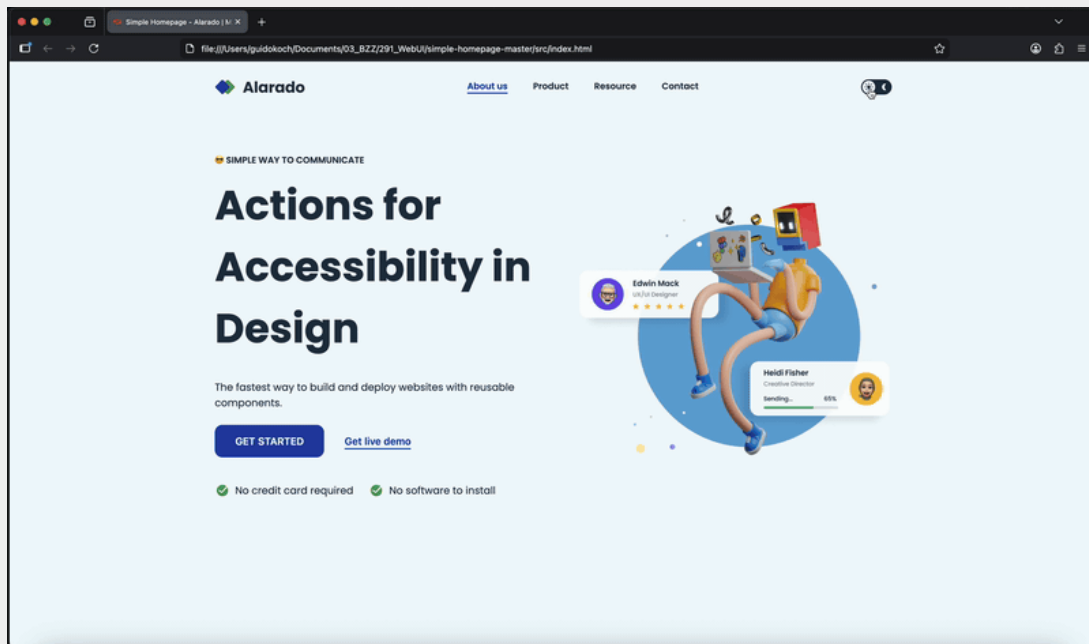


Projekt 1: Landingpage Hero Section (Light/Dark Mode) - LU01 (HTML + Grundlayout)



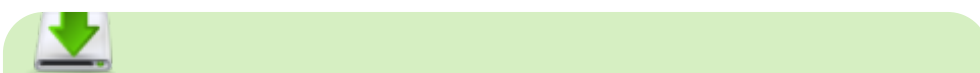
Projektziel (über mehrere LUs): Sie setzen eine Hero Section pixel-nah nach Figma um und bauen später einen Light/Dark-Mode Switch mit JavaScript.
Fokus LU01: Zuerst HTML-Struktur, danach CSS-Grundlayout (Container + Flexbox). Noch kein Feinschliff bei Fonts/Farben/Interaktionen.

Lernziele (LU01)

- Sie bauen eine saubere HTML-Struktur mit header, nav, main, section.
- Sie gruppieren UI-Teile in sinnvolle Wrapper (z.B. .content-wrapper, .button-wrapper).
- Sie setzen ein Grundlayout mit Flexbox um (Header + 2-Spalten Hero).
- Sie nutzen relative Einheiten sinnvoll (% , vh, rem) und zentrieren einen Container mit max-width.

Material

- Figma-Design + Assets (SVG/PNG)
- Starter-Template index.html (mit TODOs in LU01/LU02/LU03)
- Browser + DevTools + Live Server





Laden Sie hier das Material herunter:

landing_page_alarado.zip



Wichtig: Halten Sie sich an die TODO-Markierungen im Template. In LU01 machen Sie nur, was mit **TODO in LU01** beschriftet ist. (Google Font, Favicon, Farben/Variablen und Hover-Styles kommen später.)

Schrittweiser Aufbau (LU01)

1) HTML zuerst: Struktur & Gruppen von Elementen

1.1 Header: Logo + Navigation + Switch-Platzhalter

Aufgabe: Bauen Sie den `<header>` gemäss Template auf.

