Function Constructor & Prototypes

Wenn Objekte, welche Methoden haben, mithilfe eines Constructors gebildet werden, so nennt man diese Constructors auch Function Constructor.

```
let KerzeKonstrukteur = function (farbe, laenge, durchmesser) {
	this.farbe = farbe;
	this.laenge = laenge;
	this.durchmesser = durchmesser;
	this.brenndauerFunktion = function () {
		var minProCm = 20;
		if (this.durchmesser === 3) {
			minProCm = 37;
		} else if (this.durchmesser === 4) {
			minProCm = 68;
		} else if (this.durchmesser === 5) {
			minProCm = 100;
		} else if (this.durchmesser === 6) {
			minProCm = 135;
		} else if (this.durchmesser === 7) {
			minProCm = 175;
		} else if (this.durchmesser === 8) {
			minProCm = 215;
		} else if (this.durchmesser === 9) {
			minProCm = 260;
		} else {
			minProCm = 305;
		}
		this.brenndauer = this.laenge * minProCm;
	}
}

let kerzeModell1 = new KerzeKonstrukteur(`rot`, 8, 5);

kerzeModell1.brenndauerFunktion();
console.log(kerzeModell1);
```

// gibt auf der Console aus:

```
Kerze {farbe: "rot", laenge: 8, durchmesser: 5,
brenndauerFunktion: f, brenndauer: 800}
   1.brenndauer: 800
   2.brenndauerFunktion: f ()
   3.durchmesser: 5
   4.farbe: "rot"
   5.laenge: 8
```

Wenn der Function Constructor KerzeKonstrukteur mit dem Schlüsselwortes **new** aufgerufen wird, dann wird ein neues Objekt kerzeModell1 kreiert.

Dieses Objekt hat eine Methode brenndauerFunktion, wird nun diese Methode des Objekts kerzeModell1 aufgerufen, dann wird diesem Objekt das neue Key-Value-Paar brenndauer dazugefügt.

Mit diesem Constructor können dank das Schlüsselwortes **new** nun viele Objekte gleichen Typs angelegt werden.
Diese Objekte sind alle **Instanzen** vom Constructor KerzeKonstrukteur.
Instanzen sind Exemplare aus einer Menge gleichartiger Dinge

```
let kerzeModell2 = new KerzeKonstrukteur(`grün`, 12, 3);
let kerzeModell3 = new KerzeKonstrukteur(`blau`, 6, 6);
```

Da jetzt alle Objekte die Methode brenndauerFunktion haben und dies unpraktisch ist (u.a. benötigt solch langer Code viel Platz wenn viele Objekte angelegt werden), kann dieser Code auch “ausgelagert“ werden.

Dazu wird dem Contructor mit dem Schlüsselwort prototype ein Key-Value-Paar zugefügt, den dann alle Instanzen des Contructors haben.

```
let Kerze2 = function (farbe, laenge, durchmesser) {
        this.farbe = farbe;
        this.laenge = laenge;
t       his.durchmesser = durchmesser;
}

Kerze2.prototype.brenndauerFunktion2 = function () {
        var minProCm = 20;
        if (this.durchmesser === 3) {
                minProCm = 37;
        } else if (this.durchmesser === 4) {
                minProCm = 68;
        } else if (this.durchmesser === 5) {
                minProCm = 100;
        } else if (this.durchmesser === 6) {
                minProCm = 135;
        } else if (this.durchmesser === 7) {
                minProCm = 175;
        } else if (this.durchmesser === 8) {
                minProCm = 215;
        } else if (this.durchmesser === 9) {
                minProCm = 260;
         else {
                minProCm = 305;
        }
t       his.brenndauer = this.laenge * minProCm;
}
```

```
Kerze2.prototype.handelsmarke = true;

let anderesModell1 = new Kerze2(`weiß`, 10, 5);

anderesModell1.brenndauerFunktion2();
console.log(anderesModell1);
```

// gibt auf der Console aus:

Kerze2 {farbe: "weiß", laenge: 8, durchmesser: 5, brenndauer: 800}

1. brenndauer: 1000
2. durchmesser: 5
3. farbe: "weiß"
4. laenge: 10
5. __proto__:
    1. brenndauerFunktion2: *ƒ ()*
       handelsmarke: true
    2. constructor: *ƒ farbe, laenge, durchmesser)*
    3. __proto__: Object

Um Objekte und deren Abstammung von Constructors genauer untersuchen zu können gibt es folgende Befehle:

Object.getProtypeOf(“Objektname“)
zeigt an auf welche Prototypen ein Objekt zurückgreift

```
console.log(Object.getPrototypeOf(modell1));
```
// gibt auf der Console aus:

{brenndauerFunktion2: ƒ, handelsmarke: true, constructor: ƒ}

1. brenndauerFunktion2: *ƒ ()*
2. handelsmarke: true
3. constructor: *ƒ (farbe, laenge, durchmesser)*
4. __proto__: Object

“Constructorname“.prototype;
zeigt an welche Prototypen ein Constructor einem Objekt zuweist

```
console.log(Kerze2.prototype);
```
// gibt auf der Console aus:

{brenndauerFunktion2: ƒ, handelsmarke: true, constructor: ƒ}

5. brenndauerFunktion2: *ƒ ()*
6. handelsmarke: true
7. constructor: *ƒ (farbe, laenge, durchmesser)*
8. __proto__: Object

Dementsprechend kann man beide Befehle mit einander vergleichen, dies muss dann true sein.

```
console.log(Object.getPrototypeOf(modell1) === Kerze2.prototype);
// gibt auf der Console aus:
true
```

Eine veraltete Schreibweise dafür lautet wie folgt:
console.log(modell1.__proto__ === Kerze2.prototype);

.hasOwnProperty(“Key-Name“)
zeigt an, ob ein Objekt ein bestimmtes Key-Value-Paar hat und gibt true oder false heraus

```
console.log(modell1.hasOwnProperty(`farbe`));
// gibt auf der Console aus:
true

console.log(modell1.hasOwnProperty(`lastName`));
// gibt auf der Console aus:
false
```

instanceof
“Objektname“ instanceof “Constructorname“ zeigt an, ob ein Objekt eine Instanz eines bestimmten Constructors ist und gibt true oder false heraus

```
console.log(modell1 instanceof Kerze2);
// gibt auf der Console aus:
true
```

Object.create()

Es kann auch erst ein Prototyp angelegt werden und dann die entsprechenden Objekte mit Objekt.create() kreiert werden.

```
let studentProto = {
        studentAtDCI: true,
        stayAndStudy: function () {
                if (this.hoursPerDay > 5) {
                this.studyingHard = true;
                } else {
                this.studyingHard = false;
                }
        }
}
```

```
let everybodyAtFbw14 = Object.create(studentProto);
everybodyAtFbw14.hoursPerDay = 7;
everybodyAtFbw14.stayAndStudy();
```

Hier wird ein Objekt mit Hilfe eines Prototyps kreiert, diesem Objekt das Key-Value-Paar “housPerDay : 7“ zugewiesen und die Methode des Objekts wird aufgerufen. Dadurch wird dem Objekt dann noch der Key „studyingHard“ mit entsprechendem Wert zugewiesen.

```
console.log(everybodyAtFbw14);
// gibt auf der Console aus:

{hoursPerDay: 7}

    1.hoursPerDay: 7
    2.studyingHard: true
    3.__proto__:
        1.stayAndStudy: ƒ ()
            1.arguments: null
            2.caller: null
            3.length: 0
            4.name: "stayAndStudy"
            5.prototype: {constructor: ƒ}
            6.__proto__: ƒ ()
            7.[[FunctionLocation]]: main.js:351
            8.[[Scopes]]: Scopes[2]
        2.studentAtDCI: true
        3.__proto__: Object
```

Questions? You‘re welcome!

holger.zerbe@web.de