Version 2021a

BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
DRESDEN
UNIVERSITY OF COOPERATIVE EDUCATION

Grundlagen der Web-Programmierung

# JavaScript-Toolkits

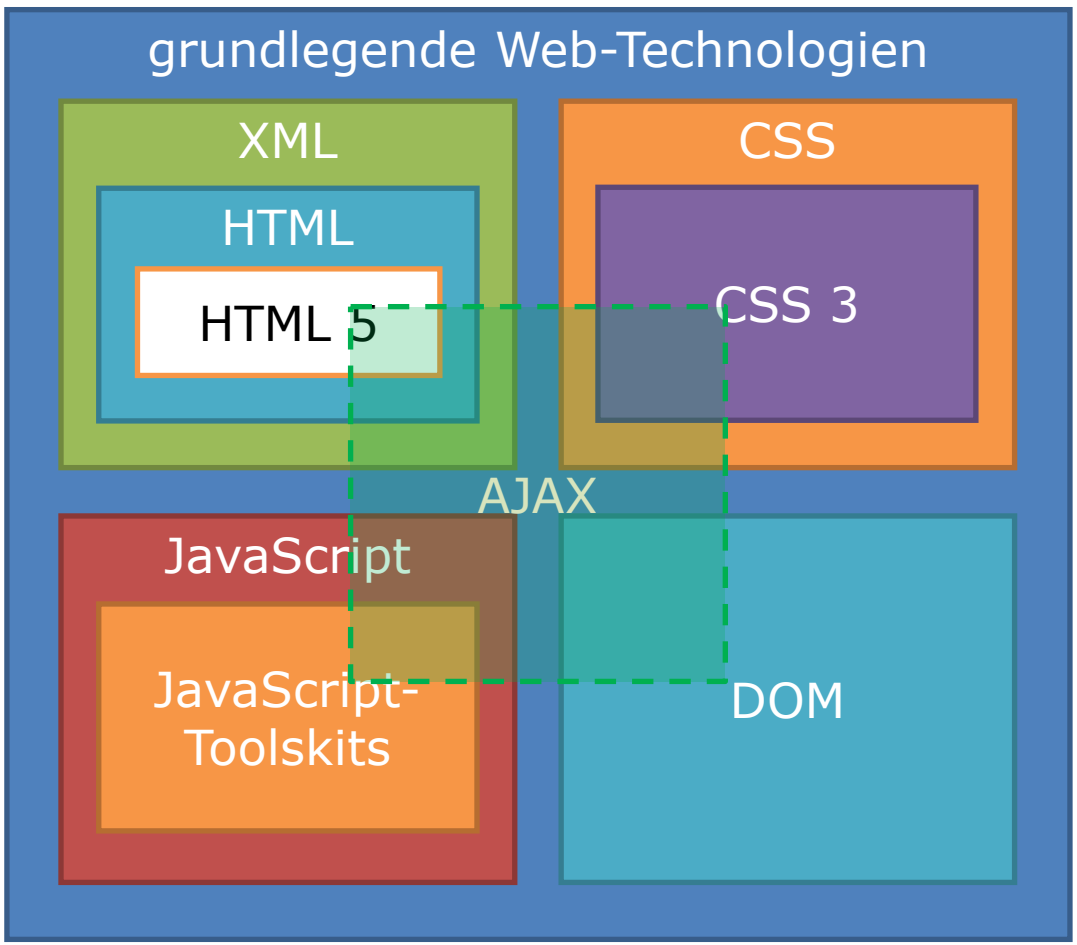
**Prof. Dr.-Ing. Tenshi Hara**
tenshi.hara@ba-sachsen.de



# Aufbau der Lehrveranstaltung

Progressive Web Apps

Search Engine Optimization





Proprietäre Standards

# Gründe für JavaScript-Toolkits

- manuelles Implementieren von JavaScript-Kode ist
  - fehleranfällig
  - zeitaufwändig
  - schwer zu handhaben
  - teilweise MutEx-belastet
- JavaScript-Frameworks (JST) erleichtern das Programmieren
  - Abstraktion und Vereinfachung regelmäßiger Aufgaben (AJAX, Ein-/Ausblenden von Elementen, Animationen, Event-Handler, …)
  - Parallelisierung von Skripten (teilweise integrierte MutEx-Prüfung)
  - Entkopplung von Client-Aktionen und Netzwerkverkehr (AJAX!)

