Kurze Repetition:

JSON, LocalStorage

LocalStorage

LocalStorage

- Innerhalb von LocalStorage können Strings abgespeichert werden
  - Es können nur primitive Objekte gespeichert werden
  - Arrays & Objekte müssen umgewandelt werden
- Daten sind auch nach dem neuladen & auch neustarten noch verfügbar
- Daten sind nur für Seiten der gleichen Domain verfügbar
- Den Inhalt eurer LocalStorage könnt ihr in der Konsole verwalten
- Ähnlich: SessionStorage ist aber nur solange verfügbar, bis Browser geschlossen wird

LocalStorage

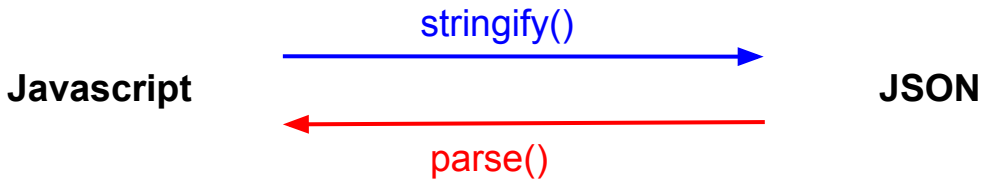
```
// Element abfragen (existiert noch nicht)
var name = localStorage.getItem('meinName');
console.log(name); // null

// Element speichern
localStorage.setItem('meinName', 'Tom');

// Element abfragen
var name = localStorage.getItem('meinName');
console.log(name); // Tom
```

JSON

Die **JavaScript Object Notation**, kurz **JSON**, ist ein kompaktes Datenformat in einer einfach lesbaren Textform zum Zweck des Datenaustauschs zwischen Anwendungen.

```
var javascriptObjekt = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');
```

```
var jsonString = JSON.stringify({ name: "John", age: 30, city: "New York"});
```



JSON Beispiel: Objekt

JSON

```
{
  "aliceblue": [240, 248, 255, 1],
  "antiquewhite": [250, 235, 215, 1],
  "aqua": [0, 255, 255, 1],
  "aquamarine": [127, 255, 212, 1],
  "azure": [240, 255, 255, 1],
  "beige": [245, 245, 220, 1],
  "bisque": [255, 228, 196, 1],
  "black": [0, 0, 0, 1],
  "blanchedalmond": [255, 235, 205, 1],
  "blue": [0, 0, 255, 1]
}
```

Javascript Objekt

```
var javascriptObjekt = {
  aliceblue: [240, 248, 255, 1],
  antiquewhite: [250, 235, 215, 1],
  aqua: [0, 255, 255, 1],
  aquamarine: [127, 255, 212, 1],
  azure: [240, 255, 255, 1],
  beige: [245, 245, 220, 1],
  bisque: [255, 228, 196, 1],
  black: [0, 0, 0, 1],
  blanchedalmond: [255, 235, 205, 1],
  blue: [0, 0, 255, 1]
};
```

JSON Beispiel: Array

JSON

```
[
  {"firstname":"Bart","name":"Simpson","age":10,"sex":"m"},
  {"firstname":"Homer","name":"Simpson","age":36,"sex":"m"},
  {"firstname":"Lisa","name":"Simpson","age":8,"sex":"w"},
  {"firstname":"Marge","name":"Simpson","age":34,"sex":"w"},
  {"firstname":"Maggie","name":"Simpson","age":1,"sex":"w"},
  {"firstname":"Hugo","name":"Simpson","age":10,"sex":"m"}
]
```

Javascript

```
var simpsons = [
  {firstname: "Bart", name: "Simpson", age: 10, sex: 'm'},
  {firstname: "Homer", name: "Simpson", age: 36, sex: 'm'},
  {firstname: "Lisa", name: "Simpson", age: 8, sex: 'w'},
  {firstname: "Marge",name: "Simpson",  age: 34, sex: 'w'},
  {firstname: "Maggie", name: "Simpson", age: 1, sex: 'w'},
  {firstname: "Hugo", name: "Simpson", age: 10, sex: 'm'}
];
```

JSON Daten im Browser speichern

```
var liste = [
  {firstname: "Bart", name: "Simpson", age: 10, sex: 'm'},
  {firstname: "Maggie", name: "Simpson", age: 1, sex: 'w'},
  {firstname: "Hugo", name: "Simpson", age: 10, sex: 'm'}
];

var json = JSON.stringify(liste);
localStorage.setItem('liste', json);
```

JSON Daten im Browser auslesen

```
var json = localStorage.getItem('liste');
if (json) {
  var liste = JSON.parse(json);
} else {
  var liste = []; // Default Wert setzen, z.B. leere Liste
}
```

Fragen zu Übungen?

