Schritt-für-Schritt Anleitung: JavaScript für Dark/Light Mode

Diese detaillierte Anleitung führt Sie durch die vollständige JavaScript-Implementierung eines Dark/Light Mode-Umschalters. Jeder Schritt wird ausführlich erklärt und mit Codebeispielen veranschaulicht.

Schritt 1: Projektstruktur vorbereiten

Bevor wir mit dem JavaScript-Code beginnen, stellen Sie sicher, dass Ihre Projektstruktur wie folgt aussieht:

```
projekt/
  ├── index.html
  ├── css/
  │   └── styles.css
  └── js/
      └── darkmode.js
```

Die HTML-Datei sollte bereits den Toggle-Switch enthalten und die CSS-Datei sollte die Farbschemata für beide Modi definieren.

Schritt 2: Die grundlegende JavaScript-Datei erstellen

Erstellen Sie eine neue Datei `darkmode.js` im Ordner `js` und binden Sie diese in Ihre HTML-Datei ein:

```
<!-- Am Ende der HTML-Datei, kurz vor dem schließenden </body>-Tag -->
<script src="js/darkmode.js"></script>
```

Schritt 3: DOM-Elemente auswählen

Als Erstes müssen wir die DOM-Elemente auswählen, mit denen wir interagieren werden:

```
// DOM-Elemente auswählen
const toggleSwitch = document.querySelector('#checkbox');
const themeLabel = document.querySelector('#theme-label');
```

Erklärung: - `document.querySelector('#checkbox')` wählt das Element mit der ID "checkbox" aus (unsere Toggle-Switch-Checkbox) - `document.querySelector('#theme-label')` wählt das Element mit der ID "theme-label" aus (der Text, der den aktuellen Modus anzeigt) - `const` deklariert eine Konstante, deren Wert nicht geändert werden kann

Schritt 4: Funktion zum Umschalten des Themes erstellen

Jetzt erstellen wir die Hauptfunktion, die den Wechsel zwischen Light und Dark Mode steuert:

```
// Funktion zum Umschalten des Themes
function switchTheme(event) {
    if (event.target.checked) {
        // Auf Dark Mode umschalten
        document.documentElement.setAttribute('data-theme', 'dark');
        themeLabel.textContent = 'Dark Mode';
    } else {
        // Auf Light Mode umschalten
        document.documentElement.setAttribute('data-theme', 'light');
```

```
        themeLabel.textContent = 'Light Mode';
    }
}
```

Erklärung: - `function switchTheme(event) { ... }` definiert eine Funktion namens "switchTheme", die ein Event-Objekt als Parameter erhält - `if (event.target.checked) { ... } else { ... }` prüft, ob die Checkbox aktiviert ist (checked) - Wenn die Checkbox aktiviert ist (Dark Mode): - `document.documentElement.setAttribute('data-theme', 'dark')` setzt das Attribut "data-theme" auf "dark" für das HTML-Element (aktiviert die Dark Mode-CSS-Variablen) - `themeLabel.textContent = 'Dark Mode'` ändert den Text des Labels zu "Dark Mode" - Wenn die Checkbox deaktiviert ist (Light Mode): - Ähnliche Aktionen, aber mit "light" statt "dark"

Schritt 5: Event-Listener für den Toggle-Switch hinzufügen

Damit die Funktion beim Klicken auf den Toggle-Switch ausgeführt wird, fügen wir einen Event-Listener hinzu:

```
// Event-Listener für den Toggle-Switch
toggleSwitch.addEventListener('change', switchTheme);
```

Erklärung: - `toggleSwitch.addEventListener('change', switchTheme)` fügt einen Event-Listener hinzu, der auf Änderungen der Checkbox achtet - Wenn sich der Zustand der Checkbox ändert (durch Klicken), wird die Funktion `switchTheme` aufgerufen - 'change' ist das Event, auf das reagiert wird (Änderung des Checkbox-Zustands) - `switchTheme` ist die Funktion, die aufgerufen wird, wenn das Event eintritt

