Das kleine Buch der HTML-/CSS-Frameworks

Jens Oliver Meiert

Das kleine Buch der HTML-/CSS-Frameworks

Jens Oliver Meiert

Dieses Buch wird verkauft unter http://leanpub.com/html-css-frameworks

Diese Version wurde veröffentlicht am 2024-11-01

Dies ist ein Leanpub-Buch. Leanpub bietet Autoren und Verlagen, mit Hilfe von Lean-Publishing, neue Möglichkeiten des Publizierens. Lean Publishing bedeutet die wiederholte Veröffentlichung neuer Beta-Versionen eines eBooks unter der Zuhilfenahme schlanker Werkzeuge. Das Feedback der Erstleser hilft dem Autor bei der Finalisierung und der anschließenden Vermarktung des Buches. Lean Publishing unterstützt den Autor darin ein Buch zu schreiben, das auch gelesen wird.

Inhaltsverzeichnis

Einleitung

Frameworks prägen das Bild der Webentwicklung der letzten zehn Jahre. Fast jeder Entwickler hat mal ein Framework eingesetzt oder setzt eins ein, und von diesen haben die meisten auch einen Favoriten (Bootstrap?); einige Entwickler haben dazu bereits ein Framework (mit)entwickelt oder entwickeln eins.

Zu einer Zeit, als nicht jedes Stylesheet gleich einen Namen und das Suffix »-Framework« erhielt, hatte ich für meinen Hausverlag O'Reilly 2015 alles zusammengetragen, was mir zur Entwicklung und dem Gebrauch von Frameworks einfiel. Dieses Wissen fußte vor allem auf meiner Erfahrung als Architekt der internen Frameworks von GMX (2003–2006) und Google (2008–2013).

Das Ergebnis war etwas, mit dem ich recht zufrieden war – als jemand, der durchaus mal Dinge »husch-husch« herunterschreibt, ist das tatsächlich nicht mit jedem meiner Werke der Fall. Da O'Reilly sich strategisch aber neu ausgerichtet hatte und das Original, *The Little Book of HTML/CSS Frameworks*, nicht für den deutschen Markt vorsah, bat ich 2018 um die Rechte, das Büchlein selbst auf Deutsch herauszubringen. Diesem Wunsch wurde entsprochen, und ich konnte mit Übersetzung und punktuellen Verbesserungen beginnen.

Was Sie in der Hand halten, ist in erster Linie immer noch eine Übersetzung der 2015er Originalausgabe. Aber auch wenn diese hier und da mal hakt – die Jahre gingen nicht spurlos am Material vorüber, und mein Deutsch, siehe spätere Bemerkungen, ächzt auch hier und da –, handelt es sich dabei, ähnlich wie bei *CSS Optimization Basics*, um ein »lebendes Buch«, und so werden Sie, wenn Sie dieses Buch z.B. über Leanpub oder Google Play Books bezogen haben, Aktualisierungen erhalten. Wenn Sie selbst Berichtigungen oder Verbesserungen vorschlagen wollen, sind diese jederzeit sehr willkommen und können in das Buch einfließen.

Mehr, nun, vom Jens der früheren Jahre. Viel Vergnügen.

Einleitung zur Originalausgabe

Dieser Tage basieren viele kommerzielle Websites auf Frameworks, und viele private Websites setzen ebenfalls Frameworks ein. Was aber sind Frameworks, warum und wann benötigen wir sie, und wie gebrauchen und bauen wir sie am besten?

Dieses kleine Buch dreht sich um Frameworks, die auf HTML und CSS basieren. Es konzentriert sich auf HTML und CSS, weil diese auch heute und auch weiterhin am Herzen jedes Webprojekts liegen. Die Prinzipien, die aufgezeigt werden, können jedoch auch auf andere Arten von Frameworks angewandt werden.

Das Ziel dieses Buchs ist, robuste und grundlegende Ideen zu Frameworks zu teilen, wobei etwas Spezifität für langfristigeren Nutzen eingetauscht wird. Wir könnten all die Frameworks analysieren,

die dort draußen gerade rumschwirren, aber wie nützlich wäre eine solche Review, wenn es darum geht, sich zu entscheiden, was man für ein Framework benutzen oder schreiben möchte – in fünf Jahren?