# Gründe gegen JavaScript-Toolkits

- i.d.R. deutlich mehr Funktionalität als benötigt
  - Übertagung unnötiger Daten
  - langsam / schlechte Performance
  - Umfang oft nicht überschaubar; Doppelungen durch Legacy-Code
- Entwickler des JST muss vertrauenswürdig sein
- oft schwer zu debuggen
  (Liegt das Problem im eigenen Kode oder im JST?)
- Abhängigkeit vom JST
  - ggf. aufwändiges Nachimplementieren fehlender Funktionalität
  - ggf. machen JST-Updates eigenen Kode kaputt oder obsolet
  - ggf. ist JST mit anderen Komponenten der Web-Seite inkompatibel

# Side-Loading

- Einbinden von extern gehosteten JST
  - erspart das Hosten
  - immer aktuelle Version
    (Version-Pinning i.d.R. möglich)
  - ggf. schnellerer Seitenaufbau beim Client, falls andere Seite bereits gleiches JST verwendet hat
- problematisch bei Kombination von Skripten
  - Cross Site Scripting (XSS Attack)
  - Cross Site Request Forgery (CSRF Attack)
- erfordert volles Vertrauen in bereitstellende Instanz
  - i.d.R. keine Prüfung des bereitgestellten JST möglich
  - plötzlicher Funktionsverlust durch Updates bei fehlendem Version-Pinning möglich

# Self-Hosting

- hosten des JST auf eigenem Server
  - Version automatisch bis zum nächsten Update fixiert
  - garantiert mit der eigenen Seite verfügbar
  - ggf. unnötige Datentransfers für den Client
- eliminiert im JST begründete Probleme bei Kombination von Skripten
  - weder für XSS noch für CSRF anfällig
    (Ausnahmen: JST lädt Komponenten nach, unsauber implementiert)
  - lokal definierte und hinterlegte, lauffähige Kombination von Skripten
- einfaches Verändern des JST möglich
  - Anpassung des JST in den Quelldateien
    (bspw. Minimierung auf Benötigtes oder Funktionalitätsänderung)
  - kein aufwändiges Überschreiben von JST-Teilen nach dem Laden
  - reduziert unnötig übertragene Daten

# Vergleich: Vanilla JS vs JST

Elementauswahl durch ID

| | Kode | Operationen pro Sekunde |
|---|---|---|
| Vanilla JS | `document.getElementById('test-table');` | 27.182.049 |
| Dōjō | `dojo.byId('test-table');` | 24.613.451 |
| Prototype JS | `$('test-table')` | 9.102.829 |
| Ext JS | `Ext.get('test-table');` | 4.374.603 |
| YUI | `YAHOO.util.Dom.get('test-table');` | 3.428.912 |
| jQuery | `$_('#test-table');` | 1.740.788 |
| MooTools | `document.id('test-table');` | 162.207 |

weit verbreitet, da leicht zu lernen

Test-Browser: Chrome 69.0.3497 unter Windows 10 (1809)

Test-System: Intel Core i5 7500T, 8 GB RAM

Quelle: vanilla-js.com


dōjō
dojo.io

# EIGENSCHAFTEN

- JST für großangelegte client-seitige Web-Entwicklung
  - schnelle, plattform-übergreifende Entwicklung von (Desktop- und Web-)Anwendungen und Web-Seiten
  - mit Fokus auf AJAX und JavaScript-getriebene Oberfläche
- abstrahiert unterschiedliche Browser-APIs auf gemeinsame Entwicklungs-API (Dōjō-API)
- auch server-seitig einsetzbar (in Kombination mit Node.js)
- Open Source → wird auch bei Google-APIs gehostet

