JavaScript Libraries

1. Einführung

1.1. Was sind Libraries

Möglichst einfach ausgedrückt ist eine Library eine Code-Sammlung wiederkehrender / mehrfach verwendeter Funktionen und Methoden. Diese soll dazu dienen, die Programmierung mit JavaScript zu vereinfachen. Durch die Verwendung einer Library müssen gewisse Teile eines Codes nicht jedes Mal von neuem geschrieben werden, sondern können durch das Aufrufen der entsprechenden Funktion innerhalb einer Library in den eigenen Code eingebunden werden. Des Weiteren dienen Libraries dazu, diese bereits existierenden Funktionen weithin zu verbreiten, so dass Programmierende diese in ihren eigenen Projekten verwenden können, ohne die Funktionen jedes Mal selbst von Grund auf schreiben zu müssen.[1] In der Regel werden innerhalb einer Library Funktionen mit einem gemeinsamen Anwendungsfokus gebündelt[2] (Bsp.: Interaktion mit Formularfeldern, Gestalten des User Interface, Datenvisualisierung, …[3]). Je nach dem welches Ziel man gerade verfolgt, sollte man auch die dem entsprechende Library wählen.

1.2. Unterschied zu Frameworks

Ein Framework stellt im Gegensatz zu einer Library einen bereits vordefinierten Rahmen / eine grundlegende Software-Architektur und besitzen bestimmte Design-Templates und eigene Funktionen, welche wiederum in einer Library zusammengefasst sind. Daher sind Frameworks um einiges komplexer als Libraries. Innerhalb des Rahmens, welcher durch das Framework vorgegeben wird, kann durch eigenen Code die gewünschte Applikation eingebaut werden, welche die beinhalteten Libraries aufrufen kann.[4] Die Beziehung zwischen der eigenen Applikation und einem Framework und einer (oder mehreren Libraries) ist in Abbildung 1 visualisiert.

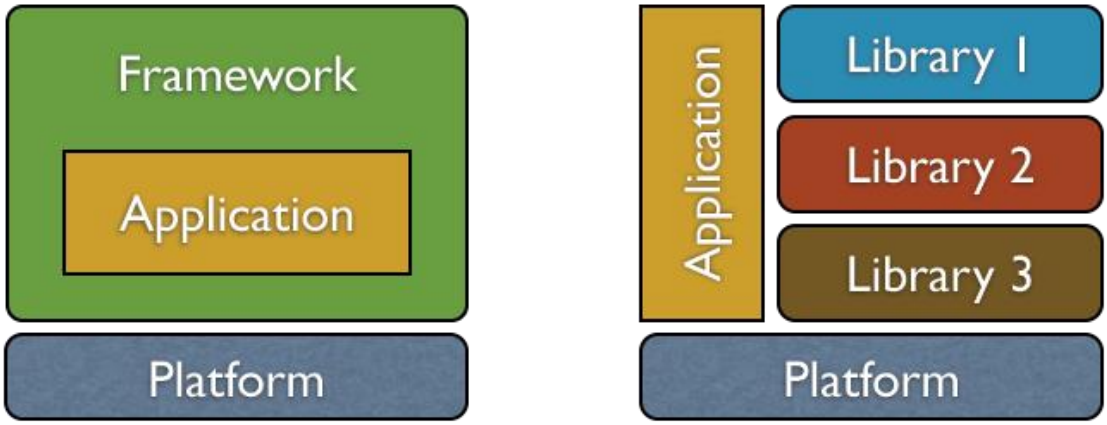

Abbildung 1: Anordnung der eigenen Applikation im Verhältnis zu einem Framework oder verschiedenen Libraries (https://tom.lokhorst.eu/2010/09/why-libraries-are-better-than-frameworks, letzter Zugriff 02.05.2023)

[1] https://kinsta.com/blog/javascript-libraries (letzter Zugriff 05.05.2023)
[2] https://www.ionos.de/digitalguide/websites/web-entwicklung/beliebte-javascript-frameworks-und-bibliotheken/ (letzter Zugriff 05.02.2023)
[3] https://kinsta.com/blog/javascript-libraries (letzter Zugriff 05.05.2023)
[4] https://www.reddit.com/r/explainlikeimfive/comments/tt6h5c/eli5_what_is_a_framework (letzter Zugriff 02.05.2023)

