JavaScript und Softwaretechniken

Model-View-Controller (MVC) in einer HTML / CSS / JavaScript Anwendung

# Hintergrund

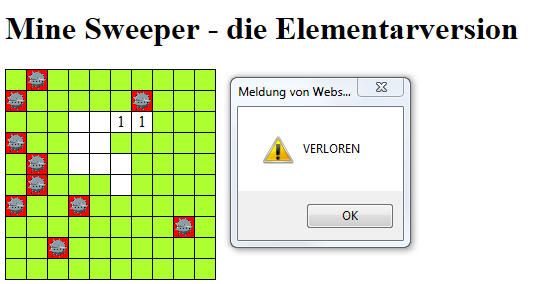
# Konventionelles Script

Im ersten Schritt werden wir mit wenig HTML und CSS sowie elementarem JavaScript Programmcode eine einfache, aber vollfunktionsfähige Version des Spiels *MineSweeper* erstellen. Im JavaScript verwenden wir dabei lediglich Variablen, Funktionen, Schleifen, einfache Objekte, Felder und Zugriffe auf das HTML DOM.

Die erste Version des Spiels wird bereits folgende Funktionen bieten:

* Die Parameter des Spielfelds wie die Größe und die Anzahl der versteckten Minen wird über HTML (*data*-) Attribute festgelegt und nicht im Programmcode fixiert.
* Beim Aufbau des Spielfelds werden die Minen zufällig versteckt. Als Hilfe für den Spieler wird automatisch ein Spielfeld aufgedeckt, unter dem sich keine Mine versteckt und in dessen unmittelbarer Nachbarschaft sich auch keine Felder mit versteckten Minen befinden.
* Beim Aufdecken eines Feldes wird die Anzahl der versteckten Minen in der unmittelbaren (8 Felder) Umgebung angezeigt.
* Wird ein Feld mit Mine aufgedeckt, ist das Spiel verloren. Wird das letzte Feld ohne versteckte Mine aufgedeckt, ist das Spiel gewonnen.
* Als Hilfe vor versehentlichem Aufdecken kann der Spieler mit der rechten Maustaste ein Feld sperren, wenn er den Verdacht hat, dass sich darunter eine Mine befindet. Diese Sperre kann jederzeit wieder aufgehoben werden.
* Die Anzeige unterscheidet nicht aufgedeckte von aufgedeckten Feldern. In aufgedeckten Feldern steht immer die Anzahl der Felder mit versteckten Minen in der unmittelbaren Umgebung.
* Ist das Spiel zu Ende, so werden auf jeden Fall die versteckten Minen angezeigt.

Zur Umsetzung werden hier die Techniken und Kenntnisse verwendet, die wir auch im Projekt *Interaktive Web Seiten mit JavaScript* im Rahmen der Projekttage 2015 eingesetzt haben. Diese werden aber im Vorfeld noch einmal erarbeitet und werden nicht vorausgesetzt.

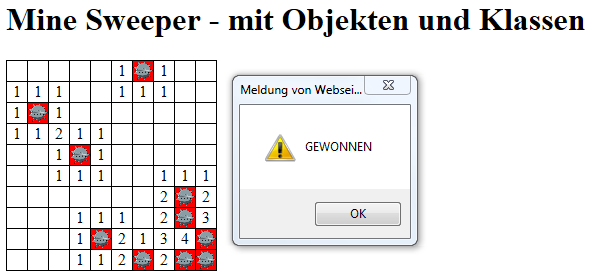


# Klassen, Objekte, Eigenschaften, Methoden

Wir werden hier nun die vorhandene Spiellogik ohne Erweiterungen mit fortgeschrittenen Funktionalitäten von JavaScript umsetzten.

Dabei lernen wir zuerst einmal das Konzept der Klassen und deren Instanzen kennen. Dazu definieren wir erst einmal für die vorhandenen so genannten anonymen Objekte (*{}*) Klassen und erzeugen dazu konkrete Instanzen (*new*). Die Eigenschaften können wir dabei unverändert übernehmen, allerdings ist es nun auch möglich, Funktionen für unsere eigenen Objekte zu definieren und zu verwenden. So entstehen im Programmcode zwei JavaScript Klassen für das Spielfeld und für jede einzelne Zelle.

