JavaScript Libraries

# Einführung

## 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, denn 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.

In der Regel werden innerhalb einer Library Funktionen mit einem gemeinsamen Anwendungsfokus gebündelt (Bsp.: Interaktion mit Formularfeldern, Gestalten des User Interface, Datenvisualisierung, …[[1]](#footnote-1)). Dementsprechend sollte man zur Programmierung die jeweils entsprechende Library wählen, je nach dem welches Ziel man gerade verfolgt.

<https://codeinstitute.net/global/blog/what-is-a-javascript-library/>

<https://www.ionos.de/digitalguide/websites/web-entwicklung/beliebte-javascript-frameworks-und-bibliotheken/>

## 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.[[2]](#footnote-2) Die Beziehung zwischen der eigenen Applikation und einem Framework und einer (oder mehreren Libraries) ist in Abbildung 1 visualisiert.

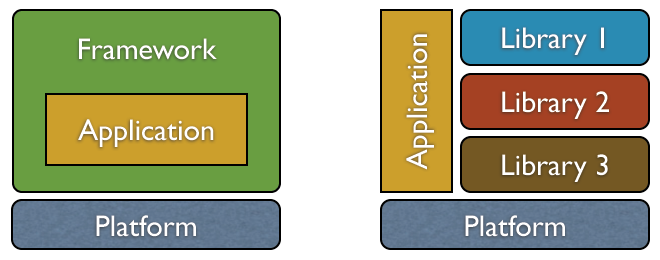


Abbildung : https://tom.lokhorst.eu/2010/09/why-libraries-are-better-than-frameworks

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 werden und ausgeführt werden kann. Die Frameworks haben somit den Kontrollfluss auch bereits vordefiniert, dadurch wird der eigene Code durch das Framework aufgerufen. [[3]](#footnote-3)[[4]](#footnote-4) Dies ist in Abbildung 2 visualisiert.



Abbildung : https://stackoverflow.com/questions/148747/what-is-the-difference-between-a-framework-and-a-library?

## 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. [[5]](#footnote-5)

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.4

# Libraries

## D3.js

D3.js ist eine JavaScript Library, welche es ermöglicht, aus Datensätzen dynamische und interaktive SVG-Visualisierungen in Webbrowsern zu erstellen.

Das Hauptprinzip der Library besteht darin, dass mit den integrierten Funktionen DOM-Elemente eines Dokumentes erstellt oder angesprochen, und danach manipuliert werden können, und zwar basierend auf Datensätzen. Die Library arbeitet dazu mit den Standards von HTML, CSS und SVG.

Um DOM-Elemente eines Dokumentes anzusprechen, wird die select() Methode verwendet, welche mit den gängigen CSS-Selektoren funktioniert. Nach dem Selektieren können die angesprochenen Elemente manipuliert werden, und zwar sowohl mit statischen wie auch mit dynamischen Eigenschaften. Eine Besonderheit der Library zeigt sich darin, dass solche Eigenschaften auch als Funktionen geschrieben werden können, welche dann, je nach zu Grunde liegenden Daten, unterschiedliche Grafiken hervorbringen.

Die zu Grunde liegenden Daten werden (in Form eines Arrays oder einer Variable) mit der Operation data() an die DOM-Elemente gebunden, und zwar jeder einzelne Datenpunkt an ein DOM-Element. Sind nicht genügend DOM-Elemente vorhanden, so werden die überzähligen Datenpunkte in der enter() Selektion gelagert, mit der neue DOM-Elemente für die Daten erstellt werden können. Sind mehr DOM-Elemente als Datenpunkte vorhanden, so werden die überzähligen DOM-Elemente in der exit() Selektion abgelegt, mit der die Elemente aus dem DOM entfernt werden können. Die default Selektion ist die Update-Selektion, dort werden die Daten gebunden und jeweils, sobald sich der Datensatz verändert, auch angepasst an das DOM weitergegeben. So passen sich die Grafiken ständig dem Datensatz an.

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

(<https://ogobrecht.com/de/posts/2015-03-01-d3js-grundlagen/>)

Daten können aus unterschiedlichen Ressourcen und Dateiformaten geladen werden. Damit D3 die Datensätze verwenden kann, müssen die Daten einfach jeweils zuerst in Arrays umgewandelt werden.