Während das Buch versucht, alles abzudecken, kratzt es so einiges aber auch nur an. Webentwicklung ist ein großes Feld geworden. Dazu, wie wir in Kürze sehen werden, dreht sich Framework-Entwicklung um Maßschneidern, und Maßschneidern wiederum hängt von den Umständen ab. Wir kennen die Situation nicht um jedes Projekt da draußen, und so können wir nicht alles verallgemeinern. Und wenn auch in einfacher Sprache geschrieben, richtet sich dieses Buch an Expertenentwickler, die Leute, die darüber entscheiden, ob und wie ein Framework verwendet werden soll, oder ob eins selbst entwickelt wird.

Es ist dabei ebenso von einem Webentwickler geschrieben worden. Ich, Jens, habe während meiner Laufbahn Frameworks für OPEN KNOWLEDGE, GMX, Aperto mit ihren Regierungs- und Geschäftskunden sowie für Google entwickelt. In dieser Zeit ist es mir nicht gelungen, die Schnelllebigkeit unserer Branche zu überlisten, aber es sind mir einige Prinzipien, Methoden und Praktiken über den Weg gelaufen, die zu haltbarerem Code führen. Davon profitierten die Frameworks, an denen ich arbeitete, und ich hoffe, davon werden Sie durch dieses Buch ebenfalls profitieren.

Anmerkung zur Übersetzung

Da der Kern dieses Buchs auf dem basiert, was ich ehemals für O'Reilly auf Englisch geschrieben habe, handelt es sich trotz einiger Ergänzungen und Neuerungen in weiten Teilen immer noch um eine Übersetzung. Wo es mir auffiel, habe ich Angleichungen vorgenommen, damit sich das Buch runder liest, aber aus Erfahrung mit dem Umgang mit zwei oder gar drei Sprachen (ich spreche auch etwas Spanisch, und das schlecht) weiß ich, dass manch eigenartige Formulierung solche Rückübersetzungen trotz vieler Schleifen überlebt. Über Verbesserungsvorschläge freue ich mich sehr.

Allgemein benutze ich in meinen Veröffentlichungen gerne ein kollektives »wir«. In diesem Buch fiel es mir als etwas *sehr* einschließend und umfassend auf, aber ich blieb in Schreibweise und Tonart nah am Original. Schließlich, ein Grund für diese Betonung der Gemeinschaft, *sind wir* hier technisch interessiert, und sehr wahrscheinlich eint uns sogar, dass wir nicht nur Entwickler, sondern Webentwickler sind. Denn das »wir« will nicht sagen, dass wir unbedingt alles gleich sehen – sondern dass wir ähnliche Berufungen und Leidenschaften haben.

Ebenfalls zur Sprache: Ich spreche oft vereinfachend von »Webentwicklern« und benutze auch in anderen Fällen durchaus die jeweils sprachliche kürzere Form. Damit beziehe ich ganz ausdrücklich *alle* ein, im Fall von Webentwicklern also sowohl Webentwicklerinnen als auch Webentwickler. Ich betrachte es als Selbstverständlichkeit, dass wir andere nicht anders behandeln oder ausschließen, weil diese ein anderes Geschlecht, eine andere sexuelle Orientierung, eine andere Religion oder andere Merkmale oder Vorlieben haben, die keinen Schaden für einen selbst oder andere Menschen bedeuten.

Danksagung

Der Dank der ursprünglichen Fassung dieses Büchleins galt Tony Ruscoe, Asim Janjua, Eric A. Meyer, Simon St.Laurent, Meg Foley, Kristen Brown und Julia Tang. Er gilt ihnen noch heute.

Schlüsselkonzepte

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Frameworks verstehen

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Was ist ein Framework?

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Warum Frameworks?

