LU03: Light Dark Mode Switch mit JavaScript

Ziel von LU03: Sie erweitern Ihre Landingpage um drei Dinge:

- Favicon verlinken
- Feature-Icons mit CSS Pseudo-Elementen umsetzen
- Light Dark Mode Switch stylen und mit JavaScript funktional machen

Wichtig: Ihr Code darf sich von den Beispielen unterscheiden. Entscheidend ist, dass Ihr Ergebnis im Browser korrekt funktioniert und Ihr Code verständlich bleibt.

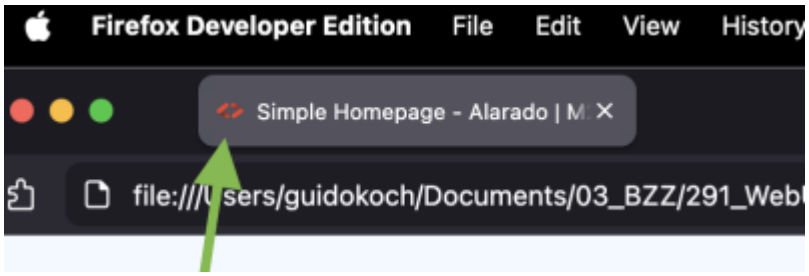
Voraussetzungen

Ihr Stand nach LU02 ist vorhanden und funktioniert im Desktop und Mobile Layout.

Hilfreiche Links

- MDN Favicon: rel="icon"
- MDN Pseudo-Elemente: ::before
- MDN classList: classList
- MDN addEventListener: addEventListener
- MDN Checkbox change event: change event

Schritt 1 Favicon verlinken



Screenshot mit Browsertab und Favicon

Aufgabe

- Ergänzen Sie im <head> Ihrer HTML-Datei ein Favicon.

- Verwenden Sie dazu ein `<link>` Tag mit `rel=„icon“`.
- Achten Sie darauf, dass der Pfad zu Ihrer Datei stimmt (z.B. `./resources/favicon.ico`).

```
<!-- Aufgabe: Favicon im head verlinken -->
<!-- Tipp: Prüfen Sie den Pfad im Browser-Netzwerk-Tab
(Network) -->
<link rel="icon" href="PFAD_ZUR_DATEI" />
```

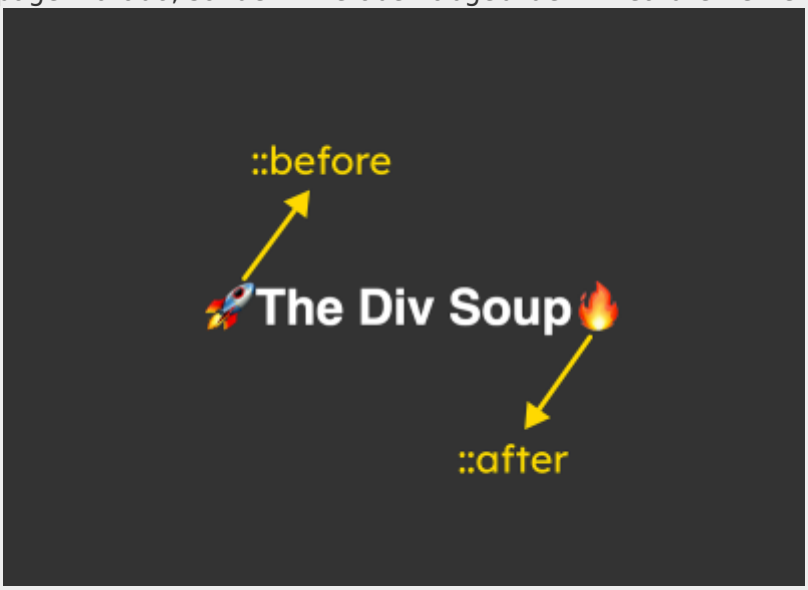
Checkliste

- Das Icon erscheint im Browser-Tab (auf Chrome oder Firefox testen, Safari zeigt das Favicon nicht an (weil File lokal gespeichert ist)).
- In DevTools → Network gibt es keinen 404 Fehler für das Favicon.

Schritt 2 Feature-Icons mit Pseudo-Elementen umsetzen

Sie haben in Ihrer Info-Zeile zwei Feature-Texte, z.B. „No credit card required„. Vor jedem Text soll ein Icon erscheinen.

