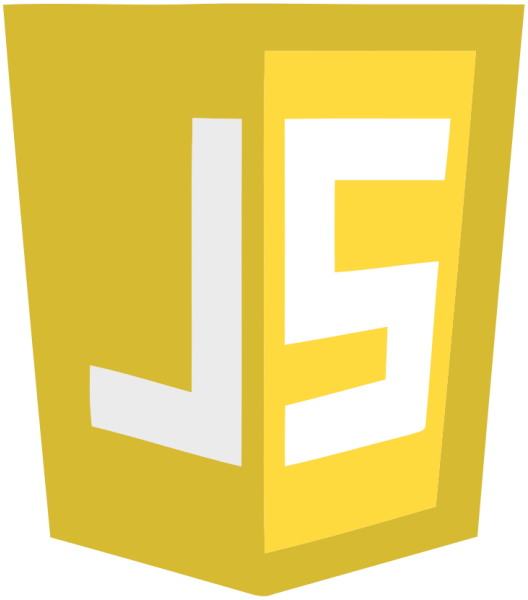
Webdesign mit JavaScript



W.Rose

Inhalt

HTML strukturiert den Inhalt
CSS verleiht ein einheitliches Design

JavaScript sorgt für Dynamik

JavaScript ist DIE Programmiersprache des Webdesigns. Sie wurde gezielt zur dynamischen Manipulation von HTML und CSS (Document Object Model) angelegt und lässt sich sehr gut integrieren und erweitern.

Als Programmiersprache kommt JavaScript mit unverzichtbaren Elementen, die Benutzereingaben verarbeiten und Daten ausgeben: Variablen und Arrays, Kontrollstrukturen wie Schleifen und Verzweigungen, mathematische und logische Operatoren und Methoden zur Ein- und Ausgabe.

Ein weiterer Vorteil von JavaScript ist die Möglichkeit, je nach Bedarf auf Erweiterungen in Form von Frameworks und Programmbibliotheken zuzugreifen. Diese Erweiterungen ermöglichen es, den Funktionsumfang zu vergrößern.

Andere Programmiersprachen wie JAVA sind vielseitiger und komplexer, aber JavaScript kommt mit einigen der besten Konzepte für Webdesign:

- JS benötigt keine Serverumgebung (**clientseitig**)
- JS kann Objekte der Webseite manipulieren (**objektorientiert**)
- JS kann Benutzereingaben umsetzen (**ereignisgesteuert**)
- JS kann für fast jeden Bedarf erweitert werden[1] **(modular)**
- JS läuft in jedem Browser **(plattformunabhängig)**

Einbindung in HTML

JavaScript ändert den Inhalt einer Webseite ereignisgesteuert (onclick) oder automatisch beim Laden der Seite (onload). Daher binden wir den <script> Block nicht im <head> ein, sondern in der Regel **vor den schließenden </body> Tag**.

```
<html>
  <head>
    <title>Projekt 1</title>
  </head>
  <body>
    <script>
     confirm('Möchtest Du lernen, wie man eine Bildergalerie
           mit JS erstellt?');
    </script>
  </body>
</html>
```

[1] JQuery; Node.js und Express.js (serverseitige Anwendungen und APIs); Lightbox (Bild- und Videoverarbeitung); Three.js (3D); Angular (Single-Page Anwendungen); React; Vue.js uvm.

Ein Verweis auf ein externes Javascript findet man hingegen oft im <head>:

```
<script src="script.js"></script>
```

Projekt 1: Eine Slideshow mit JavaScript

Für eine Slideshow braucht es grundlegende Kenntnisse in folgenden Bereichen:

- **Variablen deklarieren und kommentieren**
- **Variablen vom Typ Array (Listen bzw. HTML Sammlungen)**
- **DOM Methoden (Veränderung von HTML Elementen)**
- **Funktionen**
  - **if-Bedingungen**
  - **for-Schleifen (Iterationen)**
  - **Operatoren**

Ein fertiges Beispiel für eine solche Slideshow findest Du *hier* auf codepen.io

Variablen und Datentypen

Zu Beginn eines JavaScripts deklariert man üblicherweise die „globalen Variablen“ (var) und die „lokalen Variablen“ (let) und Konstanten (const). Dieser allgemeine Konsens erleichtert die Lesbarkeit.

Man kann den Variablen Werte unterschiedlicher Datentypen zuweisen. Unterschieden werden Zahlenwerte und Zeichenketten mit Anführungsstrichen. Daneben gibt es noch die Wahrheitswerte (Booleans), welche TRUE oder FALSE sein können.

```
var a = “Wort“;                                   (engl. string)
var b = 1.4;                                      (engl. integer)
var ready = false;                                (engl. boolean)
var imgs =  imgs[0]; imgs[1]; imgs[2]; imgs[3]; usw.   (engl. array)
```

Variablen übergeben und speichern in der Informatik Datenwerte und können, wie in der Mathematik, beliebige Namen haben. Variablen können aber auch mehrere Werte auf einmal in Listen (collections/arrays) speichern. So zum Beispiel

In Zeile 2 werden alle die URLs aller Bilder der Klasse „meineBilder“ gelistet!

