Interaktive HTML Seiten mit JavaScript

Roter Faden für die Projekttage 2015 am Gymnasium am Ölberg

Einführung

Im Informatikunterricht der Jahrgangsstufe 9 habt Ihr bereits HTML, CSS und PHP kennengelernt. Oft ging es dabei um HTML Formulare, deren Eingaben zu einem PHP Script auf einem Web Server geschickt wurden. Der PHP Programmcode wertete dann die Formulardaten aus und schickte eine neue HTML Seite zur Anzeige zum Browser zurück.

In diesem Projekt sollen einige der Möglichkeiten zur Interaktion einer HTML Seite mit dem Anwender vorgestellt werden, die ohne einen Zugriff auf einen Web Server auskommen. Wir werden hier zum einen näher auf die Fähigkeiten von CSS eingehen, die schon von sich aus eine optische Aufbereitung einer HTML Seite erlauben. Mit der Browser Programmiersprache *JavaScript* können viele weitere Effekte realisiert werden. JavaScript ist eine vollwertige Programmiersprache, die wir bei der Entwicklung kleiner Spiele etwas kennen näher lernen werden. Da JavaScript aber recht umfangreich ist, wird es bei einem ersten Einblick in die grundlegenden Konzepte bleiben müssen.

Im Zuge der vielen (nicht nur kleinen) **Übungen** werden wir auch sehen, dass es nicht immer die eine richtige Lösung für eine Anforderung gibt. Auch die Beispiellösungen zu den Übungen verstehen sich daher nur als ein möglicher Weg – der nicht zwingend der Beste sein muss.

HTML und CSS

Im ersten Abschnitt setzen wir auf dem auf, was Ihr im Informatik Unterricht bereits gelernt habt. Dazu verwenden wir *PSPad* und einfache HTML Seiten, die mit CSS verschönert werden. Wir werden dann sehen, dass man einfache optische Effekte auf einer HTML Seite auch ohne programmieren zu müssen erreichen kann. Der Anwender erhält dabei nicht nur Hilfestellungen und Hinweise zur Bedienung: gerade leichte optische Veränderungen können die Aufmerksamkeit auch bewusst auf einen bestimmten Bereich der Seite lenken.

In der ersten **Übung** wird eine einfache Liste mit den Tags *UL* und *LI* erstellt. Ansonsten gibt es nur eine HTML Seite und eine CSS Datei, die über *LINK* im *HEAD* referenziert wird. In der CSS Datei wird das Listensymbol erst einmal mit der CSS Eigenschaft *list-style* bewusst auf einen anderen Wert als in der Vorgabe festgelegt. Wir lernen dann die CSS Selektoren *:hover* und *:active* kennen und verwenden diese, um den Listeneintrag unter der Maus respektive den angeklickten Eintrag zu verändern. Neben dem *list-style* werden die CSS Eigenschaften *font-weight, font-style* und *color* vorgestellt

Die zweite **Übung** beginnt mit einem Rückblick auf CSS Klassen und die Möglichkeit, diese zu kombinieren. Das HTML Element *DIV* wird etwas eingehender betrachtet, dazu die CSS Eigenschaften *float*, *margin*, *text-align*, *width*, *height* und *background-color*. Zur Änderung der Optik werden wieder *:hover* und *:active* verwendet. Wir lernen mit den CSS Eigenschaften *display* und *visibility* verschiedene Möglichkeiten kennen um Teile der HTML Seite auszublenden. Schließlich werden die CSS Eigenschaften *background*, *background-position* und *background-size* verwendet um ein Hintergrundbild bei Bedarf einzublenden.

Als Vorbereitung für die letzte **Übung** wird noch einmal kurz das Konzept vorgestellt, wie HTML Text auf einer Seite formatiert wird und welche Rolle insbesondere das *DIV* Tag dabei spielt. In der Übung wird dann gezeigt, wie man diese Regeln gezielt vor allem mit den CSS Eigenschaften *display*, *float*, *position*, *top* und *left* umgehen und damit hier zu einem Wort auf der HTML Seite eine Detailinformation einblenden kann – so eine Art *Tooltip*, aber schon etwas anders, wie dabei erklärt wird. Insbesondere wird auf die Bedeutung der CSS Eigenschaft *position* näher eingegangen. Bei der Übung werden auch die CSS Eigenschaften *padding*, *border und background-color* eingesetzt, um unseren *Tooltip* optisch etwas aufzubereiten.