Beispiel für das Anwenden von Pseudo-Elementen (nicht so umzusetzen im Projekt Landingpage Alarado, sondern wie oben abgebildet → Feature-Texte mit grünem Haken)



Wenn wir vor oder nach einem Text ein Icon oder Emoji einblenden möchten, eignen sich Pseudo-Elemente sehr gut.

Umsetzung des im Bild oben gezeigten Designs (The Div Soup) mit HTML und CSS:

```
<body>
  <h1>The Div Soup</h1>
</body>

body {
  background-color: #000000;
}

h1 {
  color: #fff;
}

h1::before {
  content: "🚀"; /* Add a rocket icon before the text */
}

h1::after {
  content: "🔥"; /* Add a fire icon after the text */
}
```

Kurze Erklärung: Wir betten die grünen OK-Haken per CSS ein und nicht als <img>-Tag, weil sie keinen eigenen Inhalt transportieren. Sie sind reine Dekoration. So können wir das Icon einmal in einer CSS-Regel definieren und dann auf mehrere Elemente anwenden.

Aufgabe

- Ergänzen Sie bei `.icon-ok` sinnvolle Innenabstände links, damit Platz für ein Icon entsteht.
- Setzen Sie `.icon-ok` auf `position: relative`.
- Nutzen Sie `.icon-ok::before`, um das Icon einzufügen.
- Positionieren Sie das Icon mit `position: absolute` links neben dem Text.

Bei Pseudo-Elementen können Sie Icons z.B. über `content: url(…)` einfügen.

```
/* Aufgabe: Platz schaffen und Positionierung vorbereiten */
.icon-ok {
  /* z.B. padding-left setzen */
  /* position: relative */
}

/* Aufgabe: Icon einfügen und positionieren */
.icon-ok::before {
  /* content: url("PFAD_ZUM_ICON") */
  /* position: absolute */
  /* left: 0; top: ... */
```

```
}
```

Checkliste

- Vor beiden Feature-Texten steht ein Icon.
- Das Icon überlappt den Text nicht.
- Der Abstand wirkt im Desktop und Mobile sauber.

Video Switch Dark Light Mode

Video mit Erklärung zum Dark Light Mode.
Video mit Erklärung zum Dark Light Mode und dessen Umsetzung.

Schritt 3 Switch Styling einbauen

Aus der Checkbox wird ein Switch. Nutzen Sie dazu den vorgegebenen Code unten.

Jetzt stylen Sie die Checkbox als Switch. Den folgenden Code dürfen Sie 1 zu 1 übernehmen.

Aufgabe

- Ergänzen Sie im HTML im `<label class="switch">` zwei Tags (Slider und Icons).
- Fügen Sie den CSS-Block unterhalb Ihrer bestehenden Styles ein, dort wo im Template „Switch Styles“ vorgesehen sind.
- Kontrollieren Sie die Pfade zu den Icons im `url(…)`.

HTML

```
<label class="switch">
  <input type="checkbox" />
  <span class="slider"></span> <!-- Platzhalter für den
Schieber -->
  <span class="switch-icons"></span> <!-- Platzhalter für
```

```
die Icons -->
</label>
```

CSS

```
/* Following: Styles Checkbox as Switch */
.switch {
  position: relative;
  display: inline-block;
  width: 48px;
  height: 24px;
  cursor: pointer;
}

/* Hide the checkbox input */
.switch input {
  display: none;
}

/* Describe slider's look and position. */
.slider {
  position: absolute;
  cursor: pointer;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: var(--main-text-color-light);
  transition: 0.4s;
  border-radius: 20px;
}

/* Describe the white ball's location and appearance in the
slider. */
.slider:before {
  position: absolute;
  content: '';
  height: 20px;
  width: 20px;
  left: 2px;
  bottom: 2px;
  background: var(--main-text-color-dark);
  transition: 0.4s;
  border-radius: 50%;
}

/* When the checkbox is checked, shift the white ball
towards the right within the slider. */
```

```
input:checked + .slider:before {
  transform: translateX(24px);
}

.switch-icons {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background:
    4px / 16px no-repeat url('./resources/Moon_fill.svg'),
    28px / 16px no-repeat url('./resources/Sun_fill.svg');
}

/* Modify the icons once the checkbox has been selected. */
input:checked ~ .switch-icons {
  background:
    4px / 16px no-repeat
url('./resources/Moon_fill_light.svg'),
    28px / 16px no-repeat
url('./resources/Sun_fill_light.svg');
}
```

Checkliste

- Der Switch sieht aus wie ein Schalter (Slider + Kreis + Icons).
- Beim Anklicken bewegt sich der Kreis (Animation).
- Wenn Icons nicht erscheinen: Pfade in `url(…)` prüfen.

