ROADMAP

Dein Weg in die Web-Entwicklung

RIKE.DEV/
KURSE

Erfahre, welches Wissen du brauchst, um die Grundlagen der Web-Entwicklung zu erlernen. Egal, ob Quer-, Um- oder Neueinsteiger*in: Alles, was wichtig ist, findest du nach Themen gegliedert in diesem Dokument.

1. Grundlagen

Egal, ob du mit deinem Browser durch das Web surfst, an deinem Mobiltelefon durch Instagram oder Pinterest klickst oder mit deinem Auto die aktuelle Verkehrssituation prüfst: All dies wäre unmöglich ohne: **das Internet.**

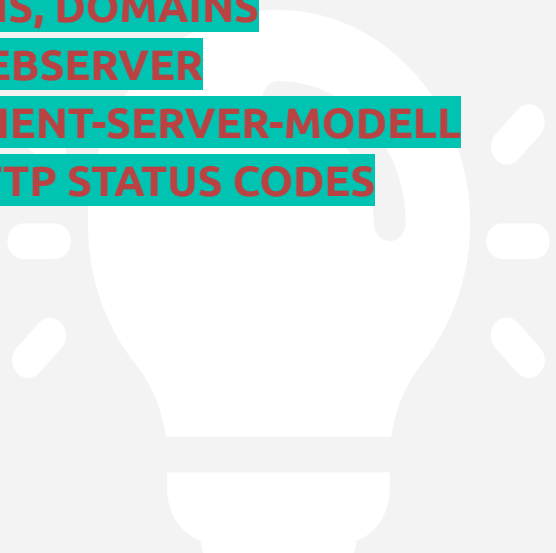
Zu wissen, wie Websites funktionieren oder sogar selbst welche entwickeln zu können, ist nicht nur sehr spannend, sondern auch fundamentaler Bestandteile vieler IT-Berufe und Businesses.

Bevor du dich also in all die damit verbundenen Technologien stürzt, ist es wichtig, das Internet als solches zu verstehen und wichtige Konzepte wie **URLs**, Protokolle wie **HTTP, TCP, IP** und **DNS**, **Webserver** und das **Client-Server-Modell** zu verinnerlichen.

Hierfür ist Interesse und Neugierde für technische Zusammenhänge nötig, aber sicher kein Informatikstudium!

KEY LEARNINGS

- INTERNET VS. WWW
- URLs
- PROTOKOLLE
- DNS, DOMAINS
- WEBSERVER
- CLIENT-SERVER-MODELL
- HTTP STATUS CODES



2. Developer Mindset & Tools

Der Alltag ist so vielfältig wie die Branchen und Anwendungsfälle, für die Web-Entwickler*innen tätig sind. Rund um die Web-Entwicklung hat sich eine Vielzahl an Tools etabliert.

Was macht eigentlich ein*e Web-Entwickler*in? Und welche Tools brauchst du für den Einstieg?

Egal ob du deinen eigenen Webshop aufbauen möchtest oder das nächste Spotify: Grundsätzlich brauchst du eine gewisse Grundkenntnis darüber, *wie Web-Entwickler*innen arbeiten und welche Tools und Programmiersprachen sie verwenden.*

Dabei ist es wichtig zu verstehen, dass ein Minimal-Setup für den Start völlig ausreicht, aber dass es sehr spannend ist, Schritt für Schritt weitere Tools kennenzulernen, die dir das Programmieren erleichtern werden.

Chat-Tools, Projekt-Management, **Versionsverwaltung** (Git) und **Entwicklungsumgebungen** machen vieles möglich, aber den Überblick nicht zu verlieren ist manchmal nicht ganz einfach!

KEY LEARNINGS

- MINDSET
- CLI
- BROWSER
- EDITOREN
- OPEN SOURCE
- STACK OVERFLOW
- VERSIONSVERWALTUNG / GIT
- KOMMUNIKATIONSTOOLS
- PROJEKTMANAGEMENT
- WEB STACKS

3. Webserver

Sie sind das Fundament des World Wide Web. Ihr Betrieb und ihre Vernetzung machen das Internet aus.