🡪 stärken / schwächen

<https://d3js.org/>

<https://en.wikipedia.org/wiki/D3.js>

<https://www.dashingd3js.com/d3-tutorial/adding-a-dom-element-using-d3-js>

<https://www.dropbox.com/s/4ygbsitfcxfm5r0/D3.js-By-Example.pdf?dl=0>

## 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.[[6]](#footnote-6) Wie einige andere Libraries auch, verwendet React ausserdem ein virtuelles DOM. Durch dieses Konzept kann der Entwicklungsprozess beschleunigt und flexibler gestaltet werden.[[7]](#footnote-7)

Um verwirrende Fehler und unvorhersehbares Verhalten von Komponenten zu vermeiden, sollte darauf geachtet werden, *pure functions* zu schreiben. D. h. die Funktion kümmert sich nur um ihre eigenen Angelegenheiten und verändert keine Objekte oder Variablen, die bereits vor ihrem Aufruf existieren. Eine Ausnahme sind dabei Eventhandler. Da sie nicht während des Renderns laufen, müssen sie nicht pure sein und können Änderungen an Variablen innerhalb eines Events vornehmen.6

Damit Daten zwischen den Renderings von Komponenten geändert werden können, müssen sie irgendwo gespeichert werden. Dafür werden States verwendet. States sind eine Art «Gedächtnis» einer Komponente. Wenn React merkt, dass sich ein solcher State verändert hat, wird die ganze Komponente neu gerendert und die Änderung wird sofort sichtbar. Geändert werden die States mit speziellen Funktionen, sogenannten Hooks, die nur während des Renderings verfügbar sind.6

Ein weiterer wichtiger Bestandteil von React sind Props, auch Properties genannt. Props werden der Funktion mitgegeben und dienen dem top-down Informationsfluss zwischen einzelnen Komponenten.6 Da React einen unidirektionalen Datenfluss (One-Way Data Binding) verwendet, können Daten nur in eine Richtung direkt geändert werden, und zwar top-down. Für den bottom-up Informationsfluss zwischen Komponenten werden sogenannte Callback-Funktionen verwendet, wodurch die übergeordneten Komponenten indirekt geändert werden können.7 Der Informationsfluss zwischen Komponenten findet zwischen der jeweils über- oder untergeordneten Komponente statt, es können grundsätzlich keine Komponenten übersprungen werden.6

React hat verschiedene Vorteile. Durch die Verwendung das Virtual DOM Konzepts profitiert React von einer verbesserten Leitung 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.[[8]](#footnote-8) 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.[[9]](#footnote-9) Fazit: React eignet sich besonders für interaktive UI-Projekte.

## Vergleich D3.js und React

Beide Frontend, beide reagieren sofort auf Änderungen der zu Grunde liegenden Daten.

Anwendungsfokus verschieden:

D3 = Datenvisualisierung mit Fokus auf DOM, hierarchisch

React = Erstellung von webbasierten GU mit Fokus auf modularen, wiederverwendbaren Komponenten

Datenverarbeitung:  
React: one-way-data-flow, top-down. Wenn Properties an einen Komponenten gegeben werden, können diese nicht mehr verändert werden.

D3: Binding geht in beide Richtungen: DOM-Elemente können auf zugehörige Daten zugreifen und sie verändern.

React hat states, d3 nicht.

Da sich die beiden Libraries jedoch nur um die Sachen im Vordergrund kümmern und nicht um Routing, Serverkommunikation, ... braucht man noch andere Frameworks wie z. B. Next.js, damit man eine ganze Applikation bauen kann.

<https://medium.com/@jcukier/d3-and-react-similarities-and-differences-6aeb49fdaf5b>

1. https://kinsta.com/blog/javascript-libraries [↑](#footnote-ref-1)
2. https://www.reddit.com/r/explainlikeimfive/comments/tt6h5c/eli5\_what\_is\_a\_framework [↑](#footnote-ref-2)
3. https://stackoverflow.com/questions/3057526/framework-vs-toolkit-vs-library/3057818#3057818 [↑](#footnote-ref-3)
4. https://web.dev/choose-js-library-or-framework/ [↑](#footnote-ref-4)
5. https://web.dev/js-libraries-vs-frameworks/ [↑](#footnote-ref-5)
6. <https://react.dev/learn> [↑](#footnote-ref-6)
7. <https://kinsta.com/de/blog/vue-vs-react/#react-hauptmerkmale> [↑](#footnote-ref-7)
8. <https://www.wilde-it.com/react> [↑](#footnote-ref-8)
9. <https://kinsta.com/de/blog/vue-vs-react/#nachteile-von-vue-vs-react> [↑](#footnote-ref-9)