

Thomas Theis

Bonuskapitel zu »Einstieg in JavaScript«

Bonuskapitel 1

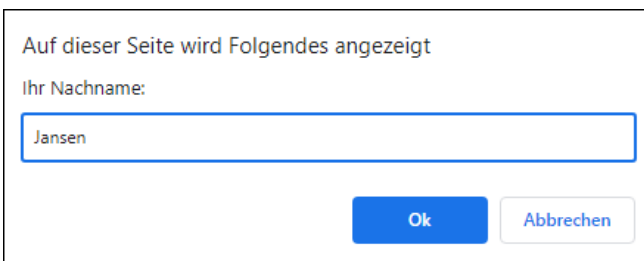
Übungen

In diesem Abschnitt finden Sie die Aufgabenstellungen zu den Übungen, auf die im Buch an den entsprechenden Stellen verwiesen wird.

Erläutern Sie Ihre eigenen Lösungen mit Kommentaren im Code. Verwenden Sie für alle Übungsaufgaben den Dokumentaufbau der Beispielprogramme, inklusive der Formatierung mithilfe der externen CSS-Datei *js5.css*. Nutzen Sie nach Möglichkeit unveränderliche Variablen. Eine Lösung zu jeder Übungsaufgabe finden Sie im Downloadpaket zum Buch, das Sie über <https://www.rheinwerk-verlag.de/einstieg-in-javascript> erreichen.

1.1 Übung »u_eingabe«

Schreiben Sie ein Programm in der Datei *u_eingabe.htm*. Nach dem Aufruf des Programms sollen zwei Eingabeaufforderungen nacheinander erscheinen. Es soll der Nachname und anschließend der Vorname eingegeben werden, wie in Abbildung 1.1 und Abbildung 1.2. Die beiden Eingaben werden in zwei Variablen des Programms gespeichert. Anschließend erscheint eine Ausgabe im Dokument, wie in Abbildung 1.3:

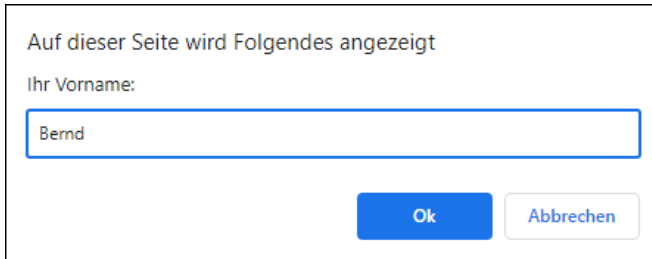


Auf dieser Seite wird Folgendes angezeigt

Ihr Nachname:

Ok Abbrechen

Abbildung 1.1 Eingabe des Nachnamens

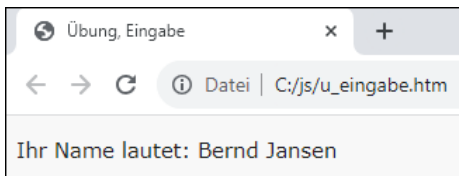


Auf dieser Seite wird Folgendes angezeigt

Ihr Vorname:

Ok Abbrechen

Abbildung 1.2 Eingabe des Vornamens



Übung, Eingabe

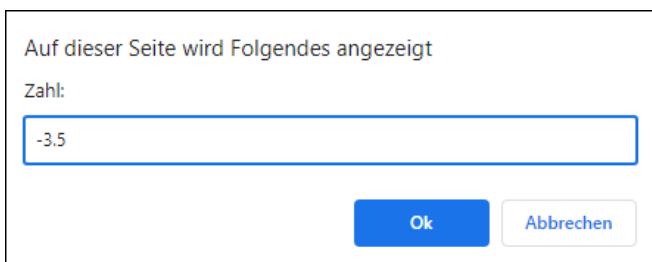
← → ↻ Datei | C:/js/u_eingabe.htm

Ihr Name lautet: Bernd Jansen

Abbildung 1.3 Ausgabe

1.2 Übung »u_zahl«

Schreiben Sie ein Programm in der Datei `u_zahl.htm`. Nach dem Aufruf des Programms soll eine Eingabeaufforderung erscheinen, wie in [Abbildung 1.4](#). Nach der Eingabe einer (gültigen) Zahl wird das Quadrat der Zahl berechnet. Zudem wird die Zahl hoch 5 gerechnet. Die Ergebnisse werden im Dokument ausgegeben, wie in [Abbildung 1.5](#).



Auf dieser Seite wird Folgendes angezeigt

Zahl:

Ok Abbrechen

Abbildung 1.4 Eingabe einer Zahl

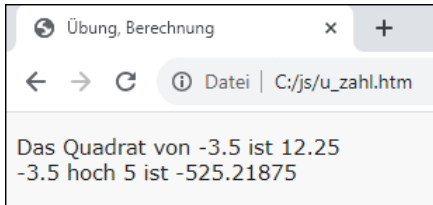


Abbildung 1.5 Ausgabe der Ergebnisse

1.3 Übung »u_if«

Schreiben Sie ein Programm in der Datei `u_if.htm`. Nach dem Aufruf des Programms sollen zwei Eingabeaufforderungen erscheinen, siehe [Abbildung 1.6](#) und [Abbildung 1.7](#). Nach der Eingabe von zwei beliebigen (gültigen) Zahlen wird ermittelt, welche der beiden Zahlen größer ist bzw. ob beide Zahlen gleich sind. Die Ergebnisse werden im Dokument ausgegeben, wie in [Abbildung 1.8](#) oder in [Abbildung 1.9](#).

Abbildung 1.6 Eingabe der ersten Zahl

Abbildung 1.7 Eingabe der zweiten Zahl

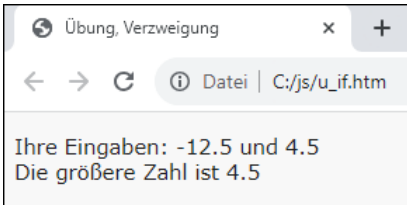


Abbildung 1.8 Ergebnis bei ungleichen Zahlen

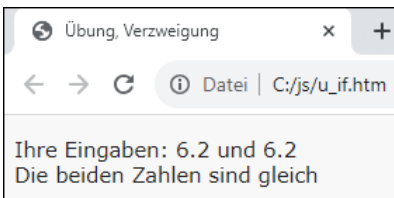


Abbildung 1.9 Ergebnis bei gleichen Zahlen

1.4 Übung »u_verknuepft«

Schreiben Sie ein Programm in der Datei `u_verknuepft.htm`. Nach dem Aufruf des Programms soll eine Eingabeaufforderung erscheinen, siehe [Abbildung 1.10](#). Nach der Eingabe einer Zahl zwischen 50.0 und 100.0 ist entweder eine Ausgabe wie in [Abbildung 1.11](#) oder eine Ausgabe wie in [Abbildung 1.12](#) zu sehen.

 A screenshot of a dialog box with the title "Auf dieser Seite wird Folgendes angezeigt". It contains the text "Zahl zwischen 50.0 und 100.0:" followed by a text input field containing the value "72.8". At the bottom right, there are two buttons: "Ok" and "Abbrechen".

Abbildung 1.10 Eingabe einer Zahl zwischen 50.0 und 100.0

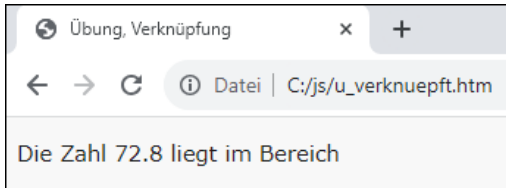


Abbildung 1.11 Zahl im Bereich

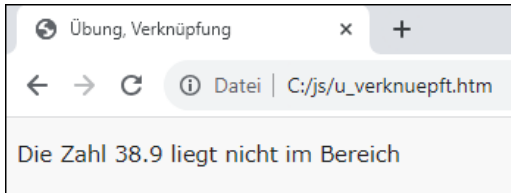


Abbildung 1.12 Zahl nicht im Bereich

1.5 Übung »u_for«

Schreiben Sie ein Programm in der Datei *u_for.htm*. Mithilfe von zwei unterschiedlichen `for`-Schleifen und kombinierten Zuweisungsoperatoren erzeugt es eine Ausgabe wie in [Abbildung 1.13](#).

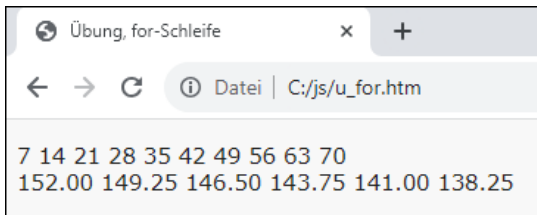
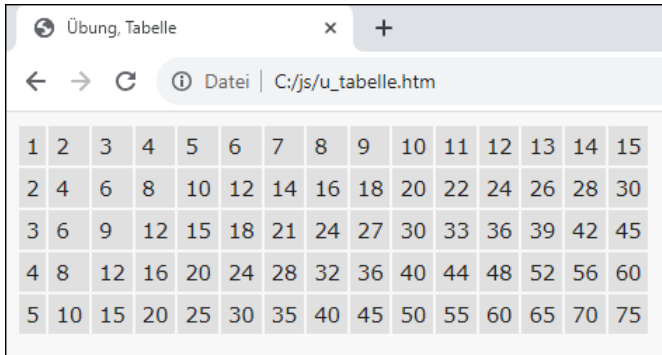


Abbildung 1.13 Zwei Schleifen

1.6 Übung »u_tabelle«

Schreiben Sie ein Programm in der Datei *u_tabelle.htm*. Es erzeugt mithilfe von zwei geschachtelten `for`-Schleifen eine Ausgabe mit einer Tabelle wie in [Abbildung 1.14](#).



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
3	6	9	12	15	18	21	24	27	30	33	36	39	42	45
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
5	10	15	20	25	30	35	40	45	50	55	60	65	70	75

Abbildung 1.14 Ausgabe einer Tabelle

1.7 Übung »u_funktion«

Schreiben Sie ein Programm in der Datei `u_funktion.htm`. Nach dem Aufruf des Programms sollen nacheinander drei Eingabeaufforderungen erscheinen. Nach der Eingabe von drei beliebigen (gültigen) Zahlen, hier ohne Abbildung, wird im Programm eine Funktion mit dem Namen `mittelwert()` aufgerufen. Diese Funktion soll von Ihnen im Kopf des Dokuments definiert werden.

Parameter der Funktion sind die drei eingegebenen Zahlen. Rückgabewert der Funktion ist der arithmetische Mittelwert, also die Summe der Zahlen, geteilt durch ihre Anzahl. Das Ergebnis wird im Dokument ausgegeben, wie in [Abbildung 1.15](#).

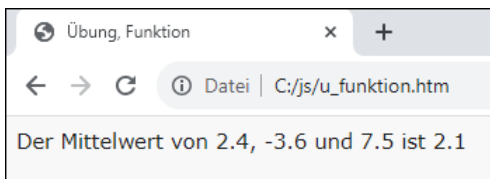


Abbildung 1.15 Funktion zur Berechnung des Mittelwerts

1.8 Übung »u_destrukturierung«

Schreiben Sie ein Programm in der Datei `u_destrukturierung.htm`. Es stellt eine Erweiterung des Programms aus der Übung `u_funktion` dar. Es wird eine eigene Funktion mit dem Namen `summe_und_mittelwert()` definiert und aufgerufen.

Parameter der Funktion sind wiederum die drei eingegebenen Zahlen. Die Funktion gibt zwei Werte zurück: die Summe der Zahlen und den arithmetischen Mittelwert. Das Ergebnis wird im Dokument ausgegeben, wie in [Abbildung 1.16](#).

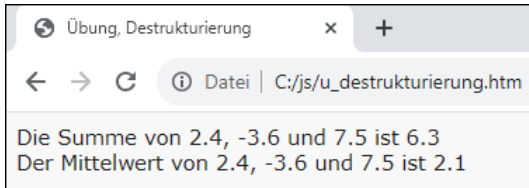


Abbildung 1.16 Funktion liefert mehrere Ergebnisse.