Javascript Delegation

Element.matches()

```
var result = element.matches(selectorString);
```

```
<div id="element" class="a b c">Text</div>
```

```
var element = document.querySelector('#element');
console.log(element.matches('.a.b.c')); //true
console.log(element.matches('.a.b'));   //true
console.log(element.matches('div.a'));  //true
console.log(element.matches('.d'));     //false
```

Elemente aus dynamischer Liste löschen

```
<ul id="parent-list">
      <li id="post-1">Item 1 <button>Löschen</button></li>
      <li id="post-2">Item 2 <button>Löschen</button></li>
      <li id="post-3">Item 3 <button>Löschen</button></li>
</ul>
```

```
<ul id="parent-list">
      <li id="post-1">Item 1 <button>Löschen</button></li>
      <li id="post-2">Item 2 <button>Löschen</button></li>
      <li id="post-3">Item 3 <button>Löschen</button></li>
</ul>
```

Elemente aus dynamischer Liste löschen

Problem:

- Ihr müsst für jedes Element, dass gelöscht werden kann, einen EventListener hinzufügen
- Für nachträglich hinzugefügte Elemente müsst ihr auch einen EventListener hinzufügen

```
<ul id="parent-list">
      <li id="post-1">Item 1 <button>Löschen</button></li>
      <li id="post-2">Item 2 <button>Löschen</button></li>
      <li id="post-3">Item 3 <button>Löschen</button></li>
</ul>
```

Elemente aus dynamischer Liste löschen

Klassischer Ansatz:

```
// Event Listener für bestehende Elemente
var buttons = document.querySelectorAll('button');
for (var i = 0; i < buttons.length; i++) {
  buttons[i].addEventlistener('click', removeItem);
}

// Hinzufügen eines neuen Elements
function addItem(text) {
  var newItem = document.createElement('li');
  newItem.innerHTML = text + '<button>Löschen</button>';
  document.querySelector('#parent-list').appendChild(newItem);
  newItem.addEventListener('click', removeItem);
}
```

```
function removeItem(event) {
  event.target.parentNode.removeChild(event.target);
}
```

```
<ul id="parent-list">
    <li>Item 1 <button>Löschen</button></li>
    <li>Item 2 <button>Löschen</button></li>
    <li>Item 3 <button>Löschen</button></li>
</ul>
```

Event Delegation

Ansatz Event Delegation:

- Event Listener auf nächst höheres Element setzen
- Auf Event Bubbling warten
- Überprüfen ob event.target gewünschtem CSS-Selektor entspricht
- Falls ja, Code ausführen

Vorteile:

- Nur 1 Event Listener nötig
- Keine Event Listener für Einträge müssen hinzugefügt oder entfernt werden
- Funktioniert mit dynamischer Anzahl von Elementen

Nachteil dieser Lösung:

- Komplizierter zu verstehen

Event Delegation

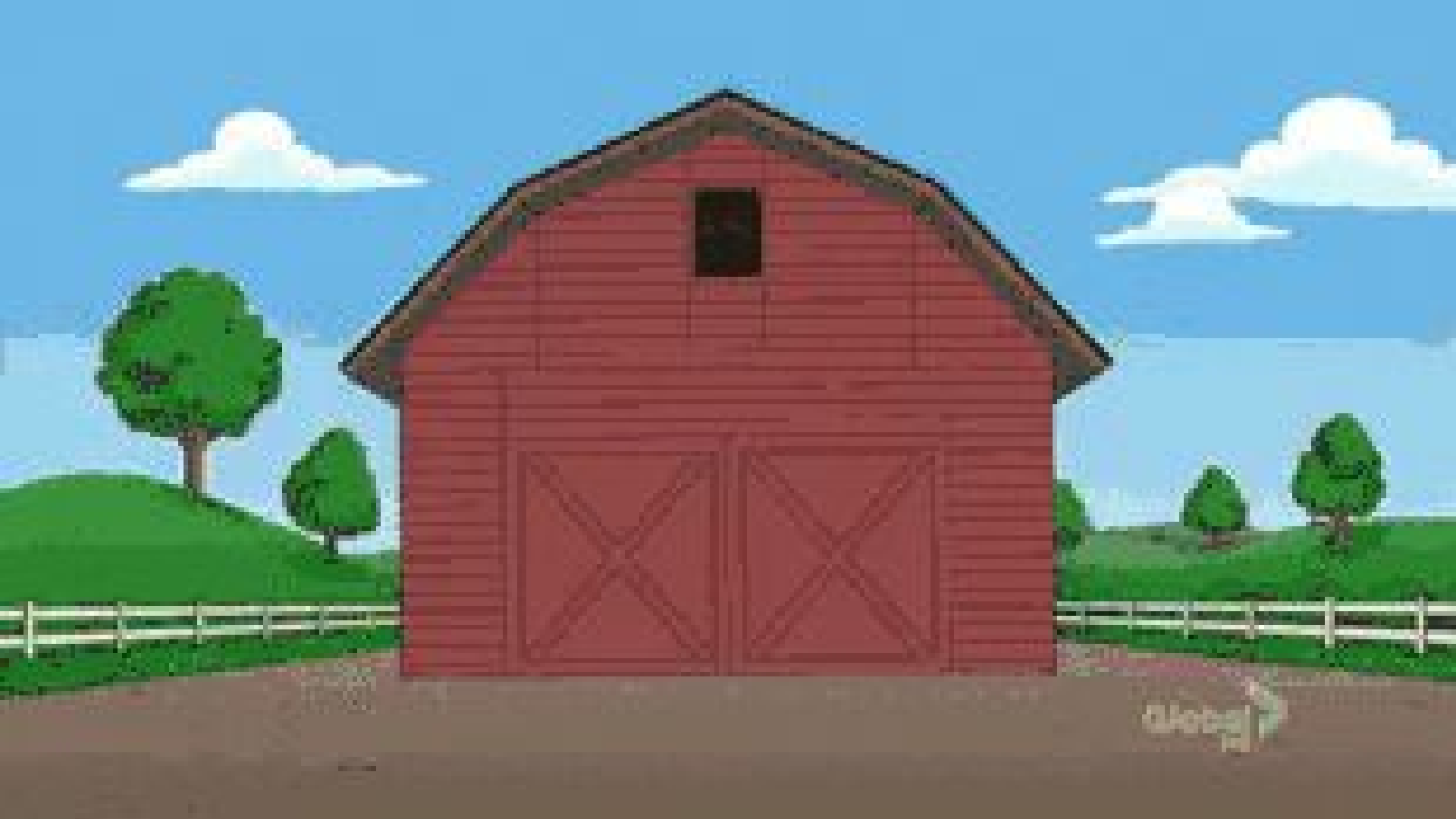
CSS Selektor der auf Button zutrifft

```
document.querySelector("#parent-list")
.addEventListener("click", function(event) {
  if (event.target && event.target.matches("li button")) {
    removeItem(event);
  }
});
```

```
<ul id="parent-list">
      <li id="post-1">Item 1 <button>Löschen</button></li>
      <li id="post-2">Item 2 <button>Löschen</button></li>
      <li id="post-3">Item 3 <button>Löschen</button></li>
</ul>
```

Delegation Demo

Listen
Rendern



Listen Render Ansatz

Ansatz Listen Rendern

- Liste der Todos wird bei jeder Änderung neu aufgebaut
- D.h. zuerst werden **ALLE** alten 's gelöscht
- Danach werden neue eingefügt
- Aufbau wird von einer eigenen Funktion gehandhabt

Vorteile:

- Wir müssen nur noch mit dem Array arbeiten und können dann die render() Funktion aufrufen
- Dadurch können wir hinzufügen, entfernen, sortieren einfacher implementieren

