

# JavaScript und Softwaretechniken

---

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

### Hintergrund

In dem Projekt *Interaktive Web Seiten mit JavaScript* der Projekttage 2015 am Gymnasium am Ölberg in Oberpleis sollte hauptsächlich ein Verständnis für das Zusammenspiel von HTML Elementen, CSS Stildefinitionen und JavaScript Programmfragmenten entwickelt werden. Damit ist es möglich, eine HTML Seite auch ohne Rückgriff auf einen Web Server auf Eingaben des Anwenders reagieren zu lassen. Selbst einfache Spiele im Browser können entwickelt werden.

JavaScript ist eine recht mächtige Programmiersprache, deren tatsächlichen Fähigkeiten nur angerissen werden konnten. In diesem Projekt soll das Konzept der Klassen vorgestellt werden. Zusätzlich wird mit der *Model-View-Controller (MVC)* eine wichtige Architektur zur Entwicklung moderner Benutzeroberflächen in einer einfachen Variante vorgestellt und umgesetzt. Auch gibt es einige weitere nützliche Informationen zur Gestaltung von HTML Seiten mit CSS.

### Konventionelles Script (4-6 Doppelstunden)

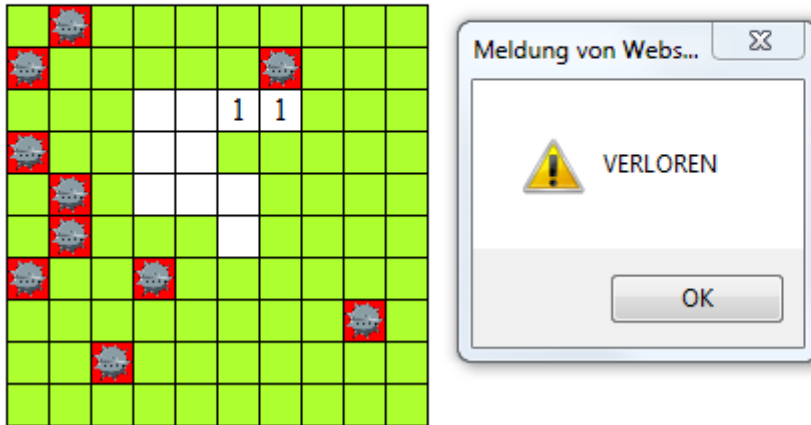
Im ersten Schritt werden wir mit wenig HTML und CSS sowie elementarem JavaScript Programmcode eine einfache, aber vollfunktionsfähige Version des Spiels *MineSweeper* erstellen. Im JavaScript verwenden wir dabei lediglich Variablen, Funktionen, Schleifen, einfache Objekte, Felder und Zugriffe auf das 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 Projekttag 2015 eingesetzt haben. Diese werden

## Mine Sweeper - die Elementarversion



aber im Vorfeld noch einmal erarbeitet und werden nicht vorausgesetzt.











### Klassen, Objekte, Eigenschaften, Methoden (2-3 Doppelstunden)

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

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*.

# Mine Sweeper - mit Objekten und Klassen

					1		1		
1	1	1			1	1	1		
1		1							
1	1	2	1	1					
		1		1					
		1	1	1			1	1	1
							2		2
			1	1	1		2		3
			1		2	1	3	4	
			1	1	2		2		



## 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 Verbinden 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 diesen 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 (2-3 Doppelstunden)

Wir wollen nun unser Spiel etwas erweitern, ohne dabei die strenge Trennung zwischen der Implementierung des Spiels und der Anzeige aufzugeben. Insbesondere ist es nach der Definition der Schnittstellen möglich, dass JavaScript Entwickler und Oberflächengestalter erst einmal relativ unabhängig voneinander ihre Arbeit beginnen können. Erst der JavaScript Code im Verbinder kombiniert beide Seiten. Wir werden dabei feststellen, dass es für ein Modell durch Sinn machen kann, dieses mit verschiedenen Anzeigeelementen zu verbinden.

Folgende neue Funktionalitäten soll unser Spiel nun erhalten:

- Der Spieler kann verschiedene Schwierigkeitsgrade wählen, die sich in der Größe des Spielfelds und der Anzahl der versteckten Minen niederschlagen.
- Es ist jederzeit möglich, das Spiel neu aufzubauen und von vorne zu beginnen.
- Auf Wunsch kann ein automatisches Aufdecken von Nachbarmfeldern eines Feldes ohne Mine in der unmittelbaren Umgebung durchgeführt werden.
- Der aktuelle Spielstand und die bisher verstrichene Zeit werden angezeigt.
- Es gibt eine Bestenliste mit den zehn besten (schnellsten) Ergebnissen pro Schwierigkeitsgrad. Der Spieler kann sich dann dort mit seinem Namen eintragen. Die Bestenliste erscheint nur, wenn das Spiel in einer Zeit gewonnen wurde, die auch tatsächlich zu einem Eintrag führen würde.

