KLASSEN

Wenn man immer wieder das gleiche Objekt verwenden muss, macht es Sinn, eine Klasse daraus zu erstellen. Damit können auch Lesbarkeit und Wartbarkeit des eigenen Quellcodes verbessert werden. Eine Klasse kann man sich wie eine Blaupause oder einen Bauplan für Objekte vorstellen. Mit einer Studentenklasse könnte man beispielsweise immer wieder Studenten erstellen und diesen Funktionen (=Methoden) geben.

Eine Klasse wird mit dem Keyword **class** erstellt. Der Name einer Klasse wird großgeschrieben und mit der **constructor**-Methode kann aus der Klasse eine Objektinstanz erstellt werden. Der Constructor kann Properties, also Eigenschaften entgegennehmen. Die Instanz kann über **this** angesprochen werden.

Beispiel:

```
class Student {
    constructor(name, id, nc) {
        this.name = name;
        this.id = id;
        this.nc = nc;
    }
}
```

new

Mit dem Keyword **new** können neue Objekte erzeugt werden.

Beispiel:

```
var student = new Student(“Jannick“, 1, 2);
```

METHODEN

Eine Klasse kann mehrere Methoden besitzen. Man kann sich eine Methode wie eine Funktion eines Objekts vorstellen.

Beispiel:

```
class Car {
      constructor(marke, baujahr, ps) {
      this.marke = marke;
      this.baujahr = baujahr;
      this.ps = ps;
}

//Methode
     drive(speed) {
          console.log(“Fährt: “ + this.marke + “ “ + speed);
      }
      honk() {
          console.log(“Hupt: “ + JSON.stringify(this))
      }

var car = new Car(“VW”, 1994, 132);
car.drive(100);

var car2 = newCar(“Porsche”, 2010, 180);
car2.drive(210);
car2.honk();
```

stringify

Wandelt ein Objekt in einen String um.

JSON

JavaScript Object Notation ist die Möglichkeit, JavaScript Objekte zu serialisieren und zu deserialisieren, sodass sie beispielsweise einfach an andere Programme gesendet oder ausgelesen werden können.

EXPORT, IMPORT UND CORS

Damit der Programmcode so einfach wie möglich erweitert werden kann, macht es Sinn, eine neue JS-Datei für eine oder mehrere Klassen zu erstellen.

export

Um die ausgelagerte Klasse wieder in der Hauptdatei zu verwenden, muss sie exportiert werden. Dafür verwendet man **export**.

Beispiel:

```
export class Car {
```

import

Innerhalb der Hauptdatei kann die Klasse durch **import** importiert werden.

Beispiel:

```
import {Car} from “./car.js“
drive(210)
honk()
```

CORS

Bei CORS geht es darum, dass zwischen zwei Domains standardmäßig keine Ressourcen geteilt werden dürfen. Mit einer CORS Policy kann man den Webserver so versehen, dass er Daten von anderen Servern erhalten darf.

VISUAL STUDIO LIVE PREVIEW FÜR LOKALEN WEBSERVER

Live Preview hostet einen lokalen Server, auf welchem man Vorschauen der eigenen Website anzeigen kann.

Port

Ein Port ist ein Teil einer Netzwerkadresse. Eine Anwendung kann immer nur auf einem Port laufen.

VERSCHIEDENE ARTEN DES IMPORTS

Um alle Klassen zu importieren, kann zum Beispiel die folgende Zeile Code verwendet werden:

```
import * as vehicles from “./car.js”
```

* = all

Weitere Import- und Export-Möglichkeiten sind auf ***developer.mozilla.org*** zu finden.