Ein **Webserver** ist ein Computer mit beliebigem Betriebssystem (also z. B. Linux), der mit dem Internet verbunden und mit Webserver-Software ausgestattet ist. Er ist dafür zuständig Dokumente und andere Dateien (Bilder, Videos, ..) an den Nutzer zu senden, der eine bestimmte **Web-Application** aufruft.

Wie heutzutage Webserver verwaltet und zugänglich gemacht werden, ist wichtiges Basiswissen auf dem Weg zur Web-Entwickler*in. Dafür solltest du selbst mal einen kleinen Webserver konfigurieren und starten.

KEY LEARNINGS

BETRIEBSSYSTEME
WEBSERVER-SOFTWARE

4. Frontend: Basics

Ein ansprechendes und funktionales Frontend ist das Tor zur Welt für jede Web-Application.

Das Frontend einer Web-Application ist das, was der Besucher deiner Website tatsächlich zu sehen bekommt, also die Benutzeroberfläche.

Mit der Weiterentwicklung der **Webbrowser** und den immer schnelleren Internet-Verbindungen in den letzten Jahren und Jahrzehnten, konnten **Benutzeroberflächen** im Web immer komplexer und dynamischer werden - sie fühlen sich heute oft eher an wie klassische Desktop-Programme, z. B. Microsoft Word.

Weiterhin ist es aber auch möglich ganz minimalistische Websites zu bauen. Hier ist es wichtig zu verstehen, welche Technologie sich für welchen Anwendungsfall am besten eignet.

KEY LEARNINGS

- HTML5
- CSS3
- CLIENTSEITIGE SKRIPTSPRACHEN
- JAVASCRIPT / JQUERY

Auf deinem Weg in die Webentwicklung wirst du dir mindestens **HTML** und CSS aneignen müssen. Ein großer Bonus und der Türöffner für dynamische und anspruchsvollere Web-Applications ist jedoch **JavaScript:** Kaum ein*e Frontend-Entwickler*in kommt um heute ohne diese Programmiersprache aus.

5. Backend: Basics

Was im Hintergrund passiert: Backend-Entwicklung ist für diejenigen spannend, die sich in die Software-Entwicklung vertiefen möchten.

Das Backend einer Web-Application bleibt für die meisten Besucher einer Website unsichtbar.

Es ist eine Software, die auf einem Webserver betrieben wird und bei etwas komplexeren Anwendungen sehr wichtige Aufgaben übernimmt: Hast du dich mal gefragt, wer eigentlich das eingegebene Passwort prüft, das du auf einer Website eingegeben hast? Wo die richtigen Daten aus einer Datenbank gefiltert werden, wenn du in einem Webshop nach roten statt nach blauen T-Shirts suchst? Oder wie Facebook berechnet, wer wen über wen kennt?

Solche **rechenintensiven, sensiblen und datengestützten Vorgänge** kann der Webbrowser meist nicht selbst übernehmen - sie werden vor der Anzeige im Frontend zuvor vom Backend durchgeführt.

KEY LEARNINGS

SERVERSEITIGE (SKRIPT-) SPRACHEN
DATENBANKEN

Für das Backend kommen ganz unterschiedliche Programmiersprachen in Frage. Denkbar sind z. B. **PHP, Python, C#, Ruby, Perl, Java, Go** und ein alter Bekannter: **JavaScript.**

Vor allem für Datenjongleur*innen bietet das Backend spannende Tätigkeitsfelder.

6. Web Frameworks

With a little help from my friends:
Selten schreiben Web-Entwickler*innen ihren kompletten Code selbst.

Dir wird auch schon aufgefallen sein: Viele Websites ähneln sich in ihrer Grundstruktur.

