

**Philosophische** Fakultät III

Sprach- , Literatur- und Kulturwissenschaften

Institut für Information und Medien, Sprache und Kultur (I:IMSK)  
Lehrstuhl für Medieninformatik

Projektseminar Mediengestaltung I: Informationsvisualisierung

Modul: MEI-M05.3

SS 2018

Leitung: Florin Schwappach

**[ Titel der Seminararbeit/Projektdokumentation]**

[Autor der Arbeit]

Matr.-Nr.: [Matrikelnummer]

[Semesterzahl und Studiengänge (z.B. 3. Semester B.A. Medieninformatik / Informationswissenschaft)]

E-Mail: [Emailadresse (z.B.: [max.mustermann@stud.uni-regensburg.de](mailto:max.mustermann@stud.uni-regensburg.de))]

[bei mehreren Autoren ggf. weitere Informationsblöcke]

Abgegeben am [Abgabetermin der Arbeit]



Inhalt

[1 Einleitung 10](#_Toc452981242)

[2 Ziele 11](#_Toc452981243)

[3 Stand der Technik 11](#_Toc452981244)

[4 Gestaltungsrichtlinien 12](#_Toc452981245)

[4.1 Sprache und Textumfang 12](#_Toc452981246)

[4.2 Inhaltliche Bestandteile 12](#_Toc452981247)

[4.3 Formatierung 15](#_Toc452981248)

[4.3.1 Seitengestaltung und Druck 15](#_Toc452981249)

[4.3.2 Typographie und Textsatz 15](#_Toc452981250)

[4.3.3 Abbildungen 16](#_Toc452981251)

[4.3.4 Tabellen 17](#_Toc452981252)

[4.3.5 Code 17](#_Toc452981253)

[4.4 Zitierweise 18](#_Toc452981254)

[4.4.1 Direkte Zitate 18](#_Toc452981255)

[4.4.2 Indirekte Zitate 19](#_Toc452981256)

[4.4.3 Sekundäre Zitate 19](#_Toc452981257)

[4.4.4 Zitierweise im Text 20](#_Toc452981258)

[4.4.5 Angaben im Literaturverzeichnis 20](#_Toc452981259)

[5 Codestruktur 22](#_Toc452981260)

[5.1 Handbücher 22](#_Toc452981261)

[5.2 Darstellung der Ergebnisse 22](#_Toc452981262)

[6 Zusammenfassung 23](#_Toc452981263)

[Literaturverzeichnis 24](#_Toc452981264)

[Anhang A: Bausteine wissenschaftlicher Arbeiten 25](#_Toc452981265)

[A1 Theoretische Arbeit 25](#_Toc452981266)

[A2 Konstruktive Arbeit 25](#_Toc452981267)

[A3 Empirische Arbeit 25](#_Toc452981268)

[Erklärung zur Urheberschaft 27](#_Toc452981269)

[Stichwortverzeichnis (optional, in der Regel nicht notwendig) 30](#_Toc452981270)

# Visuelle Kodierung der Daten

# Dokumentation der explorativen Möglichkeiten

„Bei der explorativen Analyse findet häufig eine interaktive, ungerichtete Suche nach Informationen und Strukturen statt, ohne dass Benutzer im Detail erklären könnten, was überhaupt gesucht wird.“(Preim, Dachselt (2010) S.442).

Die explorativen Möglichkeiten wurden an dem „Mantra visueller Infortmationssuche“ (Shneiderman, 1996) ausgerichtet:

1. Gewinnen eines Überblicks über den gesamten Informationsraum.
2. Zoomen zum Betrachten einer kleineren Untermenge von Daten.
3. Herausfiltern von uninteressanten Datenobjekten.
4. Auswahl eines Datenobjektes oder einer Gruppe von Daten, um Details zu erhalten.

# Dokumentation der Softwarearchitektur

Die Softwarearchitektur ist eine Mischung aus dem klassischen MVC-Pattern und dem Pattern, welches Redux verwendet[[1]](#footnote-1).

## index.html und css

Die index.html Datei gibt die Grundstruktur der Website durch HTML-Tags vor. Die css-Datei stellt die css-Klassen, um das Layout genauer zu definieren.

## Charts und Pages

Die Komponenten der einzelnen Seiten werden in Javascript geschrieben. Die Hauptmodule sind Charts und Pages. Charts sind Klassen[[2]](#footnote-2), deren Instanzen Daten visualisieren, zum Beispiel in der Form eines Sunbursts. Jede dieser Chart-Klassen erbt von der Klasse MagicCircle. Pages sind Module, deren Instanzen interaktive Seiten darstellen.

Dazu bedienen sie sich der Chartklassen. In den Pages-Klassen werden Instanzen der einzelnen Charts erzeugt, welche dargestellt werden sollen. Dort wird auch die Kommunikation zwischen diesen einzelnen Charts implementiert.

## htmlElements

Die einzelnen Charts greifen auf die Objekte des Namespaces htmlElements zu. Darin befinden sich für jeden einzelnen Chart hinterlegt relevante Variablen, wie die Breite und Höhe oder der Containertyp, z.B. SVG, indem der Chart dargestellt werden soll.

