# 1 Background

Necessary background for topic, in order to understand the problem, motivation and contribution. Approximately 10 pages.

## 1.1 JSON

## 1.2 Javascript Framework D3

Eine grundlegende Entscheidungen zu Beginn der Arbeit betraf die Frage, ob ein Javascript-Framework zur Unterstützung bei der Baumvisualisierung eingesetzt werden sollte und, wenn ja, welches. Die Wahl fiel dabei auf D3, weil es neben Unterstützung von Baum- und Graphenstrukturen auch eingebaute Funktionen für Animationen aufweist, welche, wie im Kapitel 2.4 beschrieben, eine zentrale Rolle für eine intuitive Bedienung spielen. D3 ist außerdem weit verbreitet, wodurch bei der Entwicklung auf eine reichhaltige Menge von Beispielen und Anregungen zurückgegriffen werden kann.

### 1.2.1 Hierarchies

[todo: Erklärung von heirarchies und wie man damit bäume speichern kann]

### 1.2.2 Selections

[todo: Erklärung von selections und speziell der Funktionen enter(), exit() und data()]

## 1.3 SVG

[todo: erläuterung von svg elementen, welche elemente im inneren benutzt werden können und ihre bedeutung]

# 2
# Contribution

Most important chapter of the thesis. Describes what the author contributes as research. Discusses intuition, motivation, describes and reasons about necessity of proposed elements. Defines theses based on reasonable assumptions. Discusses relevant aspects of contribution. Approximately 30 to 40 pages. Can be split into multiple chapters.

## 2.1 Baumlayout

### 2.1.1 Linear vs. Radial

Aufgrund des begrenzten Platzes, der auf mobilen Geräten typischerweise zur Verfügung steht, war früh in der Entwicklung zu entscheiden, wie Baumstrukturen möglichst platzsparend anzuordnen sind, ohne Übersichtlichkeit einzubüßen. Betrachtet wurden dabei speziell die allgemein übliche Darstellung (linear) gegenüber einer kreisförmigen (radialen) Anordnung.
Die Abbildungen 2.1 und 2.2 zeigen einen Vergleich zwischen einem Baum in linearer (Abb. 2.1) und in radialer Anordnung (Abb. 2.2). In beiden Fällen ist derselbe Baum der Höhe 4 auf gleicher Fläche abgebildet. Er besteht aus 105 Knoten, wobei alle inneren Knoten jeweils vier Nachfolger haben. Der Wurzelknoten befindet sich beim linearen Layout am oberen Rand. Die übrigen Knoten sind in drei horizontalen Linien darunter angeordnet, wobei die Kinder der Wurzel auf der obersten Linie liegen, deren Kinder auf der mittleren und deren Kinder auf der untersten Linie. Beim radialen Layout ist die Wurzel in der Mitte abgebildet und alle anderen Knoten in drei konzentrischen Kreisen darum herum. Knoten der Tiefe 1 liegen auf dem innersten Kreis, der Tiefe 2 auf dem mittleren Kreis und die Blätter auf dem äußeren Kreis. Den Mittelpunkt der Kreise bildet die Wurzel. Man sieht deutlich, dass im Fall der vertikalen Anordnung (Abb. 2.1) nicht ausreichend Platz für alle Blätter zur Verfügung steht, wodurch es zu starken Überschneidungen kommt. In der radialen Anordnung (Abb. 2.2) können hingegen alle Knoten überschneidungsfrei dargestellt werden.
Um den Grund dafür zu veranschaulichen vergleicht man den Platz, der bei den beiden Herangehensweisen auf einer einzelnen Stufe des Baumes zur Verfügung steht. Unter der Annahme,