Im **Frontend** existieren häufig ein Header mit Logo des Unternehmens und Zugriff auf Social-Media-Kanäle, ein Haupt- und Untermenü, ein Footer und immer wiederkehrende Elemente wie Buttons, Formulare (z. B. Kontaktformulare) oder Tabellenansichten.

"Versteckt" üben auch **Backends** immer wieder ähnliche Aufgaben durch: Die Registrierung und Authentifizierung von Nutzer*innen, das Bereitstellen von Daten oder das automatisierte Versenden von E-Mails.

Meist werden Frameworks für eine bestimmte Sprache oder auch einen bestimmten Zweck programmiert. Sie sind eine Art "**Werkzeugkasten**" oder "**Rahmenstruktur**". Die wichtigsten Frameworks zu kennen und auch einzubeziehen wird deine Web-Entwicklung auf ein neues Level heben!

KEY LEARNINGS

WEB FRAMEWORKS
LIBRARIES

Sind wir mal ehrlich: Die meisten Web-Entwickler*innen sind in der täglichen Arbeit ein wenig faul und greifen auf gewisse "Helferlein" zurück, die wiederkehrende Funktionalitäten bereits optimal und stabil implementiert haben - so müssen sie nicht jedes Mal von Vorne beginnen. Daher arbeiten einige Web-Entwickler*innen an sogenannten **Frameworks**, die von anderen Web-Entwickler*innen für ihre Projekte genutzt werden können.

7. Deployment

Bis hierher hast du vielleicht nur “lokal” entwickelt, d.h. an deinem eigenen Rechner: *Zeit online zu gehen, oder?*

Auf ins "richtige" Web: Jede selbst programmierte Web-Application soll natürlich irgendwann "deployed" d.h. ins öffentliche Web transferiert werden.

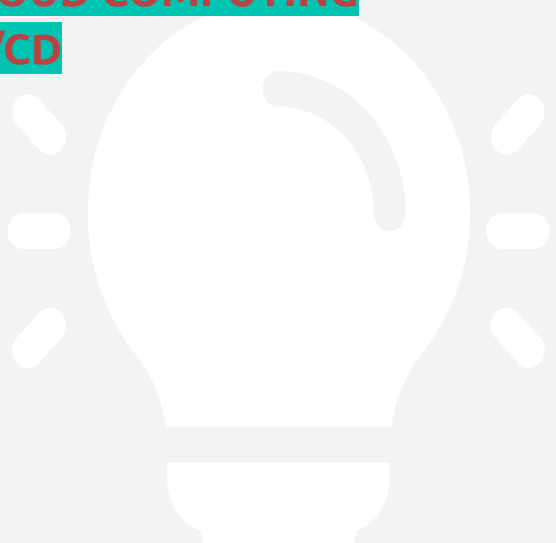
Die klassische Variante des **Webhosting** wird dabei gerade immer mehr durch **Cloud Computing**- und sogenannte "**serverless**"-Lösungen ergänzt.

Und wie für Software-Entwickler*innen üblich, etablieren viele von ihnen hochautomatisierte Prozesse, um das **Deployment** so einfach wie möglich zu machen: Das Stichwort hier ist **Continuous Integration** und **Continuous Delivery.**

In diesem Feld bewegen sich viele globale Internetkonzerne (Amazon AWS, IBM Cloud, Microsoft Azure) und noch mehr Buzzwords (z. B. Static Site Hosting, Software as a Service, u.a.), die es Wert sind, entschlüsselt zu werden.

KEY LEARNINGS

- STATISCHE VS. DYNAMISCHE WEBSITES
- WEBHOSTING
- FTP
- CLOUD COMPUTING
- CI/CD



8. Web Services und APIs

Datengestützte Webanwendungen bzw. Schnittstellen sind in Zeiten von *Big Data* unverzichtbar.

Web Services und APIs sind heute ein fundamentaler Bestandteil des Webs und dürfen daher in dieser Roadmap nicht fehlen.