1.9 Übung »u_klasse«

Schreiben Sie ein Programm in der Datei *u_klasse.htm*. Es wird eine Klasse für geometrische Rechtecke definiert, die sich in einer Zeichnung befinden. Sie verfügen über die folgenden Eigenschaften: X- und Y-Position in der Zeichnung, Breite und Höhe. Die Klasse ermöglicht mithilfe von mehreren Methoden die nachfolgende Verwendung:

```
...
<body><p>
  <script>
    const rEins = new Rechteck(50, 10, 100, 200);
    document.write(rEins + "<br>");
    rEins.verbreitern(50);
    document.write(rEins + "<br>");
    rEins.erhoehen(20);
    document.write(rEins + "<br>");
    rEins.skalierten(1.5);
    document.write(rEins + "<br>");
    rEins.verschieben(20, 80);
    document.write(rEins);
  </script></p>
</body></html>
```

Listing 1.1 Datei »u_klasse.htm«, unterer Teil

Die Ausgabe des Programms sieht aus wie in [Abbildung 1.17](#).

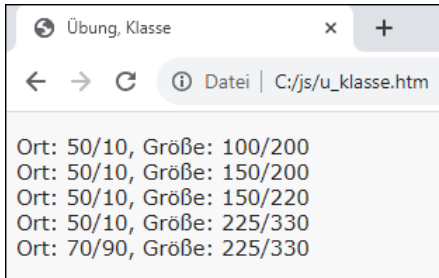


Abbildung 1.17 Ausgabe von Objekteigenschaften

1.10 Übung »u_vererbung«

Schreiben Sie ein Programm in der Datei `u_vererbung.htm`. Es stellt eine Erweiterung des Programms aus der Übung `u_klasse` dar. Es wird eine weitere Klasse für gefüllte, geometrische Rechtecke definiert, die sich in einer Zeichnung befinden. Die neue Klasse wird von der Klasse für geometrische Rechtecke abgeleitet und nutzt dabei die Möglichkeiten der Vererbung. Beide Klassen werden jeweils in einer externen JavaScript-Datei definiert.

Die neue Klasse besitzt folgende zusätzliche Eigenschaften: Füllungsfarbe, Rahmenfarbe und Rahmendicke. Mithilfe von mehreren Methoden ermöglicht die neue Klasse die nachfolgende Verwendung:

```
...
<body><p>
  <script>
    const rEins = new RechteckGefuehlt(50, 10, 100, 200, "Rot", "Schwarz", 3);
    document.write(rEins + "<br>");
    rEins.verschieben(20, 80);
    rEins.skaliieren(1.5);
    rEins.fuellungFaerben("Gelb");
    rEins.rahmenFaerben("Blau");
    rEins.rahmenSkalieren(2);
    document.write(rEins + "<br>");
  </script></p>
</body></html>
```

Listing 1.2 Datei »u_vererbung.htm«, unterer Teil

Die Ausgabe des Programms sieht aus wie in [Abbildung 1.18](#).

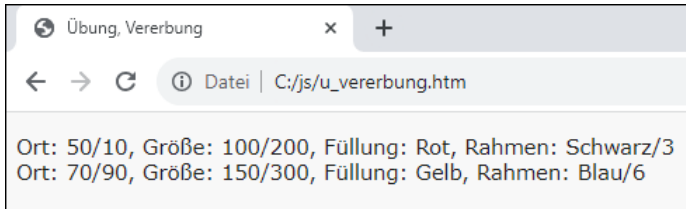


Abbildung 1.18 Vererbung

1.11 Übung »u_formular«

Schreiben Sie ein Programm in der Datei `u_formular.htm`. Es soll ein Formular mit mehreren Elementen enthalten. Nach dem Aufruf des Programms sollen zwei Zahlen in den beiden oberen Eingabefeldern eingegeben werden, wie in [Abbildung 1.19](#).

Nach dem Betätigen der Schaltfläche berechnet das Programm die Summe der beiden Zahlen und gibt sie aus. Im Ausgabefeld können keine Eingaben vorgenommen werden. Wird in einem der beiden Eingabefelder nichts oder keine gültige Zahl eingetragen, soll für dieses Eingabefeld der Wert 0 angenommen werden.

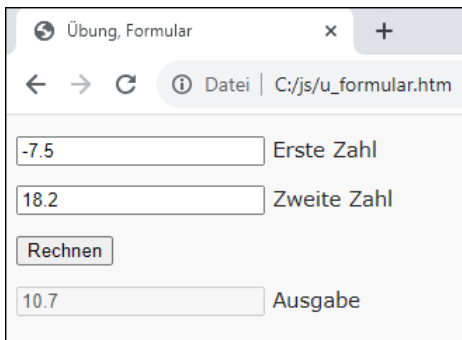


Abbildung 1.19 Formular mit Eingabe und Ausgabe

1.12 Übung »u_radio«

Schreiben Sie ein Programm in der Datei `u_radio.htm`, das auf der Übung `u_formular` basiert. Nach dem Aufruf des Programms soll mithilfe von vier Radiobuttons die Rechenoperation ausgewählt werden, wie in [Abbildung 1.20](#). Der Standardwert steht für die Addition.

Abbildung 1.20 Formular mit Radiobuttons

1.13 Übung »u_select«

Schreiben Sie ein Programm in der Datei *u_select.htm*, das wiederum auf der Übung *u_formular* basiert. Nach dem Aufruf des Programms soll die Rechenoperation diesmal mithilfe eines Auswahlmensüs vorgenommen werden, wie in [Abbildung 1.21](#). Das Auswahlmensü verdeckt kurzzeitig das Ausgabefeld. Der Standardwert ist ebenso die Addition.

Abbildung 1.21 Formular mit Auswahlmensü

1.14 Übung »u_maus«

Schreiben Sie ein Programm in der Datei *u_maus.htm*. Es wird ein Bild dargestellt. Nach einem Klick auf das Bild erscheint in einem Ausgabefeld die Information, in welchem

Viertel des Bilds der Klick erfolgte: Links oben, links unten, rechts oben oder rechts unten, wie in [Abbildung 1.22](#).

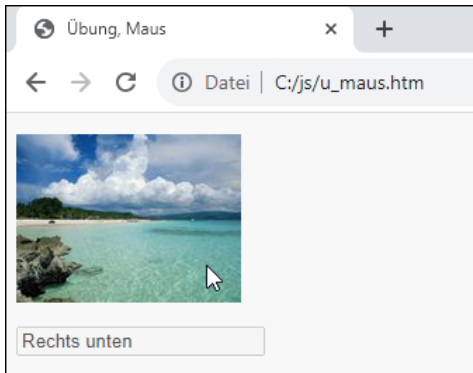


Abbildung 1.22 Maus-Ereignis

1.15 Übung »u_eindimensional«

Schreiben Sie ein Programm in der Datei *u_eindimensional.htm*. Erstellen Sie ein Feld mit 15 Elementen. Füllen Sie das Feld mit zufälligen ganzen Zahlen zwischen 20 und 30 und geben Sie die Elemente zusammen mit ihren Feld-Positionen in einer Tabelle aus. Ermitteln Sie den größten Wert des Felds und geben Sie seine Position(en) aus. Es soll aussehen wie in [Abbildung 1.23](#).

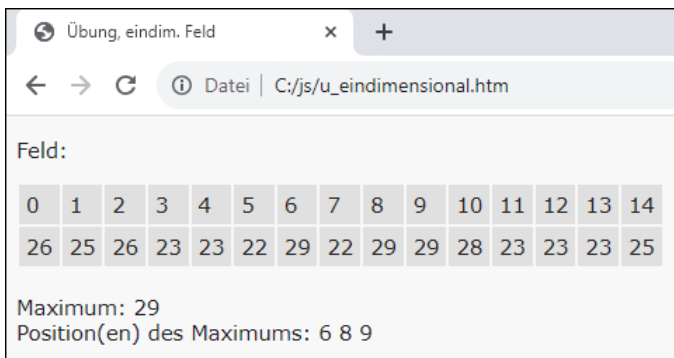
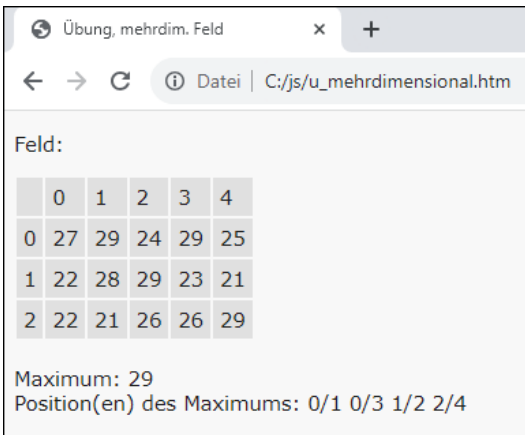


Abbildung 1.23 Eindimensionales Feld

1.16 Übung »u_mehrdimensional«

Schreiben Sie ein Programm in der Datei `u_mehrdimensional.htm`. Erstellen Sie ein zweidimensionales Feld mit 3×5 Elementen. Füllen Sie es mit zufälligen ganzen Zahlen zwischen 20 und 30 und geben Sie die Elemente zusammen mit ihren Feld-Positionen in einer Tabelle aus. Ermitteln Sie den größten Wert des Felds und geben Sie seine Position(en) aus. Es soll aussehen wie in [Abbildung 1.24](#).



Übung, mehrdim. Feld

← → ↻ Datei | C:/js/u_mehrdimensional.htm

Feld:

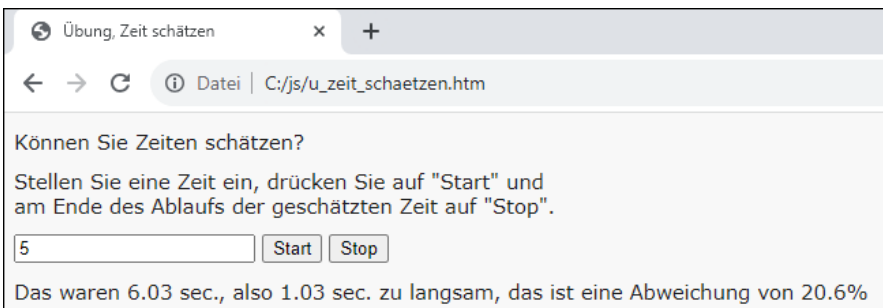
	0	1	2	3	4
0	27	29	24	29	25
1	22	28	29	23	21
2	22	21	26	26	29

Maximum: 29
Position(en) des Maximums: 0/1 0/3 1/2 2/4

Abbildung 1.24 Mehrdimensionales Feld

1.17 Übung »u_zeit_schaetzen«

Schreiben Sie ein Programm in der Datei `u_zeit_schaetzen.htm`. Erstellen Sie ein Formular mit einem Eingabefeld und zwei Schaltflächen. Nach dem Aufruf soll mithilfe des Programms trainiert werden, einen Zeitraum möglichst genau zu schätzen, z. B. fünf Sekunden. In [Abbildung 1.25](#) sehen Sie die Regeln und ein mögliches Ergebnis.



Übung, Zeit schätzen

← → ↻ Datei | C:/js/u_zeit_schaetzen.htm

Können Sie Zeiten schätzen?

Stellen Sie eine Zeit ein, drücken Sie auf "Start" und am Ende des Ablaufs der geschätzten Zeit auf "Stop".

Das waren 6.03 sec., also 1.03 sec. zu langsam, das ist eine Abweichung von 20.6%

Abbildung 1.25 Zeit schätzen

Es wird die Differenz zwischen dem Sollwert und dem Istwert berechnet. Der Sollwert ist die eingegebene Zeit in Sekunden. Der Istwert ist der Zeitraum, der zwischen dem Betätigen der Schaltfläche START und dem Betätigen der Schaltfläche STOP vergangen ist. Zudem wird ermittelt, ob die Schaltfläche STOP zu früh oder zu spät betätigt wurde. Als Letztes wird die relative Abweichung in Prozent berechnet.

Was ist die relative Abweichung? Nehmen wir an, es werden zwei Versuche gemacht, einen Zeitraum zu schätzen. Beim ersten Versuch sollen 10 Sekunden geschätzt werden, die Betätigung der Schaltfläche STOP erfolgt aber bereits nach 9.5 Sekunden. Beim zweiten Versuch sollen 5 Sekunden geschätzt werden und die Betätigung erfolgt bereits nach 4.5 Sekunden.

In beiden Fällen beträgt die absolute Abweichung 0.5 Sekunden. Bei der Berechnung der relativen Abweichung wird die absolute Abweichung aber auf unterschiedliche Sollwerte bezogen. Dies ergibt beim ersten Versuch 5 %, beim zweiten Versuch 10 %. Beim ersten Versuch ist die relative Abweichung geringer; die Schätzung ist also besser.

