JavaScript – DOM und Events - auf Benutzereingaben reagieren

Inhalt:

JavaScript ermöglicht es dem Entwickler auf Ereignisse (engl.: events) im Browser zu reagieren. Auf diese Weise kann der Benutzer mit den **Anwendungen interagieren**, über die einfachen Rückgabewerte der Funktion prompt() und confirm() hinaus. Bei diesen Ereignissen kann es sich um den Klick auf einen **Button**, die **Auswahl** eines Eintrags aus einer Liste, ein Drehen des Gerätes, eine bestimmte **Bewegung mit der Maus**, das **Absenden** eines Formulars und vieles mehr handeln.

Ein **Eventhandler** ist ein Element zur Ereignisbehandlung. Eventhandler stellen damit eine Verbindung zwischen dem HTML-Element, bei dem das Ereignis ausgelöst wird, dem Ereignis selbst und der JavaScript-Funktion her.

Neben Klicken gibt es eine ganze Reihe weiterer Standardevents in JavaScript.

Mausereignisse:

- click: einfacher Mausklick
- dbclick: doppelter Mausklick
- mouseover: Mauszeigerbewegung über einem Element
- keypress: eine Taste auf der Tastatur wird gedrückt

Tastaturereignisse:

- keyup Der Benutzer lässt eine Taste los.

Formularereignisse:

- submit Der Benutzer schickt ein Formular ab
- input der Wert eines <input>-Elements hat sich geändert

1) klassische Ereignisbehandlung:

Eventhandler sind das wichtigste Bindeglied zwischen HTML und JavaScript.

Gemeinhin beginnen die Namen von Ereignissen **mit der Silbe on**, gefolgt von einer sprechenden Beschreibung des Ereignisses.
Beispiele:

- onmouseover, onload, onclick

Dafür muss kein <script>-Tab benutzt werden, es funktioniert ohne.

1a)onclick in einem Link

Grundsätzlich gilt Folgendes: Ein Event-Handler beginnt stets mit **on**. Der Event-Handler, der zuständig ist, wenn auf einen Link geklickt wird, heißt **onclick**.

Vom Prinzip her wunderbar – Browser, die JavaScript unterstützen, führen den Code aus, folgen danach aber dem Link.

- Das href-Attribut des Links muss gesetzt werden, sonst wird er nicht angezeigt.
- Wenn kein Link aufgerufen werden soll, gibt es eine Möglichkeit, einen Link anzugeben, der keine neue Seite lädt:

Hier klicken

Erstelle eine neue HTML-Datei „onlick.html“ und erstelle die Grundstruktur (boilerplate) in VS-Code mit dem Rufzeichen und Entertaste.

Beispiel: onclick.html

Im <body> soll ein Button erstellt werden und ein onclick auf ein alert-Fenster führen.

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7   </head>
8   <body>
9     <button a href="#" onclick="alert('Farbe des Buttons ändern')">hier klicken</a>
10      </button>
11  </body>
12  </html>
```

1b)Onclick mit einer Funktion

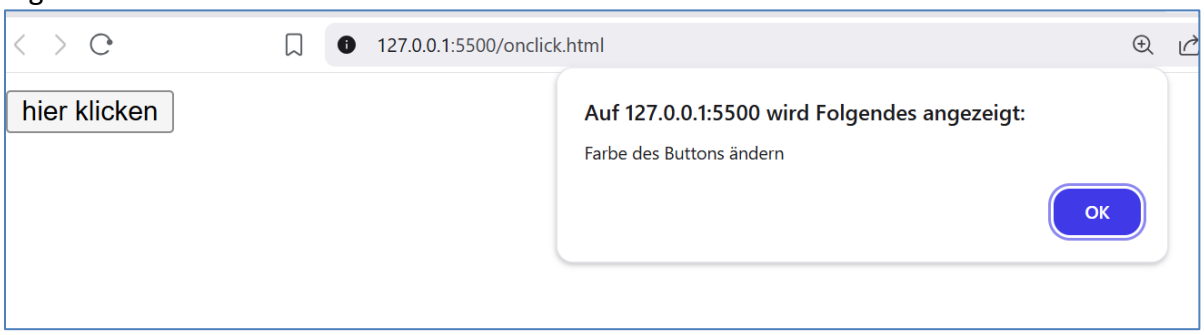
Funktion mit onclick: mit dem onclick-Ereignis auf den **Button** wird eine Funktion gestartet.

Ändere die onclick.html und erstelle die Funktion

