WBE Summary (beliebig lang)

19. September, 2024; rev. 12. Januar 2025

Linda Riesen (rieselin)

1 Vorlesung 02

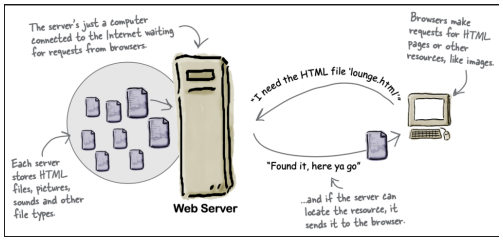

Abbildung 1: Web-Architecture

Javascript heisst so wegen marketing (hat wenig mit java gemeinsam)

- Dynamisches Typenkonzept
- Objektorientierter und funktionaler Stil möglich

1.1 REPL = Read Eval Print Loop

node ohne Argument startet die interaktive REPL

- JavaScript interaktiv
- Auto-Vervollständigung von Funktions- und Objektnamen
- _ liefert Resultat der letzten Operation
- .help gibt Hilfe zu weiteren Kommandos aus

1.2 Zahlentyp in JavaScript: Number

- 64 Bit Floating Point entsprechend IEEE 754 (wie double in Java)
- Enthält alle 32 Bit Ganzzahlen
- Konsequenz: alle Java int auch in JavaScript exakt dargestellt
- Weitere Konsequenz: oft Rechenungenauigkeit bei Zahlen mit Nachkommastellen

1.3 BIGINT

- Mit ES2020 eingeführt
- Literale mit anghängtem n
- Keine automatische Typumwandlung von/zu Number

```
typeof 12          // 'number'
typeof(12)         // 'number'
typeof 2n          // 'bigint'
typeof Infinity    // 'number'
typeof NaN         // 'number'  !!
typeof 'number'    // 'string'
```

Abbildung 2: Typeof

```
`half of 100 is ${100 / 2}`    // 'half of 100 is 50'
`erste Zeile
zweite Zeile`                  // 'erste Zeile\nzweite Zeile'
```

Abbildung 3: Tempalte String

1.4 Dynamisches Typenkonzept

- Typen werden bei Bedarf konvertiert
- Dies kann zu unerwarteten Ergebnissen führen
- Problematisch: Überladener Operator + kombiniert mit Typumwandlung

1.5 Vergleich

- == : Vergleich mit automatischer Typkonvertierung
- === : Vergleich ohne Typkonvertierung (oft vorzuziehen)
- Ebenso: != und !==

1.6 Funktionen (sind Objekte)

- Strukturierung von Programmen
- Vermeiden von Redundanz
- Namen für Code-Stücke
- Funktion kann zugewiesen werden
- Schlüsselwort function
- Parameterliste, Rückgabewert
- Die Parameterliste kann leer sein (es gibt kein overload, es wird so oder so die funktion aufgerufen)
- Die return -Anweisung kann fehlen: es wird undefined zurückgegeben
- Funktionen können an Variablen gebunden werden Diese Bindung kann jederzeit neu zugewiesen werden (ausser es ist eine Konstante)
- Funktionen auch als Parameter und Rückgabewert möglich: First Class Citizens
- Funktionsdeklarationen werden zuerst ausgewertet: Aufruf kann daher vor Deklaration stehen
- Anzahl Parameter muss nicht mit der Anzahl beim Aufruf übergebener Argumente übereinstimmen
  - Fehlende Argumente: sind undefined
  - Überzählige Argumente: werden ignoriert
  - Falls Argument undefined ist, wird Default eingesetzt
  - Default-Parameter stehen am Ende der Parameterliste

1.7 Pfeilnotation Funktionen

Genau ein Parameter: Parameterliste muss nicht geklammert werden Nur ein Ausdruck: return und Block-Klammern können entfallen