1.18 Übung »u_map«

Schreiben Sie ein Programm in der Datei `u_map.htm`. Erstellen Sie darin zunächst ein eindimensionales Feld, in dem die Namen der Wochentage gespeichert sind. Erzeugen Sie anschließend mithilfe dieses Felds und zufälligen ganzen Zahlen zwischen 20 und 30 eine Map mit 7 Elementen. Die Zahlen sollen die Höchstwerte der Temperaturen am jeweiligen Wochentag repräsentieren. Geben Sie alle Elemente der Map aus. Berechnen Sie den Durchschnitt der sieben Werte und geben Sie ihn aus, siehe [Abbildung 1.26](#).

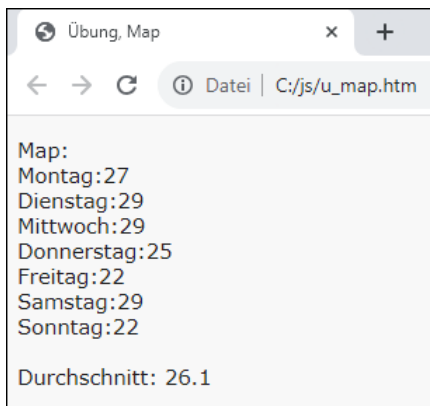


Abbildung 1.26 Map

1.19 Übung »u_svg«

Schreiben Sie ein Programm in der Datei `u_svg.htm`. Erstellen Sie einen Rahmen der Größe 400×400 Pixel. Erzeugen Sie in der linken oberen Ecke eine Grafik, wie in [Abbildung 1.27](#). Aus welchen Elementen besteht sie? Schätzen Sie die Größe und die Position der Elemente.

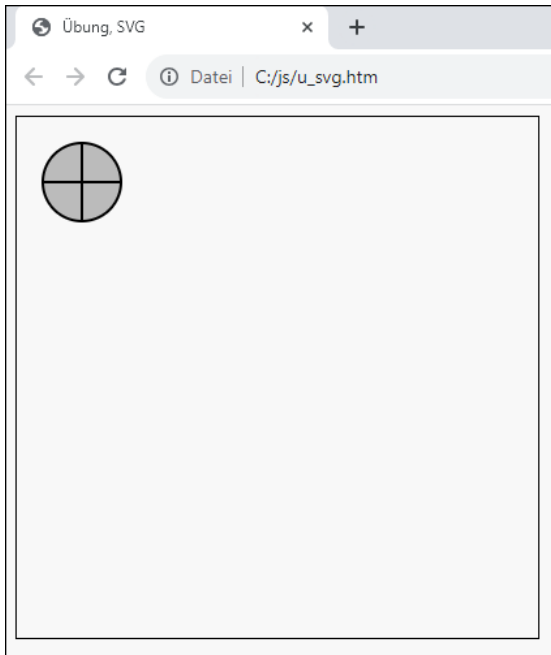


Abbildung 1.27 Startpunkt und Endpunkt

Nach einer Sekunde soll sich die Grafik innerhalb von zwei Sekunden nach rechts bewegen, bis sie in der rechten oberen Ecke angekommen ist, mit den gleichen Abständen zum Rand. Anschließend soll sie sich in zwei Sekunden zur rechten unteren Ecke bewegen, dann in zwei Sekunden zur linken unteren Ecke, zuletzt in zwei Sekunden zurück zu ihrem Ausgangspunkt in der linken oberen Ecke. Dort kommt sie also nach neun Sekunden wieder an.

Bonuskapitel 2

Reguläre Ausdrücke

Die Inhalte dieses Kapitels gehören thematisch an das Ende des Buchabschnitts 6.2, »Zeichenketten verarbeiten«.

Es wird ein Einblick in die umfangreichen Möglichkeiten von regulären Ausdrücken (englisch: *regular expressions*) gegeben. Sie dienen mithilfe von Suchmustern (englisch: *search pattern*) zum Durchsuchen und zum Ändern von Zeichenketten und werden in vielen Programmiersprachen genutzt.

Sie kommen häufig bei der Kontrolle und Auswertung von Benutzereingaben oder bei der Suche nach Dateien zum Einsatz. Wird festgestellt, dass ein Benutzer eine falsche oder unvollständige Eingabe gemacht hat, kann dies zu einer Hilfestellung und einer erneuten Eingabeaufforderung führen.

In JavaScript werden die Objekte `RegExp` und `String` gemeinsam eingesetzt, um mit regulären Ausdrücken zu arbeiten. Seit der Version 5 von HTML bieten einige Formularelemente zur Kontrolle das Attribut `pattern`, siehe Buchabschnitt 4.9.1, »Texteingaben, Suchfelder und Farben«. Als Attributwerte werden ebenfalls reguläre Ausdrücke verwendet.

2.1 Methoden

Im nachfolgenden Beispielprogramm werden einige Methoden des `String`-Objekts eingesetzt, die mit regulären Ausdrücken arbeiten:

- ▶ `search()`, liefert die Position einer Zeichenfolge im untersuchten Text, die zum Suchmuster passt
- ▶ `match()`, liefert eine oder alle Zeichenfolgen des untersuchten Textes, die zum Suchmuster passen
- ▶ `replace()`, ersetzt alle Zeichenfolgen im untersuchten Text, die zum Suchmuster passen, durch anderen Text

Der erste Teil des Programms:

```
...
<body><p>
<script>
    const tx1 = "Das ist mein Wagen oder sein Wagen.";
    const tx2 = "Dieser Text sieht anders aus.";
    document.write("Text 1: " + tx1 + "<br>");
    document.write("Text 2: " + tx2 + "<br>");
    document.write("<br>");

    document.write("Position in 1: " + tx1.search(/.ein/) + "<br>");
    document.write("Position in 2: " + tx2.search(/.ein/) + "<br>");
    document.write("<br>");
...

```

Listing 2.1 Datei »regexp_methoden.htm«, Teil 1 von 2

Zunächst werden zwei Texte erzeugt und ausgegeben.

Die Methode `search()` wird mit einem regulären Ausdruck als Suchmuster für beide Texte aufgerufen. Ein solcher Ausdruck wird innerhalb von zwei /Slashes/ notiert.

Der Punkt innerhalb des Ausdrucks steht für ein beliebiges Zeichen. Es wird also nach einer Zeichenfolge gesucht, die aus einem beliebigen Zeichen besteht, gefolgt von `ein`.

Der erste Text enthält zwei solcher Zeichenfolgen. Die Position der ersten Zeichenfolge wird ausgegeben. Dabei handelt es sich um die Position 8, also die Stelle, an der das Wort `mein` beginnt.

Der zweite Text enthält keine Zeichenfolge, die zum Suchmuster passt. Die Methode `search()` liefert dann den Wert `-1`, siehe [Abbildung 2.1](#).

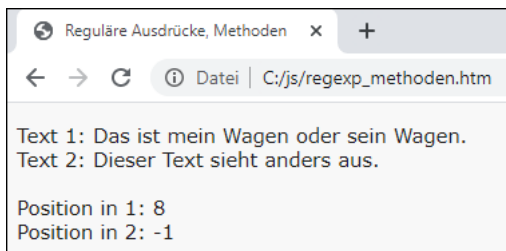


Abbildung 2.1 Methode »search()«

Es folgt der zweite Teil des Programms:

```
...
const passend = tx1.match(/.ein/);
if(passend)
    document.write("Erster Fund in 1: " + passend + "<br>");

const feld = tx1.match(/.ein/g);
if(feld)
{
    document.write("Alle Funde in 1: ");
    for(let i=0; i<feld.length; i++)
        document.write(feld[i] + " ");
}
document.write("<br><br>");

document.write(tx1.replace(/Wagen/g, "Auto") + "<br>");
document.write(tx2.replace(/Wagen/g, "Auto"));
</script></p>
</body></html>
```

Listing 2.2 Datei »regex_methoden.htm«, Teil 2 von 2

Mit dem gleichen Suchmuster wie zuvor die Methode `search()` wird die Methode `match()` aufgerufen. Sie liefert die erste Zeichenfolge, auf die das Suchmuster passt, zurück. Gibt es keine passende Zeichenfolge, wird `null` zurückgeliefert. Der Wahrheitswert von `null` ist `false`, daher würde im vorliegenden Beispiel keine Ausgabe erfolgen.

Möchten Sie alle Zeichenfolgen ermitteln, auf die das Suchmuster passt, müssen Sie am Ende des regulären Ausdrucks ein `g` notieren. Dies führt zu einer globalen Suche im gesamten Text. In diesem Fall wird ein Feld zurückgeliefert, in dem alle passenden Zeichenfolgen stehen. Gibt es keine passende Zeichenfolge, wird auch hier `null` zurückgeliefert.

Als Letztes wird die Methode `replace()` für beide Texte mit einem Suchmuster aufgerufen, das den Text `Wagen` enthält. Dieser Ausdruck wird, falls vorhanden, in beiden Texten durch die Zeichenfolge `Auto` ersetzt. Das kleine `g` sorgt wiederum für eine globale Suche. Es werden also alle Vorkommen ersetzt, nicht nur das erste.

Das Ergebnis für diesen Programmteil sehen Sie in [Abbildung 2.2](#).

Erster Fund in 1: mein
 Alle Funde in 1: mein sein

Das ist mein Auto oder sein Auto.
 Dieser Text sieht anders aus.

Abbildung 2.2 Methoden »match()« und »replace()«

In den folgenden Abschnitten wenden wir die verschiedenen regulären Ausdrücke nur für die Methode `match()` an. Sie können sie auch bei den anderen beiden Methoden einsetzen.

2.2 Suche nach Position

Im folgenden Programm wird untersucht, ob die gesuchte Zeichenfolge:

- ▶ an einer beliebigen Stelle im Text vorkommt
- ▶ am Anfang des Textes vorkommt, mithilfe des Zeichens `^`
- ▶ am Ende des Textes vorkommt, mithilfe des Zeichens `$`, oder
- ▶ genau dem Text entspricht, also sowohl am Anfang als auch am Ende des Textes vorkommt, mithilfe der beiden Zeichen `^` und `$`

Es folgt ein Beispielprogramm:

```
...
<body><p>
<script>
  let aus = "";
  if("keiner".match(/ein/)) aus += "1: keiner<br>";
  if("beide".match(/ein/)) aus += "1: beide<br>";

  if("eine".match(/^ein/)) aus += "2: eine<br>";
  if("kein".match(/^ein/)) aus += "2: kein<br>";

  if("kein".match(/ein$/)) aus += "3: kein<br>";
  if("eine".match(/ein$/)) aus += "3: eine<br>";

  if("ein".match(/^ein$/)) aus += "4: ein<br>";
  if("kein".match(/^ein$/)) aus += "4: kein<br>";
  if("eine".match(/^ein$/)) aus += "4: eine<br>";
  document.write(aus);
```

```
</script></p>
</body></html>
```

Listing 2.3 Datei »regex_position.htm«

Die regulären Ausdrücke werden unmittelbar auf die untersuchten Zeichenketten angewendet. Das Ergebnis der Methode `match()` wird mithilfe einer Verzweigung ermittelt. Es gibt nur dann eine Ausgabe des untersuchten Textes, wenn das Suchmuster gefunden wird.

Die Erläuterung der Ausgaben:

1. Die Zeichenfolge `ein` wird nur im Text `keiner` gefunden.
2. Die Zeichenfolge `ein` steht im Text `kein` nicht am Anfang.
3. Die Zeichenfolge `ein` steht im Text `eine` nicht am Ende.
4. Die Zeichenfolge `ein` steht nur im Text `ein` sowohl am Anfang als auch am Ende, entspricht also genau dem Text.

Das Ergebnis sehen Sie in [Abbildung 2.3](#).