```
<body>
    <button a href="#" onclick="handleClick()">hier klicken</a>
    </button>

    <script>
        function handleClick() {
            alert('Farbe des Buttons ändern');
        }
    </script>
</body>
</html>
```

Ergebnis:



Für eine Erweiterung dieses Beispiel, um den CSS Code zu ändern, benötigt man die nun folgende Theorie.

2)Die Methoden: querySelector und getElementById und innerHTML

DOM:

Der Begriff **DOM** steht für **Document Object Model** und bezeichnet eine Schnittstelle (API), die es JavaScript ermöglicht, auf eine HTML- oder XML-Seite zuzugreifen und sie zu verändern. Das DOM verbindet JavaScript mit der HTML-Struktur einer Webseite. Ohne DOM könnte JavaScript keine Webseiten dynamisch verändern.

Stell dir eine Webseite als ein Baumdiagramm vor. Jedes Element auf der Seite – z. B. ein , eine Überschrift <h1>, ein Absatz <p> oder ein Bild <img> – ist ein **Knoten (Node)** in diesem Baum. Dieses Baumdiagramm ist das **DOM**.

Es gibt mehrere Möglichkeiten auf diesen DOM zuzugreifen:

- querySelector
- getElementById

1. Die Methode querySelector ist eine moderne und sehr flexible Möglichkeit in JavaScript, **ein einzelnes Element im DOM auszuwählen. Damit kann man nach allen möglichen Selektoren suchen, wie z.B. id, class und auch TAGs wie <p>.**

2. **getElementById** ist eine JavaScript-Methode, mit der du ein einzelnes HTML-Element über seine eindeutige ID im DOM (Document Object Model) auswählen kannst. Anders gesagt, wird es verwendet, um ein **HTML-Element anhand seiner id** zu finden. Es übernimmt eine ID und gibt das dazugehörige Element zurück.

    - getElementById ist **schnell und effizient**, da IDs im HTML eindeutig sein müssen.
    - Sie wird am häufigsten verwendet, wenn man **gezielt ein einzelnes Element** ansprechen will.
    - Sie funktioniert **nur mit ID-Selektoren**, nicht mit Klassen (class) oder Tags (div, p usw.)

3. **innerHTML** ist eine wichtige Eigenschaft, mit der man den Inhalt eines Elements lesen und ersetzen kann. Der „innere“ Wert des Elements wird hier sichtbar gemacht. Zum Ändern eines Wertes, schreibt man nach dem Element das „innerHTML“ und dann den String für den neuen Inhalt:
   Beispiel:
   aktion.innerHTML = "Jetzt ist das neu";

Beispiel: querySelector

Weiter arbeiten mit dem Beispiel von oben: onclick.html

Hier wird auf den <button> zugegriffen.

document.querySelector('button').style.backgroundColor = 'blue';

```
8   <body>
9       <button a href="#" onclick="handleClick()">hier klicken</a>
10      </button>
11
12      <script>
13          function handleClick() {
14              alert('Farbe des Buttons ändern');
15              document.querySelector('button').style.backgroundColor = 'blue';
16          }
17      </script>
18  </body>
19  </html>
```

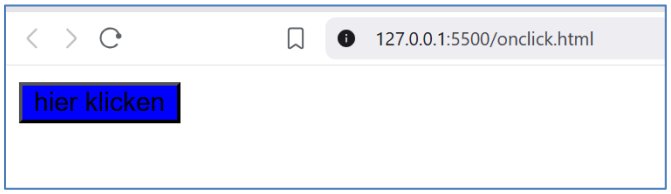
Die Methode querySelector ist sehr modern und es benötigt keine ID, die außerdem hier gar nicht vorkommt.

Zu Beginn muss „document“ angegeben sein, damit der Interpreter weiß, dass nun ein DOM-Zugriff erfolgt. Durch die Punktnotation werden die Methoden usw. angehängt.

- document – es handelt sich um einen DOM-Zugriff
- querySelector mit dem Element in der Klammer, welches benutzt werden soll
- style – Style-Eigenschaft für CSS
- backgroundColor