2 Vorlesung 03

Was	Objekt	Array
Art	Attribut-Wert-Paare	Sequenz von Werten
Literalnotation	werte = { a: 1, b: 2 }	liste = [1, 2, 3]
Ohne Inhalt	werte = { }	liste = [ ]
Elementzugriff	werte["a"] oder werte.a	liste[0]

Abbildung 4: Array vs Objects

- Objekte sind dynamische Datenstrukturen
- Sie können jederzeit erweitert werden
- Objekte können jederzeit verkleinert werden
- Mit delete wird ein Attribut entfernt
- Mit in kann überprüft werden, ob ein Attribut existiert

2.1 Methoden

- Attribute, deren Werte Funktionen sind, werden Methoden genannt
- Sie werden über das Objekt aufgerufen
- **this** ist das Objekt, über das die Methode aufgerufen wird
  - Das gilt nicht für Funktionen in Pfeilnotation

2.2 Spread Syntax

```
> let objectA = { a: 1, b: 2 }
> let objectB = { c: 100, d: 200 }

> {...objectA, ...objectB, c: 3}
{ a: 1, b: 2, c: 3, d: 200 }

> {...objectA}
{ a: 1, b: 2 }

> {...objectA} == objectA
false
```

Abbildung 5: Spread Operator

2.3 Deconstruct Objects

Teile aus (möglicherweise grossen) Objekten extrahieren Auch in Funktionsparametern möglich (spätere Lektion)

2.4 Array

- shift , unshift : Einfügen und Entfernen am Array-Anfang
- indexOf , lastIndexOf : Element im Array finden
- slice : Bereich eines Arrays ausschneiden
- concat : Arrays zusammenhängen
- at : Zugriff auf Index

2.5 Wertetypen

Zahlen, Strings und Wahrheitswerte sind Wertetypen Sie sind unveränderlich Zuweisung kann wie Kopieren behandelt werden

2.6 Referenz Datatypes

Objekte und Arrays sind Referenz-Datentypen Sie sind jederzeit veränderbar Es werden Referenzen zugewiesen

2.7 String Methoden

- slice : Ausschnitt aus einem String (vgl. Arrays)
- indexOf : Position eines Substrings (vgl. Arrays)
- trim : Whitespace am Anfang und Ende entfernen
- padStart : vorne Auffüllen bis zu bestimmter Länge
- split : Auftrennen, liefert Array von Strings
- join : Array von Strings zusammenfügen
- repeat : String mehrfach wiederholen

3 Vorlesung 04

3.1 Gültigkeitsbereich (scope)

```
let   m = 10       // variable with block scope
const n = 10       // constant with block scope
var   p = 10       // variable with function scope
```

Abbildung 6: Variable Scope

3.1.1 Globale Variablen

- Ausserhalb von Funktionen definiert
- Oder in Funktionen, aber const , let oder var vergessen
- Gültigkeitsbereich möglichst einschränken (Block, Funktion)

3.2 Pure Function

- Rückgabewert der Funktion ist ausschliesslich abhängig von den übergebenen Argumenten
- Keine weiteren Abhängigkeiten
- Keine Seiteneffekte

4 Vorlesung 05

4.1 Strict mode

- Behebt einige potenzielle Fehlerquellen in JavaScript
- Aktiviert am Anfang des Scripts / der Funktion durch üse strict"
- Im strict mode ist this bei Funktionsaufruf undefined

4.2 call / apply

- Methoden call und apply von Funktionen
- Erstes Argument: Wert von this in der Funktion
- Weitere Argumente von call : Argumente der Funktion
- Weiteres Argument von apply : Array mit den Argumenten

4.3 bind

- Noch eine Methode von Funktionen: bind
- Erzeugt neue Funktion mit gebundenem this
- Auch weitere Argumente können gebunden werden

4.4 protoype

Die meisten Objekte haben ein Prototyp-Objekt Dieses fungiert als Fallback für Attribute und Methoden Object.prototype (Vererbung) => Klassen gibts auch (als vereinfachung von prototype)