Der grösste Unterschied zwischen Libraries und Frameworks liegt in der «Inversion of Control». Dies bezeichnet die Art und Weise wie der Code aufgerufen wird. Bei der Nutzung einer Library, werden im Code die Aufrufe der Library festgehalten. An diesen Stellen wird lediglich die gewünschte Funktion aus der Library ausgeführt und die Kontrolle wird im Anschluss wieder an den eigenen Code übergeben. Bei einem Framework hingegen wird bereits im Vornherein definiert, an welchen Stellen der eigene Code eingebettet und ausgeführt werden kann. Die Frameworks haben somit auch den Kontrollfluss bereits vordefiniert. Der eigene Code wird dabei durch das Framework aufgerufen. [5][6] Dies ist in Abbildung 2 visualisiert.

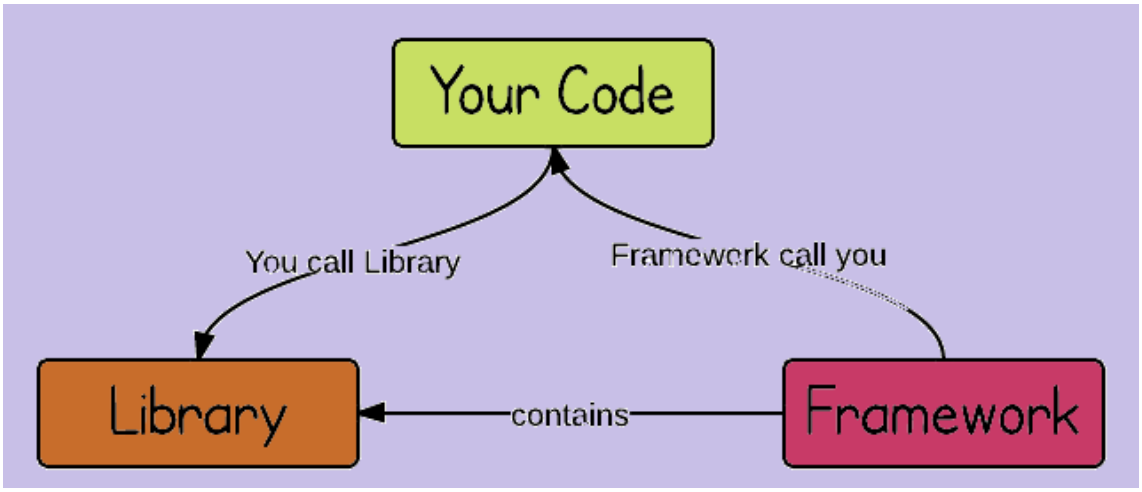

Abbildung 2: Visualisierung der Inversion of Control (https://stackoverflow.com/questions/148747/what-is-the-difference-between-a-framework-and-a-library), letzter Zugriff 02.05.2023)

1.3. Einsatzgebiete

Aufgrund der Unterschiede macht es mehr Sinn, ein Framework bereits zu Projektbeginn als Startpunkt zu verwenden. Anders als bei einer Library ist es eher schwierig ein Framework in einen bereits existierenden Code einzubauen. Dafür soll das Framework die Komplexität beim Programmieren verringern und enthält zu diesem Zweck oft zusätzliche Tools und Dokumentationen, um das Framework möglichst effektiv nutzen zu können.[7]

Libraries können zu verschiedenen Zwecken eingesetzt werden, oftmals macht es Sinn auf eine Library zuzugreifen, wenn die eigene Expertise fehlt. Man kann auf bestehende Funktionen zugreifen, anstatt diese selbst schreiben zu müssen und kann somit Zeit sparen. Das Verwenden der Library wiederum bietet eine Möglichkeit die benötigte Expertise aufzubauen und das Verwenden von Funktionen hilft den Code frei von Redundanzen zu halten.[8] Die Einsatzgebiete sind von der jeweiligen Library selbst abhängig. [9]

