# Bessere Websites

# Bessere Websites

[expand title="mehr lesen…"]

## Prinzipien der Webentwicklung

# Bessere Websites

Jens Oliver Meiert

Professionelle Webentwicklung bedeutet mehr, als bloß funktionierenden Code zusammenzuhacken. Es lohnt sich, gelegentlich mal einen Schritt zurückzutreten und darüber nachzudenken, wie man zu besseren Ergebnissen kommt.

Nach rund 20 Jahren ist die Webentwicklung zu einem reifen Berufsfeld geworden. Permanente Veränderung ist dabei eine Konstante: Immer neue technologische Standards, Möglichkeiten und Praktiken haben professionelle Webentwickler begeistert, aber auch frustriert. Der Artikel auf Seite 38 erklärt, wie man die Performance moderner Websites analysiert und verbessert. Ab Seite 46 wird beschrieben, wie man durch die clevere Trennung von lesenden und schreibenden Datenbankzugriffen (Command Query Responsibility Segregation, CQRS) Web-Anwendungen erheblich beschleunigen kann

Doch es gibt grundlegende Prinzipien, die die Arbeit in der Vergangenheit geleitet haben und die auch in Zukunft als Leitfaden dienen können [a]. Als ein Entwickler, der an kleinen und großen Projekten gearbeitet, zu Standards beigetragen und zu empfohlenen Entwicklungspraktiken

publiziert hat, will ich hier eine konkrete Sicht auf diese Prinzipien liefern.

## Fokus auf den Nutzer

Auch auf die Gefahr hin, es zum tausendsten Mal zu sagen: Webentwicklung muss sich immer auf den Nutzer konzentrieren. Das ist aus gutem Grund ein Google-Prinzip, das sich jeder Entwickler zu eigen machen sollte. Einige Beispiele sollen die enge Verknüpfung zwischen Endnutzern und Entwicklern verdeutlichen. Barrierefreiheit zum Beispiel dreht sich ausschließlich um Interaktionsmöglichkeiten der Endnutzer, muss aber bei der Entwicklung immer mitgedacht werden. Auch Codeperformance wird normalerweise von der Endnutzerseite ermittelt. Tatsächlich sind höchstens Designfragen von dieser engen Verknüpfung ausgenommen: Usability-Tests betreffen Endnutzer, aber Entwickler nicht unbedingt.

Zwei Dinge helfen bei der Konzentration auf den Nutzer. Das eine ist das Auf- und Durchsetzen von Coderichtlinien. Einen guten Einstieg bieten styleguides.io, cssguidelin.es oder das kurze Buch „The Little Book of HTML/CSS Coding Guidelines“ des Autors [b]. Das andere ist ein klarer Qualitätsanspruch.

## Fokus auf Qualität

Der Fokus auf die Qualität ist, was den Experten vom Amateur trennt, den Profi vom Hobbyentwickler. Jeder kann einen Webauftritt zusammenhacken – das Internet ist voll mit den Ergebnissen –, aber wenige können eine qualitativ hochwertige Website bauen. Es geht darum, eine Einstellung für Qualität zu kultivieren, Qualitätsziele zu setzen und Werkzeuge zu implementieren, die Qualität messen oder verbessern [c]. Das ist alles leider leichter gesagt als getan.

Letztlich geht es um die Frage: Wie kann man beurteilen, wie gut die eigene Arbeit ist, und wie lässt sie sich verbessern? Der qualitätsbewusste, ambitionierte Entwickler mag sogar auf

Exzellenz abzielen und damit auf das Qualitätsprinzip noch eins draufsetzen. Unser Arbeitsfeld weist eine Tendenz zur Schludrigkeit auf, die sich vielleicht ablegen lässt, wenn wir uns zwischendurch auf Exzellenz besinnen.

## „Keep it simple“

Es gibt kaum ein besseres Prinzip als das der Einfachheit. Es entspringt der Beobachtung, dass weniger tatsächlich mehr ist. Auch wenn nicht alle Entwickler die Kraft von „keep it simple“ sofort durchdringen und sie effektiv einzusetzen wissen: Erst mal verstanden, ist diese Kraft nicht zu unterschätzen. Sie ergibt sich letztlich aus dem Wissen, was wirklich wichtig ist und was nicht. Wer die Dinge nicht einfach hält, sagt damit, dass alles wichtig ist; das mag manchmal zutreffen, ist aber meistens falsch.

Ähnlich dem Fokus auf Qualität ist auch das Einfachhalten abstrakt. Was wichtig ist, hängt vom Kontext und den Zielen ab. Die Dinge einfach zu halten, erfordert Fokus und Erfahrung. Letztlich helfen Fokussierung und das Sammeln von Erfahrung zusammen mit dem Wissen von Kontext und Zielen dabei, dem wichtigen Prinzip der Einfachheit gerecht zu werden.