# Aufbau

- IDE beinhaltet Unterstützung für
  - Modularisierung und Abhängigkeitsprüfung
  - JavaScript- und CSS-Optimierung
  - automatisierte Generierung einer Dokumentation
  - Unit-Tests
  - Internationalisierung
  - Barrierefreiheit
  - Standard-Aufgaben (bspw. GUI-Widgets)
- unterteilt sich in
  - dōjō - JST-Kern und fast alle nicht-visuellen Module
  - dijit - GUI-Bibliothek
  - dōjōx - experimentelle dōjō- und dijit-Module
  - util - Build-Tools

# Widgets

- sind Komponenten aus JS-Kode, HTML-Schnipsel und CSS-Deklarationen
- bieten interaktive Eigenschaften für multiple Browser (aber nicht plattformübergreifend)
  - Menüs, Reiter und Hinweisdialoge (Tooltips, …)
  - sortierbare Tabellen und tabellenähnliche Container
  - dynamische Grafiken
  - 2D-Vektorgrafiken
  - animierte Effekte (Ein-/Aufblenden, Wischen, …)

# ASYNCHRONE KOMMUNIKATION

- optimiert auf AJAX mit statischem Fallback
- abstrahierter Wrapper für den XMLHttpRequest: dojo.xhr
- weitere Transportmechanismen (versteckte Iframes, …): dojo.io
- unterstützt multiple Transferkodierungen, nicht nur XML
  → insbesondere kann/soll Server mit nachgeschobenem JS antworten

# PAKETIERUNG

- Umsetzung modularer Entwicklung
  → Funktionalität kann (und soll) auf Pakete und Unterpakete geteilt werden
- Basis Paket `bootstrap` fügt alles zusammen
  - initialisiert Hierarchie
  - Unternamensräume im `dojo`-Namensraum (`dojo.io`, `dojo.event`, …)
  - weitere Namensräume können deklariert werden
    - Erweiterungen
    - Überlagern von Funktionen
    - private Funktionen zum Schutz bei Cross-Domain-Loading
  - weitere Pakete können via XMLHttpRequest nachgeladen werden
- Pakete können Abhängigkeiten deklarieren, die nachgeladen werden müssen (auch externe → Cross-Domain-Loading)
- Zusammenfassung aller benötigten Pakete und Abhängigkeiten in einer einzelnen JS-Datei möglich (Profiling)

# Speicher

client-seitiger Speicher

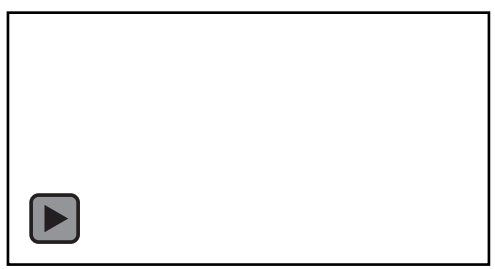
- Lesen und Schreiben von Cookies
- HTML5-Storage

server-seitiger Speicher

- CsvStore - RO-Speicher für Tabellendaten aus CSV-Quellen
- OpmlStore - RO-Speicher für hierarchische Daten aus OPML-Quellen
- YahooStore - RO-Speicher für Suchergebnisse aus der Yahoo!-Suche
- DeliciousStore - RO-Speicher für Lesezeichen von del.icio.us
- RdfStore - RW-Speicher für Zugriff auf RDF-Daten-Server mittels SPARQL (bspw. Rhizome RDF Application Server)

# Beispiel: einfliegender Text

```
<html><head>
    <title>„Hallo Welt!“ mit Dojo</title>
    <style>body{margin:20;padding:0}input{font-size:24pt}</style>
</head><body>
    <script src="https://ajax.googleapis.com/ajax/libs/dojo/1.10.4/dojo/dojo.js"
        data-dojo-config="async: true"></script>
    <input type="button" id="anker" value="Klick mich!" />
    <script>
        require(
            ["dojo/fx", "dojo/on", "dojo/dom", "dojo/domReady!"],
            function(fx, on, dom) {
                var button = dom.byId("anker");
                var slide = true;
                on(button, "click", function(evt){
                    if (slide) {
                        fx.slideTo({ node: button, left: "100", top: "100" }).play();
                        button.value = "Hallo Welt!";
                        slide = false;
                    } else {
                        fx.slideTo({ node: button, left: "20", top: "20" }).play();
                        button.value = "Klick mich!";
                        slide = true;
                    }
                });
        });
    </script>
</body></html>
```