Array.forEach()

```
var array1 = ['a', 'b', 'c'];

array1.forEach(function(element) {
  console.log(element);
});
```

```
var array1 = ['a', 'b', 'c'];

for (var i = 0; i < array1.length;
i++) {
  console.log(array1[i]);
}
```

Listen Render Ansatz

```
function renderList() {
  var liste = document.querySelector('#liste');
  liste.innerHTML = ''; // alte Liste löschen
  simpsons.forEach(function (elem, index) {
    var li = document.createElement('li'); // Neues Element erzeugen
    li.innerHTML = elem.firstname + ' ' + elem.name + ' <button>Löschen</button>'; // Daten abfüllen
    li.id = index; // Id setzen (später wichtig für's löschen)
    liste.appendChild(li); // Element an Fragment hängen
  });
}
```

Übungen zu Delegate, JSON, Render und Storage

Kopiert aus dem Google Drive Datei aus:
Module/Javascript/lektion6.zip in euer **htdocs** Verzeichnis in einen eigenen Ordner und macht:

- uebung_2_1.html
- uebung_2_2.html

uebung_2_5.html von Lektion 4 ist Vorbedingung für uebung_2_1 und 2_2 von Lektion 5

Fehlersuche

Fehler

```
console.log(document.querySelector('gibtsNicht').firstChild);
```

```
TypeError: Cannot read property 'firstChild' of null

TypeError: Cannot read property 'firstElementChild' of null

TypeError: Cannot read property 'lastChild' of null

TypeError: Cannot read property 'addEventListener' of null

TypeError: Cannot read property 'firstChild' of null
```

Fehler

```
obj.length = 777;
```

```
TypeError: Cannot set property 'length' of undefined
```

Fehler

```
var x = [1,2,3,4];

x();
```

```
TypeError: x is not a function

TypeError: undefined is not a function

TypeError: null is not a function
```

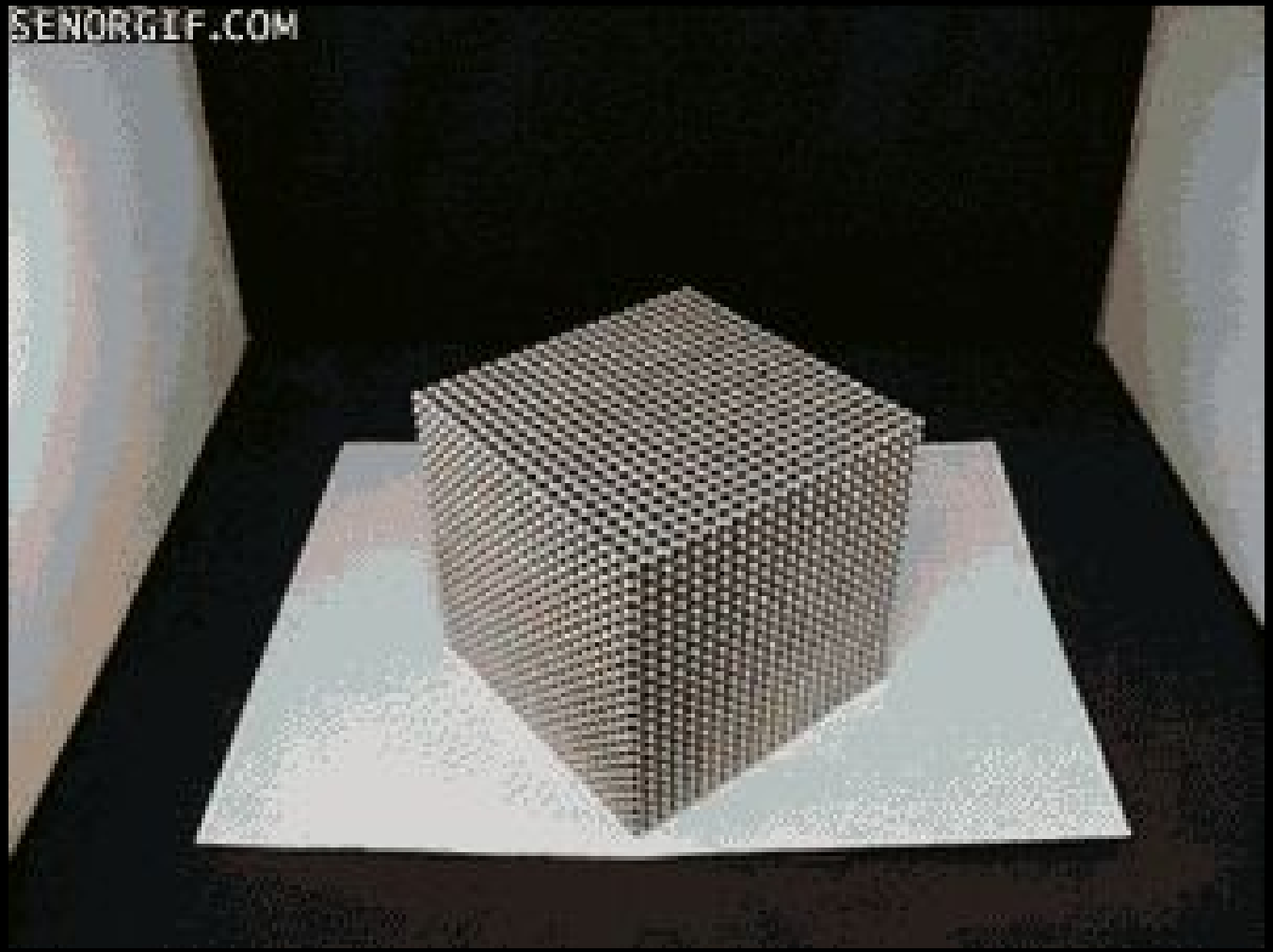
Fehler

```
foo.substring(1); // ReferenceError: "foo" is not defined
```

```
function numbers() {
  var num1 = 2,
      num2 = 3;
  return num1 + num2;
}
```

```
console.log(num1); // ReferenceError num1 is not defined.
```