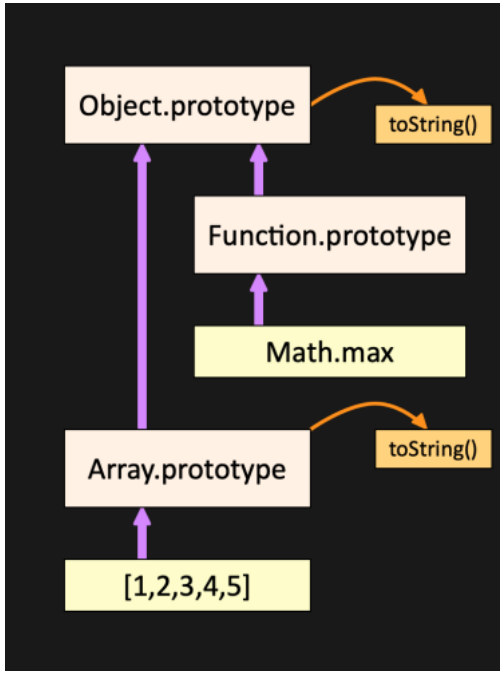

Abbildung 7: objectPrototype

Jasmine for TDD (Test driven Development)

5 Vorlesung 06

Lesen asynchron mit fs (require ('fs')) file system
Ein **Callback** ist eine Funktion, welche als Argument einer anderen Funktion übergeben wird und erst aufgerufen wird, wenn das Ereignis eingetreten ist (event listener)

Streams für grössere Dateien

Funktion	Bedeutung
`fs.access`	Zugriff auf Datei oder Ordner prüfen
`fs.mkdir`	Verzeichnis anlegen
`fs.readdir`	Verzeichnis lesen, liefert Array von Einträgen
`fs.rename`	Verzeichnis umbenennen
`fs.rmdir`	Verzeichnis löschen

Abbildung 8: File System Verzeichnisse

Funktion	Bedeutung
`fs.chmod`	Berechtigungen ändern
`fs.chown`	Besitzer und Gruppe ändern
`fs.copyFile`	Datei kopieren
`fs.link`	Hardlink anlegen
`fs.symlink`	Symbolic Link anlegen
`fs.watchFile`	Datei auf Änderungen überwachen

Abbildung 9: File System Modul

JavaScript-Code wird in einem Thread abgearbeitet Probleme vermieden, die bei paralleler Ausführung auftreten können (gemeinsame Ressourcen, mögliche Blockaden) Vorsicht: Thread darf nicht blockiert werden Browser: Tabs normalerweise mit unabhängigen Event Loops

ABLAUF

- Script wird ausgeführt
- Funktionsaufrufe → Call Stack
- Callbacks asynchroner Operationen in Event Queue(s) gelegt
- Wenn Call Stack leer, d.h. (synchrone) Aufrufe abgearbeitet:
  - Übergang in eine so genannte Event Loop
  - Callbacks aus Event Queue abgearbeitet
  - Event Queue leer: Programm beendet

Abbildung 10: Program Ablauf

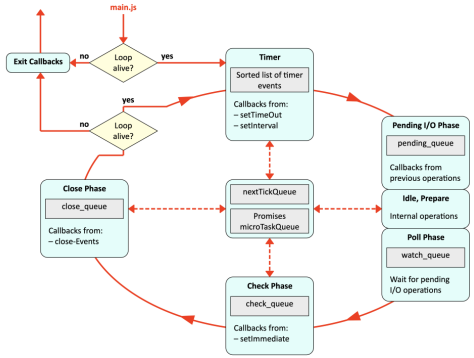

Abbildung 11: Queue

5.1 Event Emitter

const EventEmitter = require('events') const door = new EventEmitter() Verwaltet Liste von Listeners zu bestimmten Events Listener für das Event können hinzugefügt oder entfernt werden Event kann ausgelöst werden → Listener werden informiert