DOM, Events und JavaScript

Der zweite Abschnitt verwendet erst einmal nur einfachste Grundkonzepte der Programmiersprache JavaScript und konzentriert sich hauptsächlich auf das Zusammenspiel von HTML Elementen und Programmcode. Eine wichtige Rolle spielen dabei Events und das HTML DOM (*Document Object Model*). Das grundlegende Prinzip ist dabei recht einfach und wird am Anfang vorgestellt und erläutert.

Die erste **Übung** verwendet nur eine HTML Datei und ihre zugehörige CSS Datei. Der JavaScript Programmcode ist direkt in der HTML Seite als Text der *onclick* Attribute verschiedener *INPUT* Elemente (*TYPE BUTTON*) hinterlegt. Neben der Funktion *alert* werden vor allem das Konzept *this* erläutert und wichtige DOM Eigenschaften wie *className*, *parentElement* oder *nextSiblingElement* vorgestellt.

Zum besseren Einblick wird kurz auf die Entwickleroptionen eingegangen, die alle modernen Browser über F12 anbieten. Hier erhält man nicht nur Einblick in das HTML DOM und die Eigenschaften der HTML Elemente, sondern kann auch den JavaScript Programmcode näher untersuchen – diese Möglichkeit wird aber vor allem im nächsten Abschnitt relevant.

Die zweite **Übung** ist mehr oder weniger identisch zur ersten, allerdings nun mit dem Programmcode in einer separaten Datei. Am Beispiel des HTML DOM Events *onload* und der *setTimeout* Funktion wird auf den Lebenszyklus von HTML Elementen eingegangen. Vor allem werden wir sehen, dass JavaScript zu einem Zeitpunkt niemals mehr als eine Sache macht – es gibt keine Nebenläufigkeit. Neben dem JavaScript *document* Objekt wird die Funktion *getElementById* vorgestellt und auf die Bedeutung des Attributes *id* der HTML Elemente eingegangen.

Nun kommt in der nächsten **Übung** auch schon unser erstes kleines Spiel, das mit JavaScript im Browser ausgeführt wird. Wir lernen das *onmouseover* Event kennen, in etwa das JavaScript Äquivalent zum *:hover* im CSS. In JavaScript lernen wir die DOM Eigenschaften *offsetParent*, *offsetWidth*, *offsetHeight* und vor allem *style* kennen, dazu gibt es auch einen ersten Kontakt mit den JavaScript Funktionen aus der *Math* Bibliothek. Damit das Spiel nicht zu schwer für den Spieler wird, erlauben wir uns einen kleinen Kniff im CSS.

Als Vertiefung der Testmöglichkeiten mit F12 schauen wir an diesem Beispiel noch einmal etwas genauer auf die Bedeutung von *this* beim Aufruf von JavaScript Funktionen.

In der vierten **Übung** sehen wir, welche Möglichkeiten zum Verschieben von HTML Elementen bereits mit den Events *ondragstart*, *ondragover* und *ondrop* angeboten werden – auch *onmousedown* kommt dabei zum Einsatz. Wir schauen uns die Bedeutung des JavaScript Objektes *event* an und lernen, wie wir es insbesondere im Zusammenhang mit Mausbewegungen nutzen können. Dazu gibt es noch einen kleinen Einblick in die Mechanismen der Eventverarbeitung im HTML DOM.

Mit der nächsten **Übung** verändern wir dann zum ersten Mal die HTML Seite aus JavaScript heraus und fügen völlig neue HTML Elemente durch unseren Programmcode ein. Eine wichtige Rolle werden dabei die DOM Eigenschaften *parentElement*, *innerHTML* und *outerHTML* spielen.

Die abschließende **Übung** schließlich orientiert sich an einem typischen Beispiel aus der Praxis. Bereits im Browser sollen erste Prüfungen der Benutzereingaben vorgenommen werden: hier etwa müssen zwei Eingaben gültige Zahlen sein, von denen eine größer als die andere ist. Wir erfahren etwas über die HTML DOM Events *onkeyup* und *onblur* und lernen eine Möglichkeit kennen, wie im Zusammenspiel von HTML, CSS und JavaScript Fehlermeldungen angezeigt werden können. Als direktes Ergebnis unserer Prüfungen wird mit der HTML Eigenschaft *disabled* eine Schaltfläche bei Bedarf gezielt deaktiviert. Wir lernen dabei neben der *parseInt* Funktion auch die DOM Funktionen *setAttribute* und *removeAttribute* kennen.