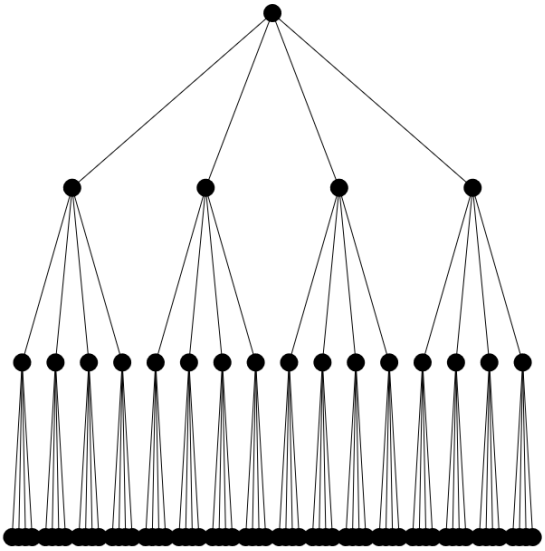

Abbildung 2.1: Lineare Baumdarstellung

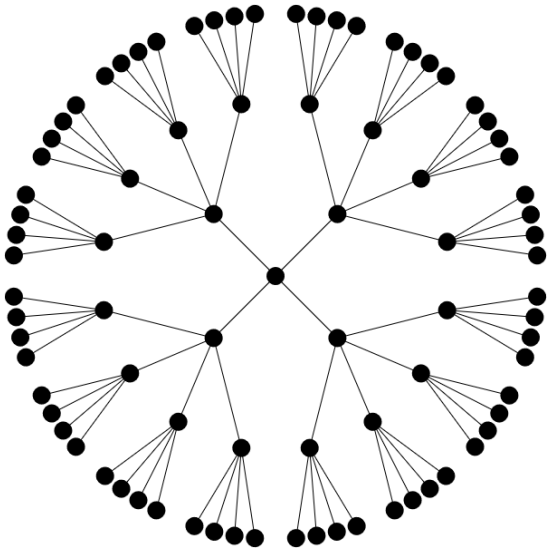

Abbildung 2.2: Radiale Baumdarstellung

dass für die gesamte Darstellung ein Quadrat mit Seitenlänge $a$ zur Verfügung steht und alle Knoten einen Durchmesser von 1 haben ergibt sich für den Platz $l_{linear}$ bzw. $l_{radial}$, der auf einer Stufe des Baumes verfügbar ist

$$\begin{aligned} l_{linear} &= a \\ l_{radial} &= 2 \cdot \pi \cdot r \qquad mit \quad 0 \leq r \leq \frac{a}{2}. \end{aligned}$$

Dabei bezeichnet $r$ den Abstand einer Stufe zum Wurzelknoten. Setzt man $l_{radial}$ und $l_{linear}$ gleich und stellt nach $r$ um, so ergibt sich

$$\begin{aligned} & l_{linear} = l_{radial} \\ \Rightarrow \quad & a = 2 \cdot \pi \cdot r \\ \Rightarrow \quad & \frac{a}{2 \cdot \pi} = r \\ \Rightarrow \quad & r \approx \frac{a}{6}. \end{aligned}$$

Da $l_{linear}$ konstant ist kann man daraus ableiten

$$l_{radial} > l_{linear} \qquad \Leftrightarrow \qquad r > \frac{a}{6}.$$

Das bedeutet, dass ab einem Abstand zur Wurzel von mehr als $\frac{a}{6}$ im radialen Layout mehr Platz für jede Stufe verfügbar ist. Außerdem gilt: Je größer der Abstand $r$, desto mehr Platzersparnis. Das ist vor allem deshalb von Interesse, weil Bäume in der Regel die Eigenschaft haben, dass mit wachsender Baumtiefe auch die Anzahl der Knoten in der jeweiligen Tiefe wächst und diese tiefer liegenden Knoten in der kreisförmigen Darstellung weiter von der Wurzel entfernt sind. Das radiale Layout bietet somit genau dann mehr Platz, wenn üblicherweise mehr Platz benötigt wird.
Aufgrund dieser Erkenntnis und der Tatsache, dass beide Darstellungsformen gute Übersichtlichkeit aufweisen, fiel die Wahl letztendlich auf das radiale Layout.