Manche Web-Applikationen sind gar nicht an ein bestimmtes Frontend gekoppelt: Es gibt Firmen oder Entwickler*innen, die bestimmte *Dienste* oder Datentools anderen Anwendungen über eine *Schnittstelle* zur Verfügung stellen – diese Dienste werden dann **Web Services**, die Schnittstellen werden **APIs** (Application Programming Interface) genannt und folgen häufig dem Architekturstil **REST**.

Insbesondere um mit der Visualisierung von Daten zu experimentieren, lieben viele Web-Entwickler*innen solche Dienste oder schreiben sie einfach gleich selbst, um ihre Backend-Fähigkeiten zu erweitern.

KEY LEARNINGS

WEB SERVICES
APIs
REST-ARCHITEKTUR
JSON

9. Content-Management- & Baukastensysteme

Braucht es eigentlich noch Web-Entwickler*innen, wenn Lösungen wie Wordpress und jimdo existieren?

Nachdem du gelernt hast, wie du eine eigene Website von Grund auf selbst baust und ins Web bringst, stellt sich spätestens jetzt für viele neue Web-Entwickler*innen eine Frage: Kann ich nicht auch einfach einen Service wie jimdo, wix oder Wordpress verwenden und mir jede Menge Arbeit ersparen? Ja, kannst du, aber auch hier gilt: **Content-Management-Systeme** (wie z. B. Wordpress, Typo3, Joomla, usw.) und **Baukastensysteme** sind nur für bestimmte Anwendungsfälle empfehlenswert und schränken dich in deiner Gestaltungsfreiheit ein.

Trotzdem ist es wichtig zu wissen, welche "**No Code**" oder "**Low Code**"-Lösungen es auf dem Markt gibt und wie sie funktionieren.

KEY LEARNINGS

CONTENT-MANAGEMENT-SYSTEME
BAUKASTENSYSTEME
NO-CODE / LOW-CODE

10. Frontend: Advanced

HTML, CSS und JavaScript sind die Fundamente des Frontends. Wenn du diese gut beherrschst, wird es Zeit für fortgeschrittene Technologien.

Jetzt wird's professionell: Sobald du deine erste einfache Web-Application gebaut hast, steigen sicher auch deine Anforderungen.

Wenn die Basics und No-Code-Lösungen nicht mehr genügen, solltest du dir anschauen, warum heute frontend-seitig häufig **Single-Page-Applications (SPA)** in JavaScript zum Einsatz kommen und dich in eines der wichtigsten modernen Frontend-Frameworks einarbeiten: Denkbar wären **Angular, React** oder **Vue.js.**

In diesem Zuge wirst du unmittelbar weitere wichtige Werkzeuge einer Frontend-Entwickler*in kennen lernen: **Package managers, Task runners, Module Bundlers, Server-Side-Rendering** und **modernes CSS**. Innerhalb deines gewählten **SPA-Frameworks** solltest du vertieft in dessen Ökosystem einsteigen, um auch komplexere Web-Applications umsetzen zu können. Innerhalb des überaus populären Frameworks React sind dies beispielsweise Tools wie **Gatsby** und **Next.js**.

KEY LEARNINGS

- SINGLE-PAGE-APPLICATIONS
- REACT
- ANGULAR
- VUE.JS
- BUILD TOOLS
- ADVANCED CSS
- SERVER-SIDE-RENDERING VS. CLIENT-SIDE-RENDERING
- TYPESCRIPT

11. Backend: Advanced

Fortgeschrittene Backend-Entwicklung verlangt, dass du dich in eine Programmiersprache einarbeitest, in der Serveranwendungen üblicherweise programmiert sind.

Auch in der Backend-Entwicklung gibt es noch viele weiterführende Konzepte zu erlernen.

Ein häufiger Anwendungsfall ist die Programmierung eines Nutzerlogins oder eine REST-Schnittstelle. Um zu lernen, wie ein Frontend mit Daten (z. B. Kundendaten, Produktdaten, Wetterdaten, die Liste ist endlos...) versorgt werden kann, solltest du so eine kleine **Backend-Anwendung** selbst entwickeln.