Das Spiel des Lebens

Am Anfang dieses Abschnitts steht die Vorstellung der Spielidee – eigentlich mehr eine Simulation als ein Spiel. Wir lernen die einfachen Regeln kennen, mit denen sich aus einer Spielgeneration die nächste berechnet – und werden diese Regeln später sogar noch weiter vereinfacht in JavaScript implementieren.

In der ersten **Übung** wird ein dynamisches Spielfeld aufgebaut, dessen Größe durch HTML Attribute festgelegt wird. Dabei erfahren wir etwas über die Erweiterbarkeit der HTML Tags und deren Attribute – vor allem über die *data-* Konvention für die Namen von Attributen. Auch in CSS gibt es eine Neuigkeit: unser tabellarisches Spielfeld wird nicht über eine HTML TABLE aufgebaut, sondern über DIV Elemente und die CSS Eigenschaft *display* mit den Werten *table*, *table-row* und *table-cell*. In JavaScript lernen wir die *for* Schleife und die *querySelector* respektive *querySelectorAll* Funktionen kennen. An einem einfachen Beispiel können wir auch sehen, welchen Einfluss kleine Änderungen im Programmcode auf die Laufzeit beim Aufbau des Spielfelds haben – bei gleichem Endergebnis.

Die zweite **Übung** erweitert das Spiel um die Möglichkeit den Zustand einzelner Zellen zu verändern. Dies geschieht recht einfach über *onclick* und eine neue CSS Klasse.

Mit der dritten **Übung** wird es dann möglich, das Spielfeld zufällig zu belegen oder zu als Ganzes zu löschen.

Mit der vierten **Übung** kommt nun die erste große Herausforderung mit JavaScript. Wir verwenden JavaScript Felder von selbst erstellen JavaScript Objekten um aus einer Spielgeneration die nächste zu berechnen und dann diese auch auf der HTML Seite darzustellen. Gerade die Verwendung von eigenen JavaScript Objekten ist ein zentraler Aspekt in der Programmierung mit JavaScript und wird daher etwas tiefer erläutert als hier im Spiel tatsächlich benötigt.

Mit der schon früher gewonnen Kenntnis über die JavaScript Abläufe und vor allem die *setTimeout* Funktion ist es dann ein leichtes, in der fünften **Übung** das Spiel so zu erweitern, dass es automatisch neue Spielgenerationen berechnet.

Zum Abschluss wird in der letzten **Übung** dann auch noch angezeigt, wie viele Generationen der Browser denn pro Sekunde wirklich berechnen kann – wenn wir die großen Browser vergleichen werden wir sehen, dass Googles Chrome bis zu 5 mal schneller sein kann als zum Beispiel der Internet Explorer von Microsoft. Die gewünschte Rate kann dabei durch den Anwender vorgegeben wer-

den – aber oft bleibt der Wunsch Vater des Gedankens. Wir werden dabei noch etwas mehr aus der *Math* Bibliothek sowie die HTML DOM Eigenschaft *textContent* kennen lernen.

Schiffe versenken

Der letzte (optionale) Abschnitt stellt eine andere Alternative zum Aufbau eines Spielfelds vor und verwendet ansonsten die Grundlagen, die in den vorherigen Abschnitten bereits erarbeitet wurden. Als Grundlage dient das wohlbekannte Spiel *Schiffe versenken* – natürlich in der Umsetzung stark vereinfacht.

In der ersten **Übung** wird wieder ein Spielfeld aufgebaut. Da dessen Größe aber mit 10x10 bekannt ist, wird dieses hier statisch in HTML auf Basis von DIV Tabellen aufgebaut. Wir lernen dabei die HTML Entität * * sowie die CSS Eigenschaften *text-align* und *vertical-align* kennen. Nachdem wir uns Gedanken über die Anzeige verschiedener Zustände der Spielzellen gemacht haben (bereits beschossen oder nicht, Treffer oder nicht) nutzen wird CSS Klassen zum Aufbau einer Legende. Im JavaScript Programmcode prüfen wir die Korrektheit des Spielfelds – statt es dynamisch aufzubauen. Wir lernen dabei die Auswahl von HTML Elementen mit *:not* kennen.