Sowohl für Eigenschaften als auch für die Funktionen einer Klasse schauen wir uns die verschiedenen Alternativen zur Deklaration an: individuell pro Objekt, gemeinsam für alle Objekte der Klasse oder gänzlich ohne Objektbezug. In diesem Zusammenhang müssen wir uns mit dem JavaScript *prototype* Konzept auseinandersetzen und lernen Neues zum bekannten *this*.



# Model-View-Controller als Idee

Auch im nächsten Schritt entwickelt sich das Spiel selbst nicht weiter. Im vorhandenen Programmcode wird allerdings ein zentrales Konzept moderner Benutzeroberflächen eingesetzt: die Trennung der Daten und Algorithmen des Programms von den Anzeige- und Bedienelementen der konkreten Benutzeroberfläche.

Der Entwickler des Programmcodes kann sich dabei vollständig um die eigentlichen Aufgaben und Datenstrukturen des Programms kümmern. Zur Gestaltung der Oberfläche ist kein tiefer Einblick in die eigentliche Programmierung notwendig. Im Idealfall können beide Bereiche erst einmal unabhängig von entsprechenden Experten entwickelt und getestet werden.

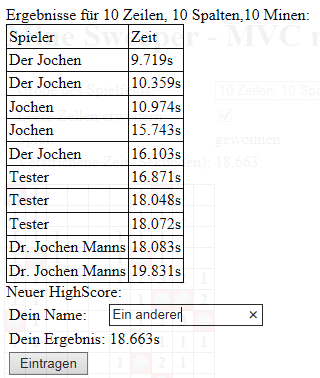
Was dann allerdings noch fehlt, ist die Verbindung zwischen den beiden vielleicht separat entwickelten Bereichen. Hier handelt es sich tatsächlich auch wieder um Programmcode, der den Spielstand in der Benutzeroberfläche sichtbar macht und Aktionen aus der Benutzeroberfläche an das eigentliche Spiel weiterleitet. Nur diese Verbinder kennen die Schnittstellen zu beiden Seiten hin, müssen sich aber weder mit den Details des Programmcodes noch der Oberflächengestaltung auseinandersetzen.

Im ersten Schritt entfernen wird alle Rückgriffe in die Oberfläche aus unseren JavaScript Klassen für Spielfeld und Zellen. Die Klassen werden damit zu den Datenmodellen (*Model*). Auch binden wird nun keine eigenen Objekte mehr an das HTML DOM, wodurch die Kombination aus HTML und CSS dann zur reinen Anzeige (*View*) dient.

Im zweiten Schritt werden wir den im ersten Schritt entfernten verbindenden Programmcode als eigenständige JavaScript Funktionen umsetzen. Nur in diesem Funktionen (*Controller*) heraus werden sowohl das Datenmodell als auch die Anzeige angesprochen, wobei der Abgleich über JavaScript Benachrichtigungen erfolgt.

Wir haben nun eine sehr einfache Implementierung einer *Model-View-Controller* (*MVC*) Architektur in unserem Spiel, auf deren Basis wird im Folgenden das Spiel erweitern werden. Tatsächlich gibt es viele Arten, MVC oder Varianten davon umzusetzen – uns reicht es hier erst einmal, die Idee kennengelernt und in einer Fassung umgesetzt zu haben.

# Model-View-Controller als Basis

* Weitere Funktionalitäten
  + Größe des Spielfelds frei wählbar
  + Neustart
  + Automatisches Auffüllen von 0-Feldern
  + Spielstandanzeige
  + Bestenliste via localStorage
* Weiterhin strenge Trennung von Anzeige und Modell
  + Mehrere (Anzeige, Verbinder) Paare für ein Modell

# User Experience

* Bestenliste optisch aufbereitet
* …