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Arten und Einsatzfälle von Frameworks

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Beliebte Frameworks

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Attribute guter Frameworks

1. Ein Framework sollte maßgeschneidert sein

2. Ein Framework sollte benutzerfreundlich sein

3. Ein Framework sollte erweiterbar sein

Frameworks benutzen

Ein Framework auswählen

Die zwei Grundregeln für den Gebrauch eines Frameworks

1. Befolgen Sie die Dokumentation

2. Überschreiben Sie keinen Framework-Code

Frameworks entwickeln

Prinzipien

Prototyp

Qualitätsmanagement

Wartung

Aktualisierungen

Dokumentation

Logistik

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Bekannte Probleme

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Mangel an Disziplin

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Mangel an einem Prototypen

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Mangel an Wartung

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Mangel an Genauigkeit

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Mangel an Mut

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Zusammenfassung

Dieser Inhalt ist in der Leseprobe nicht verfügbar. Das Buch kann bei Leanpub unter http://leanpub.com/html-css-frameworks gekauft werden.

Nachwort

Den Abschluss dieses Buchs bildet ein Artikel, der einige meiner konkreten Erfahrungen mit der Entwicklung von Frameworks widerspiegelt, und zwar der als Ideengeber und leitender Entwickler der damals ersten Google-Web-Frameworks Go (2008) und Maia (2012). Der Bericht ist eine leicht modifizierte Variante der Fassung, die Sie auch unter meiert.com/de/publications/articles/20171005 finden.

Was ich beim Entwickeln von Googles Frameworks gelernt habe

In *The Little Book of HTML/CSS Frameworks* (überarbeitet) habe ich eine Übersicht und viele Praxistipps zu Frameworks veröffentlicht; ich halte das Buch für eines meiner besten Werke, auch wenn nicht jeder seinen Empfehlungen zustimmt (in den meisten Fällen sollten wir externe Frameworks vermeiden, weil sie die Vision bzw. das erste Paradigma der Webentwicklung verletzen).

Nirgendwo habe ich jedoch bisher über die Erfahrung gesprochen, einige große Frameworks entwickelt zu haben – weder über das 2004er Framework (auch wenn wir es damals nicht so genannt haben) von GMX und Partner-Sites wie 1&1, was wahrscheinlich zur ersten europäischen Milliarden-Besucher-Website führte, die auf validem HTML und vollständig entkoppeltem CSS basierte, noch über die 2008er und 2012er Frameworks, die ich bei Google baute, die letztlich auf Zehntausenden Google-Seiten verwendet und damit einige Milliarden Mal aufgerufen wurden. Was folgt, erzählt einen Teil der Google-Geschichte.

Diese Geschichte sollte mit dem Zustand von Googles informationsbezogenen Websites in 2008 beginnen, als einige qualitätsbewusste Webentwickler, einer von ihnen ich, bei Google anfingen, zu arbeiten. Gelinde gesagt: Es war nicht schön, und wir erhielten viel Kritik von einer Vielzahl angesehener Entwickler (dies machte dann einen der Gründe aus, warum es spannend war, sich dem Webmaster-Team anzuschließen).

Von außen, und so einige andere Googler waren sich dessen bewusst, bestand die Unschönheit hauptsächlich in der schlechten Code-Qualität von Googles Websites und Produkten. Die Google-Homepages bekam dabei die meiste Aufmerksamkeit, aber auch bei Produkten, die sonst gut ankamen, wurden Qualitätsprobleme von Fachleuten nicht übersehen.

Von innen, und von hier an werde ich nur noch von Google-Websites (und nicht mehr Produkten) sprechen, war klar, dass uns effektivere Prioritäten, Prozesse und – noch wichtiger – Werkzeuge fehlten; zu diesem Zeitpunkt wurden die meisten von Googles Websites händisch erstellt und gewartet.

An diesem Problem arbeitete das Team, speziell das Kernteam in Mountain View sowie ein paar Entwickler in Zürich, um ein Content-Management-System zu implementieren, das auch in Google-Dimensionen funktionierte. Wir veröffentlichten regelmäßig Websites und Einstiegsseiten in Dutzenden Sprachen, und unsere Code-Basis, HTML-Dokumente mitsamt anderem Beiwerk, umfasste schon 2008 eine sechsstellige Zahl von Dateien.