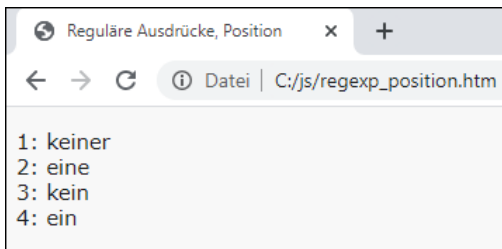


Abbildung 2.3 Position des Suchmusters

2.3 Suche nach Anzahl

Im folgenden Programm ist das Ergebnis davon abhängig, wie oft eine Zeichenfolge unmittelbar nacheinander in einem Text vorkommt:

- beliebig oft, also auch keinmal, mithilfe des Zeichens `*`
- mindestens einmal, mithilfe des Zeichens `+`,
- keinmal oder einmal, mithilfe des Zeichens `?`, oder
- so oft vorkommt wie gewünscht, also z. B. drei- bis fünfmal, mithilfe von geschweiften Klammern

Es folgt das Programm:

```
...
<body><p>
<script>
    let aus = "";
    if("ist an der".match(/ist a*/)) aus += "1: ist an der<br>";
    if("ist in der".match(/ist a*/)) aus += "1: ist in der<br>";
    if("in der".match(/ist a*/)) aus += "1: in der<br>";

    if("ist an der".match(/ist a+/)) aus += "2: ist an der<br>";
    if("ist in der".match(/ist a+/)) aus += "2: ist in der<br>";

    if("im Truck".match(/im To?r/)) aus += "3: im Truck<br>";
    if("im Tor".match(/im To?r/)) aus += "3: im Tor<br>";
    if("im Toor".match(/im To?r/)) aus += "3: im Toor<br>";

    if("im Truck".match(/im To{0,1}r/)) aus += "4: im Truck<br>";
    if("im Tor".match(/im To{0,1}r/)) aus += "4: im Tor<br>";
    if("im Toor".match(/im To{0,1}r/)) aus += "4: im Toor<br>";

    if("die Banane".match(/B(an){2,3}/)) aus += "5: die Banane<br>";
    if("das Band".match(/B(an){2,3}/)) aus += "5: das Band<br>";
    document.write(aus);
</script></p>
</body></html>
```

Listing 2.4 Datei »regexp_anzahl.htm«

Im Standardfall bezieht sich die gewünschte Häufigkeit auf ein einzelnes Zeichen unmittelbar vor der Angabe der Anzahl. Soll die Häufigkeit einer Zeichenfolge untersucht werden, muss diese in runden Klammern stehen.

Das erste Suchmuster enthält das Wort `ist`, gefolgt von einem Leerzeichen, eventuell gefolgt von keinem, einem oder mehreren `a`. Dieses Suchmuster passt auf die Texte `ist an der` und `ist in der`.

Beim zweiten Suchmuster ist mindestens ein `a` nach dem Leerzeichen Pflicht, eventuell mehrere unmittelbar nacheinander. Dies trifft nur auf den Text `ist an der` zu.

Beim dritten und beim vierten Suchmuster muss `im Tr` vorkommen, auch mit einem `o` zwischen `T` und `r`, aber nicht mit zwei oder mehr `o`.

Beim vierten und beim fünften Suchmuster wird eine bestimmte gewünschte Häufigkeit mithilfe von Zahlen ausgedrückt. Die erste Zahl stellt das Minimum, die zweite Zahl das Maximum dar. Das fünfte Suchmuster ist erfolgreich, falls nach einem B die Zeichenfolge an zwei- oder dreimal nacheinander vorkommt, so wie im Text Banane.

Das Ergebnis sehen Sie in [Abbildung 2.4](#).

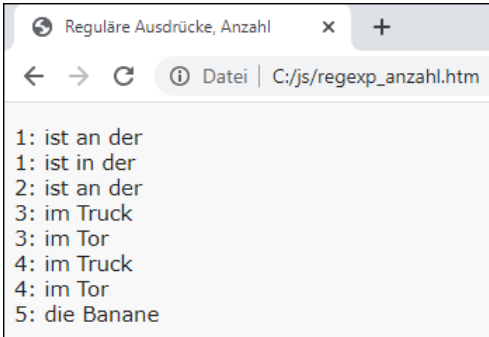


Abbildung 2.4 Anzahl von Vorkommen nacheinander

2.4 Logisches Oder

Ein Suchmuster kann auch Alternativen enthalten:

- ▶ für einzelne Zeichen, mithilfe von rechteckigen Klammern
- ▶ für Zeichenfolgen, die in runden Klammern stehen, getrennt durch das Zeichen |
- ▶ falls die Groß- und Kleinschreibung nicht beachtet wird, mithilfe des Zeichens i

Es folgt das Programm:

```
...
<body><p>
<script>
    let aus = "";
    if("ist keiner".match(/[km]eine/)) aus += "1: ist keiner<br>";
    if("ist meiner".match(/[km]eine/)) aus += "1: ist meiner<br>";
    if("ist seiner".match(/[km]eine/)) aus += "1: ist seiner<br>";

    if("ist an".match(/(an|aus)/)) aus += "2: ist an<br>";
    if("ist aus".match(/(an|aus)/)) aus += "2: ist aus<br>";
    if("ist nicht".match(/(an|aus)/)) aus += "2: ist nicht<br>";
```

```

if("ist An".match(/ist an/i)) aus += "3: ist An<br>";
if("Ist an".match(/ist an/i)) aus += "3: Ist an<br>";
if("ist Aus".match(/ist an/i)) aus += "3: ist Aus<br>";
document.write(aus);
</script></p>
</body></html>

```

Listing 2.5 Datei »regexp_oder.htm«

Das erste Suchmuster beginnt entweder mit einem `k` oder einem `m`, gefolgt von der Zeichenfolge `eine`. Dies passt nicht auf den Text `ist seiner`.

Beim zweiten Suchmuster muss entweder die Zeichenfolge `an` oder die Zeichenfolge `aus` vorkommen. Dies passt nicht auf den Text `ist nicht`.

Das dritte Suchmuster lautet `ist an`, unabhängig von der Groß- und Kleinschreibung. Dies passt auf die Texte `ist An` und `Ist an`.

Das Ergebnis sehen Sie in [Abbildung 2.5](#).

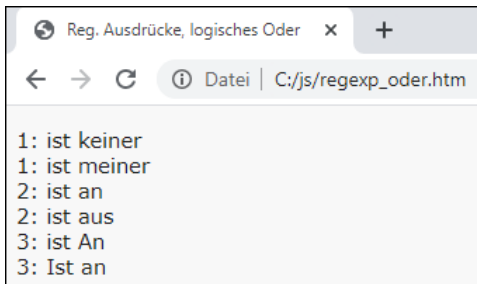


Abbildung 2.5 Logisches Oder

2.5 Gruppen von Zeichen

Bestimmte Zeichen lassen sich zu Gruppen zusammenfassen. Auf diese Weise erzeugen Sie ein Suchmuster, mit dessen Hilfe nach einem Zeichen aus der betreffenden Gruppe gesucht wird.

Es gibt unter anderem die Folgenden:

- ▶ die Gruppe aller Zeichen; es kann also ein beliebiges Zeichen vorkommen. Dies wird mit einem Punkt gekennzeichnet.
- ▶ eine Gruppe von kleinen Buchstaben ohne deutsche Umlaute oder das Eszett. Sie wird z. B. mit `[a-z]` oder `[f-p]` dargestellt.

- eine Gruppe von großen Buchstaben, ebenfalls ohne deutsche Umlaute. Sie wird z. B. mit [A-Z] oder [F-P] angegeben.
- eine Gruppe von Ziffern. Sie wird z. B. mit [0-9] oder [3-7] dargestellt.

Es folgt das Programm:

```
...
<body><p>
<script>
    let aus = "";
    if("ist mein".match(/ist .ein/)) aus += "1: ist mein<br>";
    if("ist sein".match(/ist .ein/)) aus += "1: ist sein<br>";
    if("ist ein".match(/ist .ein/)) aus += "1: ist ein<br>";

    if("kleiner".match(/[a-z]eine/)) aus += "2: kleiner<br>";
    if("Leinen".match(/[a-z]eine/)) aus += "2: Leinen<br>";

    if("Leinen".match(/[A-Z]eine/)) aus += "3: Leinen<br>";
    if("kleiner".match(/[A-Z]eine/)) aus += "3: kleiner<br>";

    if("Hauptstr. 7a".match(/[0-9]/)) aus += "4: Hauptstr. 7a<br>";
    if("Hauptstr.".match(/[0-9]/)) aus += "4: Hauptstr.<br>";
    if("Hauptstr. 5".match(/[0-27-9]/)) aus += "4: Hauptstr. 5<br>";
    document.write(aus);
</script></p>
</body></html>
```

Listing 2.6 Datei »regexp_gruppe.htm«

Das erste Suchmuster besteht aus dem Text `ist`, gefolgt von einem Leerzeichen, gefolgt von einem beliebigen Zeichen, gefolgt vom Text `ein`. Das passt nicht auf den Text `ist ein`, weil das beliebige Zeichen fehlt.

Beim zweiten Suchmuster wird erwartet, dass ein kleiner Buchstabe vor dem Text `eine` steht. Dies ist bei dem Text `Leinen` nicht der Fall.

Beim dritten Suchmuster wird erwartet, dass ein großer Buchstabe vor dem Text `eine` steht. Dies ist bei dem Text `kleiner` nicht der Fall.

Bei einem Text für das vierte Suchmuster muss in den beiden ersten Fällen mindestens eine beliebige Ziffer enthalten sein, im dritten Fall mindestens eine Ziffer aus dem Bereich von 0 bis 2 oder aus dem Bereich 7 bis 9.

Das Ergebnis sehen Sie in [Abbildung 2.6](#).

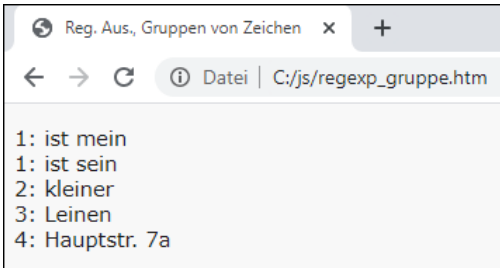


Abbildung 2.6 Gruppen von Zeichen

2.6 Suche nach Sonderzeichen

Es wurden bereits zahlreiche Sonderzeichen mitsamt ihrer Funktion innerhalb eines regulären Ausdrucks vorgestellt. Was machen Sie, wenn Sie nach einem dieser Sonderzeichen suchen müssen? Auch das geht, entweder nach einem Backslash\ oder innerhalb von rechteckigen Klammern.

Es folgt das Programm:

```
...
<body><p>
<script>
    let aus = "";
    if("27. Juli".match(/27\. Juli/)) aus += "1: 27. Juli<br>";
    if("27 Juli".match(/27\. Juli/)) aus += "1: 27 Juli<br>";

    if("kann. Das".match(/kann[.] Das/)) aus += "2: kann. Das<br>";
    if("kann? Das".match(/kann[.] Das/)) aus += "2: kann? Das<br>";
    if("kann! Das".match(/kann[.] Das/)) aus += "2: kann! Das<br>";
    document.write(aus);
</script></p>
</body></html>
```

Listing 2.7 Datei »regexp_sonderzeichen.htm«

Das erste Suchmuster enthält einen Punkt nach dem Tag des Monats. Dies wird von 27 Juli nicht erfüllt.

Beim zweiten Suchmuster darf am Ende des Satzes ein Punkt oder ein Fragezeichen stehen. Innerhalb von rechteckigen Klammern verlieren diese Sonderzeichen ihre Bedeutung. Der Satz, der mit einem Ausrufezeichen endet, passt nicht zum Suchmuster.

Das Ergebnis sehen Sie in [Abbildung 2.7](#).

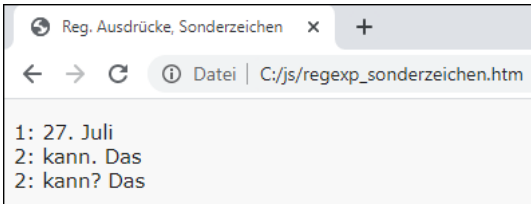


Abbildung 2.7 Suche nach Sonderzeichen

2.7 Logische Negation

Besonders bei der Suche nach Zeichen aus einem bestimmten Bereich (Buchstaben oder Ziffern) kann die logische Negation mithilfe des Zeichens `^` eingesetzt werden: Eine Suche ist genau dann erfolgreich, wenn ein Zeichen gefunden wird, das nicht aus dem angegebenen Bereich stammt.

Nachfolgend ein Beispielprogramm:

```
...
<body><p>
<script>
    let aus = "";
    if("xy".match(/[^\x]/)) aus += "1: xy<br>";
    if("xxx".match(/[^\x]/)) aus += "1: xxx<br>";

    if("15a".match(/[^\0-9]/)) aus += "2: 15a<br>";
    if("78B".match(/[^\0-9]/)) aus += "2: 78B<br>";
    if("64".match(/[^\0-9]/)) aus += "26: 64<br>";
    document.write(aus);
</script></p>
</body></html>
```

Listing 2.8 Datei »regexp_negation.htm«

Das erste Suchmuster trifft zu, falls mindestens ein Zeichen kein `x` ist. Dies ist nur für `xy` der Fall. Beim zweiten Suchmuster muss mindestens ein Zeichen keine Ziffer von `0`

bis 9 sein. Der Text 64 erfüllt diese Bedingung nicht. Das Ergebnis sehen Sie in [Abbildung 2.8](#).