5.1.1 SetTimeout

- Mit setTimeout kann Code definiert werden, der zu einem späteren Zeitpunkt ausgeführt werden soll
- Eintrag in die Timer-Liste, auch wenn Zeit auf 0 gesetzt wird
- Kann mit clearTimeout entfernt werden

5.1.2 SetInterval

- Callback alle n Millisekunden in die Callback Queue eingefügt
- Kann mit clearInterval beendet werden

5.1.3 SetImmediate

- Node.js API (im Browser nicht unterstützt)

- Callbacks, die direkt nach der Poll Phase ausgeführt werden
- Damit: Unterschied zwischen setImmediate(...) und setTimeout(..., 0)

5.1.4 Promise

- **pending** : Ausgangszustand
- **fulfilled** : erfolgreich abgeschlossen
- **rejected** : ohne Erfolg abgeschlossen
- Nur ein Zustandsübergang möglich
- Zustand in Promise-Objekt gekapselt

6 Vorlesung 07

ISO/OSI	Internet	Protokolle	typische Angaben
Application (Anwendung)	Application	HTTP FTP SMTP Telnet	URL: http://www.zhwin.ch Mailadresse: mustepet@zhwin.ch
Presentation (Darstellung)			
Session (Sitzung)			
Transport	Transport	TCP UDP	Portnummer 80 = HTTP 25 = SMTP
Network (Netzwerk)	Internet	IP	IP-Adresse 192.168.0.1
Data Link (Sicherung)	Physical / Access	Ethernet Wireless LAN Token Ring PPP/(Modem, ISDN, xDSL)	MAC-Adresse 00:0F:7F:23:45:67 Telefonnummer: 0878/123456
Physical (Bitübertragung)			

Abbildung 12: Internet Protokolle

- 1XX : Information (z.B. 101 Switching Protocols)
- 2XX : Erfolg (z.B. 200 Ok, 204 No Content)
- 3XX : Weiterleitung (z.B. 301 Moved Permanently)
- 4XX : Fehler in Anfrage (z.B. 403 Forbidden, 404 Not Found)
- 5XX : Server-Fehler (z.B. 501 Not Implemented)

6.1 EXPRESS.JS

- Minimales, flexibles Framework für Web-Apps
- Zahlreiche Utilities und Erweiterungen
- Grundlage: Node.js
- Grundlage für zahlreiche weitere Frameworks

7 Vorlesung 08

JAVASCRIPT Für den Einsatz im Browser entwickelt, hat sich durchgesetzt

Allgemeine Objekte	Browser-Objekte
Object	document
Array	window
Function	event
String	history
Date	location
Math	navigator
RegExp	...
...	...

Abbildung 13: Vorderfinierte Objekte

DOM (Document Object Model)

7.1 Website im Browser Speicher

- Browser parst HTML-Code
- Baut ein Modell der Dokumentstruktur auf
- Basierend auf dem Modell wird die Seite angezeigt
- Auf diese Datenstruktur haben Scripts Zugriff
- Anpassungen daran wirken sich live auf die Anzeige aus

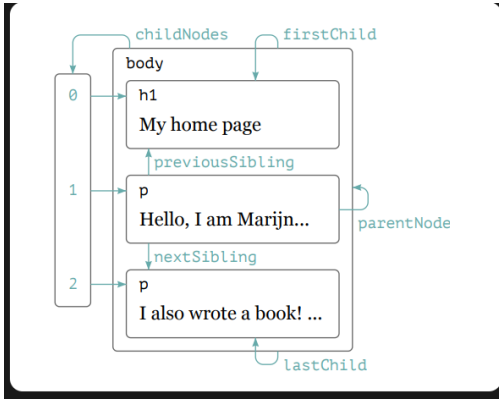

Abbildung 14: Baumstruktur

Attribut	Bedeutung
src	URL oder Pfad zur Bilddatei
alt	Alternativtext
width	Breite des Bilds
height	Höhe des Bilds
...	...