```
8  <body>
9      <button a href="#" onclick="handleClick()">hier klicken</a>
10     </button>
11
12     <script>
13         function handleClick() {
14             alert('Farbe des Buttons ändern');
15             document.querySelector('button').style.backgroundColor = 'blue';
16         }
17     </script>
18 </body>
```

Ergebnis:



Beispiel: getElementById

Info: Mit HTML-IDs kann man ein Element eindeutig identifizieren. Das wird auch in Bezug auf CSS so gehandhabt. Dafür muss man eine id verwenden.

Beispiel: ….

Hier erhält die eindeutige ID „aktion“. In dieser Seite darf es keine anderen Elemente mit der gleichen ID geben.

Der Zugriff erfolgt mit Hilfe der DOM-Methode „**getElementById()**“. Damit bekommt man ein Elementobjekt zurück, über das man dessen Inhalt und Attribute lesen, verändern und ersetzen kann.

„getElementById()“ ist die häufigste Methode im HTML-DOM, um Manipulationen durchzuführen oder Informationen vom Element zu erhalten.

Beispiel: Erstelle die Datei „2.js_innerhtml.html“:

```
<html>
<head>
<meta charset="utf-8">
<title>Javascript</title>
</head>
<body>
    <h1>Text auslesen und ändern</h1>
    <p id="tagebuch">Das ist die erste Fassung meines Tagebuches.</p>

</body>
</html>
```

Die „id“ ist das Element, über das JavaScript am einfachsten auf die Webseite zugreifen und die Inhalte eines Tags verändern kann. Zum Beispiel um den enthaltenen Text auszulesen oder ihn zu verändern.

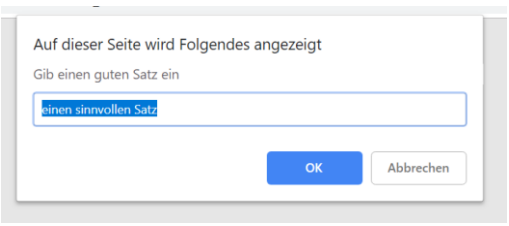
```
<body>
    <h1>Text auslesen und ändern</h1>
    <p id="tagebuch">Das ist die erste Fassung meines Tagebuches.</p>
    <script>
        tagebuch.innerHTML = "Endich fällt mir dazu was ein. Es war einmal vor
        langer Zeit....";
    </script>
</body>
```

Beim Öffnen der Seite wird JavaScript sofort ausgeführt und der bestehende Text wird sofort überschrieben.

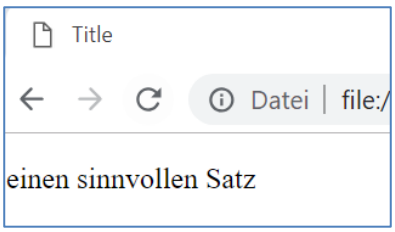
Ergebnis:

Übung:

Erstelle eine Prompt-Anweisung, in die der User einen eigenen Text eingeben kann:



Ziel:



Code:

```
<html>
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <p id="notiz">Platzhalter</p>

<script>
    var gibEin = prompt("Gib einen guten Satz ein", "einen sinnvollen Satz");
    notiz.innerHTML = gibEin;

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

Info:

notiz.innerHTML ist so etwas wie die Kurzform von:

„document.getElementById("notiz").innerHTML

Damit greift man innerhalb des Dokuments „document" (also der Webseite) auf das Element mit der Id „notiz" zu. Genauer gesagt auf das, was sich innerhalb des Tags befindet, innerHTML.

Verbesserung des Skriptes:

Der Änderung des Codes ist **NICHT** zu erkennen, da sie sofort ausgeführt wird.

Besser: Der Text wird erst geändert, wenn man auf einen Button klickt.

