Übung 3 - Einführung JavaScript

Aufbau der Webseite

- Bootstrap-Elemente verwenden - https://getbootstrap.com/css/
- Formular bauen, Elemente mit mit passenden Bootstrap-Klassen belegen

```
<form name="f1" class="form">
  <div class="form-group">
    <label for="volts">Volts</label>
    <input type="text" name="volts" class="form-control" id="volts" placeho
  </div>
</form>
```

Rechner-Logik (JS)

- Beispiel für den Umgang mit DOM-Elementen:

```
<script>
  var d = document.f1;
  var v1 = d.volts.value;
  v1 = 12;
  d.volts.value = v1;
  v1.classList.add("alert-success");
</script>
```

- Prüfen, in welche Felder der Nutzer etwas eingegeben hat

```
  var v1 = d.volts.value;
  if ( v1 ) {
    // wird ausgeführt falls etwas in das Feld geschrieben wurde
  }
```

- Entsprechend der Kombination, die Werte für die verbleibenden Felder berechnen

- `var` deklariert neue Variablen. Besonderheit in JS: `var` ist im gesamten Funktionsblock gültig und lässt sich re-deklarieren. `let` hat nur im umgebenden Block Gültigkeit, kein Re-Deklarieren möglich.

```
function testVar() {
  for( var v = 0; v < 10; v++ ) {

  }
  console.log(v); // > 10
}

function testLet() {
  for( let l = 0; l < 10; l++ ) {

  }
  console.log(l); // > undefined
}
```

- `const` muss bei Deklaration initialisiert werden und kann danach nicht mehr verändert werden.

```
const c  = 3;
c = 4; // > TypeError
```

- Mathematische Operationen und Konstanten (z.B. `sqrt()` `pow()` `PI`) sind verfügbar im Math-Paket, Grundlegende Operatoren (`+` `-` `*` `/`) können direkt verwendet werden.

Datentypen

- `string`, `number`, `boolean`, `null`, `undefined`, `object`
  - `undefined` : Variable nicht initialisiert oder Objekt-Attribut nicht vorhanden
  - `null` : Variable oder Attribut existieren, haben jedoch keinen Wert
  - `object` : JavaScript Object Notation
    - Objekte immer in Key-Value-Struktur
    - Zugriff auf Objekt-Attribute entweder mit dot `.` - oder `[]` -Notation
  - implizite Type-Conversion z.B. von `number` zu `string`
    - Identitätsoperator prüft auch auf Typgleichheit
    - `2015 == '2015' // true`
    - `2015 === '2015' // false`

Arrays

- `.length` -Attribut gibt Anzahl der Elemente zurück
- Element-Zugriff entweder mit `[]` -Notation, anders als bei Assoziativen Arrays (siehe

`object`) wird hier jedoch nur der index als Zahl angegeben. `listOfFruit[2]` gibt das dritte Element der Liste zurück

- Elemente einfügen mit `.push(elem)`. Arrays können auch wie Stacks oder Queues behandelt werden:
    - `pop()` gibt das letzte Element zurück und entfernt es. (Stack)
    - `shift()` gibt das erste Element zurück und entfernt es. (Queue)

Funktionen

```
// Aufruf: summe = addition( 2, 3 );
function addition( zahl1, zahl2 ) {
    return zahl1 + zahl2;
}

// Aufruf: summe = addition( 2, 3 );
var addition = function( zahl1, zahl2 ) {
    return zahl1 + zahl2;
}

// Funktion als Objekt-Attribut
Beispielaufruf: grundrechenarten.multiplikation( 2, 3 );
var grundrechenarten = {
    addition: function( x, y ) {
        return x + y;
    },
    subtraktion: function( x, y ) {
        return x - y;
    },
    multiplikation: function( x, y ) {
        return x * y;
    },
    division: function( x, y ) {
        return x / y;
    }
};
```