JS

```
1  var n, i, bildNr = 1;
2  let imgs = document.getElementsByClassName("meineBilder");
```

Bei der Deklaration der Variablen ist es sinnvoll, ihre Aufgabe zu kommentieren.

- Einzeilige Kommentare werden mit // angeführt.
- Mehrzeilige Kommentare durch /* eingeleitet und durch */ beendet.

Den Variablen in unserem Projekt (s.o.) werden kommen im Projekt Bildergalerie die folgenden Aufgaben zuteil:

```
1  /*  n kann den Wert +1 (vor) und -1 (zurück) annehmen. Diese Parameter werden "onclick"
2      mit den HTML-Pfeilobjekten der Galerie gesteuert.
3      i ist üblicherweise eine Zählvariable für „Iterationen“ (Programmschleifen).
4      imgs sammelt mit einer Objekt Methode alle Bilder im HTML-Dokument und bildet eine HTML-Collection. */
5  //  bildNr erhält den Startwert 1 und bildet das Ergebnis der Funktion weiter(n) ab.
6
```

Objekt Methoden und das DOM

JS kann, wie auch CSS, die Baumstruktur von HTML nutzen, um gezielt bestimmte Elemente zu verändern. Für die Selektion dieser Elemente nutzt man die objektorientierte Baumstruktur von HTML bzw. das „Document Object Model“ (DOM). Der Zugriff auf das HTML-DOM erfolgt über eine Objektmethode: z.B.: **.getElementsById() / …ByClassName().** Diese Methode sucht im HTML nach Elementen mit einem **id** oder einem **class** Attribut. Auch CSS Selektoren können mit der Objektmethode **querySelector()** wie in CSS verwendet werden. Diese neuere Methode bietet sogar umfassendere Funktionen zu Datenverarbeitung (z.B. Arraybefehle).

```
<html>
   <body>
      <img id="Bild" src="https://preview.ibb.co/gxVppG/img1.jpg" />
      <button onclick="next()">Nächstes Bild</button>

      <script> function next()
      {
      document.getElementById("Bild").src =
      "https://preview.ibb.co/iZ3Lww/img2.jpg";
      }
      </script>
   </body>
</html>
```

Dieses Skript ist eine rudimentäre DOM-Manipulation mit Bildern. Probiere es aus und beantworte die folgenden Fragen:

- Wie wird das HTML-Element in diesem Beispiel selektiert??
- Welche Methode verändert welches Attribut?
- Wodurch wird die Funktion ausgelöst?

Welche Nachteile hat dieses rudimentäre Skript?

Im oberen Beispiel wird die Objekt Methode **document.getElementsById** als Selektor genutzt. Es wird also nur ein bestimmtes Bild selektiert. Befinden sich im DOM mehrere Elemente einer Klasse (z.B. mehrere Bilder der Klasse „meine Bilder“), kann man diese mit den anderen Methoden listen und nummerieren bzw. „indizieren“ [0,1,2,3,4]. Mit Hilfe dieser Indexzahlen können wir eine Funktion

schreiben, die mehrere Bilder in Reihenfolge zeigt und beim letzten Bild wieder bei Null anfängt. Das Ergebnis dieser Funktion kann sodann *onload* oder *onclick* einer Variable übergeben werden:

onload…

```
let imgs = document.getElementsByClassName("meineBilder");
```

Nach dem Laden der HTML-Seite sucht die Objekt-Methode im DOM also jedes <img> Element der Klasse „meineBilder“ und weist jedem Bild eine Indexzahl zu. Das erste Bild der Galerie ist imgs[0], das zweite Bild ist imgs[1] usw. Der Name der Variable **imgs** ist beliebig, soll aber hier verdeutlichen, dass es sich um eine nummerierte HTML-Sammlung (*engl. HTML-collection*) von mehreren Bildern handelt. Diese Sammlung wird in der Funktion **zeigeBild(n)** weiter verwendet.

onclick…

Nach einem Klick auf ein Feld oder Button werden innerhalb der folgenden Funktionen die Attribute der Elemente weiterverwendet oder verändert. Konzentrieren wir uns zunächst auf die Schreibweise und die mit **imgs** verwendeten Attribute:

Merke: HTML-DOM-Methoden sind eingebaute JS-Werkzeuge, die HTML-Elemente dynamisch verändern können!

```
4  function weiter(n) {
5     (bildNr += n);
6     if (bildNr > imgs.length) {bildNr = 1};
7     if (bildNr < 1) {bildNr = imgs.length};
8     zeigeBild(bildNr)
9  }
```

Mit der DOM-Methode **.length** wird gezählt wie viele Bilder sich in der HTML-Sammlung befinden, denn das JavaScript muss wissen wo die Bildershow endet und sie wieder von vorne beginnt.

```
11 function zeigeBild(bildNr) {
12      for (i = 0; i < imgs.length; i++)
13      {imgs[i].style.display = "none";
14       imgs[bildNr-1].style.display = "block";}
15 }
```

Mit der DOM-Methode **.style** können die CSS-Attribute aller Bilder verändert werden. Mit der Indexzahl imgs**[0]**.style kann aber auch nur ein bestimmtes Bild verändert werden. In Zeile 11 werden alle Bilder nacheinander unsichtbar gemacht, während das aktuelle Bild in Zeile 12 sichtbar gemacht wird.

