



DANIEL MIKESCH

JAVASCRIPT 101

OBJECTS

- ▶ Ein Objekt kann ähnlich wie ein Array beliebig viele Werte speichern.
- ▶ Diese Variablen eines Objektes werden Eigenschaften oder Attribute genannt.
- ▶ Im Unterschied zu einem Array werden Werte nicht mit einem Index sondern mit Namen gespeichert.

```
var produkt = {  
  "bezeichnung": "Gartenschlauch",  
  "farbe": "grün",  
  "laenge_in_m": 20,  
  "preis": 39.9  
};
```

```
console.log(produkt.preis); // 39.9  
console.log(produkt["preis"]); //39.9
```

```
produkte.preis = 0.99;  
console.log(produkt.preis)); //0.99
```

EIGENSCHAFTEN UND METHODEN

- ▶ Variablennamen müssen nicht in Anführungszeichen stehen. ("Schöner" ist es aber)
- ▶ Objekte können auch Funktionen beinhalten. Diese werden dann Methoden genannt.
- ▶ Aufruf der Methoden erfolgt mit ()

```
var produkt = {  
  "bezeichnung": "Gartenschlauch",  
  farbe: "grün",  
  laenge_in_m: 20,  
  "preis": 39.9,  
  "log_stuff": function(_stuff){ console.log(_stuff); }  
};
```

```
produkt.log_stuff("Hey!"); // Hey!  
produkt["log_stuff"]("Heyho!"); // Heyho!
```

EIGENSCHAFTEN UND METHODEN

- Objekten können neue Attribute und Methoden auch nachträglich zugewiesen werden

```
var auto = {  
  "farbe": "rot",  
  "speed": 70  
};  
  
auto.ps = 100;  
auto.hupe = function(){  
  console.log("TRÖÖÖÖÖÖT");  
};  
console.log( auto.farbe ); // rot  
auto.hupe(); // TRÖÖÖÖÖÖT  
console.log( auto.ps ); // 100
```

EIGENSCHAFTEN UND METHODEN

- Objekte und Arrays können beliebig verschachtelt werden.

```
var produkt = {  
  "bezeichnung": "Schaufel",  
  "farben": ["rot", "grün", "gelb"],  
  "preis": { "netto": 8.5, "brutto": 10.2 }  
};
```

```
console.log(produkt.bezeichnung); //Schaufel  
console.log(produkt.farben); //["rot", "grün", "gelb"]  
console.log(produkt.farben[1]); //grün  
console.log(produkt.preis.netto); //8.5
```

JSON

- ▶ JSON: Java Script Object Notation
- ▶ JSON ist immer gültiges ausführbares Javascript.
- ▶ Gut als Austauschformat zwischen unterschiedlichen Sprachen und Systemen geeignet

```
{
  "name": "JavaScript 101",
  "kursnummer": 5101,
  "dauer": 8,
  "raum": "2.02",
  "einheiten": [
    {
      "title": "if else document",
      "beschreibung": "Beschreibungstext 123",
      "date": "2016-06-01",
      "done": true
    }, {
      "title": "loops and strings",
      "beschreibung": "Beschreibungstext 123",
      "date": "2016-06-06",
      "done": true
    }
  ]
}
```

TYPEN

- ▶ Erlaubte Typen in JSON: Boolean, Number, String, Array, Object und null
- ▶ Attributnamen müssen hier immer in doppelten Anführungszeichen stehen!
- ▶ Deklaration von Strings, Arrays, Objects nur mit den "Literals": `""`, `[]`, `{}`
- ▶ JSON kann KEINE Funktionsdeklarationen enthalten!
- ▶ JSON kann keine zyklischen Konstrukte abbilden.
- ▶ Dokumentation: <http://json.org>

JSON “HELPER” – OBJECT

Das JSON Objekt ermöglicht es aus einer bestehenden Datenkonstrukt einen JSON-String zu erzeugen oder einen String der JSON-Daten enthält auszuwerten.