Schritt 4 Dark Mode Klasse vorbereiten

Sie steuern den Dark Mode über eine CSS-Klasse auf dem `<html>` Element.

Aufgabe

- Definieren Sie eine CSS-Klasse `.theme-dark`.
- Schreiben Sie CSS-Regeln, die nur gelten, wenn `html` diese Klasse hat.
- Es sollen sich ändern:
  - Hintergrundfarbe
  - Textfarbe
  - Navigation-Linkfarben

Schreiben Sie die Regeln so, dass Sie nicht überall neue Farben kopieren, sondern Ihre Variablen verwenden.
Setzen Sie zum Testen provisorisch die Klasse `theme-dark` aufs

<html>-Element. Entfernen Sie dies wieder, sobald Sie mit der Gestaltung fertig sind: Die Klasse wird danach dynamisch mit JavaScript gesetzt.

```
/* Aufgabe: Klasse definieren */
.theme-dark {
  /* optional: color-scheme: dark; */
}

/* Aufgabe: Wenn html theme-dark hat, Farben umstellen */
html.theme-dark body {
  /* background-color: var(--...-dark); */
  /* color: var(--...-dark); */
}

/* Aufgabe: Navigation Links im Dark Mode lesbar machen */
html.theme-dark .menu li a:not(.active) {
  /* color: var(--...-dark); */
}
```

Checkliste

- Wenn die Klasse `class=„theme-dark“` im <html> steht, wird die Seite dunkel.
- Text bleibt gut lesbar.
- Navigation bleibt sichtbar.

Schritt 5 JavaScript Elemente aus dem DOM abrufen

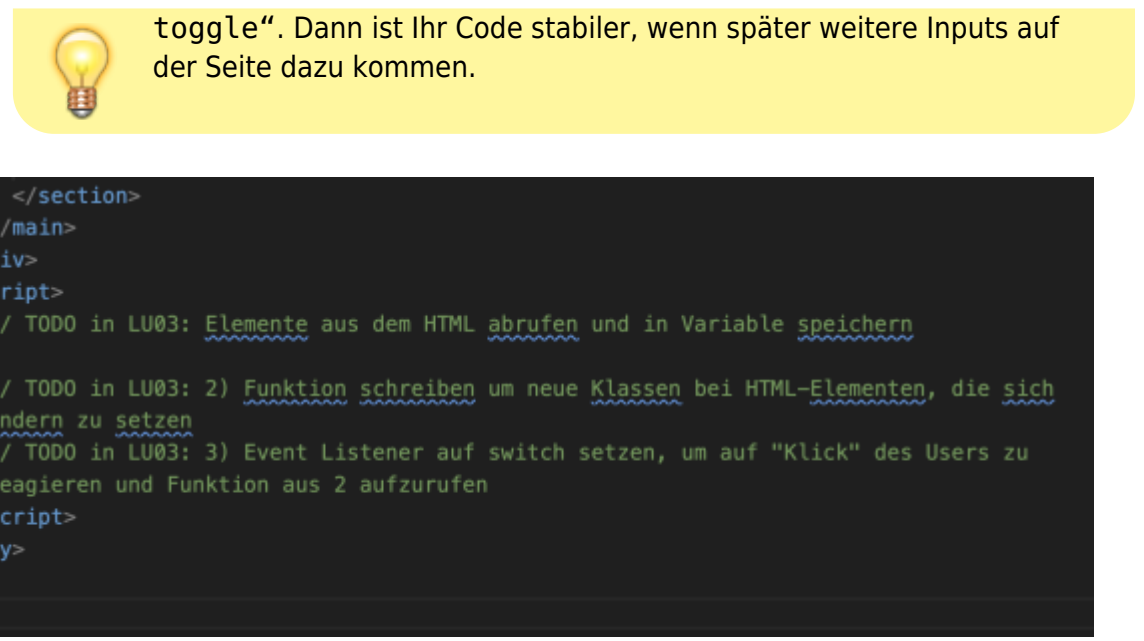
Damit Sie den Switch funktional machen können, brauchen Sie in JavaScript Zugriff auf:

- die Checkbox im Switch
- das Root-Element <html>
- das Logo-<img>

Aufgabe

- Geben Sie dem Logo eine ID, falls noch nicht vorhanden (z.B. `id=„logo“`).
- Holen Sie das Logo mit `getElementById` und speichern Sie es in einer Variable.
- Holen Sie das Root-Element mit `document.documentElement` und speichern Sie es in einer Variable (z.B. `html`).
- Rufen Sie die Checkbox ab, entweder mit einer ID oder über einen passenden Selektor, und speichern Sie sie in einer Variable (z.B. `checkbox`).

