Engineer

Progressive Enhancement

Erst mit reinem HTML nutzbare Seiten bauen, dann erweitern

Was es ist

Progressive Enhancement ist eine Methode, Webseiten schichtweise aufzubauen, beginnend mit der Schicht, die immer funktioniert. Die Grundlage ist HTML, das sämtliche Inhalte enthält und jede Aufgabe allein erfüllt: Links führen zu einem Ziel, Formulare senden an einen Server, und die Seite lässt sich lesen und benutzen, ohne dass etwas Weiteres geladen wird. Darauf folgen Stylesheets, die die Seite anordnen und angenehm gestalten. Skripte kommen zuletzt, um die Bedienung schneller oder flüssiger zu machen. Jede Schicht verbessert die darunterliegende; für ihre Aufgabe hängt die Seite von keiner Schicht oberhalb des HTML ab. Das GOV.UK Service Manual fasst die Idee so zusammen: Eine Seite zuerst mit HTML zum Funktionieren bringen.

Der Name stammt aus einem Vortrag von Steven Champeon und Nick Finck, *Inclusive Web Design For the Future with Progressive Enhancement*. Laut ihrem Aufsatz wurde er am 11. März 2003 auf der SXSW in Austin, Texas, gehalten. Der Aufsatz schreibt den Begriff als Progressive Enhancement mit Großbuchstaben und kürzt ihn mit PE ab. Er stellt ihn Graceful Degradation gegenüber, der älteren Gewohnheit, für den leistungsfähigsten Browser zu bauen und zu hoffen, dass die anderen weiterhin an die Inhalte kommen. PE beginnt laut dem Aufsatz mit einem Dokument, das die einfachsten Geräte nutzen können. Die Darstellung folgt als separate Logik, damit die einfachen Geräte nicht durch das belastet werden, was moderne Browser zusätzlich erhalten.

Der Unterschied liegt in der Arbeitsrichtung. Graceful Degradation beginnt mit dem vollständigen Erlebnis und fragt, was übrig bleibt, wenn Teile fehlen. Progressive Enhancement beginnt mit dem, was immer vorhanden ist, und fragt, was sich ergänzen lässt. In einem aktuellen Browser mit guter Verbindung können beide bei derselben Seite enden. Sie unterscheiden sich, wenn etwas

ausfällt: Eine von der Grundlage aufwärts gebaute Seite hat weiterhin ihre Grundlage. Eine von oben nach unten gebaute Seite hat das, woran ihre Autoren gedacht haben.

Es ist eine Arbeitsweise, keine Bibliothek und keine Funktion. In HTML, CSS oder JavaScript heißt nichts Progressive Enhancement, aber die Standards lassen Raum dafür. Ein Browser überspringt eine CSS-Deklaration, die er nicht versteht, und ignoriert ein Skript, dessen Typ er nicht kennt. Mit der Regel `@supports` kann ein Stylesheet prüfen, ob eine Funktion verfügbar ist, bevor es darauf baut.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Progressive Enhancement?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Oberflächen, die schnell laden, auf dem Smartphone funktionieren und sich in allen Browsern Ihrer Kunden gleich verhalten.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Full-Stack-GIS-Kartenentwicklung geleitet — Full-Stack-Kartenentwicklung von der Datenbank bis zum Browser

Full-Stack-GIS-Kartenentwicklung geleitet Kontakt aufnehmen →

Warum es wichtig ist

Die Schichten scheitern auf unterschiedliche Weise; das GOV.UK Service Manual beschreibt sie genau. CSS ist recht fehlertolerant: Ein Browser überspringt eine unbekannte Deklaration und macht weiter, obwohl ein unbekannter Selektor die gesamte Regel entfallen lässt. JavaScript ist dagegen überhaupt nicht fehlertolerant. Verwendet ein Skript eine Syntax oder Schnittstelle, die dem Browser fehlt, läuft der Rest dieses Skripts nicht. Eine Seite, die ihr Skript für die Hauptaufgabe benötigt, ist daher nur so zuverlässig wie jeder einzelne Schritt zwischen Server und Leser.

