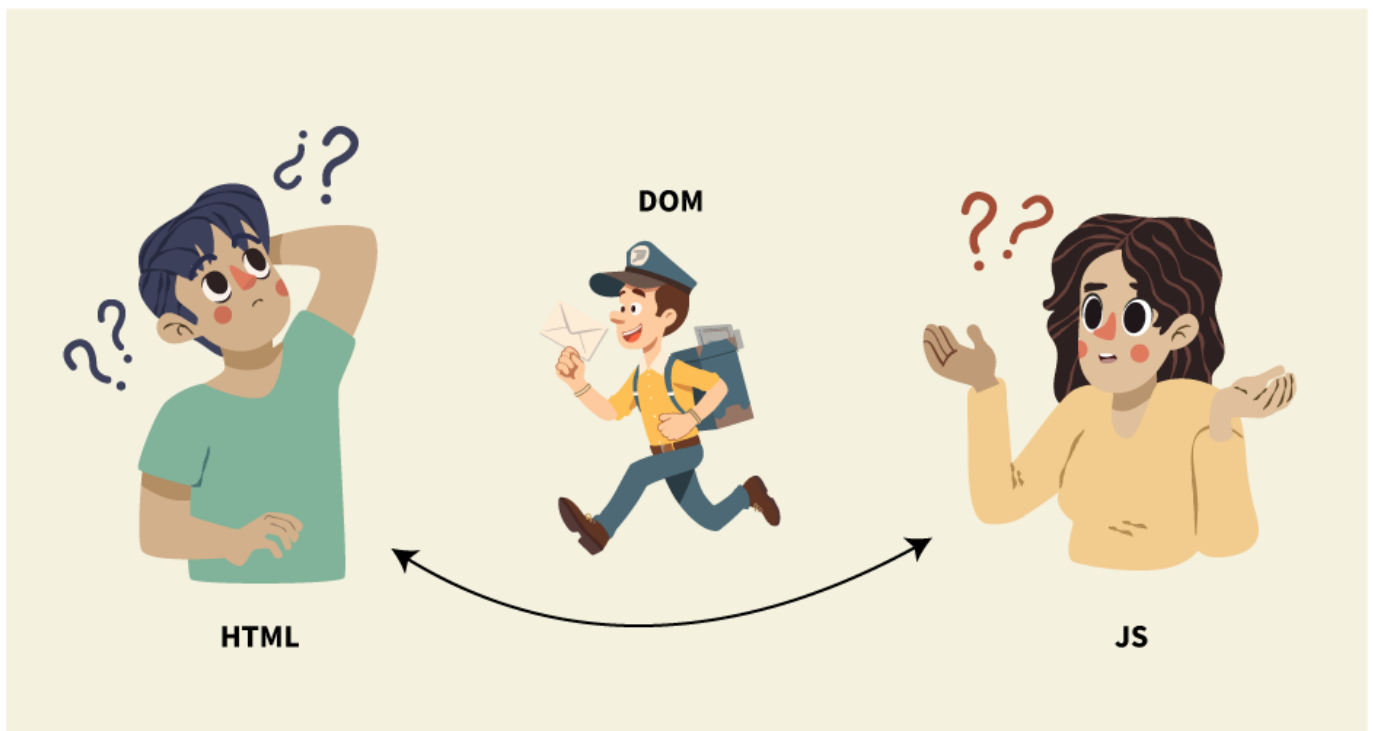


LU09a - Vue.js: HTML und JavaScript, eine Liebesgeschichte

Lernziele

- Sie verstehen, was reaktive Variablen sind, wie `ref()` funktioniert und warum wir sie brauchen.
- Sie kennen die wichtigsten Vue-Direktiven: `v-bind` und `v-on`.
- Sie können JavaScript-Werte direkt im HTML-Template mit `{{ }}` ausgeben.
- Sie sind vertraut mit der Struktur einer Vue-Komponente (`<script setup>`, `<template>`, `<style>`).