2b) Beispiel: Mit einem Klick Inhalt ändern

Erstelle folgende Datei „2.js_getelement.html“.

```
 6  </head>
 7  <body>
 8      <h1>Text auslesen und ändern</h1>
 9      <p id="tagebuch">Das ist die erste Fassung meines Tagebuches.</p>
10
11      <button onclick="neuesTagebuch()">Klicke hier, um den Text zu ändern.
        </button>
12
13      <script> //zuerst Variabe erstellen, dann die Funktion
14      var aktion = document.getElementById("tagebuch");
15      function neuesTagebuch(){
16          aktion.innerHTML = "Jetzt ist mir endlich was eingefallen. Es war
            einmal in der Hak, vor langer langer Zeit...";
17          }
18      </script>
19  </body>
20  </html>
```

- In Zeile 11 wird der Onclick-Event mittels eines Buttons ausgelöst. Dieser verwendet die Funktion, die in Zeile 15 steht.
- Vorher wird die Variable „aktion“ erstellt, die mit „getElementById“ den Inhalt der ID in Zeile 9 ausliest.
- In der Funktion wird dieser Inhalt mit einem neuen Inhalt überschrieben. Dazu wird die Methode „innerHTML“ verwendet.

Ergebnis:

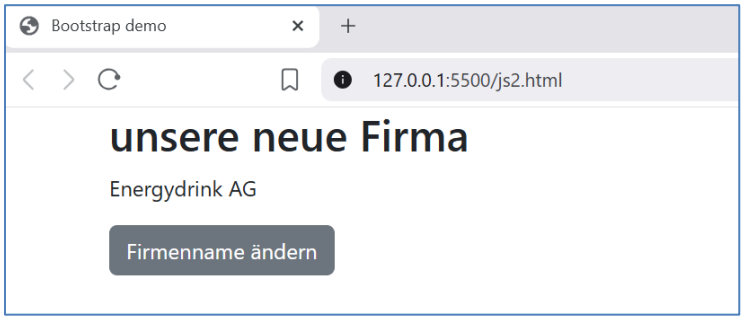
Beispiel mit Prompt – Ändern nach Button-Klick

```
<body>
<div class="container">
  <h2>unsere neue Firma</h2>
  <p id="firmenname">Energydrink AG</p>
  <button onclick="editieren()" type="button"
  class="btn btn-secondary">Firmenname ändern</button>

  <script>
    let neu = document.getElementById('firmenname')

    function editieren() {
      let neuerName = prompt('Bitte neuen Firmennamen eingeben');
      neu.innerHTML = neuerName;
    }
    </script>
</div> <!-- container ende-->
```

Ergebnis:



2c)„Value“ auslesen aus einem Formular:

Textfelder besitzen wie alle Eingabefelder eine **Eigenschaft namens „value“,** über die auf deren Wert zugegriffen werden kann. Dabei kann man mit diesem Wert wie mit einer normalen String-Variablen arbeiten. Man kann aber auch Werte in diese Eigenschaft schreiben, um den Inhalt des Textfeldes zu aktualisieren.

Textfelder enthalten immer String-Werte und geben auch nur solche zurück. In dieser Hinsicht ähneln sie „prompt-Anweisungen“. Wenn ein Textfeld einen nummerischen Wert zurückgeben soll, muss man sie es erst mit „parseInt()“ oder „parseFloat()“ in Ganz- oder Gleitpunktzahlen umwandeln.

Beispiel:
Erstelle „2.js_getelement.value.html“.