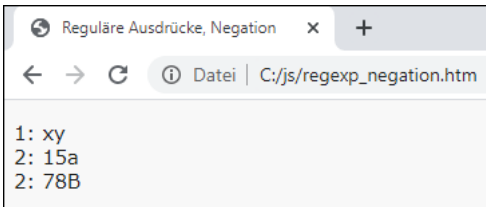


Abbildung 2.8 Logische Negation

2.8 Beispiele

Es folgen drei Beispiele mit komplexeren Suchmustern, wie sie für Eingabefelder in einem Formular vorkommen können:

- Es soll eine deutsche Postleitzahl eingetragen werden. Diese besteht aus genau fünf Ziffern, jeweils von 0 bis 9.
- Es soll ein deutsches Datum eingegeben werden. Dieses soll aus zwei Ziffern für den Tag, einem Punkt, zwei Ziffern für den Monat, wiederum einem Punkt und vier Ziffern für das Jahr bestehen. Die erste Ziffer für das Jahr darf keine 0 sein.
- Es soll ein deutsches Kfz-Kennzeichen eingetragen werden. Dieses besteht aus ein bis drei großen Buchstaben, einem Bindestrich, ein bis zwei weiteren großen Buchstaben, einem Leerzeichen und einer bis vier Ziffern. Die erste Ziffer darf keine 0 sein.

Das Beispielprogramm:

```
...
<body><p>
<script>
    let aus = "";
    const rePlz = /^[0-9]{5}$/;
    if("01644".match(rePlz)) aus += "01644<br>";
    if( "5400".match(rePlz)) aus += "5400<br>";

    const reDatum = /^[0-9]{2}\.){2}[1-9][0-9]{3}$/;
    if("30.09.2024".match(reDatum)) aus += "30.09.2024<br>";
    if( "5.11.2024".match(reDatum)) aus += "5.11.2024<br>";
    if("30.09 2024".match(reDatum)) aus += "30.09 2024<br>";
    if( "30.09.24".match(reDatum)) aus += "30.09.24<br>";
```

```

const reKfz = /^[A-Z]{1,3}-[A-Z]{1,2} [1-9][0-9]{0,3}$/;
if("HST-PU 4215".match(reKfz)) aus += "HST-PU 4215<br>";
if("M-KP 5".match(reKfz)) aus += "M-KP 5<br>";
if("NU KT 15".match(reKfz)) aus += "NU KT 15<br>";
if("NU-KT15".match(reKfz)) aus += "NU-KT15<br>";
document.write(aus);
</script></p>
</body></html>

```

Listing 2.9 Datei »regexp_beispiel.htm«

Die Postleitzahl 5400 besteht nur aus vier Ziffern.

Bei der zweiten Datumsangabe fehlt eine Ziffer beim Tag. Bei der dritten Datumsangabe steht kein Punkt nach dem Monat. Die vierte Datumsangabe hat nur zwei Stellen für das Jahr.

Das dritte Kfz-Kennzeichen enthält keinen Bindestrich, das vierte Kfz-Kennzeichen kein Leerzeichen.

Das Ergebnis sehen Sie in [Abbildung 2.9](#).

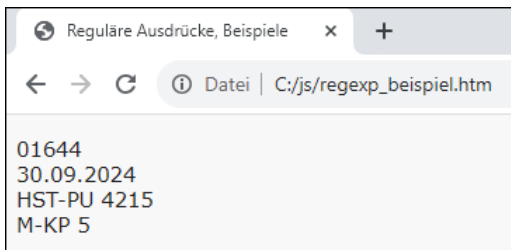


Abbildung 2.9 Beispiele Postleitzahl, Datum, Kfz-Kennzeichen

2.9 Eingabe von Zahlen

Das Programm *regexp_komma.htm* ist eine Erweiterung des Programms *zahlen_eingabe.htm* aus Buchabschnitt 2.2.3, »Eingabe von Zahlen«. Es ermöglicht mithilfe von regulären Ausdrücken die Eingabe und Ausgabe von Zahlen mit einem Dezimalkomma. Den Ablauf des Programms sehen Sie in [Abbildung 2.10](#) bis [Abbildung 2.12](#).

Auf dieser Seite wird Folgendes angezeigt

Zahl 1:

Ok Abbrechen

Abbildung 2.10 Eingabe der ersten Zahl mit Dezimalkomma

Auf dieser Seite wird Folgendes angezeigt

Zahl 2:

Ok Abbrechen

Abbildung 2.11 Eingabe der zweiten Zahl mit Dezimalkomma

Ein- und Ausgabe mit Komma x +

← → ↻ Datei | C:/js/regexp_komma.htm

2,5 + 8,75 = 11,25

Abbildung 2.12 Ausgabe des Ergebnisses mit Dezimalkomma

Es folgt das Programm:

```
...
<body>
  <script>
    const eingabe1 = prompt("Zahl 1:");
    const z1 = parseFloat(eingabe1.replace(/,/,"."));
    const eingabe2 = prompt("Zahl 2:");
    const z2 = parseFloat(eingabe2.replace(/,/,"."));
    const ergebnis = (z1 + z2).toString().replace(/\./,"");
    document.write(eingabe1 + " + " + eingabe2 + " = " + ergebnis);
  </script>
</body></html>
```

Listing 2.10 Datei »regexp_komma.htm«

Die Eingaben werden in den beiden Variablen `eingabe1` und `eingabe2` gespeichert. In diesen beiden Zeichenketten wird mithilfe der Methode `replace()` das Komma durch einen Punkt ersetzt. Anschließend werden die beiden Zeichenketten mithilfe der Funktion `parseFloat()` in eine Zahl umgewandelt.

Das Rechenergebnis wird als eine Zahl ermittelt. Sie wird mithilfe der Methode `toString()` in eine Zeichenkette umgewandelt. In dieser Zeichenkette wird der Punkt durch ein Komma ersetzt. In der Ausgabe des Ergebnisses erscheinen die ursprünglich eingegebenen Zeichenketten. Dabei ist zu beachten, dass der Punkt für reguläre Ausdrücke ein Sonderzeichen darstellt.

Bonuskapitel 3

Beispielprojekte

Die Inhalte dieses Kapitels gehören thematisch an das Ende des Buchkapitels 6, »Standardobjekte nutzen«. Sie enthalten einige Beispielprojekte mit Elementen, die Sie an unterschiedlicher Stelle kennengelernt haben: DOM-Methoden, Formulare, das `location`-Objekt, ein- und mehrdimensionale Felder sowie Zeichenketten.

3.1 Senden von Dokument zu Dokument

Das `location`-Objekt besitzt die Eigenschaft `search`. Diese kann eine sogenannte Suchzeichenkette enthalten, die beim Aufruf eines Dokuments mithilfe des Zeichens `?` an die Adresse des Dokuments angehängt werden kann.

Einzelne Informationen innerhalb der Suchzeichenkette werden durch das Zeichen `&` voneinander getrennt. Sie können aus einem Formular stammen und wie Parameter einer Funktion verwendet werden. Damit ist eine Übergabe von Informationen von einem Dokument zu einem anderen Dokument möglich, ohne ein Programm in einer Webserver Sprache wie PHP nutzen zu müssen. Einziger Nachteil: Die übergebenen Informationen sind für jeden sichtbar.

Ein Aufruf von *Google Maps* als Beispiel für eine Suchzeichenkette:

<http://maps.google.de/maps?q=Bonn&z=18>

Anschließend wird eine Karte für den Suchbegriff `Bonn` und mit dem Zoomfaktor 18 angezeigt, unter Umgehung der Google-Maps-Startseite.

Im folgenden Programm können nach dem Aufruf insgesamt drei Zahlen in einem Formular eingetragen werden, siehe [Abbildung 3.1](#). Diese drei Zahlen werden zu einer Suchzeichenkette zusammengefasst, die an ein zweites Dokument übergeben wird. Dort werden die drei Zahlen wieder aus der Suchzeichenkette entnommen und miteinander addiert, siehe [Abbildung 3.2](#).

Abbildung 3.1 Informationen senden

Abbildung 3.2 Daten empfangen und verarbeiten

Das Programm zum Senden sieht wie folgt aus:

```
... <head> ...
  <script>
    function senden()
    {
      location.href = "location_ziel.htm?z1="
        + document.getElementById("idZahl1").value + "&z2="
        + document.getElementById("idZahl2").value + "&z3="
        + document.getElementById("idZahl3").value;
    }
  </script>
</head>
<body>
  <p><input id="idZahl1" size="10"> Zahl 1</p>
  <p><input id="idZahl2" size="10"> Zahl 2</p>
  <p><input id="idZahl3" size="10"> Zahl 3</p>
```



```

<p><input id="idSenden" type="button" value="Senden"></p>
<script>
    document.getElementById("idSenden").addEventListener("click", senden);
</script>
</body></html>

```

Listing 3.1 Datei »location_quelle.htm«

Nach dem Eintragen und dem Betätigen der Schaltfläche SENDEN wird die Funktion `senden()` aufgerufen. Darin bekommt die Eigenschaft `href` einen neuen Wert. Mithilfe der Zeichen `?` und `&` sowie der Inhalte der Formularfelder wird die Suchzeichenkette zusammengesetzt.

Der Code in der Zielfeile sieht wie folgt aus:

```

...
<body><p>
    <script>
        document.write("href: " + location.href + "<br>");
        document.write("search: " + location.search + "<br>");
        const suchfeld = location.search.split("&");

        let summe = 0;
        for(let i=0; i<suchfeld.length; i++)
        {
            if(i==0)
                suchfeld[0] = suchfeld[0].substr(1);
            const teile = suchfeld[i].split("=");
            document.write(teile[0] + ": " + teile[1] + "<br>");
            if(!isNaN(teile[1]))
                summe += parseFloat(teile[1]);
        }
        document.write("Summe: " + summe);
    </script></p>
</body></html>

```

Listing 3.2 Datei »location_ziel.htm«

Die vollständige Adresse sehen Sie im Adressfeld in [Abbildung 3.2](#). Zur Kontrolle wird sie noch einmal im Dokument ausgegeben, mithilfe der Eigenschaft `href`. Die Eigenschaft

search enthält die Suchzeichenkette. Sie wird mithilfe der String-Methode `split()` zu einem Feld von Einzelinformationen zerlegt.

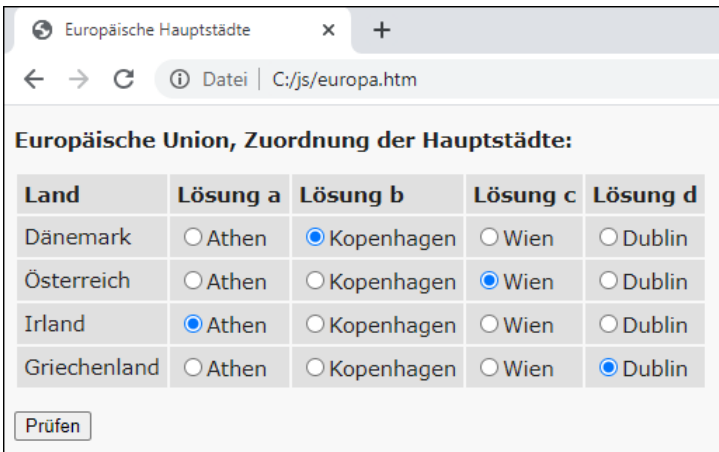
Im ersten Feldelement steckt noch das Fragezeichen. Sie können es mithilfe der String-Methode `substr()` eliminieren. Anschließend wird jede Einzelinformation in zwei Teile zerlegt, wiederum mithilfe von `split()`. Nun stehen uns die Eigenschaft-Wert-Paare zur Verfügung.

Mithilfe der Methode `isNaN()` wird geprüft, ob es sich bei dem Wert um eine Zahl handelt. Ist das der Fall, wird der Wert in eine Zahl umgewandelt, die zur Summe hinzuaddiert wird.

3.2 Lernspiel: Hauptstädte der Europäischen Union

Im Downloadpaket zum Buch finden Sie das Programm *europa.htm* mit vielen erläuternden Kommentaren. In einem Formular werden vier zufällig gewählte Länder der Europäischen Union und jeweils alle zugehörigen Hauptstädte angezeigt.