Schritt 6: Benutzereinstellung im localStorage speichern

Damit die Benutzereinstellung auch nach dem Neuladen der Seite erhalten bleibt, speichern wir sie im localStorage:

```
// Funktion zum Umschalten des Themes erweitern
function switchTheme(event) {
    if (event.target.checked) {
        // Auf Dark Mode umschalten
        document.documentElement.setAttribute('data-theme', 'dark');
        localStorage.setItem('theme', 'dark');
        themeLabel.textContent = 'Dark Mode';
    } else {
        // Auf Light Mode umschalten
        document.documentElement.setAttribute('data-theme', 'light');
        localStorage.setItem('theme', 'light');
        themeLabel.textContent = 'Light Mode';
    }
}
```

Erklärung: - `localStorage.setItem('theme', 'dark')` speichert den Wert "dark" unter dem Schlüssel "theme" im localStorage des Browsers - `localStorage.setItem('theme', 'light')` speichert den Wert "light" unter dem Schlüssel "theme" im localStorage des Browsers - Der localStorage ist ein einfacher Key-Value-Speicher im Browser, der auch nach dem Schließen des Browsers erhalten bleibt

Schritt 7: Gespeicherte Einstellung beim Laden der Seite anwenden

Damit die gespeicherte Einstellung beim Laden der Seite angewendet wird, fügen wir folgenden Code hinzu:

```
// Gespeicherte Einstellung abrufen
const currentTheme = localStorage.getItem('theme');

// Gespeichertes Theme beim Laden der Seite anwenden
if (currentTheme) {
    document.documentElement.setAttribute('data-theme', currentTheme);

    if (currentTheme === 'dark') {
        toggleSwitch.checked = true;
        themeLabel.textContent = 'Dark Mode';
    }
}
```

Erklärung: - `localStorage.getItem('theme')` ruft den gespeicherten Wert für den Schlüssel "theme" aus dem localStorage ab - `if (currentTheme) { ... }` prüft, ob ein Wert im localStorage gespeichert ist - Wenn ein Wert gespeichert ist: - `document.documentElement.setAttribute('data-theme', currentTheme)` wendet das gespeicherte Theme an - `if (currentTheme === 'dark') { ... }` prüft, ob das gespeicherte Theme "dark" ist - Wenn das gespeicherte Theme "dark" ist: - `toggleSwitch.checked = true` aktiviert die Checkbox - `themeLabel.textContent = 'Dark Mode'` ändert den Text des Labels zu "Dark Mode"

Schritt 8: Systemeinstellungen erkennen

Eine moderne Webseite sollte auch die Systemeinstellungen des Benutzers berücksichtigen. Dafür erweitern wir unseren Code:

```
// Systemeinstellung für Farbschema erkennen
function detectColorScheme() {
    // Nur wenn keine Benutzereinstellung gespeichert ist
    if (!localStorage.getItem('theme')) {
        // Prüfen, ob das System Dark Mode bevorzugt
        if (window.matchMedia('(prefers-color-scheme: dark)').matches) {
            document.documentElement.setAttribute('data-theme', 'dark');
            toggleSwitch.checked = true;
```

```
            themeLabel.textContent = 'Dark Mode';
        }
    }
}

// Systemeinstellung beim Laden überprüfen
detectColorScheme();
```

Erklärung: - `function detectColorScheme() { ... }` definiert eine Funktion zur Erkennung des bevorzugten Farbschemas - `if (!localStorage.getItem('theme')) { ... }` prüft, ob keine Benutzereinstellung gespeichert ist - `window.matchMedia('(prefers-color-scheme: dark)').matches` prüft, ob die Systemeinstellung Dark Mode bevorzugt - Wenn beide Bedingungen erfüllt sind, wird der Dark Mode aktiviert - `detectColorScheme()` ruft die Funktion beim Laden der Seite auf

Schritt 9: Auf Änderungen der Systemeinstellung reagieren

Damit die Webseite auch auf Änderungen der Systemeinstellung reagiert, fügen wir einen weiteren Event-Listener hinzu:

```
// Event-Listener für Änderungen der Systemeinstellung
window.matchMedia('(prefers-color-scheme: dark)').addEventListener('change', e ⇒
    // Nur wenn keine Benutzereinstellung gespeichert ist
    if (!localStorage.getItem('theme')) {
        if (e.matches) {
            // System wechselt zu Dark Mode
            document.documentElement.setAttribute('data-theme', 'dark');
            toggleSwitch.checked = true;
            themeLabel.textContent = 'Dark Mode';
        } else {
            // System wechselt zu Light Mode
            document.documentElement.setAttribute('data-theme', 'light');
            toggleSwitch.checked = false;
            themeLabel.textContent = 'Light Mode';
```

```
        }
    }
});
```