[5] https://stackoverflow.com/questions/3057526/framework-vs-toolkit-vs-library/3057818#3057818 (letzter Zugriff 02.05.2023)
[6] https://web.dev/choose-js-library-or-framework/ (letzter Zugriff 02.05.2023)
[7] https://web.dev/js-libraries-vs-frameworks/ (letzter Zugriff 02.05.2023)
[8] https://web.dev/choose-js-library-or-framework/ (letzter Zugriff 02.05.2023)
[9] https://kinsta.com/blog/javascript-libraries/ (letzter Zugriff 05.05.2023)

2. Libraries

2.1. D3.js

D3.js (D3 für Data-Driven Documents[10]) ist eine JavaScript Library die darauf ausgelegt ist, aus Datensätzen dynamische und interaktive SVG-Visualisierungen in Webbrowsern zu erstellen. Die Bandbreite an möglichen Diagrammen ist riesig, eine grosse Anzahl verschiedener D3-Graph-Galleries können einen guten Eindruck davon geben.[11] Ausserdem bietet D3 die Möglichkeit Visualisierungen, um animierte Transitionen und Interaktionsmöglichkeiten zu erweitern.

Das Hauptprinzip der Library besteht darin, dass Dokumente, genauer gesagt deren DOM- Elemente, datenbasiert manipuliert werden. Mit den Library-eigenen Funktionen können bestehende Elemente eines Dokumentes angesprochen, unnötige Elemente gelöscht oder neue Elemente erstellt werden. Sobald ein Element angesprochen ist, kann es manipuliert werden - und zwar datenbasiert, also indem man Daten oder ganze Datensätze an das Element bzw. die Elemente bindet. Die Properties, welche man den Elementen gibt, können dabei auch in Form von Funktionen geschrieben werden, und zwar so, dass sie sich auf den jeweils zugehörigen Datenwert beziehen. Verändert sich der Datensatz, so passen sich die Elemente automatisch entsprechend an. Ein simples Beispiel: Hat man einen Array mit Zahlen, so kann man mit D3 für jede Zahl ein Kreis-Element erstellen lassen. Den Radius jedes Kreises kann man entsprechend des Wertes seines Array-Datenwertes setzen lassen, in dem man die Array-Daten an die Kreis-Elemente bindet. Kombiniert mit den bereits erwähnten Interaktionsmöglichkeiten und Transitionen lassen sich so sehr komplexe Datenvisualisierungen umsetzen.

D3 kann sowohl mit statischen Daten arbeiten sowie auch (live) Daten aus externen Ressourcen laden. Dabei können diese Daten unterschiedliche Formate wie CSV, JSON oder XML haben - damit D3 die Datensätze verwenden kann, müssen die Daten einfach jeweils zuerst in Arrays umgewandelt werden.[12] Die Library arbeitet mit den gängigen Webstandards HTML, CSS und SVG, und unterscheidet sich von anderen Visualisierungs-Libraries vor allem durch ihre enorme Flexibilität. Sie ist nicht an proprietäre Frameworks gebunden und arbeitet auch nicht mit vorgefertigten Grafiken, die Programmierenden legen alle Elemente und deren grafische Repräsentation selbst an, was eine riesige kreative Freiheit bedeutet. Die Verwendung von D3.js erfordert keine besonderen Tools oder Plugins, jedoch ein solides Grundwissen zu den modernen Webtechnologien. Die Lernkurve ist anfangs steil, ich habe einige Stunden gebraucht, um das Grundprinzip wirklich zu verstehen. Mit diesem Verständnis ist es möglich, ganz simple Visualisierungen zu erstellen, für komplexere Datenvisualisierungen sind aber noch viele weitere Stunden nötig. Dadurch, dass die Library keinerlei kreativen Vorgaben oder Restriktionen vorgibt, kann man auch als versierte*r D3-Programmierer*in immer wieder Neues lernen. Über die Performance bei grossen Datensätzen scheiden sich die Geister: Für die einen ist die Library schnell genug[13], andere befinden sie als langsam.[14] Auf jeden Fall ist D3.js sehr weit verbreitet, sodass sich