Durch Auswahl der zugehörigen Option soll jeweils die richtige Hauptstadt ausgewählt werden. Nach der Betätigung der Schaltfläche PRÜFEN (siehe [Abbildung 3.3](#)) werden die verschiedenen Auswahlen geprüft und es werden die richtigen Lösungen zum Erlernen angezeigt (siehe [Abbildung 3.4](#)).



Europäische Hauptstädte

← → ↻ ⓘ Datei | C:/js/europa.htm

Europäische Union, Zuordnung der Hauptstädte:

Land	Lösung a	Lösung b	Lösung c	Lösung d
Dänemark	<input type="radio"/> Athen	<input checked="" type="radio"/> Kopenhagen	<input type="radio"/> Wien	<input type="radio"/> Dublin
Österreich	<input type="radio"/> Athen	<input type="radio"/> Kopenhagen	<input checked="" type="radio"/> Wien	<input type="radio"/> Dublin
Irland	<input checked="" type="radio"/> Athen	<input type="radio"/> Kopenhagen	<input type="radio"/> Wien	<input type="radio"/> Dublin
Griechenland	<input type="radio"/> Athen	<input type="radio"/> Kopenhagen	<input type="radio"/> Wien	<input checked="" type="radio"/> Dublin

Abbildung 3.3 Lernspiel, Hauptstädte der Europäischen Union

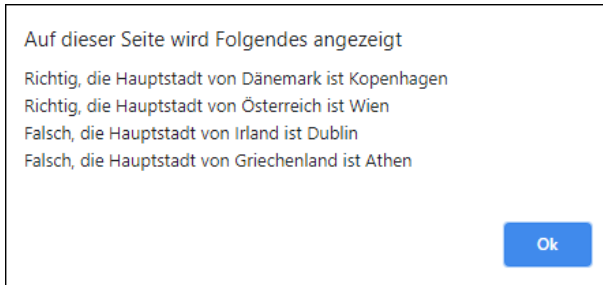


Abbildung 3.4 Bewertung und Lernen

3.3 Änderung eines Auswahlménüs

Im Downloadpaket zum Buch finden Sie das Programm *menu_wechsel.htm* mit vielen erläuternden Kommentaren. In einem Formular werden zwei Auswahlménüs angezeigt. Im ersten Auswahlménü steht eine Reihe von Ländern. Das zweite Auswahlménü enthält eine Reihe von Städten innerhalb des ausgewählten Landes.

Die ausgewählte Stadt wird in einem eigenen Absatz angezeigt, siehe [Abbildung 3.5](#). Sobald ein anderes Land ausgewählt wird, ändert sich der Inhalt des zweiten Auswahlménüs. Es enthält dann nur noch Städte des neu ausgewählten Landes, siehe [Abbildung 3.6](#).

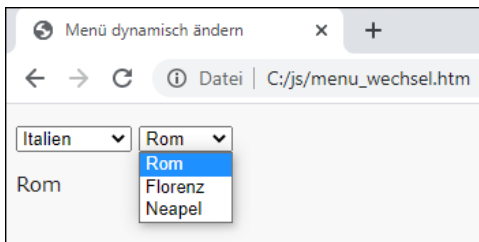


Abbildung 3.5 Änderung eines Auswahlménüs

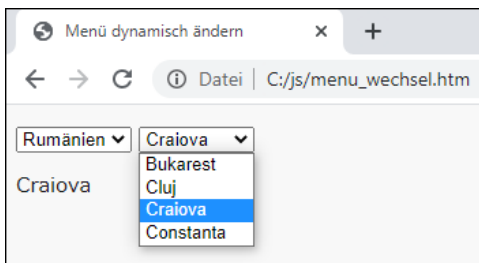


Abbildung 3.6 Nach Änderung des Landes

Bonuskapitel 4

Verschiedene Formen

Die Inhalte dieses Kapitels gehören thematisch an das Ende des Buchkapitels 10, »Drei-dimensionale Grafiken und Animationen mit Three.js«. Im Downloadpaket zum Buch finden Sie das Programm *th_wechsel.htm* mit vielen Kommentaren.

Mithilfe der Bedienleiste, die Sie in [Abbildung 4.1](#) sehen, können Sie:

- ▶ verschiedene vorgefertigte Geometrien betrachten
- ▶ alle 3D-Objekte um die verschiedenen Achsen drehen
- ▶ alle 3D-Objekte entlang der verschiedenen Achsen verschieben

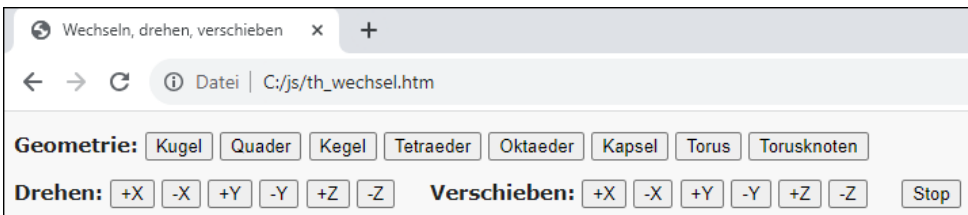


Abbildung 4.1 Bedienleiste

Sie sehen alle Geometrien in einer Gitternetzdarstellung. Dies dient der Verdeutlichung des inneren Aufbaus der 3D-Objekte, die aus vielen einzelnen Dreiecksflächen aufgebaut sind.

Die Geometrien Kugel, Kegel, Kapsel, Torus und Torusknoten bestehen zum Teil aus Rundungen. Diese werden aus Segmenten gebildet. Je mehr Segmente verwendet werden, desto »runder« werden die Teil-Geometrien, desto langsamer werden allerdings Aufbau oder Animation. Werden bei Start- und Endwinkel jeweils die Standardwerte genommen (0 und 2π), sind die Teil-Geometrien vollständig.

Alle Parameter der Geometrien besitzen Standardwerte. Winkel werden im Bogenmaß angegeben. Ein Winkel in Grad muss zur Umrechnung in das Bogenmaß mit dem Faktor $\pi/180$ multipliziert werden.

4.1 Kugel

Mithilfe eines ThreeJS-Objekts des Typs `SphereGeometry` wird eine Kugel gebildet, siehe [Abbildung 4.2](#). Die Gitternetzdarstellung ähnelt der Darstellung der Erdkugel mit Längen- und Breitengraden.

Zur Erzeugung dienen die folgenden sieben Parameter:

1. Radius, Standardwert 1
2. Anzahl der Segmente entlang der Breite, also Anzahl der sichtbaren Längengrade in westlicher oder östlicher Richtung, Standardwert 32, Minimum ist 3
3. Anzahl der Segmente entlang der Höhe, also Anzahl der sichtbaren Breitengrade in nördlicher oder südlicher Richtung, Standardwert 16, Minimum ist 2
4. Startwinkel im Bogenmaß, entlang der Breite, Standardwert 0
5. Endwinkel im Bogenmaß entlang der Breite, Standardwert 2π
6. Startwinkel im Bogenmaß entlang der Höhe, Standardwert 0
7. Endwinkel im Bogenmaß entlang der Höhe, Standardwert 2π

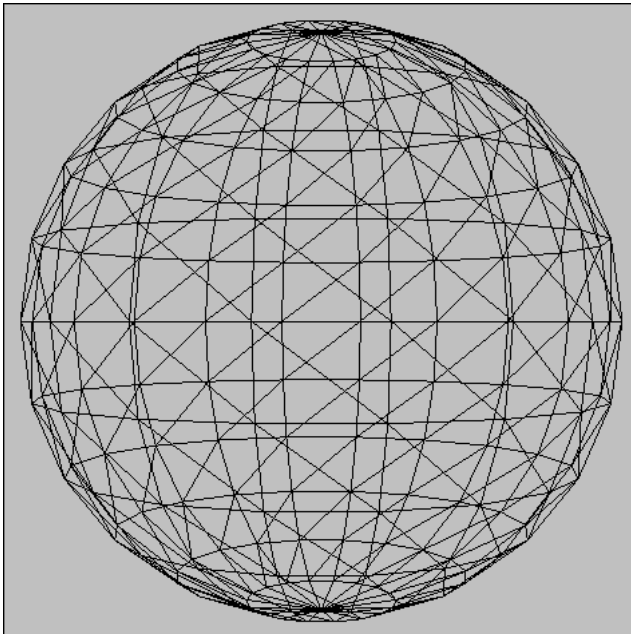


Abbildung 4.2 Kugel

4.2 Quader

Der Quader diene bereits als Einführungsbeispiel des zugehörigen Buchkapitels. Er wird mithilfe eines ThreeJS-Objekts des Typs `BoxGeometry` gebildet, siehe [Abbildung 4.3](#). Er besteht aus sechs Rechtecken.

Zur Erzeugung dienen die folgenden sechs Parameter:

1. Breite entlang der x-Achse, Standardwert 1
2. Höhe entlang der y-Achse, Standardwert 1
3. Tiefe entlang der z-Achse, Standardwert 1
4. Anzahl der Segmente in der Breite, Standardwert 1
5. Anzahl der Segmente in der Höhe, Standardwert 1
6. Anzahl der Segmente in der Tiefe, Standardwert 1

Stimmen die Breite, die Höhe und die Tiefe überein, handelt es sich um den Spezialfall eines Würfels.

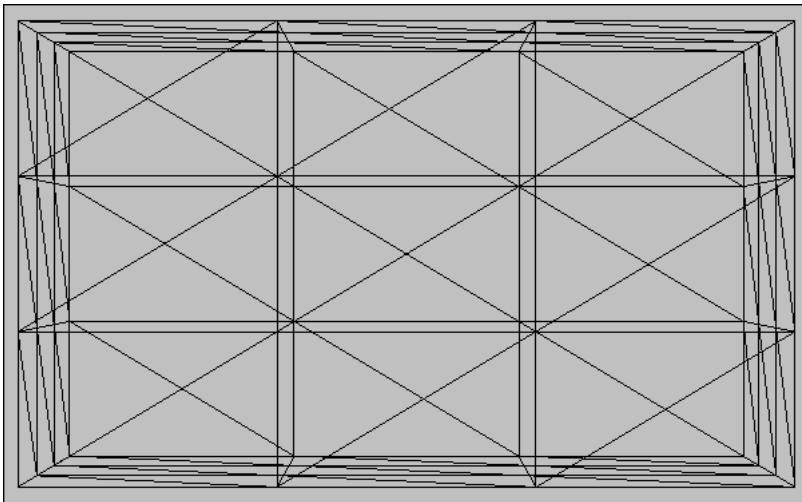


Abbildung 4.3 Quader

4.3 Kegel

Ein Kegel wird mithilfe eines ThreeJS-Objekts des Typs `CylinderGeometry` gebildet, siehe [Abbildung 4.4](#). Er besteht aus zwei kreisförmigen Deckelflächen (oben und unten) und einer Mantelfläche.

Zur Erzeugung dienen die folgenden acht Parameter:

1. Radius der oberen Deckelfläche, Standardwert 1
2. Radius der unteren Deckelfläche, Standardwert 1
3. Höhe der Mantelfläche entlang der y-Achse, Standardwert 1
4. Anzahl der Segmente der beiden Deckelflächen, entlang des Kreisumfangs, Standardwert 32
5. Anzahl der Segmente der Mantelfläche entlang der Höhe, Standardwert 1
6. Boolescher Parameter, Standardwert `false` = Deckelflächen sind sichtbar, `true` = Deckelflächen sind nicht sichtbar.
7. Startwinkel im Bogenmaß entlang des Kreisumfangs, Standardwert 0
8. Endwinkel im Bogenmaß entlang des Kreisumfangs, Standardwert 2π

Stimmen die beiden Radien überein, handelt es sich um den Spezialfall eines Kreiszylinders.

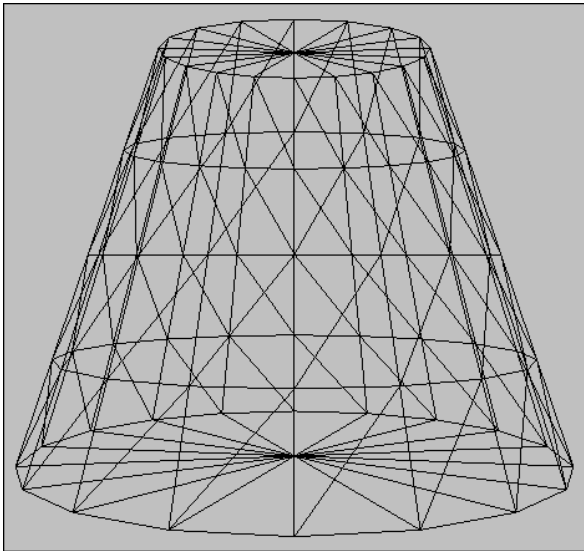


Abbildung 4.4 Kegel

4.4 Tetraeder

Ein Tetraeder wird mithilfe eines ThreeJS-Objekts des Typs `TetrahedronGeometry` gebildet, siehe [Abbildung 4.5](#). Er besteht aus vier Dreiecken. Zur Erzeugung dienen die beiden folgenden Parameter:

1. Radius, Standardwert 1
2. Anzahl der zusätzlichen Knoten, Standardwert 0, bei einem Wert größer als 0 ist es kein Tetraeder mehr.