Erklärung: - `window.matchMedia('(prefers-color-scheme: dark)').addEventListener('change', e => { ... })` fügt einen Event-Listener hinzu, der auf Änderungen der Systemeinstellung reagiert - `if (!localStorage.getItem('theme')) { ... }` prüft, ob keine Benutzereinstellung gespeichert ist - `if (e.matches) { ... } else { ... }` prüft, ob die neue Systemeinstellung Dark Mode bevorzugt - Je nach Systemeinstellung wird der entsprechende Modus aktiviert

Schritt 10: Animationseffekte hinzufügen

Um den Wechsel zwischen den Modi ansprechender zu gestalten, fügen wir Animationseffekte hinzu:

```
// Funktion zum Umschalten des Themes mit Animation
function switchTheme(event) {
    if (event.target.checked) {
        // Auf Dark Mode umschalten
        document.documentElement.setAttribute('data-theme', 'dark');
        localStorage.setItem('theme', 'dark');
        themeLabel.textContent = 'Dark Mode';

        // Animationseffekt hinzufügen
        document.body.classList.add('theme-transition');
        setTimeout(() => {
            document.body.classList.remove('theme-transition');
        }, 1000);
    } else {
        // Auf Light Mode umschalten
        document.documentElement.setAttribute('data-theme', 'light');
        localStorage.setItem('theme', 'light');
        themeLabel.textContent = 'Light Mode';
```

```
        // Animationseffekt hinzufügen
        document.body.classList.add('theme-transition');
        setTimeout(() => {
            document.body.classList.remove('theme-transition');
        }, 1000);
    }
}
```

Erklärung: - `document.body.classList.add('theme-transition')` fügt dem Body-Element die Klasse "theme-transition" hinzu - Diese Klasse kann in CSS verwendet werden, um spezielle Animationseffekte während des Themenwechsels zu definieren - `setTimeout(() => { ... }, 1000)` führt den Code innerhalb der geschweiften Klammern nach 1000 Millisekunden (1 Sekunde) aus - `document.body.classList.remove('theme-transition')` entfernt die Klasse "theme-transition" vom Body-Element

Zugehöriges CSS für die Animation:

```
.theme-transition {
    transition: background-color 1s ease, color 1s ease;
}
```

Schritt 11: Barrierefreiheit verbessern

Um die Barrierefreiheit zu verbessern, fügen wir ARIA-Attribute und Tastatursteuerung hinzu:

```
// DOM-Elemente erweitern
const slider = document.querySelector('.slider');

// ARIA-Attribute aktualisieren
function updateARIA() {
    slider.setAttribute('aria-checked', toggleSwitch.checked);
}
```

```
// Tastatursteuerung für den Toggle-Switch
slider.setAttribute('role', 'switch');
slider.setAttribute('tabindex', '0');
slider.addEventListener('keydown', (e) ⇒ {
    if (e.key ≡ 'Enter' || e.key ≡ ' ') {
        e.preventDefault();
        toggleSwitch.checked = !toggleSwitch.checked;

        // Event manuell auslösen
        const changeEvent = new Event('change');
        toggleSwitch.dispatchEvent(changeEvent);
    }
});

// Event-Listener für den Toggle-Switch erweitern
toggleSwitch.addEventListener('change', (e) ⇒ {
    switchTheme(e);
    updateARIA();
});

// ARIA-Attribute beim Laden aktualisieren
updateARIA();
```

Erklärung: - `slider.setAttribute('role', 'switch')` setzt die ARIA-Rolle auf "switch", um Screenreadern mitzuteilen, dass es sich um einen Schalter handelt - `slider.setAttribute('tabindex', '0')` macht den Schieberegler fokussierbar mit der Tabulatortaste - `slider.addEventListener('keydown', (e) ⇒ { ... })` fügt einen Event-Listener für Tastatureingaben hinzu - `if (e.key ≡ 'Enter' || e.key ≡ ' ') { ... }` prüft, ob die Enter- oder Leertaste gedrückt wurde - `e.preventDefault()` verhindert das Standardverhalten der Taste (z.B. Scrollen bei Leertaste) - `toggleSwitch.checked = !toggleSwitch.checked` ändert den Zustand der Checkbox - `const changeEvent = new Event('change')` erstellt ein neues Change-Event - `toggleSwitch.dispatchEvent(changeEvent)` löst das Change-Event manuell aus - `updateARIA()` aktualisiert die ARIA-Attribute