[10] https://d3js.org/ (letzter Zugriff 05.05.2023)
[11] https://d3-graph-gallery.com/graph/ridgeline_animation.html (letzter Zugriff 05.05.2023)
https://observablehq.com/@d3/gallery (letzter Zugriff 05.05.2023)
[12] https://observablehq.com/@d3/learn-d3-data?collection=@d3/learn-d3 (letzter Zugriff 05.05.2023)
[13] https://www.freecodecamp.org/news/d3js-tutorial-data-visualization-for-beginners/ (letzter Zugriff 05.05.2023)
[14] https://www.kofi-group.com/5-reasons-why-d3-js-is-the-best-framework-for-data-visualization/ (letzter Zugriff 05.05.2023)

viele gute und verständliche Tutorials dazu finden lassen. Die Library ist sehr mächtig und – zumindest für Menschen mit Kenntnissen in modernen Webtechnologien – für dynamische und interaktive Datenvisualisierung, zu empfehlen. Wer sich einarbeiten möchte, findet auf Observable[15], Freecodecamp[16] und D3indepth[17] ausführliche, verständliche Tutorials für Einsteiger*innen.

2.2. React

Bei React handelt es sich um eine JavaScript-Library, die für die Erstellung von webbasierten Benutzeroberflächen verwendet wird. Der Fokus von React liegt stark auf der Verwendung von Komponenten. Jede React-Applikation ist aus einzelnen Komponenten zusammengebaut, die ineinander verschachtelt werden können. Im Grunde genommen sind React-Komponenten JavaScript-Funktionen, die Markup zurückgeben. Komponenten sollten so aufgebaut sein, dass sie beliebig wiederverwendet werden können. Deshalb haben sie in den meisten Fällen auch ein eigenes CSS-File. Geschrieben werden die React-Komponenten im Normalfall mit JSX, einer Syntax-Erweiterung für JavaScript. JSX sieht HTML sehr ähnlich, ist aber ein wenig strikter und kann dynamische Daten darstellen.[18] Wie einige andere Libraries auch, verwendet React ausserdem ein virtuelles DOM. Durch dieses Konzept kann der Entwicklungsprozess beschleunigt und flexibler gestaltet werden. Ein weiteres Merkmal von React ist der unidirektionale Datenfluss (One-Way Data Binding). Das heisst, dass Daten nur in eine Richtung direkt fliessen können. Konkret bedeutet das, dass nur die übergeordnete Komponente Daten bei der untergeordneten Komponente direkt ändern kann (top-down) und nicht umgekehrt. Der umgekehrte Datenfluss (bottom-up) kann nur indirekt mit Hilfe von Callback-Funktionen stattfinden.[19]

Die beliebte Library hat verschiedene Vorteile, was dazu führt, dass sie auch von grossen Namen wie Netflix oder Airbnb eingesetzt wird. Durch die Verwendung des Virtual DOM-Konzepts profitiert React von einer verbesserten Leistung sowie verbesserten Update-Geschwindigkeit. Des Weiteren vereinfacht der unidirektionale Datenfluss die Fehlerbehebung und die wiederverwendbaren Komponenten erhöhen die Effizienz. Wenn man sich mit HTML und JavaScript bereits auskennt, ist der Einstieg in React ausserdem relativ einfach.[20] Wo es Vorteile gibt, gibt es aber auch immer gewisse Nachteile: Da sich React nur um die UI-Ebene kümmert, werden für eine vollständige Anwendung in den meisten Fällen noch weitere Tools benötigt. Auch die starke Abstraktion durch das Aufteilen der Anwendung in viele kleine Komponenten kann die Programmierung verkomplizieren. Ein weiterer oft genannter Nachteil von React ist zudem die schnelle Weiterentwicklung der Library, welche es Programmierenden erschwert Fuss zu fassen.[21] Kurz gesagt: React eignet sich besonders für interaktive Frontend-Applikationen.