Übersicht: Methoden und Eigenschaften für DOM-Manipulation

HTML Elemente finden (Methoden)

document.getElementByID(id)	***Bevor man ein Element verwenden oder ändern kann, muss man es mit diesen Methoden finden!***
document.getElementByClassName(name)	
document.getElementByTagName(name)	
document.querySelector(CSS-Selektor)	

HTML Eigenschaften verändern und Eventhandler hinzufügen

.innerHTML = neuer Inhalt	ändert beispielsweise Text
.attribute = neuer HTML-Wert	ändert beispielsweise das src-Attribut
.style.property = CSS-Wert	ändert CSS-Attribute
.onclick = function(){code}	fügt einen Eventhandler hinzu
.setAttribute (Methode)	ändert ein Attribut direkt als Methode

HTML Elemente verändern(Methoden)

document.createElement(element)	**Erstellt ein neues HTML-Element**
document.removeChild(element)	Entfernt ein HTML-Element
document.appendChild(element)	Fügt ein weiteres Element hinzu
document.replaceChild(neu, alt)	Ersetzt ein HTML-Element
document.write(text)	Schreibt direkt in das HTML Dokument[2]

Funktionen

Im Gegensatz zu den oben aufgeführten DOM-**Methoden** und grundlegenden mathematischen Operatoren die fest zu JavaScript gehören, spricht man bei eigens definierten Anweisungssequenzen[1] von **Funktionen**.

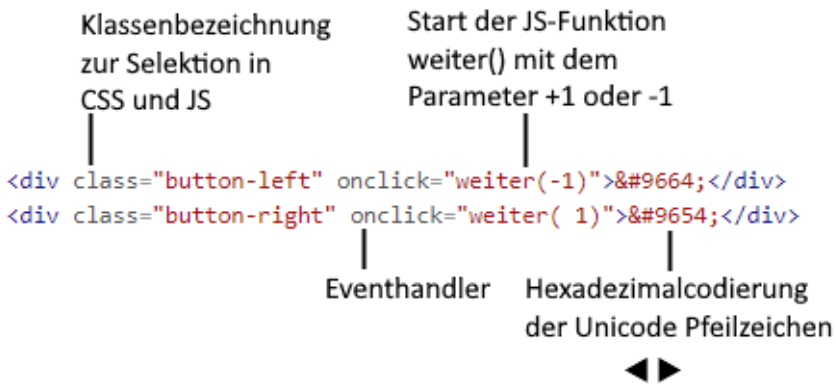
Funktionen werden in der Regel mit dem *function* Befehl definiert oder auch „initialisiert“ [2], nachdem die Variablen deklariert[3] wurden. Aufgerufen[4] werden Funktionen erst im finalen Schritt, in dem auch Parameter[5] in den Klammern[6] übergeben werden können:

English translations:

1. *code block / arguments*
2. *to initiate*
3. *to declare / define*
4. *to invoke or call*
5. *parameters*
6. *parentheses*

Definition	JS `function weiter(n)` `{zeigeBild(bildNr += n);}` *Der Zuordnungsoperator += addiert die Variablen und speichert das Resultat in der bildNr Variable*
Aufruf	HTML `<div class="button-right" onclick="weiter( 1)">` `</div>`

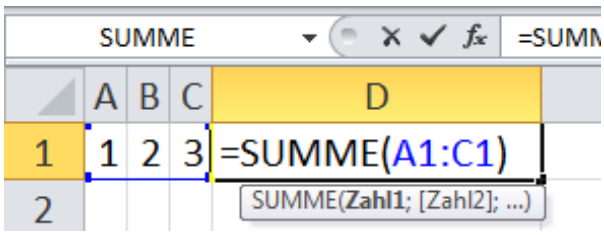
In unserem Beispiel werden die Funktionen **zeigeBild(bildNr)** und die Funktion **weiter(n)** *onclick* durch die HTML *buttons* aufgerufen. In Folge dessen wird der Wert der Variable **n** an die Funktion *weiter( )* übergeben und die Variable **bildNr verändert: : +1 (nächstes Bild) oder -1 (vorheriges Bild)**



Hast Du in Mathe schon ein Tabellenkalkulationsprogramm verwendet?