Eine Herausforderung wie auch Gelegenheit war die Diversität der Sites und Seiten, die wir erstellten. Ja, es gab eine Google-Kernidentität und Google arbeitete an einem unternehmensweiten Styleguide (die UX-Organisation um Irene Au hatte sich dem angenommen) – aber viele unserer Websites, zumindest Elemente darauf, waren einzigartig und einmalig. Diese Einzigartigkeit machte einen Grund aus, warum es noch keine solide technische Basis gab, und ich begann proaktiv, das erste Google'sche HTML-/CSS-Framework zu entwickeln: Go (es ging der Programmiersprache voraus, dessen Team der Namenskonflikt entging).

Go

Go ist speziell, was schlicht den einmaligen Umständen geschuldet war. Go ist, und selbst in minifizierter Form kriegt man davon einen Eindruck, ein kompaktes Framework. Das ist *by design*. Aber wie wurde es entworfen, und warum? Dazu sollten wir nochmal auf Googles Landschaft an Websites blicken – ein Blick, der tatsächlich Pflicht ist, wenn wir Frameworks entwerfen (aber auch benutzen).

Besagte Landschaft bestand aus einem international verteilten Team – dem Webmaster-Team – mit bunt gemischten internen Kunden, die von Googles Corporate Communications über Legal bis hin zu Marketing reichten, das eine große Zahl von unterschiedlichen und lokalisierten – bis zu 40, manchmal 50 Sprachen – Websites betrieb. Das erschuf einiges an Einzigartigkeit. Gleichzeitig geschah alle Arbeit unter dem Schirm von Google, einige Designaspekte wurden entsprechend von allen Seiten geteilt.

Sie nehmen sicher schon wahr, dass dieser übereinstimmende Teil klein war, und dass das, und nicht die hohe Zahl von Unterschieden, einen zentralen Teil von Gos Vision ausmachen musste: Go sollte nicht versuchen, »alles« abzudecken, was wir taten, sondern *gut an der Basis funktionieren*. Den kleinsten gemeinsamen Nenner finden und abbilden, das war die Idee.

Das nächste Prinzip hinter Go war *Einfachheit*, da diese Einfachheit große Gewinne versprach: Zum einen hat Einfachheit eine große Zahl großer technischer Vorteile, von höherer Geschwindigkeit über größere Robustheit bis hin zu leichterer Wartbarkeit. Zum anderen führt Einfachheit zu guter *Entwicklerbenutzerfreundlichkeit* (Entwicklerfreundlichkeit?) mitsamt, etwas das für die Arbeit an Go relevant war, einer besseren Chance auf Akzeptanz und Annahme.

Diese Chance resultierte dann in etwas, das ich auswendig illustrieren möchte, fünf oder sechs Jahre, nachdem ich die letzte Go-Seite erstellt habe; man konnte ein HTML-Dokument wie folgt schreiben:

```
<!DOCTYPE html>
<title>Google Test</title>
<link rel=stylesheet href=/css/go.css>
<h1><a href=/><img src=/logo alt=Google></a> Test</h1>
<p>Das ist nur ein Test [mit validem HTML].
```

Ebenso auswendig hervorgekramt, Go setzte dann so wenig Klassen und IDs wie möglich ein – und zwar die folgenden: `.rtl`, `.compact`, `#about`, `#nav` und `#aux`. Die Grundseitentypen waren ganze (aber dennoch limitierte) Breite, ganze Breite mit Navigation, kompakte (schmale) Seite und kompakte Seite mit Navigation. Die Zentrierung wurde durch die `.compact`-Klasse erzielt (auf `body`); das Hinzufügen einer Navigationsliste (`#nav`) erzeugte »automatisch« die Navigationsseitentypen; &c. Alle Seitentypen waren liquid und inhärent mobil-bereit. Das nur, um einen Eindruck von der Simplizität und Benutzerfreundlichkeit zu geben. (Als HTML 5 noch keimte und es noch keine logischen Eigenschaften in CSS gab, beruhte `#about` auf meiner Präferenz gegenüber `#footer` (und `<footer>`), und `#nav` spiegelte das sich zu dem Zeitpunkt noch in Planung befindliche `nav`-Element wider.)