Best Practice: Geben Sie der Checkbox eine ID, z.B. `id=„theme-`



`toggle`“. Dann ist Ihr Code stabiler, wenn später weitere Inputs auf der Seite dazu kommen.

```
      </section>
    </main>
  </div>
  <script>
    // TODO in LU03: Elemente aus dem HTML abrufen und in Variable speichern

    // TODO in LU03: 2) Funktion schreiben um neue Klassen bei HTML-Elementen, die sich
    ändern zu setzen
    // TODO in LU03: 3) Event Listener auf switch setzen, um auf "Klick" des Users zu
    reagieren und Funktion aus 2 aufzurufen
  </script>
</body>
</html>
```

Folgender Code muss im `<script>`-Tag programmiert werden.

```
// Aufgabe: DOM-Elemente abrufen und in Variablen speichern
// const checkbox = ... (Switch)
// const html = ... (Root-Element)
// const logo = ... (img)
```

Schritt 6 Funktion zum Umschalten schreiben

Jetzt schreiben Sie eine Funktion, die beim Umschalten drei Dinge erledigt:

- Entscheidet anhand der Checkbox, ob Dark Mode aktiv sein soll
- Fügt die Klasse `theme-dark` hinzu oder entfernt sie
- Tauscht das Logo aus, indem Sie `logo.src` anpassen

Nutzen Sie für das Setzen der Klasse auf dem `<html>`-Element die Methoden `classList.add` und `classList.remove`.

```
// Aufgabe: Funktion toggleTheme()
// - wenn checkbox aktiv: theme-dark setzen oder entfernen
(je nach Logik)
// - Logo wechseln: logo.src auf light oder dark setzen

function toggleTheme() {
```

```
  if (checkbox.checked) {
    // Klasse theme-dark setzen oder entfernen (je nach
Logik)
    // Logo auf passende Datei setzen (Pfad anpassen)
  } else {
    // Klasse theme-dark setzen oder entfernen (je nach
Logik)
    // Logo auf passende Datei setzen (Pfad anpassen)
  }
}
```

Checkliste

- Dark Mode verändert Background und Textfarben.
- Logo wechselt passend.
- Keine Fehlermeldung in der Console.

Schritt 7 Event Listener für den Switch

Aufgabe

- Setzen Sie auf die Checkbox einen Event Listener.
- Reagieren Sie auf das Ereignis `change` (empfohlen für Checkbox). [1)]
- Rufen Sie in der Callback-Funktion Ihre Toggle-Funktion auf.

```
// Aufgabe: checkbox.addEventListener("change", ...)
// In der Callback-Funktion: toggleTheme() aufrufen

checkbox.addEventListener('change', () => {
  // Callback-Funktion: wird aufgerufen, sobald sich der
Status der Checkbox ändert
  // Rufen Sie hier toggleTheme() auf
});
```

Schritt 8 Initialzustand beim Laden prüfen

Wenn die Seite neu lädt, soll der Zustand korrekt sein:

- Switch-Position passt zum Theme
- Logo passt zum Theme
- Farben passen zum Theme

Aufgabe

- Rufen Sie Ihre Toggle-Funktion einmal beim Laden auf, oder schreiben Sie eine kleine Initial-Funktion.
- Test: Seite neu laden, ohne zu klicken.

```
// Aufgabe: Initialzustand setzen
// - toggleTheme() beim Laden einmal ausführen
```

Schritt 9 Testen und Debugging

Testfälle

- Switch klicken: Dark Mode ein, dann wieder aus
- Seite neu laden: Zustand bleibt konsistent (keine „falsche„ Startfarbe)
- Mobile Ansicht: Layout bleibt korrekt, Switch ist je nach Ihrem CSS sichtbar oder ausgeblendet

Debugging Tipps

- DevTools Console: Fehlermeldungen lesen
- Prüfen, ob Selektoren ein Element finden [2)]
- Pfade zu Bildern prüfen (Logo, Icons, Favicon)

Wenn etwas nicht geht, geben Sie testweise Ausgaben in die Console aus, z.B. ob `checkbox.checked` true oder false ist.

[1)]
click funktioniert auch, change ist aber semantisch passender für Form-Inputs.
[2)]
Wenn eine Variable `null` ist, stimmt meist der Selektor oder die ID nicht.