Für die Bestenliste lernen wir den *localStorage* kennen. Hier können Informationen auch ohne Unterstützung eines Web Servers permanent abgelegt werden. Der Entwicklermodus (F12) vom FireFox erlaubt es sogar, Einblick in die aktuellen Daten zu nehmen.

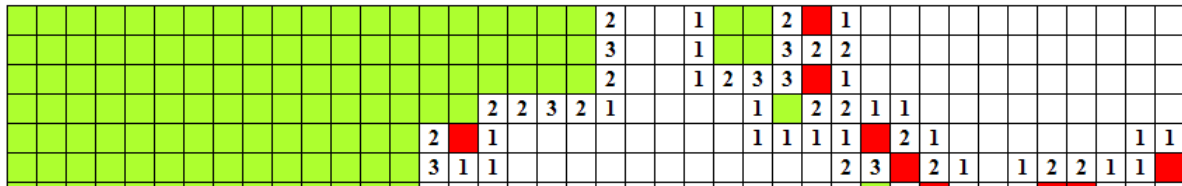
# Mine Sweeper - MVC mit erweiterten Funktionen

Größe des Spielfelds: 20 Zeilen, 40 Spalten, 100 Minen

Leere Zellen erweitern: ☒

Spiel: läuft

Verstrichene Zeit (Sekunden): 29



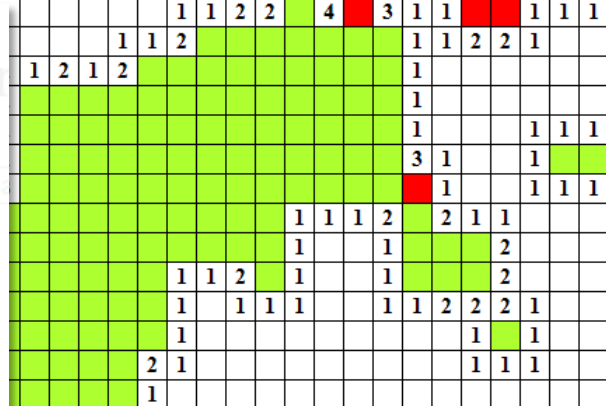
Ergebnisse für 10 Zeilen, 10 Spalten, 10 Minen:

Spieler	Zeit
Der Jochen	9.719s
Der Jochen	10.359s
Jochen	10.974s
Jochen	15.743s
Der Jochen	16.103s
Tester	16.871s
Tester	18.048s
Tester	18.072s
Dr. Jochen Manns	18.083s
Dr. Jochen Manns	19.831s

Neuer HighScore:

Dein Name:

Dein Ergebnis: 18.663s



## User Experience (2-3 Doppelstunden)

Im letzten Schwerpunkt wollen wir die Optik des Spiels etwas aufbereiten. Dabei soll versucht werden, den Programmcode des Modells weitestgehend unverändert zu lassen – lediglich *View* und *Controller* wären als anzupassen. Wir werden dabei aber feststellen müssen, dass dies nicht immer ganz einfach ist und oft mindestens neue Benachrichtigungen aus dem Modell erforderlich macht.

Dieser Abschnitt ist vom Ziel offen und der individuellen Phantasie überlassen. Man könnte zum Beispiel die Einstellungen etwas hübscher machen.

Größe des Spielfelds:

Leere Zellen erweitern:

Auch die Bestenliste hat jede Menge Luft nach oben – das folgende Beispiel kann man auch noch nicht wirklich schön nennen:

# Mine Sweeper - Oberflächengestaltung

## Die besten Ergebnisse für 9 Zeilen, 9 Spalten, 10 Minen

Größe des Spielfelds:

Einfach

Neu aufbauen

Leere Zellen:

ja

Spiel:

gewonnen

Verstrichen:

13.607

	1	1	
	2		
	3		
	2		
	1	1	
	1	1	
	1		
1	1	2	1
1		1	

Spieler	Zeit
Jochen	6.824s
Jochen	9.92s
Jochen	12.991s
Jochen	15s
Jochen	15.887s
Jochen	16.335s
Jochen	16.663s
Jochen	16.711s
Jochen	17.504s
Jochen	17.719s

### Neuer HighScore

Dein Ergebnis: 13.607s

Dein Name:

Mal wieder ein Test

×

Eintragen