Wie befriedigte Go dann die vielen Bedürfnisse, die in den Projekten mit unseren internen Kunden aufkamen? Wie auch schon durchscheinen mag: Go tat das nicht. Eines der Designziele war tatsächlich, dies gar nicht erst zu probieren. Go definierte den Kern von Googles Websites auf eine Weise, die so einfach wie möglich gehalten wurde, und erkannte die Einzigartigkeit und systemische Nicht-Managebarkeit all der unterschiedlichen Google-Sites durch zwei andere Entscheidungen an:

1. das Definieren anderer populärer Seitenelemente (wie Formulare) in einer Bibliothekserweiterung, »Go X« (man importierte in diesem Fall go-x.css, was leicht zu merken war);
2. das Teilen der Verantwortung und das Überlassen von allem anderen an die Webentwickler, die an dem jeweiligen Projekt arbeiteten (so dass diese Go oder Go X in ihrem default.css-Projekt-Stylesheet importierten und ihr eigenes Ding machten).

Auf rasche Akzeptanz und Annahme hoffte ich nicht nur durch Benutzerfreundlichkeit – die Stylesheet-URL verinnerlichen und in ein einfaches HTML-Dokument mit Google-Logo packen –, sondern auch durch ein paar weitere Faktoren, die an qualitätsverwandte Logistik erinnern:

- ein Template-Generator (da dies prä-CMS war, half er dem Team, konsistente Go-Vorlagen zu erstellen);
- eine Mailingliste für Ankündigungen, auf der das ganze Team vertreten war;
- eine Mailingliste für die Framework-Entwicklung;
- Dokumentation;
- improvisierte Videos (Asim Janjua und mir machten diese besondere Freude);
- Code-Richtlinien, die die Verwendung steuerten.

Aus technischer Sicht halte ich Go für eines der elegantesten Dinge, die ich bisher entwickelt habe. Go war einfach, leicht, robust, erweiterbar, skalierbar und wartbar. Bis heute erscheint es basal und

unspektakulär, und genau das war sorgfältig überlegt und funktionierte außerordentlich gut. Fast ein Jahrzehnt später gibt es wenig, das ich ändern würde (ich würde neuere HTML- und CSS-Features einsetzen, um Go noch kompakter zu machen).

Bei alldem aber muss ich auch sagen, dass ich aus technischer und leitender Perspektive damit gescheitert war, größere Akzeptanz für Go zu finden. Ich hatte Fehler gemacht, andere erfahrene Entwickler im Team einzubeziehen und besser mit ihnen zusammenzuarbeiten. Ich hatte versäumt, Gos Vorteile näher zu erläutern und damit mehr Unterstützung in Team und Management zu gewinnen. Manche Reibung im Team und einige verlorene Zeit dabei, Go zum Standard zu machen, waren das Ergebnis davon. Ich musste lernen.

Maia

Vorwärts. Eine der dringendsten Fragen, die ich mir 2012 stellen musste, war, wann wir ein neues Framework bauen würden. (Wann würden Sie?) In meinem Kopf war das Denken bislang zu *aktualisieren, iterieren*, ohne Versionierung, weil Webdesign, ebenso Webentwicklung, ein Prozess ist – und trotzdem wich ich das erste Mal von dieser Position ab. Warum?

Als wir – immer noch Googles Webmaster-Team – an einem neuen Design für alle informationsbezogenen Google-Sites arbeiteten, war absehbar, dass dies Konsequenzen für unseren Einsatz von Frameworks haben würde. Als Leiter des Projekts mutete dies für mich erst nach einem Update für Go an. Ich änderte diese Ansicht jedoch angesichts folgender Überlegungen:

- Die *Struktur* unserer Anforderungen und unserer Seiten war dabei, sich stark zu ändern.
- Die *Komplexität* unserer Seiten würde sich vergrößern.
- Die Designphilosophie und Integrität von Go konnten bewahrt werden (was sicher nicht meinen vertrauensvollsten und teamorientiertesten Schritt bedeutete – ich befürchtete, dass ich einen wichtigen Erfolg und Verantwortung verlieren würde).
- Es konnte nützlich und legitim sein, ein leichtes Framework (Go) als *Fallback* in der Hinterhand zu haben.