Schritt 12: Vollständiger JavaScript-Code

Hier ist der vollständige JavaScript-Code mit allen besprochenen Funktionen:

```
// DOM-Elemente auswählen
const toggleSwitch = document.querySelector('#checkbox');
const themeLabel = document.querySelector('#theme-label');
const slider = document.querySelector('.slider');
const currentTheme = localStorage.getItem('theme');

// Funktion zum Umschalten des Themes mit Animation
function switchTheme(event) {
    if (event.target.checked) {
        // Auf Dark Mode umschalten
        document.documentElement.setAttribute('data-theme', 'dark');
        localStorage.setItem('theme', 'dark');
        themeLabel.textContent = 'Dark Mode';

        // Animationseffekt hinzufügen
        document.body.classList.add('theme-transition');
        setTimeout(() => {
            document.body.classList.remove('theme-transition');
        }, 1000);
    } else {
        // Auf Light Mode umschalten
        document.documentElement.setAttribute('data-theme', 'light');
        localStorage.setItem('theme', 'light');
        themeLabel.textContent = 'Light Mode';

        // Animationseffekt hinzufügen
        document.body.classList.add('theme-transition');
        setTimeout(() => {
            document.body.classList.remove('theme-transition');
        }, 1000);
    }
}
```

```
// ARIA-Attribute aktualisieren
function updateARIA() {
    slider.setAttribute('aria-checked', toggleSwitch.checked);
}

// Systemeinstellung für Farbschema erkennen
function detectColorScheme() {
    if (!localStorage.getItem('theme') && window.matchMedia('(prefers-color-schem
        document.documentElement.setAttribute('data-theme', 'dark');
        toggleSwitch.checked = true;
        themeLabel.textContent = 'Dark Mode';
        updateARIA();
    }
}

// Tastatursteuerung für den Toggle-Switch
slider.setAttribute('role', 'switch');
slider.setAttribute('tabindex', '0');
slider.addEventListener('keydown', (e) => {
    if (e.key === 'Enter' || e.key === ' ') {
        e.preventDefault();
        toggleSwitch.checked = !toggleSwitch.checked;

        // Event manuell auslösen
        const changeEvent = new Event('change');
        toggleSwitch.dispatchEvent(changeEvent);
    }
});

// Event-Listener für den Toggle-Switch
toggleSwitch.addEventListener('change', (e) => {
    switchTheme(e);
    updateARIA();
});

// Gespeichertes Theme beim Laden der Seite anwenden
if (currentTheme) {
```

```
    document.documentElement.setAttribute('data-theme', currentTheme);

    if (currentTheme === 'dark') {
        toggleSwitch.checked = true;
        themeLabel.textContent = 'Dark Mode';
    }
    updateARIA();
}

// Systemeinstellung beim Laden überprüfen
detectColorScheme();

// Event-Listener für Änderungen der Systemeinstellung
window.matchMedia('(prefers-color-scheme: dark)').addEventListener('change', e =>
    if (!localStorage.getItem('theme')) {
        if (e.matches) {
            document.documentElement.setAttribute('data-theme', 'dark');
            toggleSwitch.checked = true;
            themeLabel.textContent = 'Dark Mode';
        } else {
            document.documentElement.setAttribute('data-theme', 'light');
            toggleSwitch.checked = false;
            themeLabel.textContent = 'Light Mode';
        }
        updateARIA();
    }
});
```

Schritt 13: Testen und Debuggen

Nach der Implementierung sollten Sie Ihren Code gründlich testen:

1. **Grundfunktionalität testen:**
2. Funktioniert der Toggle-Switch?
3. Ändert sich das Farbschema beim Umschalten?

4. Wird der Text des Labels aktualisiert?
5. **localStorage testen:**
6. Wird die Einstellung gespeichert?
7. Wird die gespeicherte Einstellung beim Neuladen angewendet?
8. **Systemeinstellungen testen:**
9. Wird die Systemeinstellung berücksichtigt, wenn keine Benutzereinstellung gespeichert ist?
10. Reagiert die Webseite auf Änderungen der Systemeinstellung?
11. **Barrierefreiheit testen:**
12. Kann der Toggle-Switch mit der Tastatur bedient werden?
13. Sind die ARIA-Attribute korrekt gesetzt?
14. **Animationen testen:**
15. Sind die Übergänge zwischen den Modi sanft und ansprechend?