Vergleiche Funktionen in Excel und JavaScript unter folgenden Gesichtspunkten:

- Gibt es in Excel / Calc auch Definitionen, Funktionen, Parameter und Variablen?
- Welche Gemeinsamkeiten gibt es bei der Schreibweise?



a) Methoden und das Math Objekt

Mit den Objekt-Methoden haben wir bereits einige objektgebundene Methoden kennengelernt, die oft für dynamisches HTML verwendet werden. Darüber hinaus gibt es aber noch weitere Methoden die in kleinen Programmen oder in Formularen oft genutzt werden:

Noten : Punkte in Noten umrechnen
Rose

HTML

```
1  <h1>Notenumrechner</h1>
2  <form onsubmit="event.preventDefault(); processForm();">
3    <span>Maximale Punktzahl:</span>
4    <input type="number" id="maxPoints" required>
5    <br><br>
6    <span>Erreichte Punktzahl:</span>
7    <input type="number" id="achievedPoints" required>
8    <br><br>
9    <button type="submit">Note berechnen</button>
10 </form>
```

- **Benutzereingaben:** prompt(); <input> (HTML);
- **Benutzerausgabe:** .alert(); .confirm(); console.log();
- **Formularauswertungen:** .eval(); return[3]
- **Das Math-Objekt:** Math.random; Math.floor; uvm.

Beispiele welche Eingaben und Ausgaben verwende sind dieser Notendurchschnittsberechner und der praktische Punkte-Noten-Konverter.

Die folgenden Skripte kannst du in den <script> Block einer HTML-Datei oder in den JS-Bereich eines Pens (*codepen.io*) kopieren, um sie zu testen:

Mit der Methode **eval()** kann der Inhalt der Klammer zum Beispiel eine Berechnung ausgewertet werden.

```
alert(eval(3+4*6));
```

Siehe auch das Beispiel auf Seite 12!

Mit **document.write()** können Inhalte oder Parameter in einen HTML-Block geschrieben werden:

```
//Definition (initiation)
function noten(name,mitarbeit,klausur) {
 document.write("<br>"+name+"s Mitarbeit: "+mitarbeit+".<br>");
 document.write(""+name+ "s Klausuren: "+klausur+".<br>");
 }
//Aufruf (invocation)
 noten("Jana",2,3);
 noten("Sven",4,4);
 noten("Ali",3,2);
```

Mit dem **Math.** Objekt sind in JavaScript komplexe Berechnungen möglich.

```
ergebnis=Math.sqrt(25);
alert(ergebnis);
```

Die Methode **sqrt()** berechnet die Quadratwurzel (*engl. square root*)

Die Methode **pow(x,y)** berechnet x^y die y-Potenz von x (*engl. x to the power of y*)

Teste auch die anderen oben aufgeführten mathematischen Funktionen und kommentiere die folgende Funktion:

```
Math.floor((Math.random()*99)+1);
```

Ergänze die Methoden-Liste von Seite 7 und führe eine Art informatisches Vokabel- oder Werkzeugheft!

[3] Hierbei handelt es sich nicht um Methoden, sondern um Schüsselwörter innerhalb von Funktionen.

b) Funktionen definieren

In unserem Galerie-Projekt werden mit Hilfe von Verzweigungen und Operatoren zwei Funktionen definiert:

- **weiter(n)** Die Variable bildNr wird mit dem Parameter n und dem Operator += verändert. Entweder um +1 (vor) oder um -1 (zurück). Die if-Funktion prüft, ob die neue bildNr in der HTML Sammlung [imgs] vorhanden ist. Falls nicht, springt die bildNr zum ersten oder letzten Bild. Anschließend wird die Funktion zeigeBild aufgerufen und der veränderte Parameter bildNr an die folgende Funktion übergeben.

- **zeigeBild(n)** Die for-Schleife macht mit Hilfe einer Iteration (Zählschleife) zunächst alle Bilder unsichtbar und schließlich das richtige Bild sichtbar, indem die CSS-Attribute mit der DOM-Methode **.style.display** manipuliert werden.

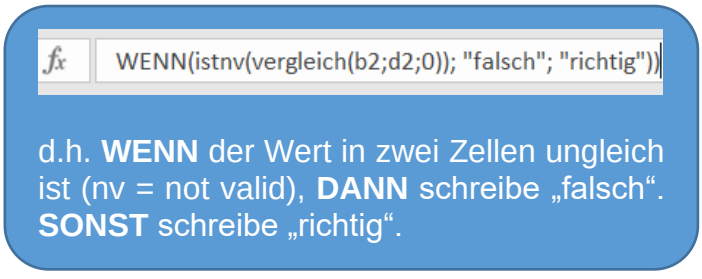
JS

```
4  function weiter(n) {
5    (bildNr += n);
6    if (bildNr > imgs.length) {bildNr = 1};
7    if (bildNr < 1) {bildNr = imgs.length};
8    zeigeBild(bildNr)
9  }
10 function zeigeBild(bildNr) {
11     for (i = 0; i < imgs.length; i++)
12     {imgs[i].style.display = "none";
13      imgs[bildNr-1].style.display = "block";}
14 }
```

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y

if – Abfragen

if then else Abfragen unterbrechen die linare Ausführung der Befehle und überspringen Anweisungen – abhängig von den Ergebnissen der Abfrage. Auch die Tabellenkalkulation kennt derartige Abfragen. Hier ein Beipiel aus einer Excel-Vokabelabfrage:



d.h. **WENN** der Wert in zwei Zellen ungleich ist (nv = not valid), **DANN** schreibe „falsch“. **SONST** schreibe „richtig“.

