

WEB

INHALT

- DOM Event Model
- API
- XHR

DOM EVENT MODEL (LEVEL 3)

DREI MÖGLICHKEITEN EIN EVENT ZU REGISTRIEREN:

1. `EventTarget.addEventListener`
2. HTML attribute
3. DOM element properties

EVENTTARGET.ADDEVENTLISTENER

Kein Support in IE 6-8: EventTarget.attachEvent

```
// Assuming myButton is a button element
myButton.addEventListener( 'click',
                           function(){alert('Hello world');}
                           );
```

HTML ATTRIBUTE

```
<button onclick="alert('Hello world!')">
```

DOM ELEMENT PROPERTIES

Leider nur ein Eventhändler pro Element und pro Event

```
// Assuming myButton is a button element  
myButton.onclick = function(event){alert('Hello world');};
```

EVENTS:

- click, dblclick
- key/ -down, -up
- mouse/ -down, -enter, -move, -leave
- load, unload
- scroll

WEB-API

APPLICATION PROGRAM INTERFACE (API)

- Programmierschnittstelle
- Anbindung an das System ermöglichen
- z.B an Datenbanken, WebServern, oder Modulen
- Hat nichts mit dem DOM zu tun

HERAUSFORDERUNG

schnelle abfragen udn analysieren von Daten

- Facebook, another shitty social media platform
- Daten in der Cloud
- NSA

BISHER GELÖST:

1. Request an den Server
2. Server holt Daten aus der Datenbank
3. Serverseitig wird html mit den Daten erstellt bzw zusammengefügt
4. komplette HTML-Seite geht zurück an den Client
5. User drückt nochmal auf F5
6. Beginne bei Punkt 1



BESSER:

- Server bietet Datenschnittstelle an.

```
GET servername:port/api/blog/11 //Den Blogpost mit der ID 11 laden
GET servername:port/api/blogs //Alle Blogposts laden
PUT servername:port/api/user/11 //updated User mit der ID 11
DELETE servername:port/api/user/11 //loesche User mit der ID 11
```

GET SERVERNAME:PORT/BLOG/11

```
{  
  name: "codekitty's awesome blog",  
  date: "25.05.1987",  
  content: "This is my first awesome blog entry..so much awesomness"  
}
```

BESSER:

- Server bietet Datenschnittstelle an.
- Client erhält nur einmal HTML und cached dieses.
- Es werden nur noch Daten nachgeladen.

WICHTIG:

- GET: laden von Ressourcen
- PUT: updaten von Ressourcen
- POST: neu anlegen einer Ressource
- DELETE: löschen einer Ressource



XMLHTTPREQUEST

XHR

- XHR steht für XMLHttpRequest
- Neue API ist `HTTP fetch` (aber ist dank MS noch nicht standardisiert)
- Aktuell: XHR2

XHR

- Bisher: Request an den Server mithilfe des form-Tags
- individuelle Erstellung des Request mit XHR2
- Festlegung von HTTP-Methode, Adresse,
ResponseType

BEISPIEL: (EMPFANGEN VON DATEN)

```
var xhr = new XMLHttpRequest();  
xhr.open('GET', 'http://server/api/');  
xhr.responseType = 'json';  
xhr.onload = function() {  
    var data = xhr.response;  
    if (data !== null) {  
        console.log(data); // Parsed JSON object  
    }  
};  
xhr.send(null);
```

BEISPIEL: (SENDEN VON DATEN)

```
var formData = new FormData();  
formData.append('username', 'johndoe');  
formData.append('id', 123456);  
  
var xhr = new XMLHttpRequest();  
xhr.open('POST', '/server', true);  
xhr.onload = function(e) { ... };  
  
xhr.send(formData);
```