Warum nicht bei der Programmiersprache bleiben, die wir bereits aus dem Frontend kennen? Eine Backend-Anwendung, die in JavaScript geschrieben ist und ein populäres Framework wie Express.js nutzt, kommt heute auch in großen Unternehmen zum Einsatz.

KEY LEARNINGS

- DATENBANKANBINDUNG
- AUTHENTICATION / AUTHORIZATION
- MICROSERVICES

Auch innerhalb der Backend-Entwicklung kommen häufig Web Frameworks zum Einsatz.

Viele Backend-Entwickler*innen entscheiden sich zunächst für eines von ihnen, beispielsweise **Django (Python), Ruby on Rails (Ruby), Symfony (PHP), Express.js (JavaScript), ASP.NET (C#)** oder **Spring Boot (Java)** und bauen ihr Wissen dann weiter aus.

12. Developer Workflows & Best Practices

Was gibt's sonst noch zu lernen?

Ganz zu Beginn dieser Roadmap haben wir bereits über die wichtigsten Tools von Web-Entwickler*innen gesprochen. Dieses Thema wird abgerundet durch **Best Practices** rund um Web Developer Workflows.

Ein gutes Webprojekt enthält beispielsweise eine Reihe an **automatisierten Tests**, die sicherstellen, dass bei einer Weiterentwicklung alte Implementierungen nicht einfach kaputt gehen. Viele Web-Entwickler*innen schreiben daher Tests, um ihre Code-Base abzusichern.

Ebenfalls wichtige Faktoren bei einem wirklich professionellem Webprojekt sind die Themen **Performance, Barrierefreiheit** und **Web-Security**.

Zudem solltest du dir die Themen **Containerisierung** mit Docker und Kubernetes anschauen: Zumindest von den Grundlagen sollte ein*e Web-Entwickler*in schon einmal gehört haben.

KEY LEARNINGS

- TESTING
- STAGING
- WEB-SECURITY
- PERFORMANCE
- LOGGING & MONITORING
- BARRIEREFREIHEIT
- DEVOPS
- DOCKER
- KUBERNETES

13. Trends & Inspiration

Keep in touch!

Wie geht's weiter?

Wenn du bis hierher gekommen bist, hast du dir bereits das wichtigste **Grundlagenwissen** angeeignet. Doch die Web-Entwickler*innen-Community steht praktisch nie still: Immer wieder gibt es neue "Rising Stars", neue Trends und Themen zu entdecken.

Um einen Einblick in aktuell heiß diskutierte Themen zu bekommen, ist es ebenso gut, die wichtigsten Newsletter, Aocial-Media-**Communities**, Influencer*innen, Websites und Portale zu kennen. Es gibt zudem eine große Anzahl an Videos, Tutorials und anderen Ressourcen rund um das Thema zu entdecken.

Und wenn du doch eine andere Richtung einschlagen möchtest, gibt es jede Menge **Berufe**, bei denen du mit deinem neu erlernten Grundlagenwissen punkten kannst: Online-Redakteur*in, SEO-Expert*in, Data Scientist, UX-Designer*in, IT-Projektmanager*in und viele mehr.

KEY LEARNINGS

JAMSTACK
PROGRASSIVE WEB APPS
GRAPHQL
CODE GENERATORS

WEB DEVELOPER COMMUNITIES & RESSOURCEN

VERWANDTE BERUFSGRUPPEN

Dir hat diese Roadmap gefallen?

Du willst direkt loslegen?

Schau mal vorbei auf **rike.dev/kurse** und buche jetzt deinen Anfänger- oder Fortgeschrittenen-Kurs!

Foto: unsplash.com/@wocintechchat
Design: slidesgo.com/theme/minimal-charm

E-Mail: kurse@rike.dev