Abbildung 16: image

7.1.1 Baumstruktur

- Jeder Knoten hat ein nodeType -Attribut
- HTML-Elemente haben den nodeType 1

child nodes liefert aber kein echtes array (Nur Array ähnliches objekt)

NodeType	Konstante	Bedeutung
1	Node.ELEMENT_NODE	Elementknoten
3	Node.TEXT_NODE	Textknoten
8	Node.COMMENT_NODE	Kommentarknoten

Abbildung 15: Node Types

7.2 Elementobjekte

7.3 Attribute

- class (in html)
- data- = eigene attribute

8 Vorlesung 09

8.1 Event Handling

```
<p>Click this document to activate the handler.</p>
<script>
  window.addEventListener("click", () => {
    console.log("You knocked?")
  })
</script>
```

Abbildung 17: eventHandler addEventListener

- Events können auch an DOM-Elementen registriert werden
- Nur Ereignisse, die im Kontext dieses Elements auftreten, werden dann berücksichtigt
- Remove: removeEventListener
- Ereignisse werden für Knoten im DOM-Baum registriert
  - Reagieren auch, wenn Ereignis an untergeordnetem Knoten auftritt
  - Alle Handler nach oben bis zur Wurzel des Dokuments ausgeführt
  - Bis ein Handler stopPropagation() auf dem Event-Objekt aufruft
  - Element, bei welchem das Ereignis ausgelöst wurde: event.target
  - Element, bei welchem das Ereignis registriert wurde: event.currentTarget
- Aufruf von preventDefault() auf Event-Objekt verhindert Default-Verhalten
- Ereignisse keydown und keyup

  - Modifier-Tasten als Attribute des Event-Objekts
  - Achtung: keydown kann bei längerem Drücken mehrfach auslösen
- Mausklicks: mousedown , mouseup , click , dblclick
- Mausbewegung: mousemove
- Berührung (Touch-Display): touchstart , touchmove , touchend
- Ereignis-Typ: scroll Attribute des Event-Objekts: pageYOffset , pageXOffset
- Focus und Lade ereignisse (werden nicht propagiert)
  - Fokus erhalten/verlieren: focus , blur
  - Seite wurde geladen: load
    * Ausgelöst auf window und document.body
    * Elemente mit externen Ressourcen (img) unterstützen ebenfalls load -Events
    * Bevor Seite verlassen wird: beforeunload

8.2 jQuery

Aufruf	Bedeutung	Beispiel
$(*Funktion*)	DOM ready	$(function() { .... });
$("*CSS Selektor*") .aktion(arg1, ...) .aktion(...)	Wrapped Set - Knoten, die Sel. erfüllen - eingepackt in jQuery Obj.	$(".toggleButton").attr("title") $(".toggleButton").attr("title", "click here") $(".toggleButton").attr({title : "click here", ...}) $(".toggleButton").attr("title", function(){...}) .css(...) .text(...) .on("click", function(event) { ...})
$("*HTML-Code*")	Wrapped Set - neuer Knoten - eingepackt in jQuery Obj. - noch nicht im DOM	$(" ... ").addClass(...) .appendTo("*Selektor*") $(" ... ").length $(" ... ")[0]
$(DOM-Knoten)	Wrapped Set - dieser Knoten - eingepackt in jQuery Obj.	$(document.body) $(this)

Abbildung 18: jQuery Overview

8.3 Bilder und Grafiken

- Einfache Grafiken mit HTML und CSS möglich
- Zum Beispiel: Balkendiagramme
- Alternative für Vektorgrafiken: SVG
- Alternative für Pixelgrafiken: Canvas: Element canvas als Zeichenbereich im Dokument

8.4 Weitere Browser APIs

8.4.1 Web Storage

- Speichern von Daten clientseitig
- Einfache Variante: Cookies (s. spätere Lektion)
- Einfache Alternative: LocalStorage
- Mehr Features: IndexedDB (nicht Stoff von WBE)

