weiterverwendet werden kann. In diesem Fall erhält man „undefined“, weil man lediglich den Code mit einer Anweisung ausgeben lässt.

Gerne wird eine Funktion wie eine Variable definiert:

```
var output = function(){
        document.write("Hallo Welt");
        }
```

```
1  <!doctype html>
2  <head>
3  <title>Funktion Einstieg</title>
4  <script>
5      var output = function(){
6      document.write("Hallo Welt");
7      };
8  </script>
9  </head>
10 <body>
11     <script>
12         output();
13     </script>
14 </body>
15 </html>
```

Speicher als „2.funktion.html“

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 natürlich 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);
            }
calculate(5, 6);                                // hier stehen Werte, Argumente
```

Beispiel:

Syntax:

```
function sagHallo (argument) {
        document.write(“Hallo “ + argument);
        }
```

Es handelt sich bei „argument“ um eine neue Variable, wobei das Schlüsselwort „var“ 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 (in Anführungszeichen).

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

```
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 (wie oben gezeigt)

Erstelle die Funktion „verbinden“ und der Ausgabe mittels „document.write“.
Hier wird eine weitere Möglichkeit genutzt, nämlich dass man mit Variablen arbeitet.
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 „2.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.

Übung: Verbessere das Beispiel, indem die Variablen „stadt“ und „land“ mit einer „prompt“-Anweisung vom Benutzer eingegeben werden können.
Speicher es unter 2.funktion_par2.html.

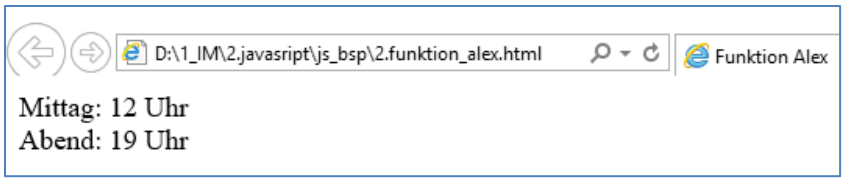
Gültigkeitsbereich von Variablen

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.

Lokale Variable werden **innerhalb von Funktionen** beziehungsweise vergleichbaren Konstruktionen explizit **mit var deklariert** und können dann nur dort benutzt werden. Beispiel: „var satz“ im nachfolgenden Beispiel bzw. „var x“ im „alex()“-Beispiel. Man kann mehrere Variable mit demselben Namen in verschiedenen Funktionen deklarieren, da jede dieser Variablen ihren eigenen lokalen Gültigkeitsbereich hat.
Gültigkeitsbereich von Variablen: Dazu gibt es im Gegensatz auch „**globale Variablen**“, die im gesamten Programm gültig und bekannt sind, auch innerhalb von Funktionen (siehe „var y“ im „alex()“ Beispiel, da „y“ im <body“ erzeugt wird). Beispielsweise sind im nachfolgenden Beispiel die beiden Variablen im <body> „var a und b“ solche globale Variablen.

Hausübung: function alex()

Erstelle im <head> eine Funktion namens „function alex()“ mit einer Variablen „var x = 19;“ und einem „document.write(“Abend: “ + x + “ “ + y + “
“);“
Im <body> soll die Variable „x“ den Wert „12“ erhalten und erst hier die zweite Variable „y“ den Wert „Uhr“ erhalten. Nun sollen mittels document.write ausgegeben werden (“Mittag: “ + x + “ “ + y + “
“). Erst danach soll die Funktion aus dem <head> mit „alex();“ angezeigt werden. Speichere unter „2.funktion_alex.html“.
Das Ergebnis soll so aussehen:



Erklärung:
In der Funktion alex() wird mit x=19 eine lokale Variable erzeugt, die die globale Variable aus dem <body> ausblendet. Die Variable y wird in beiden Ausgaben verwendet, da sie in der Funktion ja nicht lokal erzeugt bzw. abgeändert wird.