Eine schwierige Beziehung



Stellen Sie sich vor: HTML und JavaScript sitzen im selben Raum, aber sie sprechen nicht miteinander. JavaScript muss jedes Mal den Umweg über den DOM nehmen – wie eine Nachricht, die über mehrere Umwege zur Empfängerin gelangt.

```
// JavaScript sucht sich sein Element mühsam zusammen...  
const panel = document.querySelector('.panel');  
  
// ...und greift dann von aussen ein  
panel.classList.add('open');  
panel.setAttribute('aria-expanded', 'true');
```

HTML weiss nichts von JavaScript. JavaScript weiss nichts von HTML. Das funktioniert – aber es ist schwerfällig, fehleranfällig und wird mit jedem weiteren Baustein unübersichtlicher.

Die Liebesbeziehung: Wie Vue.js die beiden zusammenbringt



Vue.js kommt ins Spiel und bringt die beiden näher zusammen. HTML und JavaScript sprechen plötzlich **direkt** miteinander. Das Template ist kein stummes HTML mehr – es ist angereichert mit JavaScript-Logik, direkt dort wo sie gebraucht wird.

Video: Einführung in Vue.js

[Video Einführung Vue.js](#)

Zuschnitt (5 Min.) aus dem YouTube-Video von Loris Galler: *«Alles was du über Vue.js wissen musst!»*

Das Video erklärt die Grundlagen von Vue.js: Was ist ein JavaScript-Framework, wie funktioniert der komponentenbasierte Ansatz, was bedeutet Reaktivität – und

wie sorgt Data Binding für dynamische Benutzeroberflächen.

ref() - Daten, die Vue beobachtet

```
const isOpen = ref(false);
```

Mit `ref()` wird eine Variable **reaktiv**. Sobald sich ihr Wert ändert, weiss Vue sofort, welche Teile des Templates (=HTML) neu gerendert werden müssen - ganz ohne dass Sie selbst mit `querySelector` nach Elementen suchen.

: (v-bind) - HTML-Attribute an JavaScript-Werte binden

```
<!-- Langform -->  
<div v-bind:class="{ open: isOpen }">  
  
<!-- Kurzform – das gebräuchlichere -->  
<div :class="{ open: isOpen }">
```

Ohne den Doppelpunkt wäre `class="{ open: isOpen }"` ein gewöhnlicher Text-String. Mit dem Doppelpunkt wird der Ausdruck rechts als JavaScript ausgewertet - `isOpen` entscheidet direkt, ob die Klasse gesetzt wird oder nicht. Das ersetzt das manuelle `classList.add()` und `classList.remove()`.