Davon gibt es viele. Dieselbe Seite zählt auf, was das Laden oder Ausführen von CSS oder JavaScript verhindern kann: ein vorübergehender Netzwerkfehler, ein Werbeblocker, ein ausgefallenes Content Delivery Network, ein DNS-Fehler, ein Fehler in einem Browserupdate oder eine Firewall im Unternehmen beziehungsweise auf dem eigenen Rechner. Nichts davon hängt mit einem alten Browser zusammen, und der Leser kann nicht erkennen, welche Ursache zutrifft. Er sieht einen Knopf, der nichts tut. Das Handbuch nennt keine Zahlen zur Häufigkeit der einzelnen Fälle; sein Argument beruht auf der Liste, nicht auf einem Prozentsatz.

Manche Leser schalten Funktionen auch absichtlich aus. Das Handbuch sagt, einige Nutzer deaktivierten bewusst Browserfunktionen, und diese Entscheidung solle respektiert werden. Der HTML-Standard hält fest, dass viele Nutzer Skripting deaktivieren. Deshalb und weil sich deklarative Funktionen oft leichter pflegen lassen, empfiehlt er Autoren, sie nach Möglichkeit anstelle von Skripten zu verwenden. Sein Beispiel ist das Element `details`, das einen Seitenabschnitt ohne eine Zeile JavaScript ein- und ausblendet. Außerdem fordert der Standard Autoren dazu auf, Anwendungen bei fehlender Skriptunterstützung geordnet auf einfachere Funktionen zurückfallen zu lassen.

Für britische Regierungsdienste macht das Handbuch daraus eine Regel: Sie sollen allein mit HTML funktionieren. Wo JavaScript verwendet wird, soll es HTML und CSS lediglich verbessern, sodass Menschen den Dienst bei einem Skriptfehler weiterhin nutzen können. Diese Regel gilt für die Dienste einer Regierung, nicht für das ganze Web. Ihre Begründung lässt sich aber auf jede Website übertragen, deren Autoren ihre Leser nicht auswählen können.

Auch das Team gewinnt. Wenn eine Erweiterung ausfällt, verliert der Leser die Verbesserung, nicht die Möglichkeit, seine Aufgabe zu erledigen. Ein Skriptfehler wird dadurch zur Unannehmlichkeit statt zum Ausfall. Eine neue Funktion lässt sich an Browser ausliefern, die sie unterstützen, ohne einen Plan für die anderen zu brauchen: Diese behalten die bisherige Seite.

Wie es in der Praxis aussieht

Das eigene Beispiel des HTML-Standards für ein geordnetes Verhalten ohne Skripte ist eine Tabelle, die sich über ihre Spaltenüberschriften neu sortieren lässt. Es eignet sich gut, um den Ansatz zu prüfen. Die Grundlage ist eine Tabelle mit gewöhnlichen Links als Überschriften. Ein Klick fordert vom Server dieselbe Tabelle, nach dieser Spalte sortiert, an; der Server liefert sie in dieser Reihenfolge zurück. Das funktioniert in jedem Browser, mit oder ohne Skript.

```
<table id="orders">
  <thead>
    <tr>
      <th><a href="?sort=date">Date</a></th>
      <th><a href="?sort=total">Total</a></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>2026-10-20</td>
```

```
      <td>120</td>
    </tr>
    <tr>
      <td>2026-10-24</td>
      <td>45</td>
    </tr>
  </tbody>
</table>
<script type="module" src="sort.mjs"></script>
```