If-Bedingungen in Javascript sind ähnlich aufgebaut:

if (Bedingung erfüllt?) {Anweisungen} else { Anweisungen }

Die ***if***-Abfragen fragen beide die Anzahl der Bilder der Klasse „meineBilder“ mit Hilfe der Methode ***.length*** ab. Sind die Bedingungen nicht erfüllt, geht es im Skript einfach weiter. Das else ist hier redundant und enfällt.

```
9       if (bildNr > imgs.length) {bildNr = 1};
10      if (bildNr < 1) {bildNr = imgs.length};
```

- Wenn der neue Wert der Variable bildNr größer ist als die Anzahl der Bilder in der HTML-Sammlung, dann beginne wieder von vorn.
- Wenn der neue Wert der Variable bildNr kleiner ist als 1, dann beginne mit dem letzten Bild der Liste. Beispielsweise Bild 7 in einer Liste der Länge 7 [0,1,2,3,4,5,6].

for – Schleifen und Iterationen

Die for-Schleife dient dazu das CSS-Attribut (Zeile 13) der Unsichtbarkeit auf mehrere HTML-Objekte anzuwenden. Der Vorgang wird solange wiederholt, bis die **Zählvariable i** (0-6) nicht mehr kleiner ist als die Gesamtzahl der Bilder in der HTML-Sammlung (7).

```
12      for (i = 0; i < imgs.length; i++) //Iteration
13      {imgs[i].style.display = "none";}
14      imgs[bildNr-1].style.display = "block";
```

Nehmen wir an, es befinden sich 8 Bilder in der Sammlung.

1. Die for-Schleife beginnt mit dem Wert i = 0.
2. Ist 0 < 8 ? Das ist wahr!
3. Dann führe die folgende Anweisung durch und erhöhe i um 1.
4. Das Bild imgs[0] wird per CSS unsichtbar.
5. Die Zählvariable wird erhöht i = 1 (*engl. incremented*).
6. Ist 1 < 8 ? Auch das ist wahr!

…Einige Wiederholungen (engl. iterations) später….

7. Zählvariable i = 7
8. Ist 7 < 7 ? Das ist falsch!
9. Die for-Schleife ist damit beendet und es geht weiter in Zeile 14.
10. Das Bild [**bildNr-1**] aus der HTML-Sammlung wird sichtbar.

Zeile 14: Hä? Warum **imgs[bildNr-1]** ?

Weil Bild 1 dem Bild[0] im Array entspricht.
Weil Bild 7 dem Bild[6] im Array entspricht.

Die Methoden zählen unterschiedlich!

Projekt 2: Noten und Punkte berechnen

In diesem Projekt werden ein paar Konzepte eingeführt, die im Quiz-Projekt weiter vertieft werden:

- **Variablen, Arrays und Datentypen**
  - **parseInt und parseFloat und .value**
  - **Die Verkettung von Zeichen und Werten**
- **EVA**
  - **Eingaben übergeben und Verarbeiten (return)**
  - **Ergebnisse ausgeben (return + alert)**

Es gibt verschiedene Wege ein Input vom User zu erfragen:

- Mit JS-Funktionen wie prompt() oder confirm()
- Mit HTML <form> und <input> Elementen und einem JS – Selektor
- Mit HTML Elementen (z.B. Buttons) deren Wert (.innerText) übergeben wird.

Normalerweise sind HTML-Formulare für die Weiterarbeit mit serverbasierten Sprachen wie PHP und SQL gedacht. Das Standardverhalten eines Formulars sollte daher für JavaScripts mit der Methode preventDefault() unterbunden werden!

HTML

```html
<h2>Notenberechnung</h2>
<form onsubmit="event.preventDefault();">
  Name :<input type="text" id="name" required><br>
  Klasse:<input type="text" id="klasse" required><br>
  Note 1 <input type="number" id="note1" min="0.75" max="6" step="0.25" required><br>
  Note 2 <input type="number" id="note2" min="0.75" max="6" step="0.25" required><br>
  Note 3 <input type="number" id="note3" min="0.75" max="6" step="0.25" required><br>
  Note 4 <input type="number" id="note4" min="0.75" max="6" step="0.25" required><br>
  <br>
  <button type="button" onclick="berechneNote()">Berechnen</button>
</form>
```

Das folgende JS überprüft auf Knopfdruck die Eingaben in den Input-Feldern.

Zunächst werden die Input-Felder anhand ihrer id selektiert und die eingegebenen Werte mit der Methode **.value** übergeben.

JS

```js
function berechneNote() {

    var name = document.getElementById("name").value;
    var klasse = document.getElementById("klasse").value;
```

Erfolgt keine Konvertierung, geht JavaScript bei den Werten davon aus, dass es sich um beliebige Zeichenketten vom Typ „String“ handelt. Zum Beispiel um einen Namen. Das kann bei Berechnungen zu seltsamen Ergebnissen führen!

