JavaScript-Schulung: Ablaufplan

Thematische Einleitung

Schulungsinhalte.pdf: Inhalte, Lernziele und Vorgehen der Schulung
https://www.wpvs.de/: Hinweis auf die Onlinefolien zum Nachschlagen

Folie 3: Geschichte und heutige Einsatzgebiete von JavaScript
- Als clientseitige Skriptsprache im Browser entworfen
- Dank node.js auch als leichtgewichtigtes Serverbackend z.B. für Cloudanwendungen
- Dank Cordova/Phonegap zur Entwicklung mobiler Anwendungen
- Dank Tessel oder Espruino zur Entwicklung eingebetteter IoT-Devices
- Dank Electron und anderer Technologien zur Entwicklung von Desktopanwendungen

JavaScript in HTML einbinden

1.1 JavaScript und HTML in einer Datei
- Anlegen einer minimalen HTML-Seite mit einem `<button>` und einem `<div>`
- Innerhalb des `<div>` soll angezeigt werden, wie oft der Button bereits geklickt wurde
- Zunächst `inline` mit `<script>` und `onclick=""`

1.2 Zweiter Button (Lösung)
- Aufgabe: Einbauen eines zweiten Buttons, der den Zähler auf Null zurücksetzt
- Benutzung der Browser-Konsole (`n` ausgeben, `n` hochsetzen, `onRestButtonClicked()` aufrufen)
- Debuggen der Funktion `onRestButtonClicked()`
- Hinweis, dass der Debugger nur startet, wenn die Entwicklungswerkzeuge offen sind

1.3 JavaScript-Code auslagern
- Verschieben des JavaScript-Codes in eine eigene Datei
- Registrieren des Ereignisbehandlers mit `addEventListener`
- `onLoad`-Ereignis des `window`-Objekts bei Programmstart abfangen

Grundlegende Spracheigenschaften von JavaScript

2.1 Mehrere JavaScript-Dateien (Lösung)
- Aufgabe: Je Button eine eigene JavaScript-Datei ausprogrammieren
- Hinweis, dass die Funktionen außerhalb `window.addEventListener(…)` liegen müssen
- `window`-Objekt als globaler Namensraum für globale Variablen und Funktionen
- Verwendung von `var` und `let` zur Deklaration lokaler Variablen
- JS.Scratch: `var` oder `let` vor einer Variable vergessen

2.2 if-Abfrage
- JavaScript-Code wieder in eine Datei namens `counter.js` verschieben
- Zusätzlichen Button zum Herunterzählen einbauen
- Zusätzliche Funktion zur Ausgabe des Zählers
- Zusätzliche Funktion zur Sicherstellung, dass der Zählen zwischen 0 und 10 liegt
- Zähler direkt bei Programmstart anzeigen

2.3 Tastaturereignisse mit switch-Anweisung
- Event-Listener für das `keyup`-Ereignis einbauen
- Zunächst `event`-Objekt auf der Konsole ausgeben, um die Tastaturcodes zu erfahren
- Danach `switch`-Anweisung für `"ArrowUp"`, `"ArrowDown"`, `"+"`, `"-"` und `" "`

2.4 for-Schleife (Lösung)

- Zweiten Button und zweites Ausgabefeld zur Berechnung der Fakultät (noch ohne Funktion)
- JS.Scratch: Berechnung der Fakultät rekursiv und per `for`-Schleife
- Aufgabe: Zweite Codedatei zur Berechnung der Fakultät ausprogrammieren
- Bonusaufgabe: Berechnung der Fakultät mit Enter auslösen

Pause

Definition komplexer Datenstrukturen

3.1 Vorlage für Aufgabenverwaltung

- Kopieren der Vorlage in einen neuen Ordner als Grundlage für eine neue App
- Struktur des Quellcodes kurz erklären (Eingabefelder, Tabelle, …)
- Globale Variable zum „Speichern“ der Aufgaben, Initialisierung des Schaubild
- JS.Scratch: Definition einer Aufgabenliste mit Listen und Objekten

3.2 Listen und Objekte

- Achtung: Arrow-Syntax für Funktionen verwenden, damit später `this` richtig zugewiesen wird!
- Vordefinierte Platzhalterwerte des Schaubilds entfernen
- Neue Funktion `appendTaskToTable()` zum Anzeigen eines weiteren Eintrags ausprogrammieren
- Neue Funktion `updateChart()` zum Neuaufbau des Schaubilds ausprogrammieren
- `click`-Event das Buttons abfangen, um einen neuen Datensatz zu speichern und anzuzeigen
- Aufgabe: Neuen Eintrag „speichern“ und anzeigen, sowie Eingabefelder zurücksetzen

3.3 Klasse Task

- JS.Scratch: Objekte mit Konstrukturfunktionen und Klassen erzeugen (ohne Vererbung)
- Anlegen einer neuen Codedatei mit einer Klasse `Task`
- Ermittlung des Prio-Textes anhand eines Getter-Properties in der Klasse `Task`
- Aufgabe: Anpassen aller Codestellen in der Datei `main.js`

3.4 Klasse Main

- Kapselung aller Inhalte der Datei `main.js` in einer Klasse namens `Main`
- Attribute hierzu in einen Konstruktor verschieben
- Methoden hierzu ohne das `function`-Schlüsselwort definieren
- Unterschied zwischen `function()` und Arrow-Functions bzgl. `this` erklären
- Aufgabe: `this.` vor alle releventanen Codestellen schreiben, mit der Browserkonsole gut testen!
- Bonusaufgabe: Eingabeprüfung einbauen, damit keine leeren Datensätze angelegt werden

Pause

Fortgeschrittene Themen

4.1 Kapselung mit AMD-Modulen

- Probleme, die durch das in den Browsern fehlende Modulsystem von JavaScript bestehen
- Funktionsweise von ES6-Modulen mit `import`- und `export`-Anweisungen kurz erklären
- Alternativer Lösungsansatz im Browser benötigt, da `import`/`export` dort nicht funktionieren
  `define([abhängigkeiten], function(abhängigkeiten) { return … }`
- Asynchrones Nachladen von Quellcode, daher der Name „Asynchronous Module Definition“)
- Einbinden von `require.js` per CDN-Link
- Umstellen der Dateien `task.js` und `main.js` zu AMD-Modulen