BERUFSAKADEMIE SACHSEN
STAATLICHE STUDIENAKADEMIE
DRESDEN
UNIVERSITY OF COOPERATIVE EDUCATION

# jQuery

jquery.com

# Eigenschaften (1/2)

- große JavaScript-Bibliothek zur DOM-Manipulation
  - vereinfachte Syntax zur Adressierung, Auswahl und Veränderung von DOM-Elementen
  - erweitert DOM um Ereignisbehandlung (Event-Zuweisung und Callback werden an einer Stelle definiert)
- vereinfachter Umgang mit visuellen Effekten durch gezielte CSS-Manipulation
- Open Source → wird auch bei Google-APIs gehostet
- weit verbreitet und beliebt
  - einfach zu erlernen
  - integriert in Wordpress, VisualStudio, Nokia Web-Runtime, …

# Eigenschaften (2/2)

- speziell auf AJAX zugeschnittene Objekteigenschaft: Promise/Future
- Zugriff auf HTML5 Utilities (insb. Feature Detection)
- Kompatibilitätsfunktionen für ältere Browser
  - Nachbau von HTML5-Standardfunktionalität
  - Nachbau von CSS3-Standardfunktionalität
  - Nachbau von JavaScript-Funktionen (`$.inArray()`, `$.each()`, …)

Achtung: Kompatibilitätsgarantie beschränkt sich auf „Current − 1“-Version der gängigen Browser

# Grundprinzipien

- Separation von JavaScript und HTML durch simples Anhängen von Events an DOM mittels JS anstatt HTML-Events (welche wiederum JS aufrufen)
- „kurz und knapp"-Mantra
  - Verkettung/Aneinanderreihung von Funktionen
  - Abkürzungen und Pseudonyme für Funktionen
- Eliminieren von Browser-Inkompatibilität
  - Abstraktion der Browser-eigenen JavaScript-Implementierung durch multiple Fallback-Mechanismen
  - konsistente Schnittstelle für den Entwickler
- Erweiterbarkeit
  - einfaches Hinzufügen neuer Events, Elemente und Methoden
  - Wiederverwendbarkeit in Form von Plug-Ins

# FUNKTIONEN

- Objektmethoden
  - jQuery/$ agiert als Factory
  - entsprechend DOM-Auswahl werden Elemente selektiert
  - dienen der DOM-Manipulation und Effektausführung
  - Beispiel: `$(".bsp").css("border-color", "red");`
- statische Werkzeugmethoden
  - jQuery/$ agiert als Utility-Identifier
  - aus Browser-Engine werden statische Funktionen ausgeführt
  - sind nicht DOM-Element-spezifisch
  - Beispiel: `$.ajax({url: "test.js", dataType: "script"});`
- Ausgangspunkt (nach vollständigen Laden des DOM):

```
$(function(){/* initialer Kode */});
```

# Formularübertragung – typischer AJAX-Fall

```
$.ajax({
   type: 'POST',
   url: '/pfad/beispiel.php',
   data: {
      vorname : 'Max',
      nachname : 'Mustermann',
   },
}).then(function(msg) {
   alert('Daten an Server gesendet: ' + msg);
}).catch(function(xmlHttpRequest, statusText, errorThrown) {
   alert(
      'Formularverarbeitung fehlgeschlagen.\n\n'
      + 'XmlHttpRequest: ' + JSON.stringify(xmlHttpRequest)
      + ',\nStatus-Text: ' + statusText
      + ',\nFehler: ' + errorThrown);
});
```

# DATUMSAUSWAHL

HTML-Kode:

```
<input type="date" name="date" id="datepicker"
   value="2019-03-18" size="10" maxlength="10"
   autocomplete="off"/>
```

zugehöriger jQuery-Kode:

```
<script type="text/javascript" language="javascript">
   $(function(){
      $("#datepicker")
      .datepicker({
         changeMonth : true,
         changeYear : true,
         dateFormat : "yy-mm-dd",
         minDate : "-360D",
         maxDate : 0,
         firstDay : 1
      });
   });
</script>
```