Nun zu meinem Eindruck: Ich finde React eine spannende Library und es hat mir Spass gemacht mich damit auseinanderzusetzen. Besonders hilfreich für das konkrete

[15] https://observablehq.com/@d3/learn-d3?collection=@d3/learn-d3 (letzter Zugriff 05.05.2023)
[16] https://www.freecodecamp.org/news/d3js-tutorial-data-visualization-for-beginners/ (letzter Zugriff 05.05.2023)
[17] https://www.d3indepth.com/ (letzter Zugriff 05.05.2023)
[18] https://react.dev/learn (letzter Zugriff: 20.04.2023)
[19] https://kinsta.com/de/blog/vue-vs-react/#react-hauptmerkmale (letzter Zugriff 05.05.2023)
[20] https://www.wilde-it.com/react (letzter Zugriff: 23.04.2023)
[21] https://kinsta.com/de/blog/vue-vs-react/#nachteile-von-vue-vs-react (letzter Zugriff: 28.04.2023)

Lernen fand ich den Abschnitt ‘Learn’ auf der Website von React selbst (https://react.dev/learn). Einerseits gibt es dort für einen schnellen Einstieg den Abschnitt ‘Quick Start’, welcher unter anderem ein Tutorial für Tic-Tac-Toe enthält. Andererseits gibt es den Abschnitt ‘Learn React’, in welchem man detailliert an React herangeführt wird. Bei jedem Unterthema gibt es ausserdem eine bis mehrere Challenges wo man das im Kapitel Gelernte anwenden kann. Durch die bereits vorhandenen HTML-, CSS- und JavaScript-Kenntnisse fiel es mir auch nicht sonderlich schwer, die Syntax von JSX grundsätzlich zu verstehen. Je mehr ich jedoch ins Detail ging, desto komplexer empfand ich es. Einen groben Überblick kann man sich folglich schnell verschaffen, wenn man jedoch in die Tiefe gehen möchte, benötigt man definitiv mehr Zeit. Das Erstellen einer eigenen Mini-Applikation von Grund auf erwies sich als etwas komplizierter als erwartet, da das auf der React-Seite auch eher spärlich beschrieben ist. Das Erstellen des Grundbausteins war zwar schnell möglich, ich empfand es aber als störend, dass mir nicht ganz klar war, wofür all die Dateien sind, die mit dem Befehl ‘npx create-react-app’ automatisch erstellt wurden bzw. was diese genau machen. Beim Testen meiner Mini-Applikation musste ich zudem noch lernen, dass man nicht einfach die index.html-Seite im Browser öffnen kann, sondern dass man die Applikation mit dem Befehl ‘npm start’ starten muss. Nach diesen Anfangsschwierigkeiten ging es dann aber gut, die Mini-Applikation umzusetzen.

2.3. Vergleich D3.js und React

Im Grunde genommen geht es bei beiden Libraries um die Visualisierung des Frontends. Dadurch werden für das Erstellen einer vollständigen Webapplikation in den meisten Fällen noch weitere Tools benötigt, die sich um Routing, Serverkommunikation, etc. kümmern können. Ebenso reagieren beide auf Änderungen der ihnen zu Grunde liegenden Daten mit sofortigen Anpassungen in der Visualisierung. Jedoch haben die beiden unterschiedliche Anwendungsfoki und wenden zur Visualisierung andere Methoden an. Bei D3.js geht es vorwiegend um die grafische Darstellung eines zugrunde liegenden Datensatzes. Dazu werden direkt die DOM-Elemente mit ihrer Hierarchie angesprochen. Bei React geht es grundsätzlich um die Erstellung eines webbasierten User Interfaces. Dazu wird die Applikation zuerst in unterschiedliche Komponenten gegliedert. Die Visualisierung erfolgt durch das Ansprechen dieser Komponenten.[22] Ein weiterer Unterschied ist die Bearbeitung des DOMs. D3.js ändert jedes Mal direkt das gesamte DOM, während React mit Hilfe eines virtuellen DOMs eine Liste mit den minimalen Änderungen sammelt und diese erst danach am richtigen DOM ausführt.[23]

[22] https://medium.com/@jcukier/d3-and-react-similarities-and-differences-6aeb49fdaf5b (letzter Zugriff 05.05.2023)
[23] https://megagon.ai/react-d3-a-starters-guide (letzter Zugriff: 05.05.2023)