CALL, APPLY & BIND - FUNCTIONS

Im Javascript sind Funktionen auch Objekte, deshalb können sie auch mit Methoden angesprochen werden.

.call()
Mit .call() werden Funktionen als Objekte aufgerufen und ausgeführt.
Der Befehl beispielfunktion.call() macht also das Gleiche wie der direkte Funktionsaufruf beispielfunktion().

Mit .call() können Parameter übergeben werden, als erster Parameter wird das Objekt, auf den die Methode angewandt wird, übergeben.
Beispiel: beispielfunktion.call(Objekt)

```
let cheeseburger = {
        typ: `cheeseburger`,
        anzahl: 1,
        gekocht: function () {
                this.anzahl++;
                console.log(`Es sind jetzt ${this.anzahl} ${this.typ} vorhanden.`)
        },
        verspeist: function () {
                if (this.anzahl > 0) {
                        this.anzahl--
                        console.log(`Es wurde ein ${this.typ} verspeist, es sind jetzt
                        ${this.anzahl} Stück vorhanden.`)
                } else {
                        console.log(`Sorry, nichts mehr zum Essen da!`)
                }
        }
};

let pommes = {
        typ : `Pommes Frites`,
        anzahl : 0
};


cheeseburger.verspeist.call(cheeseburger);
cheeseburger.verspeist.call(cheeseburger);
```

// das this-keyword in den Methoden .gekocht() und .verspeist() des Objektes cheeseburger verweist hier auf den von .call() übergebenen Parameter cheeseburger

// auf der Console wird ausgeben (2 Aufrufe, 2 Ausgaben):
Es wurde ein cheeseburger verspeist, es sind jetzt 0 Stück vorhanden.
Sorry, nichts mehr zum Essen da!

```
cheeseburger.gekocht.call(cheeseburger);
```
// das this-keyword in den Methoden .gekocht() und .verspeist() des Objektes cheeseburger verweist hier wieder auf den von .call() übergebenen Parameter cheeseburger

// auf der Console wird ausgeben:
Es sind jetzt 1 cheeseburger vorhanden.

```
cheeseburger.gekocht.call(pommes);
```
// Hier verweist das this-Keyword der Methoden .gekocht () und .verspeist() des Objektes cheeseburger auf den von .call() übergebenen Parameter pommes

Es werden also die Methode .gekocht() des Objekts cheeseburger bei dem Objekt pommes angewandt (das Objekt pommes selber hat nicht diese Methode!).

// auf der Console wird ausgegeben:
Es sind jetzt 1 Pommes Frites vorhanden.

Werden mit .call() Parameter übergeben, so sieht der Syntax wie folgt aus: .call(Objekt, “Parameter1“, “Parameter2“, “Parameter3“ …), zuerst also das zu benutzende Objekt und dann die weiteren Parameter.

.apply()
Die Methode .apply() macht genau das Gleiche wie .call(), jedoch ändert sich der Syntax. Die weiteren Parameter (auch wenn es nur einer ist), werden hier als ein Array übergeben. Der Syntax sieht wie folgt aus:
.appl(Objekt, [“Parameter1“, “Parameter2“, “Parameter3“])

.bind()
Mit .bind() kann eine neue Funktion erstellt werden, dabei wird eine bestehende Funktion genutzt und ihre Parameter werden dauerhaft eingebunden.

```
function addition(summand, zahlDerAdditionen) {
        let summe = 0;
        for (let i=0; i<zahlDerAdditionen; i++ ) {
                summe += summand;
        }
        return summe;
}
```

```
let ergebis = addition(23,4);
console.log(ergebis);
```
// hier wird die Funktion additoin() mit den Parametern summand und anzahlDerAdditonen aufgerufen.
Die Schleife in der Funktion wird 4 mal (anzahlDerAdditionen) durchlaufen und jedesmal wird 23 (summand) addiert.

// auf der Console wird ausgegeben:
92

```
addiere25 = addition.bind(this, 25, 5);
```
// Hier wird eine neue Funktion erstellt. Dazu wird die alte Funktion addition() benutzt und mit den Parametern 25 (summand) und 5(anzahlDerAdditionen) dauerhaft addiere25 (neue Funktion) zu gewiesen.

```
console.log(addiere25());
```

// auf der Console wird ausgegeben:
125

Zusammenfassung:

- Funktionen sind auch Objects (mit Methoden und Variablen)

- call: Methode einer Funktion, die die Funktion aufrufen
  mit einem anderen kontext (ändert das this-keyword
  innerhalb der Funktion)

- apply: das Gleiche wie call, nur mit Arrays

- bind: erstellt eine Funktion, die eine vorhandene Funktion als
  Vorbild hat mit vordefinierten Parametern.

Questions? You‘re welcome!

holger.zerbe@web.de