Funktionen mit Rückgabewert (return)

Man kann nicht nur Parameter an eine Funktion senden, sondern auch ein Ergebnis von einer Funktion zurückerhalten. Dieses Ergebnis nennt man den **Rückgabewert** einer Funktion. Dabei kann es sich um eine Zeichenkette, eine Zahl, einen Wahrheitswert oder ein beliebiges anderes Objekt handeln. Man kann den Rückgabewert unmittelbar ausgeben, speichern, um ihn im weiteren Verlauf des Programms zu nutzen z.B. mit anderem Code kombinieren.

Rückgabewert einer Funktion mit „return“
Um einen Funktionswert zurückzuliefern verwendet man das Schlüsselwort „return“ gefolgt von dem gewünschten Rückgabewert.
Die Anweisung return sollte immer die letzte Anweisung in einer Funktionsimplementierung sein. Wenn der Interpreter diese Anweisung vorfindet, verlässt er unmittelbar die Funktion und macht mit dem Code hinter dem Funktionsaufruf weiter. Gegebenenfalls nachfolgend in der Funktionsimplementierung notierte Anweisungen werden nicht mehr ausgeführt. Diese wären dann sogenannter unerreichbarer Code (unreachable code) und solcher ist immer (!) ein logischer Fehler im Programm oder Skript.

Übung: 2.funktion_verdoppeln.html
Erstelle eine Funktion „function verdoppeln(zahl)“, die das Argument „zahl“ entgegennimmt und als Ergebnis „zahl * 2“ zurückliefert. Der Rückgabewert dieser Funktion ist also immer doppelt so groß wie ihr übergebener Wert.

```
<head>
<meta charset="utf-8">
<title>Funktion Verdoppeln</title>
<script>
function verdoppeln (zahl) {
    var ergebnis = zahl * 2;
    return ergebnis;
}
</script>
</head>
<body>
<script>
var zahl=3;
var Lösung = verdoppeln(zahl);
document.write("Die Zahl " + zahl + " wurde verdoppelt: " + Lösung);
</script>
</body>
```

Im <body> wird die Funktion aufgerufen und mit document.write() ausgegeben.

Ergebnis:

Übung:

In diesem Programm gibt es eine Funktion, die zwei Zahlen als Parameter benötigt. Sie liefert als Rückgabewert die Summe der beiden Zahlen, denn Funktionen können zwar mehrere Argumente entgegennehmen, aber immer nur einen Wert zurückliefern.

```
<!doctype html>
<head>
<title>Funktion Parameter Rückgabe</title>
<script>
function summe(a, b)
    {
    var ergebnis = a + b;
    return ergebnis;
    }
</script>
</head>

<body>
<script>
    var x = 3, y = 5;
    z = summe(x, y);
    document.write("Summe: " + z + "<br>");
    document.write("Summe: " + summe(14, 5));
</script>

</body>
</html>
```

Speichere unter „2.funktion_parameter_rückgabe.html“

Die Funktion summe() erwartet zwei Parameter. Diese beiden Parameter werden innerhalb der Funktion addiert. In der nächsten Anweisung steht das Schlüsselwort return.

a)Beim ersten Aufruf der Funktion summe() werden die aktuellen Werte von x und y an die Funktion übergeben. Die Summe wird zurückgeliefert und in der Variablen z gespeichert. Diese wird anschließend ausgegeben.

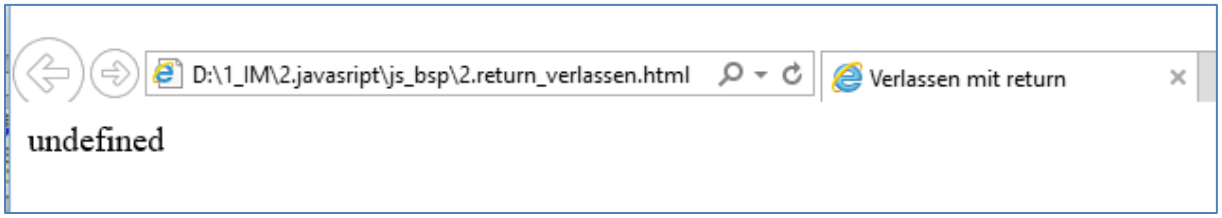
b)Zur Berechnung und Ausgabe der zweiten Summe wird die Methode document.write() aufgerufen. Innerhalb des Aufrufs wird wiederum die Funktion summe() aufgerufen. Es werden zwei Zahlenwerte übergeben, nämlich hier 14 und 5. Der Rückgabewert wird direkt in die Ausgabe eingebaut.

