JavaScript Libraries

# Einführung

## Was sind Libraries

Möglichst einfach ausgedrückt ist eine Library eine Code-Sammlung wiederkehrender / mehrfach verwendeter Funktionen und Methoden. Dies 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 existierenden Funktionen weithin zu verbreiten.

In der Regel werden innerhalb einer Library Funktionen mit einem gemeinsamen Anwendungsfokus gebündelt (Bsp.

<https://www.khanacademy.org/computing/computer-programming/html-css-js/using-js-libraries-in-your-webpage/a/whats-a-js-library>

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

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

https://www.reddit.com/r/explainlikeimfive/comments/tt6h5c/eli5\_what\_is\_a\_framework/

## Unterschied zu Frameworks

Ein Framework hingegen stellt einen bereits vordefinierten Rahmen, eine grundlegende Software-Architektur und besitzen bestimmte Design-Templates und eigene Funktionen, welche wiederum in einer Library zusammengefasst sind. Innerhalb dieses Rahmens wird durch den eigenen Code die gewünschte Applikation eingebaut, welche die beinhalteten Libraries aufrufen kann.

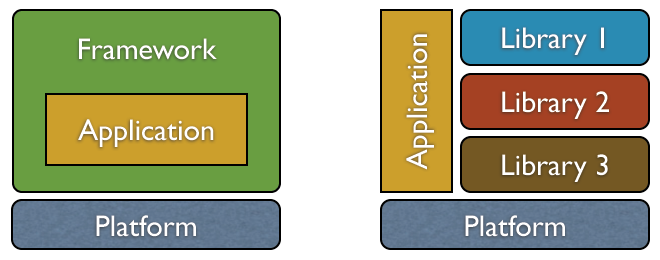


Abbildung : https://tom.lokhorst.eu/2010/09/why-libraries-are-better-than-frameworks

Der grösste Unterschied zwischen Libraries und Frameworks liegt in der «Inversion of Control». Dies bezeichnet die Art und Weise wie der Code aufgerufen wird.



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

https://stackoverflow.com/questions/3057526/framework-vs-toolkit-vs-library

<https://web.dev/js-libraries-vs-frameworks/>

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

<https://stackoverflow.com/questions/724438/what-is-the-difference-between-a-class-library-and-a-framework3>

## Einsatzgebiete

<https://web.dev/choose-js-library-or-framework/>

# Libraries

## 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. Das Spezielle an React ist der starke Fokus auf 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.

Um verwirrende Fehler und unvorhersehbares Verhalten von Komponenten zu vermeiden, sollte man darauf achten *pure functions* zu schreiben. D. h. die Funktion kümert sich nur um ihre eigenen Angelegenheiten und verändert keine Objekte oder Variablen, die bereits vor ihrem Aufruf existieren. Bei gleichen Eingaben sollte eine *pure function* immer dasselbe Ergebnis zurückgeben. Eine Ausnahme sind dabei Eventhandler. Da sie nicht während des Renderns laufen, müssen sie nicht pure sein und können deshalb Änderungen vornehmen. Damit Daten geändert werden können, müssen sie jedoch erst 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.

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. Für den bottom-up Informationsfluss zwischen Komponenten werden sogenannte Callback-Methoden verwendet. Der Informationsfluss zwischen Komponenten kann nur zwischen der jeweils über- oder untergeordneten Komponente stattfinden, es können grundsätzlich keine Komponenten übersprungen werden.

Hooks 🡪 wichtig oder evtl. Weglassen?

Quelle: <https://react.dev/learn>

Stärken und Schwächen von React …

## Vergleich D3.js und React