Möchte man mit Daten und Werten rechnen, ist es in JS notwendig „Strings“ zu „Zahlen“ zu konvertieren. Hierzu gibt es die Methoden parseInt (Integer bzw. Ganzzahl) und parseFloat (Fließkommazahl)

```
6    var note1 = parseFloat(document.getElementById("note1").value);
7    var note2 = parseFloat(document.getElementById("note2").value);
8    var note3 = parseFloat(document.getElementById("note3").value);
9    var note4 = parseFloat(document.getElementById("note4").value);
```

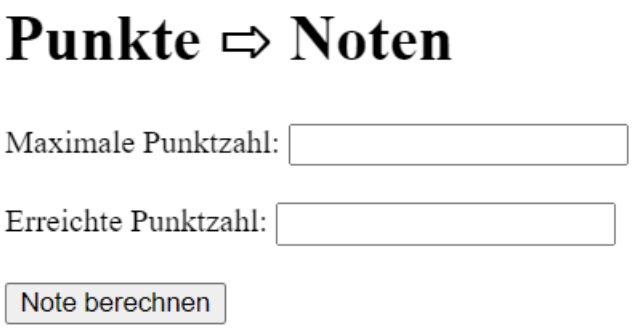
Nun, wo die Schulnoten als Fließkommazahlen (2.25; 2.75; 1.00; 3.25) vorliegen und in Variablen gespeichert sind, kann der Mittelwert durch eine schlichte Division berechnet werden:

```
19   var mittelwert = (note1 + note2 + note3 + note4) / 4;
```

Für die Ausgabe nutzen wir hier die Alert-Funktion und kombinieren Zeichenketten in Anführungszeichen mit den Werten der Variablen. Für die Kombination verschiedener Datentypen verwendet man das Pluszeichen. Das „\n“ erzeugt einen Zeilenumbruch.

```
21   alert("Schüler: " + name + "\n" +
22       "Klasse: " + klasse + "\n" +
23       "Mittelwert: " + mittelwert);
```

Wie viele Punkte hätten zur nächstbesseren Note gefehlt. Solch ein JavaScript schafft Fakten:



Was ist bei der Arbeit mit HTML <form> Elementen in Kombination mit JavaScript in Zeile 2 zu beachten? (S.12)

Was passiert in Zeile 9?

HTML

```html
1  <h1>Punkte &#x21E8; Noten</h1>
2  <form onsubmit="event.preventDefault();">
3    <span>Maximale Punktzahl:</span>
4    <input type="number" id="maxPunkte" required>
5    <br><br>
6    <span>Erreichte Punktzahl:</span>
7    <input type="number" id="Ergebnis" required>
8    <br><br>
9    <button onclick="Konvertierung();">Note berechnen</button>
10 </form>
```

Von welchem Typ sind die beiden Arrays?

Wozu wird der Index [i] in der Schleife zweifach genutzt?

In welcher Reihenfolge werden die Funktionen aufgerufen?

```javascript
var schwellenwerte = [96, 92, 88, 84, 79, 75, 71, 66, 62, 58, 54, 50, 40, 30, 20, 0];
var noten = [
  "sehr gut (plus)", "sehr gut", "sehr gut (minus)",
  "gut (plus)", "gut", "gut (minus)",
  "befriedigend (plus)", "befriedigend", "befriedigend (minus)",
  "ausreichend (plus)", "ausreichend", "ausreichend (minus)",
  "mangelhaft (plus)", "mangelhaft", "mangelhaft (minus)",
  "ungenügend"
];

function punkteZuNote(maximalePunktzahl, erreichtePunktzahl) {
  var prozent = (erreichtePunktzahl / maximalePunktzahl) * 100;

  for (var i = 0; i < 16; i++) {
    if (prozent >= schwellenwerte[i]) {
      Schulnote = noten[i];
      break;
    }
  }

  return Schulnote;
}

function Konvertierung() {
  var maximalePunktzahl = parseFloat(document.getElementById('maxPunkte').value);
  var erreichtePunktzahl = parseFloat(document.getElementById('Ergebnis').value);

  var Schulnote = punkteZuNote(maximalePunktzahl, erreichtePunktzahl);
  alert('Die erreichte Note ist: ' + Schulnote);
}
```

Projekt 3: Ein Quiz mit JavaScript

Für ein ansprechendes Quiz mit Score, braucht es Kenntnisse in folgenden Bereichen:

- **Objekte und Arrays erstellen**
  - **Klassen und Konstruktoren**
  - **What is *this*? *This* is a refernce**
- **DOM-Manipulation**
  - **Dynamische Inhalte**
  - **Eventhandler**
- **Eingaben auswerten (Vergleichen und Zählen)**

Ein fertiges Beispiel für das Quiz Projekt findest Du *hier* auf codepen.io

Warum kompliziert, wenn es auch einfach geht?

