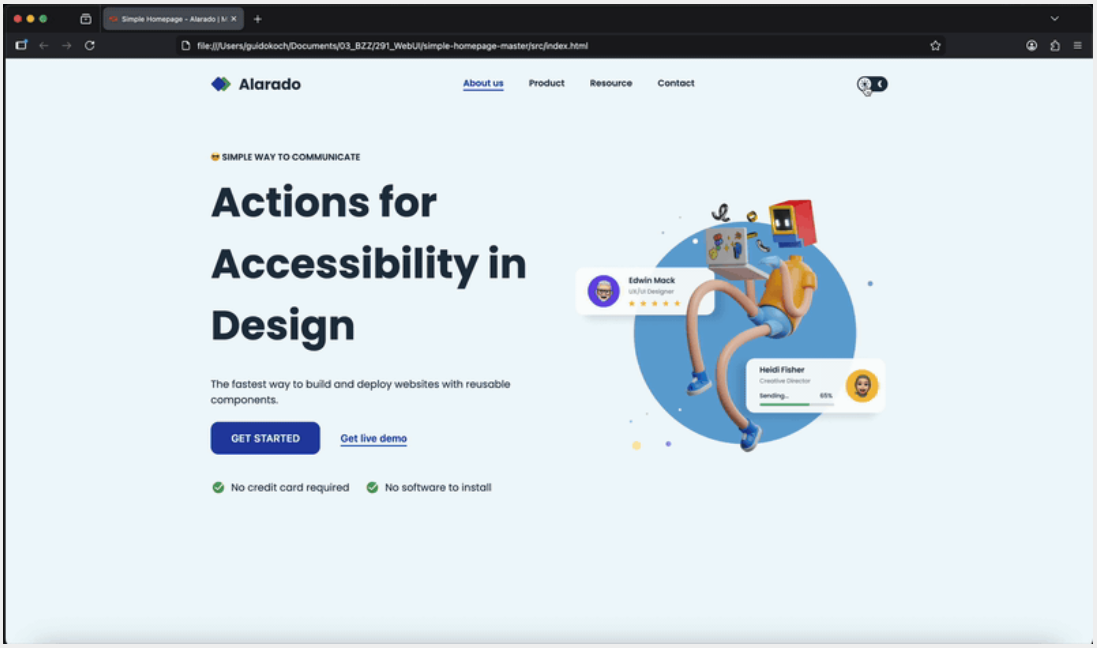
Projekt 1: Landingpage Hero Section (Light/Dark Mode) – LU01 (HTML + Grundlayout)



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.

→ Vorwissen: UI-Elemente lesen (Figma → HTML)

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

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 (ca. 35–45 Min.)

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
- `header` = Kopfbereich der Seite (Logo, Navigation, Controls) - `nav` = Navigation (Links als Liste) - `main` = Hauptinhalt (pro Seite 1×) - `section` = thematisch zusammenhängender Abschnitt (z.B. Hero)

Beispiel: Header-Grundgerüst (Sie passen Texte/Links an Ihr Design an):

```
<header>
  <img src="./resources/logo-light.svg" alt="Alarado Logo" id="logo" />

  <nav class="menu" aria-label="Hauptnavigation">
    <ul>
      <li><a href="#" class="active">About us</a></li>
      <li><a href="#">Product</a></li>
      <li><a href="#">Resource</a></li>
      <li><a href="#">Contact</a></li>
    </ul>
  </nav>

  <label class="switch">
    <input id="theme-toggle" type="checkbox" />
  </label>
</header>
```

Checkpoint (2 Min.)

- 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`)

Beispiel: Hero-Content-Wrapper

```
<section id="hero-section">
  <div class="content-wrapper">
    <h3> Simple way to communicate</h3>
    <h1>Actions for Accessibility in Design</h1>
    <p>The fastest way to build and deploy websites with reusable
components.</p>

    <div class="button-wrapper">
      <button class="highlight">Get Started</button>
      <button class="active">Get live demo</button>
    </div>

    <div class="info-wrapper">
      <span class="icon-ok">No credit card required</span>
      <span class="icon-ok">No software to install</span>
    </div>
  </div>

  <!-- Bild kommt in 1.3 -->
</section>
```

Checkpoint (2 Min.)