Eine Funktion vorzeitig verlassen mit „return“

Sobald der JavaScript-Interpreter in einer Funktion auf das Schlüsselwort „return“ trifft, verlässt er die Funktion, auch wenn darin noch weiterer Code folgt.
Zum Beispiel wird „return“ immer wieder verwendet, um eine Funktion vorzeitig zu verlassen, wenn ungültige Argumente übergeben wurden.
Beispiel:
Die nachfolgende Funktion liefert den 5 Buchstaben eines Wortes. Aber wenn das Wort kürzer als 5 Buchstaben ist, soll die Funktion verlassen werden.

```
3   <head>
4   <meta charset="utf-8">
5   <title>Verlassen mit return</title>
6   <script>
7   function fuenfteBuchstabe (name){
8       if(name.length < 5) {
9           return;
10      }
11      var wort = "Der fünfte Buchstabe lautet " + name[4];
12      return wort;
13  }
14  </script>
15  </head>
16  <body>
17  <script>
18  var name = "Herbert";
19  var Lösung = fuenfteBuchstabe(name);
20  document.write(Lösung);
21  </script>
22  </body>
```

Die Funktion „fuenfterBuchstabe“ liefert einen String zurück, in dem sie den fünften Buchstaben des Namens ausgibt. Ist der Name kürzer als fünf Buchstaben (name.length < 5) dann wird die Funktion über return direkt verlassen. Das Ergebnis wäre „undefined“. Dass bedeutet, dass der Interpreter nicht zur zweiten Anweisung gelangt, die den fünften Buchstaben ausgeben sollte.
In der Zeile 8 wird überprüft, ob die Länge des Namens kleiner als 5 ist.
In der Zeile 18 wird mit „var name = “ der Name festgelegt.
Übung b) gib den Namen „Max“ statt „Herbert“ ein und betrachte das Ergebnis im Browser.



Übung: Würfel mit Zufallszahl

In der Funktion „wuerfel“ wird ein Zufallswert zwischen 0 und 6 berechnet, auf eine Ganzzahl gerundet und anschließend per „return“ zurückgegeben. Dieser Wert wird der Variablen „augen“ zugewiesen und ausgegeben.
Erstelle ein Dokument „2.funktion.wuerfel.html“. Benötigt werden die Funktionen „Math.random()“ und „Math.round()“.

```html
<!doctype html>
<head>
<title>Funktion Würfel</title>
<script>
    var wuerfel = function(){
        value = Math.random() * 6;
        value = Math.round(value);
        return value;
    };
</script>
</head>
<body>
    <script>
        var auge = wuerfel();
        document.write(auge);
    </script>
</body>
</html>
```

Quellen:

https://www.youtube.com/watch?v=cAiSbkWvBQU (Mario, 2022)

Nick Morgan, in: JavaScript kinderleicht; 2015, dpunkt.verlag GmbH, Heidelberg, S.115-126

Thomas Theis, in: Einstieg in JavaScript, 2., aktualisierte Auflage 2016; Rheinwerk Verlag GmbH, Bonn, 2016

Florian Franke, Johannes Ippen, in: Apps mit HTML5, CSS3 und JavaScript; 2015, Rheinwerk Verlag, Bonn; S. 102-117

Stephan Elter, in: Programmieren lernen mit JavaScript; 2017, Rheinwerk Verlag, Bonn