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Test</title>
        <script>
            function antwort1()
            {
                alert("Richtig!");
            }
            function antwort2()
            {
                alert("Leider falsch");
            }
        </script>
    </head>
    <body>
        <h2>Frage: Wie lautet die Antwort auf die große Frage nach dem
            Leben, dem Universum und Allem?</h2>
        <p>
            <a href="javascript:antwort2()">Urknall</a> <br>
            <a href="javascript:antwort2()">Gott</a> <br>
            <a href="javascript:antwort1()">42</a>
        </p>
    </body>
</html>
```

1. Überführe das Skript in einen Pen oder eine HTML-Datei.
2. Erweitere das Quiz um drei weitere Fragen.
3. Erkläre die Funktionsweise und ihre Nachteile in einem <!-- Kommentar --!>

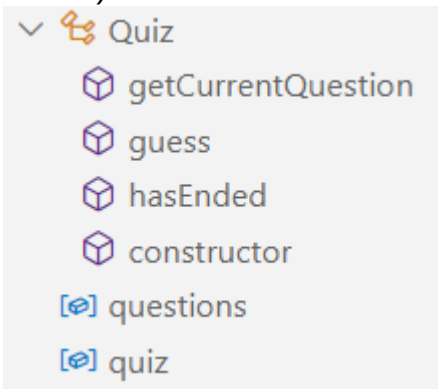
Neue JS Objekte und Arrays definieren

JavaScript ist objektorientiert. Objekte haben Eigenschaften und Werte und sie beinhalten Methoden, Funktionen und Variablen. Die können wir am Beispiel des Quiz-Objekts veranschaulichen:

Was gehört zum Quiz-Objekt?

- Quiz (Objekt)
  - Startbedingungen
  - Quizfragen [Objekt-Array]
    - Fragetexte (Eigenschaft)
    - Antwortmöglichkeiten [String-Array]
    - Lösung (Eigenschaft)
  - Punktestand (Funktion)
  - Auswertung der Benutzereingabe (Funktion)
  - Fragenindex (Funktion)
  - End-Abfrage (Funktion)

Der Editor Visual Studio Code zeigt diese Zugehörigkeiten im Bereich „Outline“ an und unterscheidet Objektklassen, Methoden und Eigenschaften bzw. Arrays.




```
1  // Erzeugung eines Quiz-Objekts bzw. einer neuen Klasse (constructor)
2  class Quiz {
3      constructor(questions) {
4          this.score = 0;
5          this.questions = questions;
6          this.currentQuestionIndex = 0;
7      }
8      // Funktionen, die mit den Attributen des Quiz-Objekts arbeiten, nachdem dieses erzeugt wurde.
9      getCurrentQuestion() {
10         return this.questions[this.currentQuestionIndex];
11     }
12     guess(answer) {
13         if (this.getCurrentQuestion().isCorrectAnswer(answer)) {
14             this.score++;
15         }
16         this.currentQuestionIndex++;
17     }
18     hasEnded() {
19         return this.currentQuestionIndex >= this.questions.length;
20     }
21 }
```

Welche Eigenschaften und Methoden werden hier definiert?

Ordne die oben gennanten Bestandteile des Quiz-Objekts den Anweisungen in Zeile 1 – 21 zu.

Worauf bezieht sich **this**?

Klassen und Konstruktoren

Mit Konstruktoren ist es möglich nicht nur ein Objekt (Quiz) mit seinen Eingenschaften und Methoden zu definieren. Es können auch mehrere Objekte einer Klasse (***class***) nach einer Konstruktionsvorlage (***engl. blueprint***) erzeugt werden. Das nächste Objekt (Question) ist dafür ein gutes Beispiel.

Den Objekten der Klasse Question ist gemein, dass sie sich aus einem Fragetext (Eigenschaft), 4 Antwortmöglichkeiten [Array] und einer Lösung zusammensetzen. Sie unterscheiden sich nur in ihren Datenwerten vom Typ String (Texte).

Die neue Methode ***constructor*** einer Klasse erfüllt den Zweck einer Konstruktionsvorlage für neue, gleichartige Objekte. In älteren Skripten findet man noch Konstruktor-Funktionen. Diese waren kaum von anderen Funktionen zu unterscheiden. Aus dieser Zeit stammt auch die noch immer gängige Praxis zur besseren Unterscheidung, die Namen von Objekten groß zu schreiben.

Was ist *this*?

```
23  class Question {
24      constructor(text, choices, answer) {
25          this.text = text;
26          this.choices = choices;
27          this.answer = answer;
28      }
29      isCorrectAnswer(choice) {
30          return this.answer === choice;
31      }
32  }
```

In JavaScript verweist das Schlüsselwort ***this*** normalerweise auf das Objekt, dass die Funktion aufruft. Dies gilt insbesondere für Funktionen, die als Methoden eines Objekts definiert sind. In solchen Fällen verweist ***this*** auf das Objekt selbst und ermöglicht den Zugriff auf dessen Eigenschaften und Methoden.

Darüber hinaus kann ***this*** auch im Zusammenhang mit der Verwendung von Konstruktoren auftreten. Wenn eine Funktion als Konstruktor mit dem ***new*** Schlüsselwort aufgerufen wird, können Instanzen (Kopien) dieser Klasse generiert werden, auf welche ***this*** verweist.

In unserem Fall wurde oben die Klasse „Question“ mit drei Eigenschaften und einer Methode konstruiert, die Eingaben des Users prüft. Jede Instanz dieser Klasse erwartet eine Fragetext, ein Array an Auswahlmöglichkeiten und die Lösung.