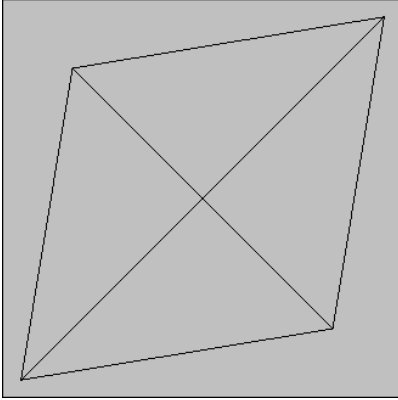


Abbildung 4.5 Tetraeder

4.5 Oktaeder

Ein Oktaeder wird mithilfe eines ThreeJS-Objekts des Typs `OktahedronGeometry` gebildet, siehe [Abbildung 4.6](#). Er besteht aus acht Dreiecken. Zur Erzeugung dienen die beiden folgenden Parameter:

1. Radius, Standardwert 1
2. Anzahl der zusätzlichen Knoten, Standardwert 0, bei einem Wert größer als 0 ist es kein Oktaeder mehr.

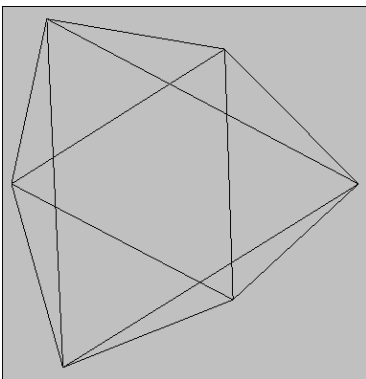


Abbildung 4.6 Oktaeder

4.6 Kapsel

Eine Kapsel wird mithilfe eines ThreeJS-Objekts des Typs `CapsuleGeometry` gebildet, siehe [Abbildung 4.7](#). Sie besteht aus einem Kreiszylinder als Mittelstück mit zwei Halbkugeln als Deckel.

Zur Erzeugung dienen die folgenden vier Parameter:

1. Radius der Deckel, Standardwert 1
2. Länge des Mittelstücks, Standardwert 1
3. Halbe Anzahl der Segmente der beiden Deckelflächen, entlang der Höhe, Standardwert 4, ein Wert von 4 ergibt 8 Segmente an jedem Deckel
4. Anzahl der Segmente entlang des Kreisumfangs, Standardwert 8

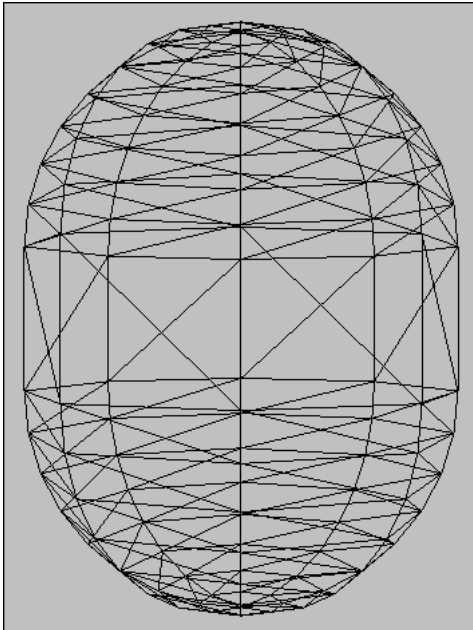


Abbildung 4.7 Kapsel

4.7 Torus

Die Form eines Torus ähnelt einem Schlauch (englisch: *tube*). Ein Torus wird mithilfe eines ThreeJS-Objekts des Typs `TorusGeometry` gebildet, siehe [Abbildung 4.8](#). Zur Erzeugung dienen die folgenden fünf Parameter:

1. Außenradius des Torus, Standardwert 1
2. Durchmesser des Torus, Standardwert 0.4
3. Anzahl der radialen Segmente, Standardwert 12
4. Anzahl der tubularen Segmente, Standardwert 48
5. Winkel im Bogenmaß, Standardwert 2π

Der Außenradius sollte mindestens so groß sein wie der Durchmesser. Zum besseren Verständnis sollten Sie Parameter 3 und 4 einige Male ändern und die Auswirkungen betrachten.

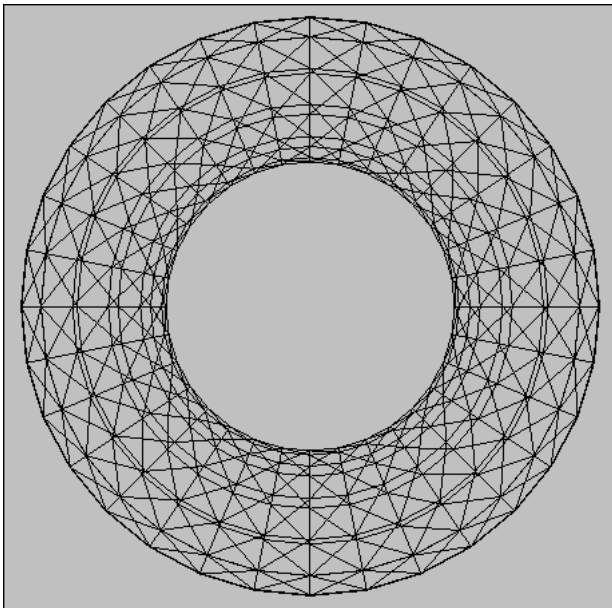


Abbildung 4.8 Torus

4.8 Torusknoten

Bei einem Torusknoten (siehe [Abbildung 4.9](#)) handelt es sich um einen dreidimensional verknoteten Torus, auf den bestimmte mathematische Bedingungen zutreffen, siehe <https://de.wikipedia.org/wiki/Torusknoten>.

Zur Erzeugung dienen die folgenden sechs Parameter:

1. Radius des Torus, Standardwert 1
2. Radius des Schlauchs, Standardwert 0.4

3. Anzahl der tubularen Segmente, Standardwert 64
4. Anzahl der radialen Segmente, Standardwert 8
5. Der Wert p bestimmt, wie oft sich die Geometrie um ihre Rotationssymmetrieachse windet, Standardwert 2
6. Der Wert q bestimmt, wie oft sich die Geometrie um einen Kreis im Inneren des Torus windet, Standardwert 3

Zum besseren Verständnis sollten Sie die einzelnen Parameter einige Male ändern und die Auswirkungen betrachten.

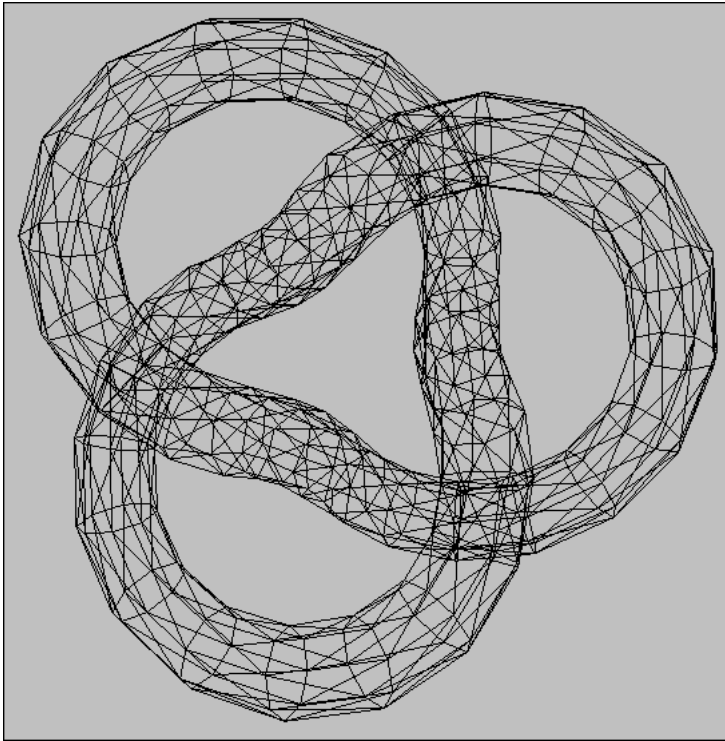


Abbildung 4.9 Torusknoten

Bonuskapitel 5

Dokumente mit mehreren Seiten

Die Inhalte dieses Kapitels gehören thematisch an das Ende des Buchkapitels 12, »Mobile Apps mit Onsen UI«. Es gibt verschiedene Möglichkeiten zur Aufteilung und Darstellung eines Dokuments, das mehrere Seiten umfasst. Sie können mit Tabs in einer Tableiste, mit einem Navigator oder mit einem Karussell arbeiten.

5.1 Aufteilung von Seiten mit Tabs

Die einzelnen Unterseiten eines Dokuments können mithilfe von Tabs erreicht werden, die in einer Tableiste stehen.

Jede Unterseite befindet sich in einer Komponente des Typs `ons-page`. Diese Komponente steht entweder in einer externen Datei oder in einem `template-Container`. Letztergenannte sind in HTML zur Speicherung von Inhalten vorgesehen, die nicht nach dem Laden eines Dokuments, sondern erst nach einer Anforderung angezeigt werden.

Zwischen den einzelnen Seiten kann entweder durch Antippen der Tabs oder durch seitliches Wischen gewechselt werden. Ein Beispiel mit drei Tabs, die zu drei verschiedenen Seiten führen, sehen Sie in [Abbildung 5.1](#).

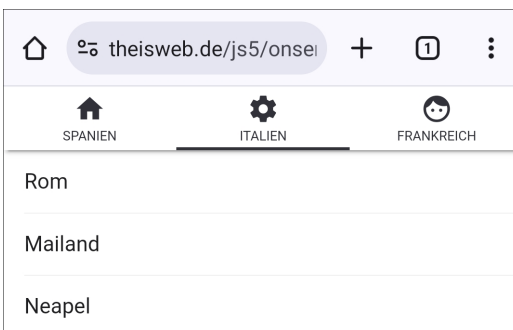


Abbildung 5.1 Tableiste mit drei Tabs

Dazu das Programm:

```
...
<body>
<ons-page>
  <ons-tabbar swipeable position="top">
    <ons-tab page="onsen_tab_spa.htm" label="Spanien"
      icon="md-home"></ons-tab>
    <ons-tab page="onsen_tab_ita.htm" label="Italien"
      icon="md-settings" active></ons-tab>
    <ons-tab page="onsen_tab_fra.htm" label="Frankreich"
      icon="md-face"></ons-tab>
  </ons-tabbar>
</ons-page>

<template id="onsen_tab_spa.htm">
  <ons-page>
    <ons-list>
      <ons-list-item>Madrid</ons-list-item>
      <ons-list-item>Barcelona</ons-list-item>
      <ons-list-item>Sevilla</ons-list-item>
    </ons-list>
  </ons-page>
</template>

<template id="onsen_tab_ita.htm">
  <ons-page>
    <ons-list>
      <ons-list-item>Rom</ons-list-item>
      <ons-list-item>Mailand</ons-list-item>
      <ons-list-item>Neapel</ons-list-item>
    </ons-list>
  </ons-page>
</template>
</body></html>
```

Listing 5.1 Datei »onsen_tab.htm«

Das Dokument beinhaltet eine Komponente des Typs `ons-page`. Diese Hauptseite beinhaltet die Tableiste, die zum Wechsel zwischen den einzelnen Unterseiten dient. Die Tableiste ist eine Komponente des Typs `ons-tabbar`. Das Attribut `swipeable` ermöglicht

das Wischen. Lassen Sie das Attribut `position` weg, wird die Tableiste am unteren Rand platziert. Im vorliegenden Programm wird sie am oberen Rand platziert.