### 2.1.2 Polarkoordinaten

D3 bietet zur Berechnung einer sinnvollen Knotenanordnung von Bäumen die Funktion $d3.tree()$ an, welche unter Verwendung des Reingold-Tilford Algorithmus [todo: quelle] allen Knoten eines Baumes x- und y-Koordinaten jeweils im Bereich von 0 bis 1 zuordnet. Es liegt dann in der Hand des Programmierers, diese sinnvoll zu interpretieren. Um die Knoten, wie in Abbildung 2.2 gezeigt in konzentrischen Kreisen anzuordnen, eignen sich Polarkoordinaten ausgezeichnet. Einem Vorschlag aus der Dokumentation von D3 folgend [todo: quelle] wird die y-Koordinate als Radius $\rho$, die x-Koordinate als Polarwinkel $\varphi$ in Radiant interpretiert. [todo: bild]

$$\rho = y$$
$$\varphi = 2 \cdot \pi \cdot x$$

Zur Darstellung auf dem Bildschirm müssen $\rho$ und $\varphi$ anschließend in kartesische Koordinaten $x_{screen}$ und $y_{screen}$ umgerechnet werden. Die allgemeinen Umrechnungsformeln ergeben sich als

$$x_{screen} = \rho \cdot \cos(\varphi)$$
$$y_{screen} = \rho \cdot \sin(\varphi)$$

Bei dieser Umrechnung wird allerdings noch nicht berücksichtigt, dass die gegebenen Polarkoordinaten auf generischen Koordinaten im Bereich $[0,1]$ basieren. Zur korrekten Positionierung müssen noch eine Skalierung auf die verfügbare Breite (width) $w$ und Höhe (height) $h$, sowie eine Verschiebung in die Mitte der Anzeige vorgenommen werden. Es entstehen die endgültigen Formeln:

$$x_{screen} = \rho \cdot \cos(\varphi) \cdot w + \frac{w}{2} \tag{2.1}$$
$$y_{screen} = \rho \cdot \sin(\varphi) \cdot h + \frac{h}{2}. \tag{2.2}$$

## 2.2 Anzeigen der Knoten mit D3

[todo: Beschreibung der JSON struktur, wie sie umgewandelt wird in eine hierarchy. diskussion über canvas vs svg]
Nachdem mit dem radialen Layout (Kapitel 2.1.1) eine erste Maßnahme zum Einsparen von Platz ergriffen wurde, stellt sich als nächstes die Frage, wie die Übersichtlichkeit weiter verbessert werden kann.

## 2.3 Reduzieren der angezeigten Knoten

### 2.3.1 Relevante und irrelevante Knoten

Da Knoten auf dem Bildschirm später Informationen in Form von Text beinhalten sollen, ist es abzusehen, dass jeder einzelne Knoten mehr Platz einnehmen wird, als z.B. in Abbildung 2.2 gezeigt ist. Es bietet sich an, immer nur aktuell relevante Knoten ein- und irrelevante auszublenden. Das erfordert je nach Eingabe dynamische Änderungen an der Anzeige der Baumstruktur (Kapitel 2.3.2). Zunächst muss jedoch entschieden werden, welche Knoten aktuell relevant

