Dark Mode

- Dark Mode: Gängige Methoden

Dark Mode: Gängige Methoden

1. Über `prefers-color-scheme` automatisch dem System folgen

Der Browser erkennt, ob das Betriebssystem auf Hell oder Dunkel gestellt ist.

Beispiel

```
body {
  background: white;
  color: black;
}

@media (prefers-color-scheme: dark) {
  body {
    background: #121212;
    color: white;
  }
}
```

Vorteile

- Sehr einfach
- Kein JavaScript nötig
- Nutzt die Nutzerpräferenz automatisch

Nachteile

- Nutzer kann auf deiner Seite nicht unbedingt separat umschalten
- Weniger flexibel

---

2. Per CSS-Klasse auf `html` oder `body`

Du setzt z. B. eine Klasse wie `dark` auf das `<html>`- oder `<body>`-Element und definierst dafür eigene Styles.

Beispiel

```
<html class="dark">
```

```
body {
  background: white;
  color: black;
}

.dark body {
  background: #121212;
  color: white;
}
```

Vorteile

- Sehr verbreitet
- Einfach per JavaScript umschaltbar
- Gut mit Frameworks kombinierbar

Nachteile

- Man braucht etwas JS für den Toggle
- Bei vielen Komponenten kann das CSS unübersichtlich werden

---

3. Mit CSS Custom Properties (Variablen)

Oft die sauberste Lösung: Du definierst Farbvariablen und überschreibst sie im Dark Mode.

Beispiel

```
:root {
  --bg: white;
  --text: black;
}

.dark {
  --bg: #121212;
  --text: white;
}

body {
  background: var(--bg);
  color: var(--text);
}
```

Vorteile

- Sehr wartbar
- Farben zentral definiert
- Ideal für größere Projekte und Design-Systeme

Nachteile

- Anfangs etwas mehr Struktur nötig
- Man muss konsequent mit Variablen arbeiten

---

4. Kombination aus Systempräferenz + manuellem Toggle

Das ist in der Praxis oft die beste Lösung.

Typischer Ablauf

- Standardmäßig: `prefers-color-scheme` nutzen
- Optional: Nutzer kann selbst hell/dunkel wählen
- Auswahl in `localStorage` speichern
- Beim nächsten Besuch wiederherstellen

Beispielidee

- Wenn der Nutzer nichts gewählt hat → System folgen
- Wenn der Nutzer manuell gewählt hat → diese Wahl hat Vorrang

Vorteile

- Beste UX
- Flexibel
- Nutzer behält Kontrolle

Nachteile

- Etwas mehr Implementierungsaufwand
- Man sollte auf „Flash of wrong theme“ beim Laden achten

---

5. Framework-spezifische Lösungen

Viele Frameworks bringen eigene Patterns mit:

- **Tailwind CSS**: `dark:`-Klassen, z. B. `dark:bg-black`
- **Bootstrap**: teils über Data-Attribute oder Theme-Konfiguration
- **Material UI / Chakra UI / etc.**: Theme Provider, Light/Dark Tokens

Vorteil

- Schnell integrierbar
- Oft bereits gut dokumentiert

Nachteil

- Abhängig vom Framework
- Weniger „roh“ verständlich als pures CSS

Technisch gängige Umsetzungsmuster

A. Nur CSS

Gut für einfache Websites:

- `prefers-color-scheme`
- CSS-Variablen
- kein manueller Schalter

B. CSS + JavaScript Toggle

Gut für die meisten Websites:

- Theme-Klasse setzen (`dark`)
- Theme im `localStorage` speichern

- Optional Systempräferenz als Fallback

C. Design Tokens / Theming-System

Gut für große Anwendungen:

- Farben, Abstände, Komponenten über Tokens
- Light/Dark als Theme-Varianten
- oft mit Component Libraries

Worauf man achten sollte

1. Nicht nur Hintergrund und Text ändern

Auch wichtig:

- Buttons
- Links
- Borders
- Inputs
- Cards
- Modals
- Tabellen
- Code-Blöcke
- Icons
- Schatten

2. Kontrast

Dark Mode ist nicht einfach „schwarz mit weißem Text“.
Besser:

- leicht aufgehellte Dunkeltöne statt purem Schwarz
- nicht reinweißes Textweiß
- ausreichender Kontrast nach WCAG

3. Bilder und Logos

- Manche Logos funktionieren auf dunklem Hintergrund nicht
- Eventuell alternative Assets nutzen
- Bei Fotos sparsam mit Filtern sein

4. FOUC / Flash beim Laden vermeiden

Wenn das Theme erst nach dem Laden per JS gesetzt wird, sieht man kurz das falsche Theme.
Lösung:

- Theme sehr früh im `<head>` setzen
- gespeicherte Präferenz vor dem Rendern anwenden

5. Browser-UI anpassen

Mit

```
<meta name="color-scheme" content="light dark">
```

oder in CSS:

```
:root {
  color-scheme: light dark;
}
```

können native Form-Controls und Scrollbars besser zum Theme passen.

Empfehlung für die Praxis

Für moderne Websites ist meistens diese Kombination am sinnvollsten:

- **CSS Custom Properties** für Farben
- `prefers-color-scheme` als Standard
- **manueller Toggle** per JS
- **Speicherung in** `localStorage`

Das ist heute wahrscheinlich der gängigste und robusteste Ansatz.