Interaktive HTML Seiten mit JavaScript

Roter Faden für die Projekttage 2015 am Gymnasium am Ölberg

# Einführung

* Eigentlich zwei Aspekte: Interaktivität & Programmieren mit JavaScript
* Hier: JavaScript pur 🡺 Grundlagen, Verständnis entwickeln
* Beispiellösungen: es gibt KEINE einzig wahre / richtige Lösung!

# 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 erreichen kann ohne programmieren zu müssen. Der Anwender erhält dadurch nicht nur Hilfestellungen und Hinweise zu 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, wobei hier aber mit den CSS Eigenschaften *display* und *visibility* verschiedene Möglichkeiten eingesetzt werden um Teile der HTML Seite auszublenden. Schließlich werden die CSS Eigenschaften *background*, *background-position* und *background-size* verwendet um ein Hintergrundbild zeitweise 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 der HTML DOM Events *onload* und *setTimeout* 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 Mehrläufigkeit. 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 werden – 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

* Spielidee vorstellen
* Oberflächengestaltung entwickeln
* (Übung) Spielfeld statisch aufbauen
* Algorithmus zur Verwaltung der Schiffe entwickeln
* (Übung) Schiffe verstecken
* Algorithmus zum Suchen entwickeln
* (Übung) Algorithmus implementieren
* (Übung) Cheat einbauen
* (Übung) Erneutes Verstecken ohne Browser Refresh

# Im richtigen Leben

* JavaScript Frameworks
* SPA & AJAX für Daten und HTML