Beginne mit der Erstellung eines Formulars, das zwei Inputfelder enthält. Diese haben fixe Werte „value“, nämlich „Herr“ und „Huber“. Später, mit Speichern in eine Datenbank, würde man hier Werte finden, die der Kunde eingeben würde.

```
<form>
    <label for="anrede" >Anrede</label>
    <input type="text" name="anrede" value="Herr" id="anrede">
     <br> <br>
    <label for="familienname" >Familienname</label>
    <input type="text" value="Huber" id="familienname">
</form>
```

```
<form>
    <label for="anrede" >Anrede</label>
    <input type="text" name="anrede" value="Herr" id="anrede">    <br> <br>
    <label for="familienname" >Familienname</label>
    <input type="text" value="Huber" id="familienname">
  </form>
```

Darunter erstelle einen <script>-Bereich, in dem diese beiden fixen Werte in Variablen übergeben werden. Die Zuordnung erfolgt mittels der ID der Felder:

```
var anr = document.getElementById("anrede").value;
var fam = document.getElementById("familienname").value;
```

```
16          var anr = document.getElementById("anrede").value;
17          var fam = document.getElementById("familienname").value;
```

Diese Werte werden in eine neue Variable „ergebnis“ gespeichert und dann ausgegeben:

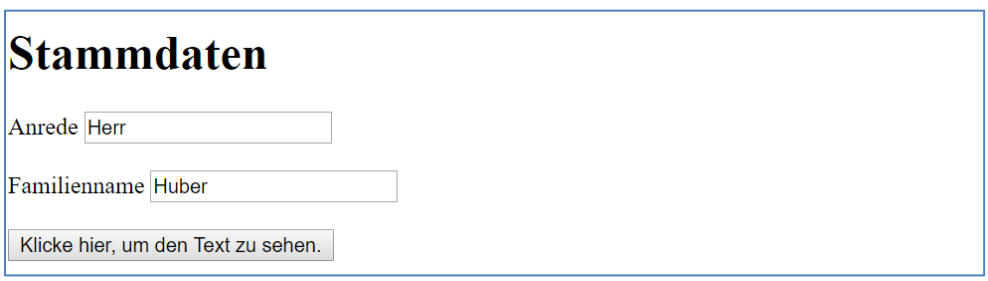
```
18
19          var ergebnis = "Sehr geehrter " + anr + " " + fam + ".";
20
21          function hallo() {
22              document.write(ergebnis)
23                 };
24      </script>
```

Code:

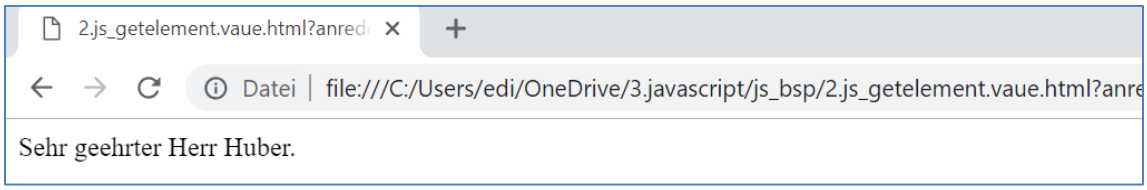
```
 1
 2  <html lang="de">
 3  <head>
 4      <title>Wert auslesen</title>
 5  </head>
 6  <body>
 7  <h1>Stammdaten</h1>
 8      <form>
 9          <label for="anrede" >Anrede</label>
10          <input type="text" name="anrede" value="Herr" id="anrede">
             <br> <br>
11          <label for="familienname" >Familienname</label>
12          <input type="text" value="Huber" id="familienname">
13      </form>
14
```

```
13      </form>
14
15      <script> //Das script muss nach der betroffenen ID stehen, sonst
        ist sie leer
16          var anr = document.getElementById("anrede").value;
17          var fam = document.getElementById("familienname").value;
18
19          var ergebnis = "Sehr geehrter " + anr + " " + fam + ".";
20
21          function hallo() {
22              document.write(ergebnis)
23                 };
24      </script>
25
26      <br>
27      <button type="button" onclick="hallo()">Klicke hier, um den Text
        zu sehen.</button>
28
29  </body>
30  </html>
```

Ergebnis:



Ergebnis:

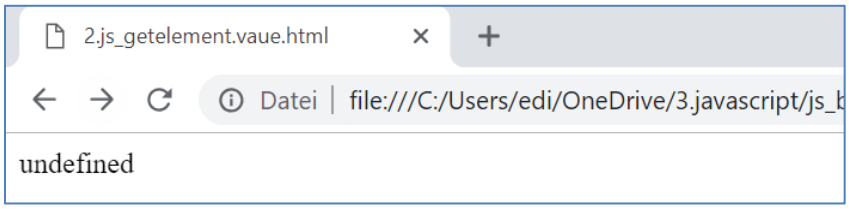


Beachte:
Die Variablen müssen NACH den input-Feldern erstellt werden, sonst wären sie leer.