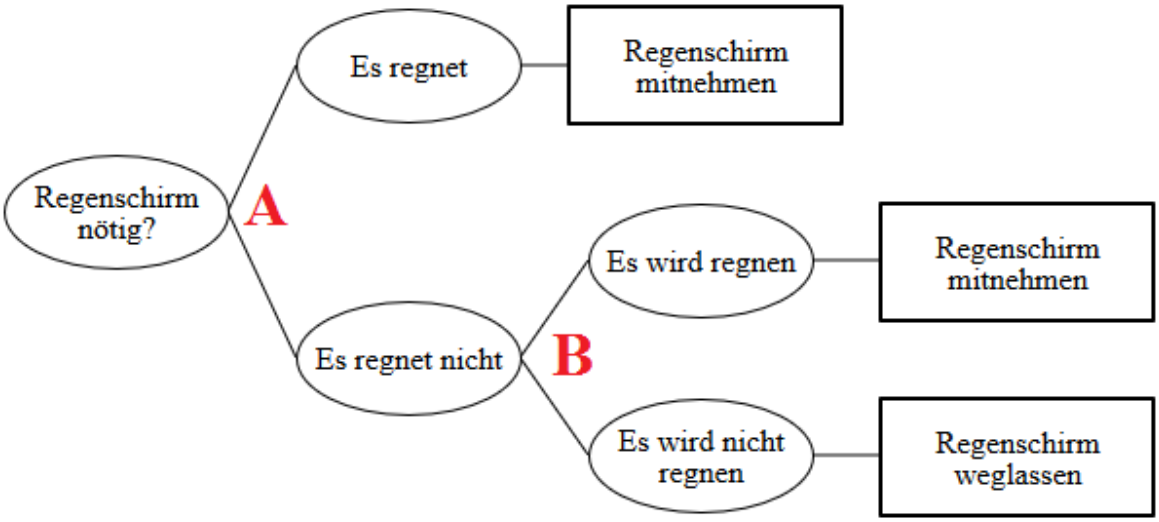


Abbildung 2.3: Ein simpler Entscheidungsbaum

oder irrelevant sind. In Abbildung 2.3 ist beispielhaft ein Entscheidungsbaum zu sehen, der in stark vereinfachter Form die Entscheidungsfindung zur Frage, ob ein Regenschirm nötig ist, zeigt. Innere Knoten sind als Ellipsen dargestellt und Blätter, welche endgültige Ergebnisse repräsentieren, sind rechteckig. Die Beschriftung der Knoten zeigt an, welche Aussage oder Frage sie symbolisieren. Die beiden Stellen, an denen Entscheidungen getroffen werden müssen, sind mit *A* und *B* markiert. In Situation *A* gilt es zu entscheiden, ob es regnet oder nicht. Um diese Entscheidung treffen zu können, ist nicht relevant, ob es zu einem späteren Zeitpunkt regnen wird oder nicht und welche Ergebnisse sich daraus ableiten lassen. Bei *B* wiederum sind vorherige Entscheidungsmöglichkeiten nicht von Interesse, ebenso wie die Folgen davon, ob es regnen wird oder nicht. Man kann sagen, dass bei jeder Verzweigung nur die zur Verfügung stehenden Alternativen relevant sind und angezeigt werden müssen. Da es jedoch nicht nur darum geht, so viel Platz wie möglich zu sparen, sondern auch darum, eine übersichtliche und intuitiv Verständliche Darstellung zu finden ist es hilfreich, außerdem noch den Knoten, von dem die Verzweigung ausgeht zu zeigen. Bei *A* also den Knoten „Regenschirm nötig“, bei *B* „Es regnet nicht“. Auf diese Art ist es leichter die Orientierung zu behalten, selbst wenn ein Großteil des Baumes nicht sichtbar ist.

### 2.3.2 Navigieren durch den Baum

Durch das Ausblenden vieler der Knoten muss nun eine Möglichkeit geschaffen werden, von einer Entscheidung zur nächsten zu navigieren, wobei immer wieder irrelevante Knoten aus- und relevant gewordene eingeblendet werden. Das wird möglich gemacht, indem die Baumanzeige auf Berührung hin dynamisch aktualisiert wird. Es gilt dabei, dass der Knoten, von dem die aktuelle Verzweigung ausgeht, in der Mitte des Bildschirms zu sehen ist und die nachfolgenden Entscheidungsmöglichkeiten im Kreis darum herum angeordnet werden. Abbildung 2.4 zeigt die Reaktion des Baumes auf verschiedene Nutzereingaben. Auf der linken Seite ist der Ausgangszustand zu sehen und rechts der Nachfolgezustand. Links ist der mittlere Knoten