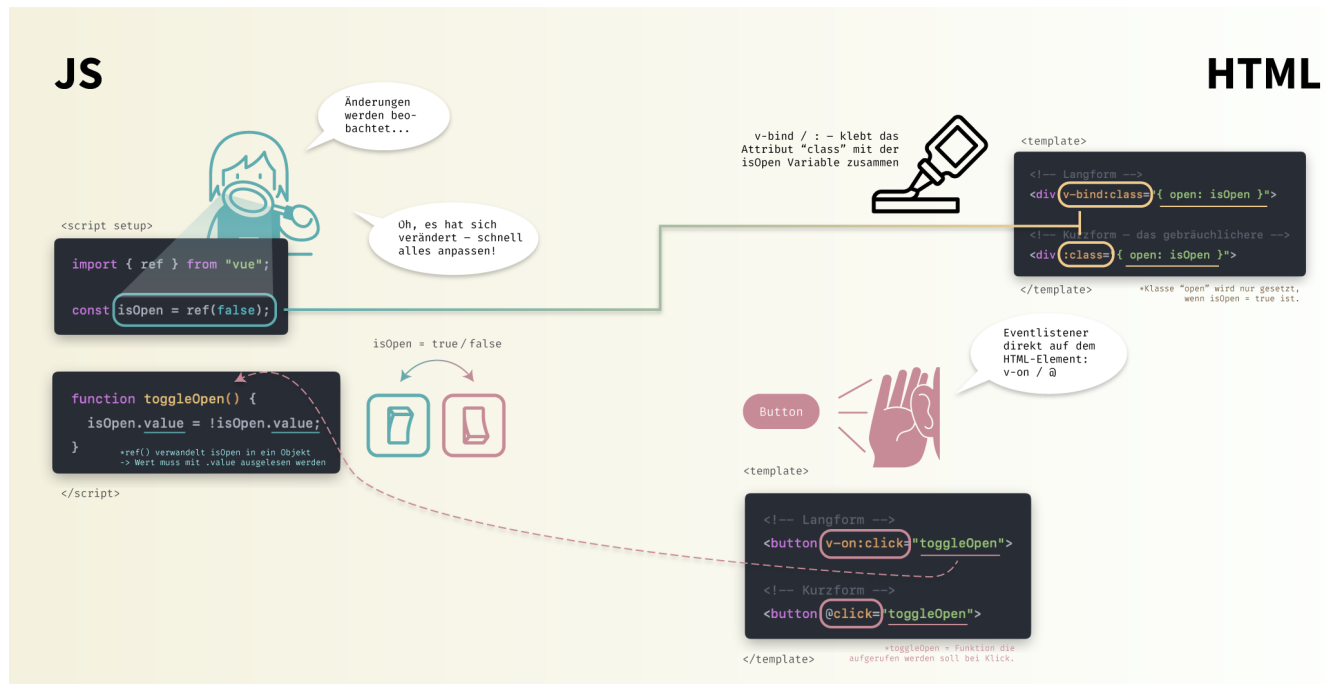
Das funktioniert mit **jedem** HTML-Attribut:

```
:href="url"  
:disabled="isLoading"  
:aria-expanded="isOpen"
```

@click (v-on) - Events direkt am Element deklarieren

```
<!-- Langform -->  
<button v-on:click="toggleOpen">  
  
<!-- Kurzform -->  
<button @click="toggleOpen">
```

@ ist die Kurzform von v-on : . Anstatt in JavaScript erst querySelector aufzurufen und dann addEventListener zu setzen, schreibt man das Event direkt am Element im Template - dort wo es passiert, dort wo es hingehört.



{{ }} - Ein Fenster direkt ins JavaScript

```
<h1>{{ faqTitle }}</h1>
```

Doppelte geschweifte Klammern betten einen JavaScript-Wert direkt ins HTML ein. Ändert sich der Wert der ref() -Variable, aktualisiert sich die Anzeige automatisch - ohne manuelles element.textContent = ...

Moustache / Schnauz: {{ faqTitle }}

JavaScript Werte direkt im HTML



Das Muster dahinter: Von getrennten Welten zu einer einzigen

Alle diese Konzepte folgen derselben Idee:

Vanilla JS	Vue.js
HTML und JS sind voneinander getrennt	HTML und JS sind direkt verbunden
JS sucht das Element (querySelector)	Binding direkt im Template (:, @)
Zustand wird aus dem DOM abgelesen	Zustand ist in ref() gespeichert - einer klaren Variable
Manuell aktualisieren	Automatisch durch Reaktivität

ref() ist das Fundament - sie macht die Daten reaktiv. {{ }}, : und @ sind die Verbindungen, über die diese Daten direkt mit dem HTML kommunizieren. Zusammen ergibt das eine kohärente Einheit, in der Struktur und Logik nicht mehr getrennt sind.



Merksatz: In Vanilla JS ¹⁾ geht JavaScript zu HTML. In Vue geht HTML zu JavaScript. Das ist der Paradigmenwechsel - und der Grund, warum die Beziehung plötzlich so viel einfacher wird. ☐

1)

Vanilla JS bezeichnet reines JavaScript ohne Frameworks, Libraries oder zusätzliche Tools - nur das, was der Browser von Haus aus versteht. Der Begriff «Vanilla» kommt aus dem Englischen und steht für «pur» oder «ohne Extras» - wie Vanilleeis ohne Toppings.

From:
<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/de/modul/m291/learningunits/lu09/theorie/a_html_js_relationship

Last update: **2026/05/10 20:26**