Diese Punkte führten mich dazu, Bemühungen zu unterstützen und dann anzuführen, ein neues Google-Web-Framework zu entwickeln, ein Framework, dass ich »Maia« taufte (es ist ebenfalls noch in Gebrauch, obwohl ich nicht weiß, ob es gewartet und wie sehr es modifiziert wurde) – »Maia« wegen eines Faibles für griechische Mythologie: Maia, Mutter von Hermes.

Die Geschichte von Maia ist anders als die von Go, insbesondere dadurch, dass Maia eine offiziell abgesegnete Teamunternehmung war, mit einem Team von fünf Webmastern, für deren Arbeit ich verantwortlich war. Der Prozess und die Organisation waren ähnlich zu dem, das ich in *The Little Book of HTML/CSS Frameworks* beschreibe – angefangen bei der Motivation, eine zugeschnittene Lösung zu den eigenen (Design-)Problemen zu schaffen, über die Gründung eines Team, das über entsprechende Ziele und Werkzeuge verfügte, bis hin zur Kommunikation an eine nochmal größere Organisation. Ich glaube, dass *The Little Book of HTML/CSS Frameworks* ein nützliches Buch ist, um mehr über solche Entwicklung zu erfahren, aber es folgen hier die Eckpfeiler für die Arbeit an Maia:

- das Definieren von Framework-Designzielen und -Entwicklungsprinzipien;
- das Zusammenstellen und Organisieren eines kleinen Teams von Entwicklern und Designern (mit den großartigen Tony Ruscoe und Zacky Ma);
- das Aufsetzen eines funktionierenden Prototypen, der auch alle Framework-Dokumentation beinhaltete (eine wichtige und bewährte Vorgehensweise);
- das Einrichten von Mailinglisten für Entwickler und Benutzer, für wichtige Ankündigungen.

Abgesehen von den komplexeren visuellen und technischen Anforderungen war die Entwicklung von Maia *als Team* der größte Unterschied zur Arbeit an Go, und zusammen mit stärkerem Management-Support wohl der wesentliche Faktor dabei, weitreichende Akzeptanz zu gewährleisten und zu beschleunigen.

Es fällt auf, dass ich Maias Geschichte knapp halte, sie gar abrupt beende; aber genau der Umstand, dass Maia Teamarbeit war, lässt es mich bevorzugen, hierüber – wenn das möglich wäre – *mit* dem Team zu schreiben.

Erkenntnisse

Das Geschilderte ist schon mit einigen Lektionen versehen, aber dies ist, was ich betonen möchte.

1. Auch wenn ich das nicht direkt bei Google lernte – ich bin Idealist –, will ich es doch erwähnen, weil es mir wichtig erscheint: Nicht nur durch die Fokussierung und Segmentierung von Framework-Verantwortlichkeiten (wie bei Go und Go X getan) *ist es absolut möglich, zugeschnittene, effiziente und trotzdem benutzbare Frameworks zu bauen und zu warten.* (Das ist auch ein Kernpunkt in *The Little Book of HTML/CSS Frameworks*. Es gibt keinen Grund, unnötigen Code und schlechte Qualität hinzunehmen.)
2. Um eine hochwertige Lösung zu bauen, *ist Kommunikation wichtiger als Konsens.* Ich habe meine Probleme mit Demokratie als Entscheidungswerkzeug, da ich diese selten die beste Lösung erbringen sehe – und im Technischen gibt es oft so etwas wie ein »bestes«. Aber ich habe auch Probleme gehabt, als ich ein Team vernachlässigte, um einfach die Lösungen voranzutreiben, die nur ich für am besten erachtete. Eine Alternative ist, die beidseitige Kommunikation zu verbessern: dem Team zuhören, andere Experten aus dem Team involvieren, Unterstützung vom Management einholen, &c.
3. Dies kam gar nicht auf, als ich über Maia sprach, und ich widerspreche auch meinem vergangenen Selbst: *Es ist kritisch, zu vermeiden, ein zusätzliches Framework zu bauen.* Die Vision muss sein, zu iterieren und zu warten – und nicht »Fire and Forget«. Ich hatte nichts großartig probiert, nicht gekämpft; ich hatte den Gewinn mitgenommen, den ich mit Go hatte, und dann den bequemen Weg beschritten, halbherzig ein kleines Team zu führen, um Konfrontation innerhalb der Organisation zu vermeiden (Unternehmenspolitik hatte dann auch mich erreicht, so scheint es). Das ist ein sehr anderer Schnack so plötzlich, aber ich hatte den wohl technisch inferioren Pfad genommen, als ich uns zugestand, an einem anderen Framework zu arbeiten, und damit einiges von dem geopfert, von dem wir zuvor so profitierten und was Go auszeichnete.