Innerhalb der Tableiste stehen mehrere Tabs, die mithilfe von Komponenten des Typs `ons-tab` erzeugt werden und in der Darstellung jeweils die gleiche Breite einnehmen. Der Wert des Attributs `page` verweist auf die einzelnen Unterseiten. Diese stehen entweder innerhalb eines `template`-Containers oder in einer externen Datei.

Die Attribute `label` und `icon` dienen zur Kennzeichnung eines Tabs. Das Attribut `active` wird für denjenigen Tab gesetzt, der zum Start des Dokuments vorausgewählt ist und dessen zugehörige Unterseite angezeigt wird.

Im vorliegenden Beispiel beinhalten die Unterseiten jeweils eine Liste mit Einträgen. Die ersten beiden Unterseiten stehen in `template`-Containern. Die Verknüpfung zum Attribut `page` des zugehörigen Tabs wird über das Attribut `id` hergestellt. Die dritte Unterseite steht in einer externen Datei. Hier wird die Verknüpfung zum Attribut `page` des zugehörigen Tabs über den Dateinamen hergestellt. Es folgt der Inhalt der externen Datei:

```
<ons-page>
  <ons-list>
    <ons-list-item>Paris</ons-list-item>
    <ons-list-item>Lyon</ons-list-item>
    <ons-list-item>Marseille</ons-list-item>
  </ons-list>
</ons-page>
```

Listing 5.2 Datei »onsen_tab_fra.htm«

Nach dem Antippen eines Tabs erscheint die zugehörige Seite, siehe [Abbildung 5.2](#).



Abbildung 5.2 Nach Antippen des Tabs »Frankreich«

5.2 Hierarchie von Seiten mit einem Navigator

Ein Navigator dient dazu, mehrere Unterseiten eines Dokuments in einer festgelegten Hierarchie anzusteuern. Dabei handelt es sich um eine häufig verwendete Technik beim Aufbau von Dokumenten, die umfangreiche Informationen und Aktionsmöglichkeiten bietet. Ähnlich wie bei einer Tableiste stehen die Unterseiten innerhalb eines `template-Containers` oder in einer externen Datei.

Nach dem Aufruf des Programms erscheint zunächst eine Startseite. Sie befindet sich auf der obersten Ebene der Seiten. Von dort aus kann man, typischerweise über die Elemente einer Liste, auf die nächste Ebene gelangen, also auf untergeordnete Seiten. Von dort aus kann man wiederum auf die nächste Ebene gelangen usw. Für die schrittweise Rückkehr zur obersten Ebene mit der Startseite wird eine spezielle Zurück-Schaltfläche genutzt. Beim Wechsel zwischen den Ebenen können unterschiedliche Arten von Animationen zum Einsatz kommen.

In [Abbildung 5.3](#) sehen Sie eine Startseite, von der sechs weitere Seiten in zwei untergeordneten Ebenen angesteuert werden können. Nur die Startseite befindet sich in der Hauptdatei. Die sechs Unterseiten der zweiten und der dritten Ebene stehen in externen Dateien.

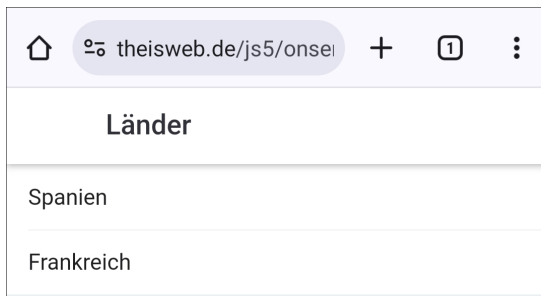


Abbildung 5.3 Navigator, Startseite

Dazu das Programm:

```
...
<body>
<ons-navigator id="nvg" animation="slide"
  page="onsen_nav_land.htm"></ons-navigator>
<script>
```

```

const ziel = ["onsen_nav_spa.htm", "onsen_nav_fra.htm",
  "onsen_nav_spa_mad.htm", "onsen_nav_spa_bar.htm",
  "onsen_nav_fra_par.htm", "onsen_nav_fra_mar.htm"];

function weiter(nr)
{
  document.getElementById("nvg").pushPage(ziel[nr]);
}
</script>
</body></html>

```

Listing 5.3 Datei »onsen_nav.htm«

Die Komponente des Typs `ons-navigator` dient zur Steuerung der Navigation. Ihr Attribut `id` wird zum Wechseln der Seiten benötigt.

Die Animation beim Wechseln wird mithilfe des Attributs `slide` eingestellt. Mögliche Werte sind `fade`, `lift`, `slide` und `none`. Der Wert des Attributs `page` verweist auf die Startseite, die in der externen Datei `onsen_nav_land.htm` steht.

Es folgt die Startseite:

```

<ons-page>
  <ons-toolbar>
    <div class="left">&nbsp;</div>
    <div class="center">Länder</div>
  </ons-toolbar>
  <ons-list>
    <ons-list-item tappable onclick="weiter(0)">
      Spanien</ons-list-item>
    <ons-list-item tappable onclick="weiter(1)">
      Frankreich</ons-list-item>
  </ons-list>
</ons-page>

```

Listing 5.4 Datei »onsen_nav_land.htm«

Das Antippen der Listenelemente innerhalb der Startseite führt zu einer sichtbaren Reaktion und zum Aufruf der Funktion `weiter()`. Innerhalb dieser Funktion wird die Methode `pushPage()` für die Komponente des Typs `ons-navigator` aufgerufen. Dabei wird

der Name der gewünschten Unterseite übergeben. Hier stammt er aus einem Feld mit Dateinamen.

Technisch gesehen liegen alle Seiten auf einem Stapel. Es ist immer nur die oberste Seite des Stapels zu sehen. Zu Beginn ist das die Startseite. Die Methode `pushPage()` schiebt die gewünschte Unterseite als neue oberste Seite auf den Stapel. Sie verdeckt damit die bisherige oberste Seite.

In den Unterseiten finden Sie eine Komponente des Typs `ons-back-button`. Sie dient als Schaltfläche zur Rückkehr zur jeweils vorherigen Ebene. Dabei wird die aktuell oberste Seite vom Stapel geholt, und die darunterliegende Seite ist wieder die oberste Seite des Stapels.

Nachfolgend sehen Sie den Aufbau der Unterseite mit den Infos über Spanien, siehe auch [Abbildung 5.4](#). Die andere Unterseite der zweiten Ebene hat einen vergleichbaren Aufbau und wird hier nicht dargestellt.

```
<ons-page>
  <ons-toolbar>
    <div class="left"><ons-back-button></ons-back-button></div>
    <div class="center">Spanien</div>
  </ons-toolbar>
  <ons-list>
    <ons-list-item tappable onclick="weiter(2)">
      Madrid</ons-list-item>
    <ons-list-item tappable onclick="weiter(3)">
      Barcelona</ons-list-item>
  </ons-list>
</ons-page>
```

Listing 5.5 Datei »onsen_nav_spa.htm«

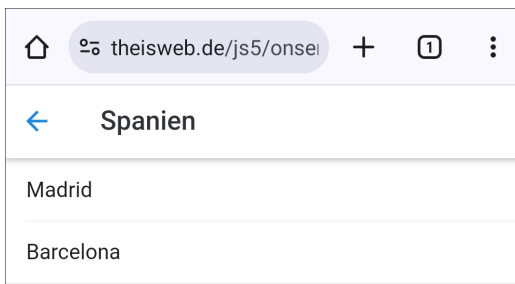


Abbildung 5.4 Navigator, zweite Ebene, »Spanien«

Es folgt die Unterseite mit den Infos über Barcelona, siehe auch [Abbildung 5.5](#). Die anderen Unterseiten der dritten Ebene haben einen vergleichbaren Aufbau und werden hier nicht dargestellt.

```
<ons-page>
  <ons-toolbar>
    <div class="left"><ons-back-button></ons-back-button></div>
    <div class="center">Barcelona</div>
  </ons-toolbar>
  <ons-list>
    <ons-list-item>Infos über Barcelona</ons-list-item>
  </ons-list>
</ons-page>
```

Listing 5.6 Datei »onsen_nav_spa_bar.htm«

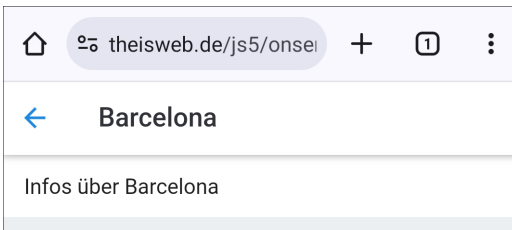


Abbildung 5.5 Navigator, dritte Ebene, »Barcelona«

5.3 Rundlauf in einem Karussell

Ein Karussell beinhaltet mehrere Einträge innerhalb einer einzigen Seite. In jedem Eintrag können bestimmte Inhalte der Seite dargestellt werden. Zwischen den einzelnen Einträgen kann entweder durch Antippen einer Schaltfläche in der Navigationsleiste oder durch seitliches Wischen gewechselt werden. In [Abbildung 5.6](#) sehen Sie die Startseite eines Karussells mit insgesamt drei Einträgen.

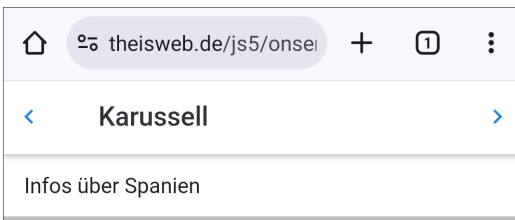


Abbildung 5.6 Karussell, Startseite

Dazu das Programm:

```
...
<body>
<ons-page>
  <ons-toolbar>
    <div class="left">
      <ons-toolbar-button id="idZurueck">
        <ons-icon icon="md-chevron-left"></ons-icon>
      </ons-toolbar-button>
    </div>
    <div class="center">Karussell</div>
    <div class="right">
      <ons-toolbar-button id="idWeiter">
        <ons-icon icon="md-chevron-right"></ons-icon>
      </ons-toolbar-button>
    </div>
  </ons-toolbar>

  <ons-carousel fullscreen swipeable auto-scroll id="kar">
    <ons-carousel-item style="background-color:#c0c0c0;">
      <ons-list>
        <ons-list-item>Infos über Spanien</ons-list-item>
      </ons-list>
    </ons-carousel-item>
    <ons-carousel-item style="background-color:#d0d0d0;">
      <ons-list>
        <ons-list-item>Infos über Italien</ons-list-item>
      </ons-list>
    </ons-carousel-item>
    <ons-carousel-item style="background-color:#e0e0e0;">
      <ons-list>
        <ons-list-item>Infos über Frankreich</ons-list-item>
      </ons-list>
    </ons-carousel-item>
  </ons-carousel>
</ons-page>

<script>
```

```

document.getElementById("idZurueck").addEventListener("click",
    function() {document.getElementById("kar").prev();});
document.getElementById("idWeiter").addEventListener("click",
    function() {document.getElementById("kar").next();});
</script>
</body></html>

```

Listing 5.7 Datei »onsen_karussell.htm«

Die beiden Schaltflächen in der Navigationsleiste werden mithilfe von Komponenten des Typs `ons-toolbar-button` erstellt. Die Betätigung der Schaltflächen führt zu den beiden Funktionen `zurueck()` und `weiter()`. Innerhalb dieser Funktionen werden für das gesamte Karussell die Methoden `prev()` und `next()` zum Wechsel des Karussell-Eintrags aufgerufen.

Das Karussell wird mithilfe einer Komponente des Typs `ons-carousel` erstellt. Das Attribut `id` dient zur Identifikation für die genannten Methoden. Das Attribut `fullscreen` sorgt dafür, dass sich die Karussell-Einträge jeweils über den gesamten Bildschirm erstrecken. Das Attribut `swipeable` ermöglicht das Wischen zum Wechseln der Karussell-Einträge. Das Attribut `auto-scroll` sorgt dafür, dass das Karussell nach dem Loslassen nach einem Wischvorgang automatisch zur nächstgelegenen Kante scrollt. Auf diese Weise sind die Einträge immer vollständig sichtbar. Diese sind Komponenten des Typs `ons-carousel-item`.

In [Abbildung 5.7](#) sehen Sie die letzte Seite des Karussells.

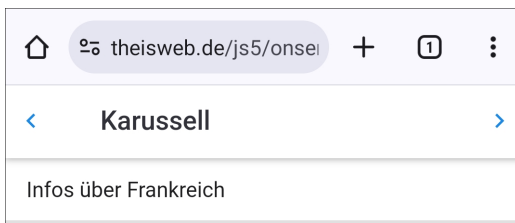


Abbildung 5.7 Karussell, letzte Seite