Divide & Conquer



Divide and Conquer

Problemlösungsstrategie die schneller zum Ziel führt

```
document.querySelector('#liste')
.addEventListener('onclick', function(event) {
  event.target.parentNode.removeChild(event.target);
});
```

Problem: Element wird nicht entfernt!

Divide and Conquer

```
document.querySelector('#liste')
.addEventListener('onclick', function(event) {
  console.log('test');
  event.target.parentNode.removeChild(event.target);
});
```

2 mögliche Resultate

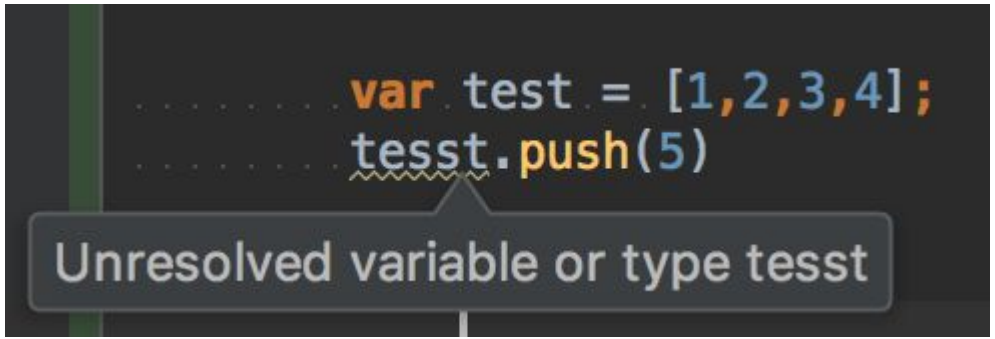
'test' erscheint NICHT auf der Konsole:
=> Methode wird gar nie aufgerufen,
=> Problem liegt am Eventlistener

'test' erscheint auf der Konsole:
=> Methode wird aufgerufen
=> Code zur entfernung des Elements funktioniert nicht richtig

IDE Tricks

```
var test = [1,2,3,4];
tesst.push(5)
```

Gelb unterstrichen = Variable wurde nicht initialisiert



IDE Tricks

```
var teest = [1,2,3,4];
```

Grauer Text = Variable wird nicht verwendet