Das Skript ist die Erweiterung. Es findet dieselben Links, unterbindet die Navigation und ordnet die Zeilen direkt auf der Seite neu. Eine Anfrage an den Server samt Antwort entfällt.

```
const table = document.querySelector("#orders");
const body = table.tBodies[0];
const collator = new Intl.Collator("en", { numeric: true });

function sortBy(column) {
  const text = (row) => row.cells[column].textContent;
  const rows = [...body.rows];
  rows.sort((a, b) => collator.compare(text(a), text(b)));
  body.append(...rows);
}

for (const link of table.tHead.querySelectorAll("a")) {
  link.addEventListener("click", (event) => {
    event.preventDefault();
    sortBy(link.closest("th").cellIndex);
  });
}
```

Fehler vor dem Anbinden eines Listeners lassen den jeweiligen Link funktionsfähig. Kommt `sort.mjs` nie an, arbeiten die Links wie zuvor. Bricht das Skript unterwegs mit einer Ausnahme ab, führen die Links der Spaltenüberschriften, für die noch kein Listener angebunden wurde, weiterhin zum Server. Ein Browser ohne Unterstützung für Modulskripte behandelt `type="module"` als unbekannten Skripttyp und ignoriert es, wie der HTML-Standard festlegt. Ein Modulskript ohne `async` wird ausgeführt, nachdem der Browser das gesamte HTML geparst hat. Die Tabelle ist also vorhanden, wenn das Skript nach ihr sucht. Dieses Beispiel kann nicht auf die normale Navigation zurückfallen, wenn ein bereits angebundener Klick-Handler nach dem Abbruch der Navigation durch `preventDefault()` eine Ausnahme auslöst.

Der Aufwand liegt auf dem Server: Die Grundlage braucht einen Endpunkt, der sortieren kann, und beide Sortierwege müssen übereinstimmen. Das ist der übliche, einmalig zu tragende Preis dieses Ansatzes. Der beabsichtigte Nutzen ist eine Seite, deren Hauptaufgabe über den Ausweichweg zum Server nutzbar bleibt. Ein angebundener Handler muss diesen Ausweichweg erhalten, wenn seine eigene Arbeit fehlschlägt.

Diese Website benennt ihre Grundlage ausdrücklich. Ihre Erklärung zur Barrierefreiheit beschreibt vollständige Seiten aus HTML, CSS und SVG ohne ausführbares JavaScript. Der Suchcode ist erhalten, aber deaktiviert; es gibt also keinen Filter innerhalb der Seite, von dem etwas abhängt. Die Erklärung nennt auch eine verbleibende Grenze: Die kleinen Beschriftungen beim Überfahren oder Fokussieren von Symbolen lassen sich nicht per Taste schließen. Dafür wäre Browsercode nötig, den die Website nicht ausliefert. Deshalb erfüllt sie 1.4.13, Inhalte bei Überfahren oder Fokussieren, nur teilweise.

Wo es schiefgeht

Ein `noscript`-Block ersetzt die Grundlage. Das Element `noscript` zeigt seinen Inhalt nur bei deaktiviertem Skripting. Der HTML-Standard formuliert deutlich, das noscript-Element sei ein grobes Werkzeug. Skripting kann aktiviert sein, während das Skript der Seite trotzdem ausfällt. Dann erhält der Leser weder die skriptgesteuerte Seite noch die Ausweichlösung. Beginnen Sie mit einer Seite, die ohne Skript funktioniert, und lassen Sie das Skript sie verändern. So macht es das eigene Beispiel des Standards: Sein Skript ergänzt ein Formular um ein laufend aktualisiertes Ergebnis und entfernt dessen Absendeknopf.

Eine nicht unterstützte Funktion stoppt das ganze Skript. Weil die JavaScript-Schicht nicht fehlertolerant ist, kann ein einziges neueres Syntaxelement in einem Bundle alle darin enthaltenen Erweiterungen außer Betrieb setzen, auch solche, die funktioniert hätten. Halten Sie die Grundlage vom Skript unabhängig, damit ein Fehler nur die Erweiterung kostet. Laden Sie modernen Code mit `type="module"`, das ältere Browser überspringen. Falls ältere Browser überhaupt eine erweiterte Version brauchen, geben Sie ihnen ein eigenes klassisches Bundle mit `nomodule`, das Browser mit Modulunterstützung nicht ausführen.