Diesmal ist bereits die zweite **Übung** eine echte Herausforderung und es gibt eine Vielzahl von Möglichkeiten der Umsetzung. Ziel der Übung ist es, dass der erste Spieler seine Schiffe auf dem Spielfeld versteckt – wie gewohnt ein 5er, ein 4er, zwei 3er und ein 2er. Die Beispiellösung der Übung verwendet das Ziehen und Fallenlassen mit der Maus um diese Aufgabe zu meistern. Wichtig ist es dabei zu beachten, dass jedes Schiff in zwei Orientierungen (längs und quer) positioniert werden kann. Je nach Orientierung natürlich nicht an jeder Stelle auf dem Spielfeld. Im Ansatz mit Ziehen und Fallenlassen muss während des Ziehens bereits geprüft werden, ob ein Schiff in einer bestimmten Orientierung fallengelassen werden kann. Erst das Fallenlassen markiert dann die entsprechenden Spielzellen. Zusätzlich soll es dem ersten Spieler möglich sein, eine einmal getroffene Entscheidung zu revidieren und ein Schiff an eine andere Stelle zu versetzen – möglicherweise sogar mit einer anderen Orientierung. Solange das Spielfeld vom ersten Spieler aufgebaut wird muss dieser natürlich seine Auswahl sehen können. Ist er mit seinen Verstecken zufrieden, schaltet er auf den Suchmodus für den zweiten Spieler um.

Die Suche durch den zweiten Spieler wird dann die dritte **Übung** sein. Beim Anklicken einer Spielzelle muss auf einen möglichen Treffer geprüft werden. Wenn es einen Treffer gibt, soll weiterhin untersucht werden, ob ein Schiff als Ganzes versenkt wurde. Mit dem Versenken des letzten Schiffs ist das Spiel beendet und der zweite Spieler soll nun die Anzahl der Versuche angezeigt bekommen. Wir werden hier mit JavaScript Objekten arbeiten, die an die einzelnen HTML Elemente der Spielzellen gebunden werden. Einen Treffer zu erkennen wird dann sehr einfach – versenkte Schiffe sind schon etwas kniffeliger.

Damit das Spiel nicht zu schwer und frustrierend wird, bauen wir in der vierten **Übung** die Option ein, sich bei jedem Versuch anzeigen zu lassen, wie viele Treffer in den (3 bis 8) umliegenden Zellen existieren. Als praktisch erweisen sich dabei die Funktionen *min* und *max* aus der *Math* Bibliothek.

Die letzte **Übung** erlaubt einen Neustart des Spiels, ohne einen Browser Refresh durchführen zu müssen. Hier können wir lernen, wie man die im Spielverlauf durchgeführten Veränderungen an den HTML Elementen ordnungsgemäß rückgängig macht – in den meisten Fällen geht es um die *className* Eigenschaft sowie eigene Erweiterungen in Form von angehängten JavaScript Objekten.

Programmiertechniken am Beispiel *Schiffe versenken*

Bei der Programmierung von Software für Benutzeroberflächen verwendet man häufig Techniken wie *Model-View-Controller* (*MVC*). In der **Übung** zu diesem optionalen Abschnitt für Fortgeschrittene nehmen wir uns noch einmal das Spiel *Schiffe versenken* vor und lernen das Grundkonzept von MVC kennen – zumindest in einer der häufiger in der Praxis eingesetzten Ausprägungen, denn auch hier gibt es nicht die eine Lösung für alle Anforderungen.

Wir werden dabei schon ein ganzes Stück tiefer in die Programmiersprache JavaScript eintauchen. Darüber hinaus lernen wir JavaScript Klassen kennen und definieren eigene Funktionen für Objekte und Klassen. Auch *this* wird wieder eine wichtige Rolle spielen. Konzepte wie die *prototype* Eigenschaft von Funktionen oder die Sichtbarkeit von Variablen werden vorgestellt und erklärt. Gerade die Sichtbarkeit von Variablen ist einer der wesentlichen Aspekte, in denen sich JavaScript grundlegend von PHP unterscheidet.

Im richtigen Leben

Wir haben uns in diesem Projekt direkt mit der Programmierung von JavaScript in HTML Seiten auseinandergesetzt. In der Praxis verwendet man im Allgemeinen sogenannte Frameworks, die oft spezialisiert für besondere Anforderungen fertige Bibliotheken anbieten – zum Beispiel zum Arbeiten mit dem HTML DOM und insbesondere CSS Klassen.

Zusätzlich kann über JavaScript auch direkt mit einem Web Server und dort zum Beispiel einem PHP Script kommuniziert werden. Daten oder HTML Fragmente werden aus der Antwort des Scripts dann in die HTML Seite eingemischt. Gerade dieses Zusammenspiel zwischen Client Programmierung im Browser und Unterstützung durch so genannte Web Dienste macht die Stärke moderner Web Anwendungen aus.

In diesem Zusammenhang werden wir kurz auf die Abkürzungen *SPA* (*Single-Page-Application*) und *AJAX* (*Asynchronous-JavaScript-And-XML*) und deren Bedeutung eingehen.