## Store

Die Daten welche die Charts brauchen, um die Informationen darzustellen befinden sich im Ordner Store. Dazu zählen unter anderem die csv-Datei crimeCorrelation.csv, welche die Korrelationen zwischen den einzelnen Verbrechen im Tabellenformat enthält oder die JSON-Datei crimes.json, welche alle relevanten Daten der FBI-Kriminalstatistik im JSON-Format darstellt. Die Klassen greifen direkt auf diese Dateien zu. Ebenfalls enthält der Store die Config-Datei config.js, welche Werte, die klassenübergreifend relevant, aber keine FBI-Statistiken sind, innerhalb von Objekten speichert. Dazu zählen z.B. die Farben die für jedes Verbrechen hinterlegt sind. Ebenfalls enthält die die config.js die Pfade zu sämtlichen Skripten die geladen werden.  
Im Store befindet sich ebenfalls die Datei commonfunctions.js. Darin befinden sich Funktionen, welche Klassenübergreifend relevant sind. Zum Beispiel greift die Funktion getCrimeColor(crime) auf die config-Datei zu und gibt die Farbe zurück, welche für ein bestimmtes Verbrechen hinterlegt ist.

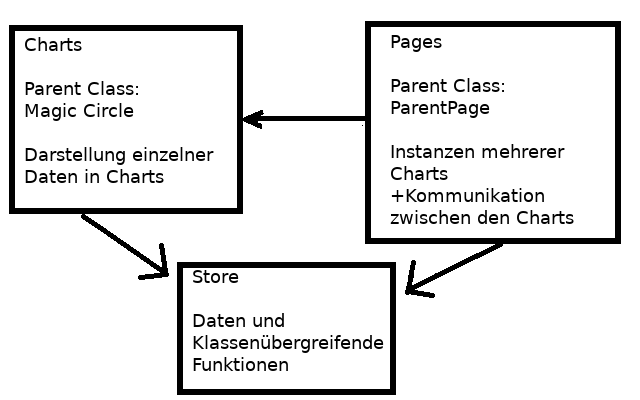


Abbildung 1 Pages enthalten die Instanzen der Charts. Beide greifen auf den Store zu.

## Statemachine

Die einzelnen Seiten werden in einer StateMachine verwaltet. In der Klasse StateMachine werden die Instanzen der Seiten erstellt. In jedem Zustand wird eine andere Seite dargestellt. Der Zustand wird mittels der Methode switchState(state) gewechselt.

Die Instanz der StateMachine befindet sich in der Datei components.js.  
In der Datei action.js befinden sich die Funktionen, welche switchState(state) mit unterschiedlichen Zuständen als Parameter aufrufen und somit die StateMachine in einen neuen Zustand versetzen. Diese Actions werden in der Datei listener.js aufgerufen. In dieser Datei werden die DOM-Elemente mit Click-Listenern ausgestattet. Sobald ein DOM-Element geklickt wird, wird die passende Seite dargestellt, indem die passende Action aufgerufen wird und die StateMachine den Zustand wechselt.

## C:\Users\b64\AppData\Local\Microsoft\Windows\INetCache\Content.Word\StateMachine.png

Abbildung 2 Der Aufruf der Seiten erfolgt nach dem Observer-Listener Pattern.

## 

## Zuordnung der Pages zum DOM-ELement

Damit jede Page und jeder Chart weiß, in welchem Element des DOMs sie dargestellt werden soll, erhält jede Page im Konstruktor eine id. Diese id gibt sie an ihre Charts weiter. Die id ist die css-id des DOM-Elements. Mittels der commonfunctions-Methode getPageById(id) wird das DOM-Element herausgesucht und geladen.

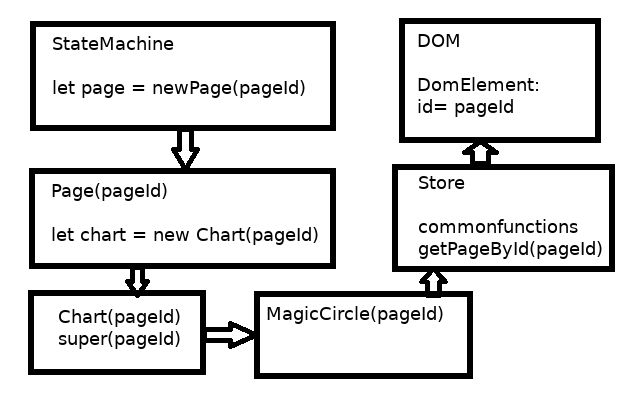


Abbildung 3 Die Zuordnung der Elemente zum DOM-Container erfolgt mittels einer id

## Scriptloader

Das Scriptloader-Modul lädt alle Scripts und Bibliotheken, indem es auf die in der scripts.js hinterlegten Pfade zurückgreift. Diese Methode wird in der index.html aufgerufen, um die Anwendung zu initialisieren.

|  |  |
| --- | --- |
| Charts | Enthält die Charts. Diese stellen Daten graphisch dar. |
| Pages | Enthält die Pages. Diese sind für die Instanziierung der Charts und die Kommunikation zwischen den Charts verantwortlich, |
| htmlelements | htmlelements.js . Enthält Layout-Daten für die Charts. |
| StateMachine | Enthält StateMachine, Components, Listener, Actions |
| Scripts | Enthält ScriptLoader und Pfade zu den Scripts  und Bibliotheken. |
| Store | Enthält Daten, Config - und css Datei |
| libs | Enthält Bibliotheken |

Abbildung 4 Die wichtigsten Komponenten im Überblick.

1. Redux(2018). Three Principles. Einsehbar unter: <https://redux.js.org/introduction/three-principles>, zuletzt abgerufen am 26.06.2018 [↑](#footnote-ref-1)
2. Es handelt sich dabei nicht, um Klassen im OOP-Sinn, sondern die der ECMAScript 2015 - Syntax. [↑](#footnote-ref-2)