## Langfristiges Denken (und Vermeidung von Hypes)

Ebenfalls auf der Liste bewährter Prinzipien befinden sich langfristiges Denken und Nachhaltigkeit. Auch das ist nicht ganz so einfach zu fassen – vielleicht funktionieren Prinzipien genau deshalb, weil sie zum Nachdenken über die eigene Arbeit zwingen? Langfristiges Denken bedeutet, Projekte in geplanter, nachhaltiger Manier anzugehen. Konkret heißt das:

– das Anstreben qualitativ hochwertiger Inhalte und Dienste (und die Angewohnheit, über Kampagnenseiten und Landingpages

hinauszublicken);

– das Schaffen benutzbarer und ansprechender Designs (und ihre stetige Verbesserung mittels iterativer Prozesse);

– das Schreiben von robustem Code und seine Wartung sowie das Vermeiden von Code, der mit großer Wahrscheinlichkeit kurz- und mittelfristig schon wieder hinfällig sein wird [d];

– das absolute und leidenschaftliche Ablehnen von „Fire-and-Forget“ [e].

Es gibt einen einfachen Weg, sich langfristiges Denken anzueignen und andere davon zu überzeugen: indem man genau notiert, wie teuer kurzfristiges Denken ist durch all die Extra-Arbeit, die es verursacht. Das führt dann auch gleich zum nächsten Prinzip:

## „Don’t repeat yourself“ (oder: jeder Code braucht Wartung)

DRY, „don’t repeat yourself“, ist vermutlich das wichtigste Prinzip für wartbaren Code. Es ist deshalb so wichtig, weil es die Menge an Code reduziert und die Zahl der Stellen vermindert, die man im Blick behalten muss [f]. Weniger Code bedeutet weniger Zeit, die auf Wartung verwendet werden muss.

Wiederholung lässt sich auf zwei Arten vermeiden: keinen Code duplizieren und keine Dateien kopieren. Wenn etwas bereits existiert, muss es einen Weg geben, es wiederzuverwenden. Das erfordert Selbstdisziplin, Bewusstsein, Fleiß und Automatisierung. Schwierig ist die Situation bei der automatisierten Vermeidung von Wiederholungen.

## Verantwortungsbewusst entwickeln

Was die Kampagne „Code Responsibly“ (vom Autor einstmals initiiert) schon auf Jahre bewirbt, stellt ein wichtiges Leitprinzip dar: Webentwicklung bedeutet Verantwortung [g].

Das heißt: ständiges Lernen, Achten auf Barrierefreiheit und Performance, semantisch korrekte Auszeichnung der Inhalte, Codevalidierung, Fokus auf Wartbarkeit, Zusammenarbeit mit anderen, gute Dokumentation, Achten auf Qualität sowie die Bereitschaft, anderen etwas beizubringen.

Verantwortungsbewusste Webentwicklung beginnt aber vielleicht schon mit dem Bekenntnis zu Qualität – und zu Professionalität. Dazu gehören Wissen und Fähigkeiten, Urteilsvermögen und Kompetenz sowie natürlich Integrität und angemessenes Verhalten.

## Den Überblick behalten

Überblick bedeutet, sein Arbeitsfeld, dessen Grenzen und auch seine Nachbarn zu kennen. Das Arbeitsfeld sollte durch Webstandards [h] und darauf aufsetzende Empfehlungen [i] gut markiert sein. Die Nachbarn umfassen alles, was der Nützlichkeit und Qualität von Sites und Apps dient: Design, Usability und User Experience, Barrierefreiheit, Psychologie und Mensch-Maschine-Interaktion sowie die Inhalte der Sites und Apps. Eine technisch gesunde Website ist unabhängig von ihren Inhalten, eine großartige Website kann diese Grenze unscharf machen.

Es gibt aber auch Grenzen, derer sich Entwickler gewahr sein müssen, insbesondere bei User-Agents. Webentwickler haben einige unnötige und deshalb eher blöde Abenteuer unternommen, wenn sie probiert haben, sich sämtlicher Barrierefreiheits- [j] und User-Experience-Probleme [k] anzunehmen. Wir müssen auch in der Lage sein, zu erkennen, wann etwas nicht mehr unser Problem ist.

Obwohl all diese Prinzipien eher abstrakt sind, muss man aufpassen, was „absolute Wahrheiten“ anbelangt: Oft kommt es halt doch darauf an. Diese abstrakten Prinzipien geben jedoch eine Richtung vor und helfen dabei, bessere Entscheidungen zu treffen. Welche das letztlich sind, liegt in der Hand des

Entwicklers: Webentwicklung hat immer einen Zweck. (odi) Jens Oliver Meiert ist Philosoph und Entwickler (Google, W3C, O'Reilly). Er ist spezialisiert auf maßgeschneiderten Code für komplexe internationale Websites.

## Bessere Web-Apps

- Jeremy Keith, Design Principles
- Jens Oliver Meiert, The Little Book of HTML/CSS Coding Guidelines
- Jens Oliver Meiert, The Little Book of Website Quality Control
- Einführung in Wartbarkeit
- The Problem of „Fire and Forget" in Web Design
- CSS, DRY, and Code Optimization
- Code Responsibly
- W3C Standards
- Web Fundamentals
- Joe Clark, When accessibility is not your problem
- window.scrollTo() or: When to Stay Clear of User Agents

[/expand]