8.4.2 Local Storage

- Bleibt nach Schliessen des Browsers erhalten
- In Developer Tools einsehbar und änderbar
- Alternative solange Browser/Tab geöffnet: sessionStorage

8.4.3 History

- Zugriff auf den Verlauf des aktuellen Fensters/Tabs
- length : Anzahl Einträge inkl. aktueller Seite
- back() : zurück zur letzten Seite
- Seit HTML5 mehr Kontrolle über den Verlauf:
- pushState() , replaceState() , popstate -Event

8.4.4 Geolocation

8.4.5 Web Workers

Laufen parallel zum Haupt Script: aufwändige Berechnungen blockieren nicht dien Event Loop

```
var options = { enableHighAccuracy: true, timeout: 5000, maximumAge: 0 }

function success(pos) {
  var crd = pos.coords
  console.log(`Latitude : ${crd.latitude}`)
  console.log(`Longitude: ${crd.longitude}`)
  console.log(`More or less ${crd.accuracy} meters.`)
}

function error(err) { ... }

navigator.geolocation.getCurrentPosition(success, error, options)
```

Abbildung 19: Geolocation

```
// squareworker.js
addEventListener("message", event => {
  postMessage(event.data * event.data)
})

// main script
let squareWorker = new Worker("code/squareworker.js")
squareWorker.addEventListener("message", event => {
  console.log("The worker responded:", event.data)
})
squareWorker.postMessage(10)
squareWorker.postMessage(24)
```

Abbildung 20: Web Workers

9 Vorlesung 10

9.1 Formular:

- Formularelement geändert: change
- Eingabe in Textfeld: input
- Tastendrücke in aktivem Textfeld: keydown , keypress , keyup
- Formular absenden: submit

9.1.1 GET Methode

- Formulardaten an URL angehängt (Browser-Adressleiste)
- Genannt: Query String (Teil der URL)
- Vor allem für passive Anfragen (zum Beispiel eine Suche)
- Funktionen: encodeURIComponent , decodeURIComponent

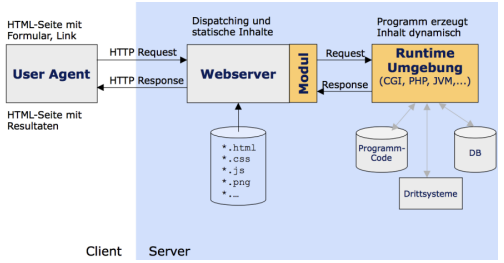

Abbildung 21: Serverseitige Programmlogik

- Die GET-Methode ist ungeeignet für Anmeldedaten und andere sensitiven Informationen

9.1.2 POST Methode

- Im Formular wird method=post"gesetzt
- Formulardaten im Body des HTTP-Request gesendet
- Um Daten anzulegen oder zu aktualisieren
- Für sensitive Daten (Login): POST und HTTPS

9.2 Cookies

- HTTP als zustandsloses Protokoll konzipiert
- Cookies: Speichern von Informationen auf dem Client
- Response: Set-Cookie -Header, Request: Cookie -Header
- Zugriff mit JavaScript möglich (ausser HttpOnly ist gesetzt)

9.3 Session

Cookies auf dem Client leicht manipulierbar Session: Client-spezifische Daten auf dem Server speichern Identifikation des Clients über Session-ID (Cookie o.a.) Gefahr: Session-ID gerät in falsche Hände (Session-Hijacking)

9.4 Ajax und XMLHttpRequest

Idee: Informationen asynchron im Hintergrund laden Technische Grundlage: XMLHttpRequest -Objekt

9.5 Fetch API

Interface for fetching resources Gibt Promise zurück Nach Server-Antwort erfüllt mit Response-Objekt

10 Vorlesung 11

10.1 Frameworks und Bibliotheken

Bibliothek: Kontrolle beim eigenen Programm, Funktionen u klassen der Bibliothek verwendet.
Framework: Rahmen für die Anwendung, Kontrolle beim Framework