- Logo als `img` einfügen (+ sinnvoller `alt`)
- Navigation als `nav.menu` mit `ul > li > a`
- Switch als Platzhalter einfügen (nur `label + input type=„checkbox“`)

Repetition: Semantische Grundstruktur

1. header = Kopfbereich der Seite (Logo, Navigation, Controls)
2. nav = Navigation (Links als Liste)
3. main = Hauptinhalt (pro Seite 1x)
4. section = thematisch zusammenhängender Abschnitt (z.B. Hero)

Checkliste

- Ist die Navigation eine Liste (`ul/li`)?
- Gibt es pro Link ein `a`-Element (nicht nur Text)?
- Hat das Logo einen passenden `alt`-Text?

1.2 Hero Section: Inhalt links, Bild rechts

Aufgabe: Ergänzen Sie in `<section id=„hero-section“>`:

- Textblock: h3, h1, p
- Buttons in `.button-wrapper` gruppieren
- Features/Infos in einer Gruppe (z.B. `.info-wrapper`)

Checkliste

- Sind Buttons in einem Wrapper gruppiert?
- Sind "Features" in einem Wrapper gruppiert?
- Ist die Struktur logisch lesbar, auch ohne CSS?

—

1.3 Hero-Bild

Aufgabe: Fügen Sie das Bild gemäss Template ein. Nutzen Sie `srcset` für 1x/2x (Retina).

Repetition: `srcset` kurz erklärt

Mit `srcset` kann der Browser je nach Bildschirmauflösung die passende Datei wählen (z.B. @2x).

Sie liefern mehrere Varianten – der Browser entscheidet.

Beispiel: Bild mit `srcset`

```

```

Checkliste

- Stimmt der Pfad (`./resources/...`), und lädt das Bild?
- Haben Sie einen sinnvollen `alt`-Text?

—

2) CSS danach: Grundlayout

2.1 Container zentrieren: `main-wrapper`

Aufgabe: Setzen Sie den Hauptcontainer mittig und begrenzen Sie die Breite (gemäss Figma).

Repetition: Container-Zentrierung

- max-width begrenzt die Breite
- margin: 0 auto zentriert horizontal
- padding sorgt für Innenabstand (damit Inhalt nicht am Rand klebt)

Beispiel:

```
.main-wrapper {  
  max-width: 1100px; /* Richtwert – gemäss Figma anpassen */  
  margin: 0 auto;  
  padding: 24px;  
}
```

—

2.2 Header mit Flexbox

Aufgabe: Positionieren Sie Logo (links), Navigation (mitte) und Switch (rechts).

Repetition: Flexbox Basics

- display: flex → Elemente in einer Reihe
- justify-content: space-between → links/mittig/rechts verteilen
- align-items: center → vertikal zentrieren
- gap → Abstand zwischen Items

—

2.3 Navigation: Menu-Items als Flex-Reihe

Aufgabe: Lassen Sie die Menu-Items nebeneinander erscheinen (ohne Styling der Farben – das kommt in LU02).

—

2.4 Hero Section: 2-Spalten Flexbox

Aufgabe: Textblock links, Bild rechts – nebeneinander.

—

2.5 Bild soll responsiv schrumpfen

Aufgabe: Das Bild soll proportional kleiner werden, wenn das Fenster kleiner wird.

Repetition: Responsive Images

- width: 100% → Bild füllt den Container
- height: auto → Seitenverhältnis bleibt korrekt