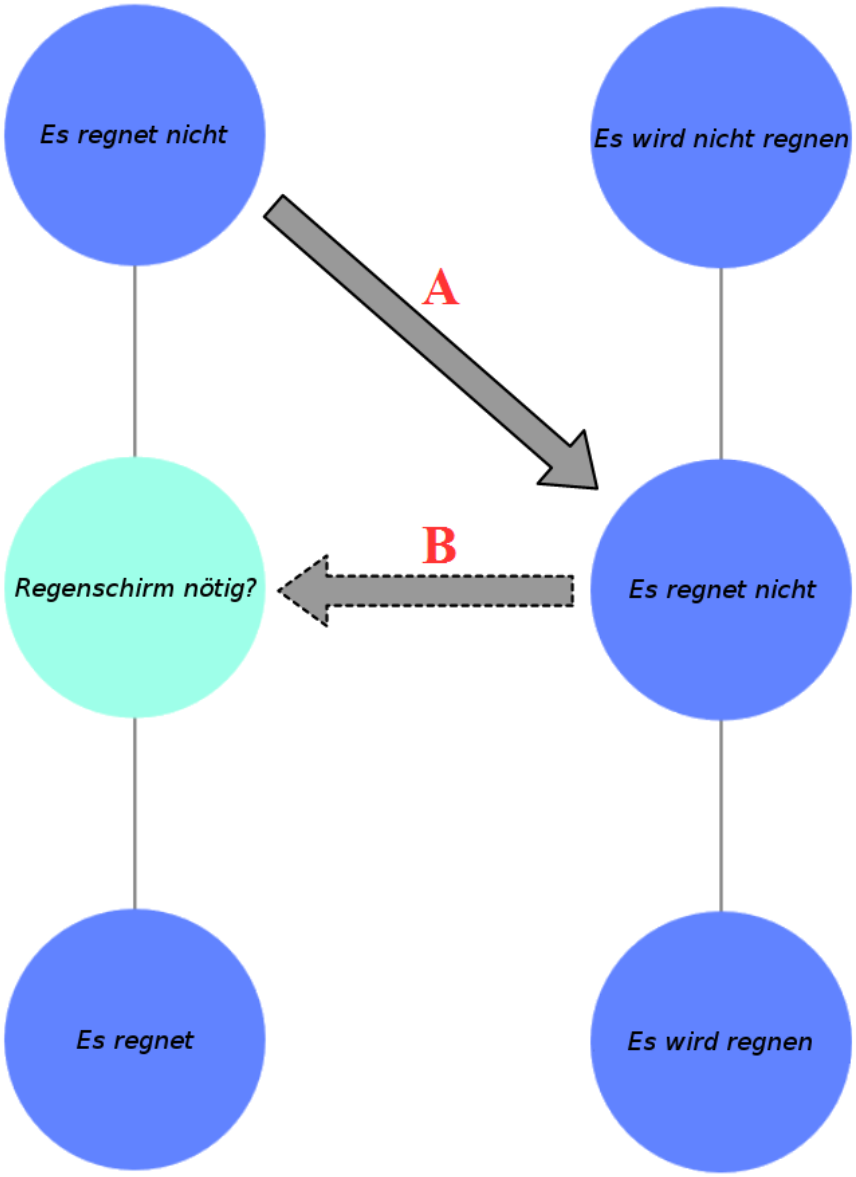


Abbildung 2.4: Beispiel zur Baumnavigation

mit „Regenschirm nötig?“ beschriftet und dessen Nachfolger mit „Es regnet nicht“ sowie „es regnet“. Rechts sieht man „Es regnet nicht“ in der Mitte, „Es wird nicht regnen“ und „Es wird regnen“ als Folgeknoten. Die mit *A* und *B* beschrifteten Pfeile symbolisieren Reaktionen auf Eingaben, die im Folgenden genauer beschrieben werden.
Durch Antippen eines der äußeren Knoten wird dieser in die Mitte verschoben und es zeigen sich dessen Nachfolger. Tippt man zum Beispiel in Abbildung 2.4 links „No red flags“ an, dann wird dieser, wie Pfeil *A* zeigt, zum mittleren Knoten, die anderen verschwinden und es erscheinen die neuen Nachfolgeknoten. Möchte man einen Schritt zurück machen, kann der Mittelknoten berührt werden und man wird wieder zur links gezeigten Situation geleitet (illustriert durch Pfeil *B*).
Um diese Aktualisierungen der Anzeige zu erreichen macht man Gebrauch von *hierarchies* und *selections* in D3 und besonders von deren Funktionen *data*(), *enter*() und *exit*() (siehe Kapitel

1.2). Wird ein Knoten angetippt,

## 2.4 Einsatz von Animationen