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

—

1.3 Hero-Bild: ''img'' mit ''srcset''

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`

```
<img
  class="hero-image"
  src="./resources/hero-image-simple-homepage.png"
  srcset="./resources/hero-image-simple-homepage.png 1x,
          ./resources/hero-image-simple-homepage@2x.png 2x"
  alt="Illustration zur Landingpage"
/>
```

Checkpoint (1 Min.)

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

—

2) CSS danach: Grundlayout (ca. 35–45 Min.)

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

Beispiel:

```
header {
  display: flex;
  align-items: center;
  justify-content: space-between;
}
```

—

2.3 Navigation: Menu-Items als Flex-Reihe

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

```
.menu ul {
  list-style: none;
  display: flex;
  gap: 24px;
}
```

—

2.4 Hero Section: 2-Spalten Flexbox

Aufgabe: Textblock links, Bild rechts – nebeneinander.

```
#hero-section {
  display: flex;
  align-items: center;
  justify-content: space-between;
  gap: 32px;
  margin-top: 48px;
}
```

—

2.5 Bild soll responsiv schrumpfen

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

> **Repetition: Responsive Images**
> - `max-width: 100%` → nie breiter als der Container - `height: auto` → Seitenverhältnis bleibt korrekt

```
.hero-image {
  max-width: 100%;
  height: auto;
  width: 520px; /* Richtwert – gemäss Figma */
}
```

—

2.6 Buttons: als Gruppe mit Flexbox

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

```
.button-wrapper {
  display: flex;
  gap: 16px;
  margin-top: 24px;
}
```

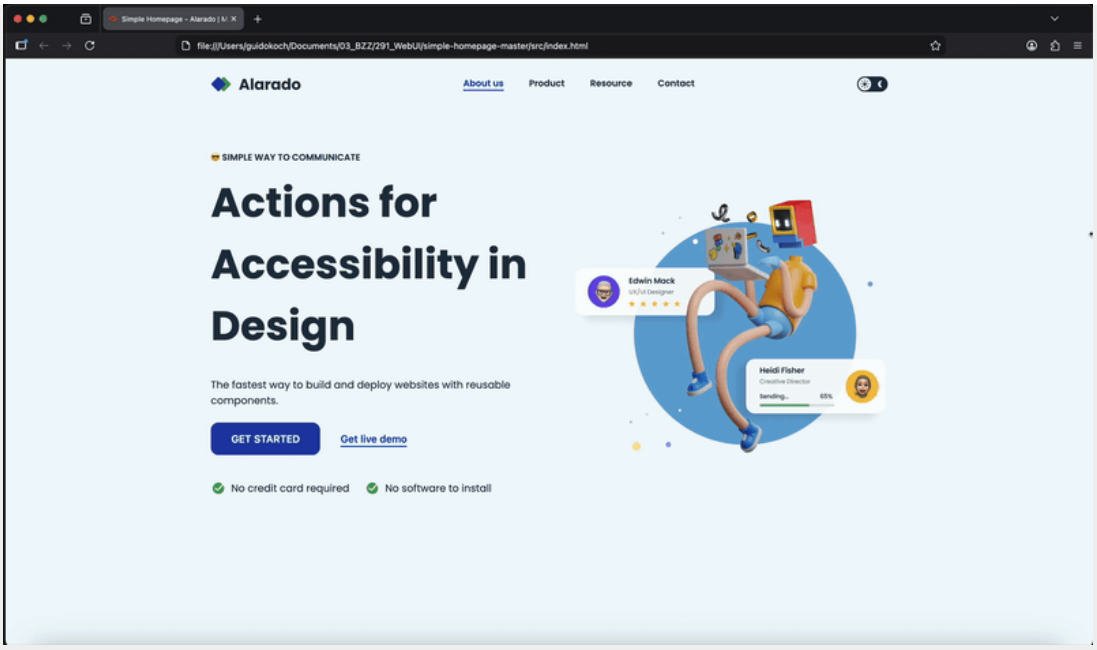
—

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;
}
```

—



Hinweis: Das echte Responsive-Layout (Media Query + Hamburger) machen Sie in LU02/LU03.

Mini-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 (`max-width:100%`, `height:auto`)
- Keine Fehler in der Console

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`.
Sie zeigen kurz im Browser: - Header korrekt angeordnet - Hero in 2 Spalten - Bild skaliert responsiv

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