JSON.stringify(data)

Erzeugt aus Daten einen JSON-String

JSON.parse(string)

Erzeugt aus einem JSON-String ein Datenkonstrukt

JSON.parse()

```
var data = JSON.parse("[1, \"yay\", {\"farbe\": \"rot\"}]");  
console.log(data); // [1, "yay", {farbe: "rot"}]
```

JSON.stringify()

```
var a = { "nr": 42, "farbe": "rot" };  
console.log(JSON.stringify(a));  
// {"nr":42,"farbe":"rot"}
```

```
a.durchsichtig = false;  
console.log(JSON.stringify(a));  
// {"nr":42,"farbe":"rot","durchsichtig":false}
```

TIMER FUNKTIONEN

- ▶ Wenn wir Funktionen erst nach einer bestimmten Zeit aufrufen wollen ermöglicht uns das die Funktion **setTimeout**
- ▶ Wenn wir Funktionen immer wieder in einem bestimmten gleichbleibenden Intervall aufrufen wollen ermöglicht uns das die Funktion **setInterval**
- ▶ Möchten wir diesen Vorgang wieder stoppen gibt es die Funktionen **clearTimeout** und **clearInterval**

START

`setTimeout(function, delay)` führt die Funktion einmal nach delay Millisekunden aus

`setInterval(function, interval)` führt die Funktion immer wieder nach interval Millisekunden aus

```
setTimeout( function(){  
    console.log("Es sind 5 Sekunden vergangen!");  
}, 5000);
```

```
var count = 0;  
setInterval( function(){  
    count++;  
    console.log("Es sind " + count + " Sekunden vergangen!");  
}, 1000);
```

ABBRECHEN

`clearTimeout(handle)` verhindert den Aufruf der Funktion (wenn sie nicht schon ausgeführt wurde)

`clearInterval(handle)` stoppt den Vorgang des immer wieder Aufrufens der Funktion

```
var handle_timeout = setTimeout( function(){  
    console.log("Es sind 5 Sekunden vergangen!");  
}, 5000);  
clearTimeout(handle_timeout); //Nein lieber doch nicht
```

```
var count = 0;  
var handle_interval = setInterval( function(){  
    count++;  
    console.log("Es sind " + count + " Sekunden vergangen!");  
    if(count > 10) clearInterval(handle_interval);  
}, 1000);
```

DAS EVENT OBJECT

<https://developer.mozilla.org/en-US/docs/Web/Events>

Das dem Handler mitübergabene Event ist ein Objekt und beinhaltet Informationen zum aufgetretenen Event.

Unterschiedliche Eventtypen (MouseEvent, KeyboardEvent, ...) haben unterschiedliche Eigenschaften

- ▶ welches Element hat das Event ausgelöst (Event.target)
- ▶ X/Y-Koordinaten (z.B.: bei Klick) (Event.pageX)
- ▶ Gedrückte Taste (z.B.: bei KeyDown) (Event.charCode)

EVENT OBJECT

```
<button id="myButton">Speichern</button>
```

```
var button = document.querySelector("#myButton");  
button.addEventListener("click", function(_event) {  
    var elm = _event.target;  
    var button_text = elm.textContent;  
    console.log("Der Button " + button_text + " wurde geklickt");  
});
```

EVENT OBJECT – PREVENT DEFAULT

```
<a id="myLink" href="www.google.com">Go somewhere</a>
```

```
var link = document.querySelector("#myLink");
link.addEventListener("click", function(_event) {
    _event.preventDefault(); //verhindert die Default Aktion des Browser
    var elm = _event.target;
    var link_text = elm.textContent;
    console.log(link_text + " wurde geklickt aber es passiert nix!");
});
```

EVENT OBJECT – PREVENT DEFAULT

```
<form id="myForm" action="doSomething.php">
  <input type="text">
  <input type="submit">
</form>
```

```
var link = document.querySelector("#myForm");
link.addEventListener("submit", function(_event) {
  _event.preventDefault(); //verhindert die Default Aktion des Browser
  console.log("Das Formular wird noch nicht Abgeschickt!");
});
```