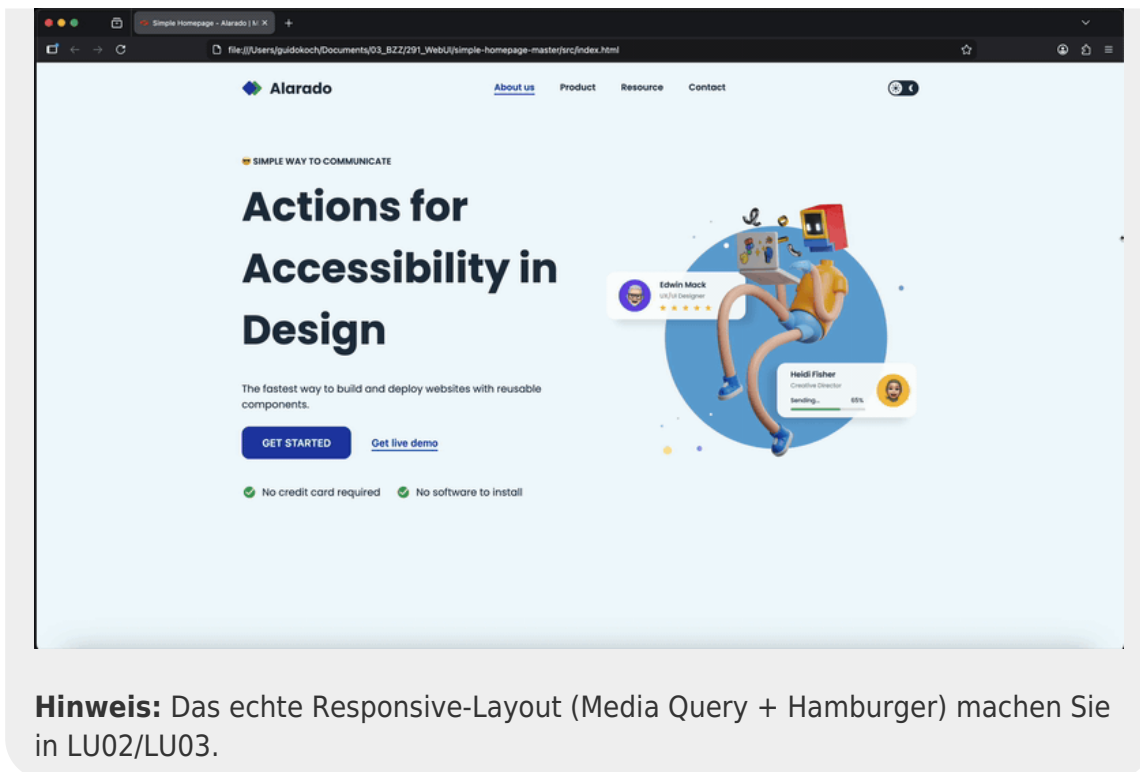
2.6 Buttons: als Gruppe mit Flexbox

Aufgabe: Buttons nebeneinander, mit Abstand. (Details wie Farben/Fonts in LU02)

2.7 Hamburger-Buttons (für LU01: ausblenden)

Aufgabe: Im Desktop sollen Hamburger-Buttons unsichtbar sein (damit nichts “stört”).

```
.hamburger-button,  
.hamburger-button-close {  
  display: none;  
}
```



Checkliste "LU01 fertig"

- HTML: Logo + Nav-Liste + Switch-Platzhalter sind vorhanden
- HTML: Hero-Text, Button-Gruppe, Info-Gruppe sind vorhanden
- HTML: Bild mit srcset lädt korrekt
- CSS: .main-wrapper zentriert mit max-width
- CSS: Header als Flexbox (3 Bereiche verteilt)
- CSS: Hero als Flexbox (2 Spalten)
- CSS: Bild schrumpft responsiv (width:100%, height:auto)

Debugging-Checkliste

- Öffnen Sie DevTools → Console (gibt es rote Fehlermeldungen?)
- Prüfen Sie Pfade zu ./resources/... (404?)
- Prüfen Sie im Elements-Tab: stimmen Klassen/IDs genau?
- Test: Browserfenster kleiner ziehen → Bild bleibt proportional

Abgabe (Ende LU01)

Zwischenabgabe: Commit mit Message z.B. LU01: HTML structure + basic flex layout.

Ausblick (LU02 / LU03)

- LU02: Google Font, Farbvariablen (':root'), Typografie, Link-/Button-States, Media Query
- LU03: Favicon, Icons via `::before`, Switch-Styling + JS für Theme-Toggle

From:

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

Permanent link:

https://wiki.bzz.ch/de/modul/m291/learningunits/lu01/aufgaben/a_landingpage?rev=1769380260

Last update: **2026/01/25 23:31**