Schritt 14: Erweiterte Funktionen

Hier sind einige erweiterte Funktionen, die Sie implementieren können:

1. Automatischer Wechsel basierend auf Tageszeit

```
// Automatischer Wechsel basierend auf Tageszeit
function autoSwitchByTime() {
    const hour = new Date().getHours();

    // Zwischen 20 Uhr und 7 Uhr Dark Mode aktivieren
    if (hour ⩾ 20 || hour < 7) {
        document.documentElement.setAttribute('data-theme', 'dark');
        toggleSwitch.checked = true;
        themeLabel.textContent = 'Dark Mode (Auto)';
```

```
    } else {
        document.documentElement.setAttribute('data-theme', 'light');
        toggleSwitch.checked = false;
        themeLabel.textContent = 'Light Mode (Auto)';
    }
    updateARIA();
}

// Checkbox für automatischen Wechsel
const autoSwitch = document.querySelector('#auto-switch');
autoSwitch.addEventListener('change', (e) ⇒ {
    if (e.target.checked) {
        // Automatischen Modus aktivieren
        localStorage.setItem('theme-auto', 'true');
        autoSwitchByTime();

        // Timer für stündliche Überprüfung
        setInterval(autoSwitchByTime, 3600000); // 1 Stunde
    } else {
        // Automatischen Modus deaktivieren
        localStorage.removeItem('theme-auto');

        // Zurück zum manuellen Modus
        if (toggleSwitch.checked) {
            document.documentElement.setAttribute('data-theme', 'dark');
            localStorage.setItem('theme', 'dark');
            themeLabel.textContent = 'Dark Mode';
        } else {
            document.documentElement.setAttribute('data-theme', 'light');
            localStorage.setItem('theme', 'light');
            themeLabel.textContent = 'Light Mode';
        }
    }
});

// Automatischen Modus beim Laden wiederherstellen
if (localStorage.getItem('theme-auto') ⩵ 'true') {
```

```
    autoSwitch.checked = true;
    autoSwitchByTime();
    setInterval(autoSwitchByTime, 3600000); // 1 Stunde
}
```

2. Benutzerdefinierte Farbschemata

```
// Farbauswahl-Elemente
const primaryColorPicker = document.querySelector('#primary-color');
const accentColorPicker = document.querySelector('#accent-color');

// Farben beim Ändern anwenden
primaryColorPicker.addEventListener('input', updateCustomColors);
accentColorPicker.addEventListener('input', updateCustomColors);

// Benutzerdefinierte Farben anwenden
function updateCustomColors() {
    const primaryColor = primaryColorPicker.value;
    const accentColor = accentColorPicker.value;

    // CSS-Variablen überschreiben
    document.documentElement.style.setProperty('--primary-color', primaryColor);
    document.documentElement.style.setProperty('--accent-color', accentColor);

    // Farben speichern
    localStorage.setItem('primary-color', primaryColor);
    localStorage.setItem('accent-color', accentColor);
}

// Gespeicherte Farben beim Laden anwenden
if (localStorage.getItem('primary-color') && localStorage.getItem('accent-color')
    primaryColorPicker.value = localStorage.getItem('primary-color');
    accentColorPicker.value = localStorage.getItem('accent-color');
    updateCustomColors();
}
```

Zusammenfassung

In dieser Schritt-für-Schritt-Anleitung haben Sie gelernt, wie Sie einen vollständigen Dark/Light Mode-Umschalter mit JavaScript implementieren können. Sie haben folgende Funktionen implementiert:

1. Grundlegende Umschaltfunktion zwischen Light und Dark Mode
2. Speicherung der Benutzereinstellung im localStorage
3. Erkennung und Berücksichtigung der Systemeinstellungen
4. Animationseffekte für sanfte Übergänge
5. Verbesserung der Barrierefreiheit mit ARIA-Attributen und Tastatursteuerung
6. Erweiterte Funktionen wie automatischer Wechsel und benutzerdefinierte Farbschemata

Mit diesem Wissen können Sie nun einen modernen, benutzerfreundlichen Dark/Light Mode-Umschalter in Ihre eigenen Webprojekte integrieren.