Zum Abschluss wünsche ich, dass hier einige Dinge dabei sind, die für Ihre eigenen Framework-Entscheidungen nützlich sind, so dass Sie ernten können, was wir geerntet haben, ohne die Fehler zu machen, die ich gemacht habe. Selbst wenn ich diese lediglich anschnitt.

Mit herzlichen Wünschen an das alte Webmaster-Team. Die Arbeit mit euch allen war eine große Ehre und ein großes Vergnügen. Ich denke gerne daran zurück.

Über den Autor

Jens Oliver Meiert ist auf Leitung und Kommunikation in den Bereichen Webentwicklung und -design spezialisiert. Er war technischer Leiter in Unternehmen wie GMX, Aperto und Google, sowie verantwortlich für technische Kommunikation bei sum.cumo.

Jens ist Verfasser von knapp einem Dutzend Büchern (einige davon für seinen Hausverlag, O'Reilly) und Gastautor bei verschiedenen Magazinen, u.a. Smashing Magazine, A List Apart und heise.

Für mehr zu hochwertiger Webentwicklung folgen Sie Jens auf seiner Website (meiert.com) oder auf Twitter (@j9t).

Weitere Werke von Jens Oliver Meiert:

Upgrade Your HTML (2019–2024)

> The *Upgrade Your HTML* series is about one thing: Picking examples of HTML in the wild, and explaining how to make that code better. Kindly. Constructively. Thoroughly, as finding a balance between detail and brevity permits.

Erhältlich bei Amazon, Apple Books, Kobo, Google Play Books und Leanpub.

The Web Development Glossary 3K (2023)

> What is a BHO? CQRS? An EMD? What is Goanna? Hooking? Sharding? How about dynamic color, the phoenix server pattern, or the rules of ARIA? Covering more than 3,000 terms and concepts, and including explanations from Wikipedia and MDN Web Docs, *The Web Development Glossary 3K* provides an overview of web development unlike any other book or site.

Erhältlich bei Apple Books, Kobo, Google Play Books und Leanpub. (Testen Sie das Glossar online unter WebGlossary.info!)

The Little Book of Little Books (2021)

> *The Little Book of Little Books* consists of lovingly polished editions of *The Little Book of HTML/CSS Frameworks* (originally published in 2015), *The Little Book of HTML/CSS Coding Guidelines* (2015), and *The Little Book of Website Quality Control* (2016).

Erhältlich bei Amazon, Apple Books, Kobo, Google Play Books und Leanpub.

CSS Optimization Basics (2018)

> Are you unsure about your style sheets' quality, or whether you've maxed out your options? *CSS Optimization Basics* covers the necessary mindsets, discusses the main optimization methods, and presents useful resources to write higher-quality CSS.

Erhältlich bei Amazon, Apple Books, Kobo, Google Play Books und Leanpub.

On Web Development (2015)

> *On Web Development* bundles 134 articles and the last 11 years of technical writings by Jens Oliver Meiert (meiert.com). Freshly reordered and commented, the articles cover processes and maintenance, HTML and CSS, standards, as well as development and design in general; they include coding basics and principles, carefully scathing criticism, and tips and tricks and trivia.

Erhältlich bei Amazon.

Über *Das kleine Buch der HTML-/CSS-Frameworks*

Autor: Jens Oliver Meiert

Herausgeber: Frontend Dogma, c/o Jens Oliver Meiert, Apartado de correos 3, 36070 Pontevedra, Spanien

Kontaktieren Sie +34-610859489 oder info@frontenddogma.com für Fragen und weitere Informationen.

Folgen Sie Frontend Dogma auf Mastodon.

[1.3.38]