Die Erstellung der Variablen im <script> Bereich kann hier nicht im <head> erfolgen, da dann die Elemente aus dem <body> noch nicht vorhanden wären. Daher muss dieser Teil danach geschrieben sein.

```
13      </form>
14
15      <script> //Das script muss nach der betroffenen ID stehen, sonst
        ist sie leer
16          var anr = document.getElementById("anrede").value;
17          var fam = document.getElementById("familienname").value;
18
19          var ergebnis = "Sehr geehrter " + anr + " " + fam + ".";
```

Sonst wäre das Ergebnis folgendes:



Aufgabe: Inputfelder frei

Lösung

```
<body>

    <form>
        <label for="anrede">Anrede</label>
        <input type="text" name="anrede" id="anrede">    <br> <br>
        <label for="familienname">Familienname</label>
        <input type="text" id="familienname">
    </form>
```

```
    <script>
    function hallo() {
        var anr = document.getElementById("anrede").value;
        var fam = document.getElementById("familienname").value;

        var ergebnis = "Sehr geehrter" + " " + anr + " " + fam + "!";

        document.write(ergebnis);
    };
    </script>
```

```
    <br>
    <button type="button" onclick="hallo()">Klicke hier, um den Text zu sehen.</button>

</body>
</html>
```

2d)mit value den Wert lesen und Formular überprüfen

Verwende das Formular weiter und überprüfe den Namen. Wenn er kleiner als 3 Buchstaben ist, soll eine Warnmeldung ausgegeben werden.

Speicher die neue Datei als „2.js_getelement.formular.html“.

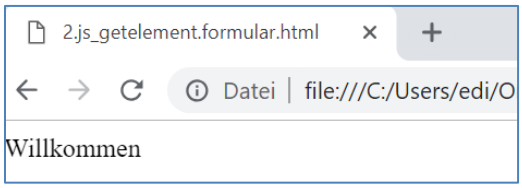
```
var fam = document.getElementById("familienname");

    function checkUser() {
      if (fam.value.length < 3) {
        document.write("Der Name muss mindestens 3 Zeichen haben");
        }
       else {
            document.write("Willkommen");
            }
       };
```

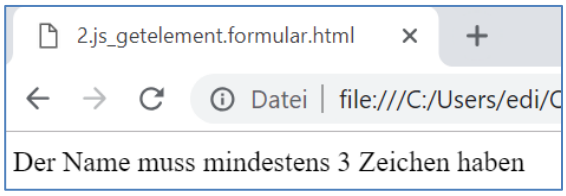
ABER nicht vergessen: den Namen der Funktion im Button ändern:

```
26      </script>
27      <br>
28      <button type="button" onclick="checkUser()">Klicke hier, um den
        Text zu sehen.</button>
29
```

Ergebnis: wenn ok



wenn nicht ok:



3) Beispiel: Schuhe zählen (getElementById)

Berechnung der Anzahl der Schuhepaare, die im vollen Regal (Anzahl der Laden mal Anzahl der Paare je Etage) und neben dem Regal herumstehen.

Welche Variablen benötigt man?

- regalLaden
- paareJeLade
- paareNebenDemRegal
- paare = paareNebenDemRegal + (regalLaden * paareJeLade)

Aufgabe:

Die vorgegebenen Werte sollen in einer Berechnung ein Resultat ergeben, welches in einem Satz ausgegeben werden soll. Im <head> soll dafür die Berechnung in einem <script> erstellt werden.

Verwende „parseInt()“ um mit den Daten rechnen zu können.