```
34  let questions = [
35      new Question("Wer war nicht Schulleiter am ConvoS?", ["Hans Grote", "Hans Wiesmann", "Zita Bruski", "Wolfgang Lücking"],
36      new Question("Wie lautet die Antwort auf die große Frage nach dem Leben, dem Universum und Allem?", ["Gott", "42", "Hä
37      new Question("Womit wurde dieses Quiz erstellt?", ["HTML", "PHP", "Javascript", "CSS"], "Javascript"),
38      new Question("Gibt es eine Programmiersprache für Anfänger?", ["Javascript", "Python", "C#", "Scratch"], "Scratch"),
39      new Question("Ist das ConvoS die beste Schule in Soest?", ["Ja", "Nein", "Eher Nicht", "Das ist Ansichtssache"], "Ja")
40  ];
```

Das User Interface (DOM-Manipulation)

Das dritte und letzte Objekt ist für die Eingabe, Verarbeitung und Anzeige (EVA) der Daten zuständig (engl. user interface oder frontend). Auch dieses Objekt wird einmalig als Objektliteral notiert. Das heißt, es werden Eigenschaften und Methoden innerhalb von geschweiften Klammern {} definiert und auch aufgerufen!

```
const QuizUI = {
```

Funktion 1: displayNext wird definiert: Wenn hasEnded wahr ist, wird die Funktion displayScore aufgerufen. Sonst werden die Funktionen displayQuestion, displayChoices und displayProgress aufgerufen.

```
  displayNext: function() {
    if (quiz.hasEnded()) {
      this.displayScore();
    } else {
      this.displayQuestion();
      this.displayChoices();
      this.displayProgress();
    }
  },
```

Funktion 2: refreshIDs - Diese Funktion hat die Aufgabe bestimmte Texte im HTML zu überschreiben. Er erwartet dazu beim Aufruf die Ziel-id im HTML und einen neuen Text als String.

```
  refreshIDs: function(id, text) {
    var element = document.getElementById(id);
    element.innerHTML = text;
  },
```

Funktion 3: displayQuestion - Diese Methode überschreibt beispielsweise den Text des HTML-Elements mit der id="question" mit dem Text der aktuellen Frage.

```
  displayQuestion: function() {
    this.refreshIDs("question", quiz.getCurrentQuestion().text);
  },
```

Funktion 4: displayScore - Diese Methode erstellt das HTML des Schlussbildschirms innerhalb des , indem es den Wert der Variable score mit HTML kombiniert, dass am Schluss angezeigt wird.

```
displayScore: function() {
  var gameOverHTML = "<h1>Game Over</h1>";
  gameOverHTML += "<h2> Your score is: " + quiz.score + "</h2>";
  this.refreshIDs("quiz", gameOverHTML);
},
```

Funktion 5: guessHandler - Hier wird jedem Button ein onclick-Ereignis verliehen. Dann wird die Methoden "guess()" des Quiz-Objekts aufgerufen um die Antwort zu prüfen und ggfls. die Werte des Scores und des Index zu erhöhen, bevor mit die Methode "displayNext()" erneut aufgerufen wird.

```
guessHandler: function(id, guess) {
  var button = document.getElementById(id);
  button.onclick = function() {
    quiz.guess(guess);
    QuizUI.displayNext();
  };
},
```

Funktion 6: displayChoices - Der guessHandler(id, guess) wird durch die Methode displayChoices() 4x aufgerufen. Die Schleife durchläuft dabei die id der Buttons "guess1" bis "guess4" und die Antwortmöglichkeiten "choices[1]" bis "choices[4]" um diese der Funktion guessHandler(id, guess) als Parameter zu übergeben.

```
displayChoices: function() {
  var choices = quiz.getCurrentQuestion().choices;
  for (var i = 0; i < choices.length; i++) {
    this.refreshIDs("guess" + i, choices[i]);
    this.guessHandler("guess" + i, choices[i]);
  }
},
```

Funktion 7: displayProgress - Diese Funktion aktualisiert den Text im HTML Frage x von y , indem es mit Hilfe der Methode refreshIDs bzw. .innerHTML die Werte der Variablen mit Text kombiniert.

```
  displayProgress: function() {
    var currentQuestionNumber = quiz.currentQuestionIndex + 1;
    this.refreshIDs("progress", "Frage " + currentQuestionNumber + " von " +
quiz.questions.length);
  }
};

QuizUI.displayNext();          //Ruft zu Beginn das Quiz mit der ersten Frage auf.
```

Vorlage: https://codepen.io/jasonchan/pen/wMaEwN

Quellen:

Wörterbuch Scratch – JavaScript: https://leopardjs.com/manual

Online-Tutorial mit einfachen Beispielen: https://www.w3schools.com/JS/default.asp

Alternatives Turorial: CSS, HTML und Javascript mit {stil}

JS cheat sheet: https://htmlcheatsheet.com/js/