10.2 Serversite Architekturmuster

10.2.1 Model-View-Controller (MVC):

- Models
  - repräsentieren anwendungsspezifisches Wissen und Daten
  - ähnlich Klassen: User, Photo, Todo, Note
  - können Observer über Zustandsänderungen informieren
- Views
  - bilden die Benutzerschnittstelle (z.B. HTML/CSS)
  - können Models überwachen, kommunizieren aber normalerweise
  - nicht direkt mit ihnen
- Controllers
  - verarbeiten Eingaben (z.B. Clicks) und aktualisieren Models

```jsx
/* JSX */
const element = (
  <div style="background:salmon">
    <h1>Hello World</h1>
    <h2 style="text-align:right">from SuiWeb</h2>
  </div>
)

/* SJDON */
const element =
  ["div", {style: "background:salmon"},
    ["h1", "Hello World"],
    ["h2", {style: "text-align:right"}, "from SuiWeb"] ]
```

Abbildung 22: Vergleich JSX SJDON

10.2.2 Single Page Apps (SPAs)

- Neuladen von Seiten vermeiden
- Inhalte dynamisch nachgeladen (Ajax, REST)
- Kommunikation mit Server im Hintergrund
- UI reagiert schneller (Usability)

10.3 DOM-Scripting und Abstraktionen

Basierend auf DOM-Funktionen, DOM-Funktionen in einer Funktion elt gekapselt

- Möglichkeit, eigene Elemente zu definieren
- Implementiert mit HTML, CSS und JavaScript
- Implementierung im Shadow DOM verstecken

10.3.1 Abstraktionen

React.js, vue.js

10.4 JSX und SJDON

10.4.1 JSX

- Von React-Komponenten verwendete Syntax

- JavaScript-Code mit XML-Teilen
- HTML-Tags in Kleinbuchstaben
- Eigene Komponenten mit grossen Anfangsbuchstaben
- Neue Komponente kann dann als Tag im JSX benutzt werden
- HTML-Elemente können die üblichen Attribute haben
- Wenige Ausnahmen, z.B.: class -Attribut heisst className in JSX

10.4.2 SJDON

SJDON – Simple JavaScript DOM Notation

10.5 Eigene Bibliothek: SuiWeb

- Datengesteuert
- Input: Daten der Applikation
- Output: DOM-Struktur für Browser
- nur one way bindung

11 Vorlesung 12

stupid fantasy framework language. stupid.
Interne Repräsentation und das DOM Komponenten und Properties Darstellung von Komponenten Defaults und weitere Beispiele

12 Vorlesung 13

12.1 Zustand von Komponenten

In React: Klassen

12.1.1 Hooks (State-Hooks)

- useState liefert Zustand und Update-Funktion
- Initialwert wird als Argument übergeben
- Zustandsänderung führt zum erneuten Rendern der Komponente
- Properties werden als Argument übergeben (props -Objekt)

12.2 Komponenten-Design

12.2.1 Container Komponente

- Daten-Verwaltung von Daten-Darstellung trennen
- Container-Komponente zuständig, Daten zu holen
- Daten per props an Render-Komponenten weitergegeben

12.2.2 Effect Hook

Lifecycle-Methoden In Funktionskomponenten: Effect Hooks

12.3 Optimierungsansätze

- Klarer und einfacher Datenfluss:
  - Daten nach unten weitergegeben (props)
  - Ereignisse nach oben weitergegeben und dort behandelt
- Properties werden nicht geändert, Zustand ist veränderbar
- Zustand wird von Komponente verwaltet
- Es ist von Vorteil, die meisten Komponenten zustandslos zu konzipieren

13 Vorlesung 14

- React DOM
  - Performs the actual
  - rendering on a web page
- React Component API
  - Data to be rendered
  - Lifecycle methods
  - Events: respond to user interactions
  - JSX: syntax used to describe UI structures