Ein neuer Selektor reißt gültige Regeln mit. Trifft ein Browser auf einen unbekannten Selektor, verwirft er die gesamte Regel einschließlich ihrer Deklarationen. Eine neue Pseudoklasse gemeinsam mit allgemein bekannten

Selektoren aufzulisten kann deshalb Stile auch aus Browsern entfernen, die diese sonst angewandt hätten. Geben Sie einem möglicherweise nicht unterstützten Selektor eine eigene Regel.

Stile der Ausweichlösung greifen in das erweiterte Layout ein. Breitenangaben für Browser ohne ein neueres Layoutsystem können diesem System in den Browsern entgegenwirken, die es besitzen. CSS Conditional Rules Level 3 zeigt die Lösung mit einem Flex-Layout und einem Block `@supports not (display: flex)` für die Alternative. Die Spezifikation weist darauf hin, dass die Breitenangaben das Flex-Layout beeinträchtigen könnten. Deshalb gehören sie ausschließlich in die Stile für Browser ohne diese Unterstützung.

Bilder werden per Skript ausgewählt. Der HTML-Standard weist darauf hin: Nennt eine Seite in ihrem Markup kein Bild und überlässt die Auswahl einem Skript, wird bei deaktiviertem Skripting überhaupt kein Bild geladen. Ersetzt ein Skript ein bereits im Markup benanntes Bild, kann der Browser zwei Downloads beginnen. Geben Sie die Kandidaten im HTML mit `srcset` und `sizes` oder mit `picture` an und überlassen Sie die Auswahl dem Browser.

Verhalten ohne Grundlage darunter. Die Strategieliste von Champeon und Finck fordert dazu auf, für jedes Verhalten eine Ausweichlösung vorzusehen: ein echtes `href` hinter jedem Klick-Handler und eine Prüfung auf dem Server hinter jeder Prüfung im Browser. Ein Bedienelement, das nur als Klick-Handler existiert, oder ein ausschließlich im Browser geprüftes Formular hat überhaupt keine Grundlage. Verwenden Sie zuerst echte Links und echte Formularübermittlungen, prüfen Sie immer auf dem Server und behandeln Sie die Prüfung im Browser als zusätzlichen Komfort.

Es gilt als Thema für alte Browser. Glaubt ein Team, die Grundlage sei nur für alte Browser da, lässt es sie weg, sobald diese verschwunden sind. Die Ursachenliste von GOV.UK handelt von Netzwerken, Blockern, Firewalls und fehlerhaften Updates; all das trifft auch aktuelle Browser. Erhalten Sie die Grundlage als genutzten und getesteten Weg: Laden Sie die Seite gelegentlich mit blockierten Skripten und führen Sie ihre Hauptaufgabe vom Anfang bis zum Ende aus.

Weiterführende Lektüre

- Progressive Enhancement einsetzen — die Anleitung des GOV.UK Service Manuals, zuletzt aktualisiert am 27. September 2024: die drei Schichten, ihre jeweiligen Ausfälle und warum Regierungsdienste allein mit HTML funktionieren sollen.
- HTML-Standard: Skripting — das Element `script` mit `type="module"` und `nomodule` sowie das Element `noscript` mit dem Rat, sich nicht darauf zu verlassen.
- CSS Conditional Rules Module Level 3 — der Candidate Recommendation Draft des W3C vom 15. August 2024, der `@supports` mit `not`, `and` und `or` definiert.
- HTML-Standard: Bilder — `srcset`, `sizes` und `picture` sowie der Grund, warum eine skriptgesteuerte Bildauswahl Leser mit deaktiviertem Skripting ausschließt.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Arbeitsablauf, die vorhandenen Schnittstellen und die Nutzer, die die Veränderung benötigen. Wir können eine prüfbare Lieferung definieren und die Übergabe so gestalten, dass Ihr Team die gelieferten Bestandteile warten kann.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Progressive Enhancement?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/progressive-enhancement/