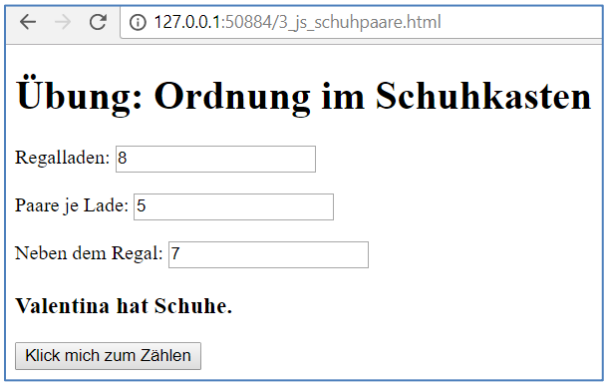
Code 1. Variante:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        function zaehleSchuhe() {
            var regalLaden = document.getElementById("laden").value;
            var paareJeLade = document.getElementById("paare-je-lade").value;
            var paareNebenDemRegal = document.getElementById("neben-dem-regal").value;
            var paare = paareNebenDemRegal + paareJeLade * regalLaden;
            document.getElementById("ausgabe").innerHTML= paare;
        }
    </script>
</head>
<body>
    <h1>Übung: Ordnung im Schuhkasten</h1>
    <p>Regalladen: <input type="text" id="laden" value="8"></p>
    <p>Paare je Lade: <input type="text" id="paare-je-lade" value="5"></p>
    <p>Neben dem Regal: <input type="text" id="neben-dem-regal" value="7"></p>
    <h3>Valentina hat <span id="ausgabe"> </span> Schuhe.</h3>
    <button type="button" onclick="zaehleSchuhe()">Klick mich zum Zählen</button>
</body>
</html>
```

Mit „getElementById()“ kann man die Eingabefelder finden und ihre Eigenschaften, ihren „value“ auslesen.
Man könnte das Eingabefeld statt dem verwendeten „type=“text“ auch „number“ verwenden.
Die Eingabefelder müssen eine ID haben, um sie mit „getElementById“ finden zu können.

Beachte: in JavaScript hat man die CamelCase-Methode bei den Namen verwendet, in HTML haben wir die Bindestriche benutzt.

Ergebnis:



Leider funktioniert die Berechnung nicht. Der Grund liegt im „value“. Im „value“ steht nämlich keine Zahl, sondern ein String. **Denn alle Werte, die aus den Eingabefeldern stammen, sind Strings.**

Daher muss man alle Eingabewerte explizit in Zahlen umwandeln. Dazu hat JavaScript die Funktionen „parseInt()“ und parseFloat()“ parat.

- parseInt() erzeugt aus einem String eine Ganzzahl
- parseFloat() erzeugt aus einem String einen Flaot-Wert

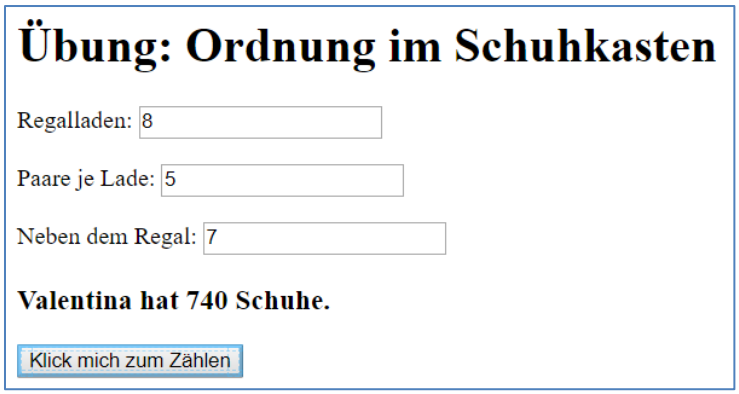
Hier soll „parseInt()“ verwendet werden, da die Anzahl der Schuhe immer ganzzahlig sein wird.

Diese drei Zeilen sind daher vor der Berechnung einzufügen: (also in der Zeile 11, damit die Berechnung danach erfolgt)

Code 2: nach Verbesserung

```
regalLaden = parseInt(regalLaden);
paareJeLade = parseInt(paareJeLade);
paareNebenDemRegall = parseInt(paareNebenDemRegal);
```

Ergebnis:



Man kann die Werte auch direkt in der Anzeige ändern.