FED 24/25

Kleurtjes met has en lightdark intro

Support voor :has()

First 🥳

Yes 👍

Yes 👍

De grote browsers werken samen om elk jaar tegelijk op dezelfde manier nieuwe technieken te introduceren: wpt.fyi/interop-2024

De actuele support voor :has() op caniuse: caniuse.com/css-has → 92.5% (not bad :)

Support voor :light-dark()

Yes 👍

Yes 👍

First 🥳

De grote browsers werken samen om elk jaar tegelijk op dezelfde manier nieuwe technieken te introduceren: wpt.fyi/interop-2024

De actuele support voor :light-dark() → 81% :(caniuse.com/mdn-css_types_color_light-dark

auto, light en dark met :has() en light-dark()

Om **auto, light en dark mode** te implementeren was altijd best veel JS nodig.

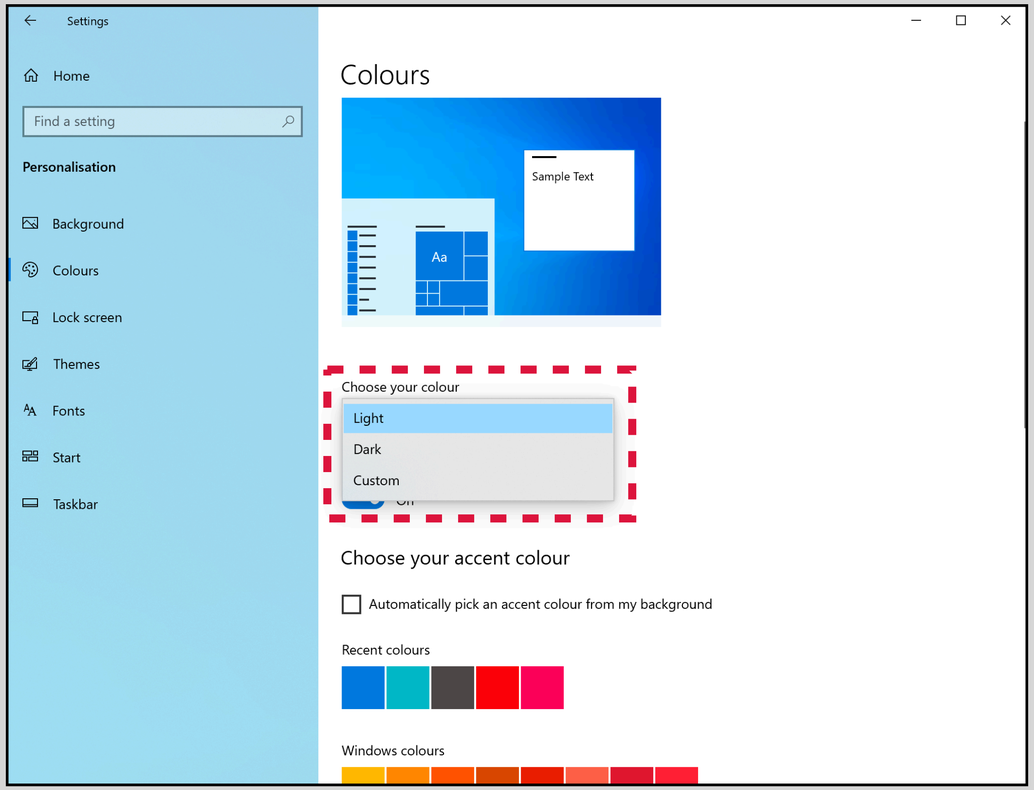
Met de relatief nieuwe **:has() en light-dark()** features kan dat met veel minder code en **alleen met CSS**.

start:
codepen.io/shooft/pen/mdZjZNW

auto, licht en donker:
codepen.io/shooft/pen/xxozyVW

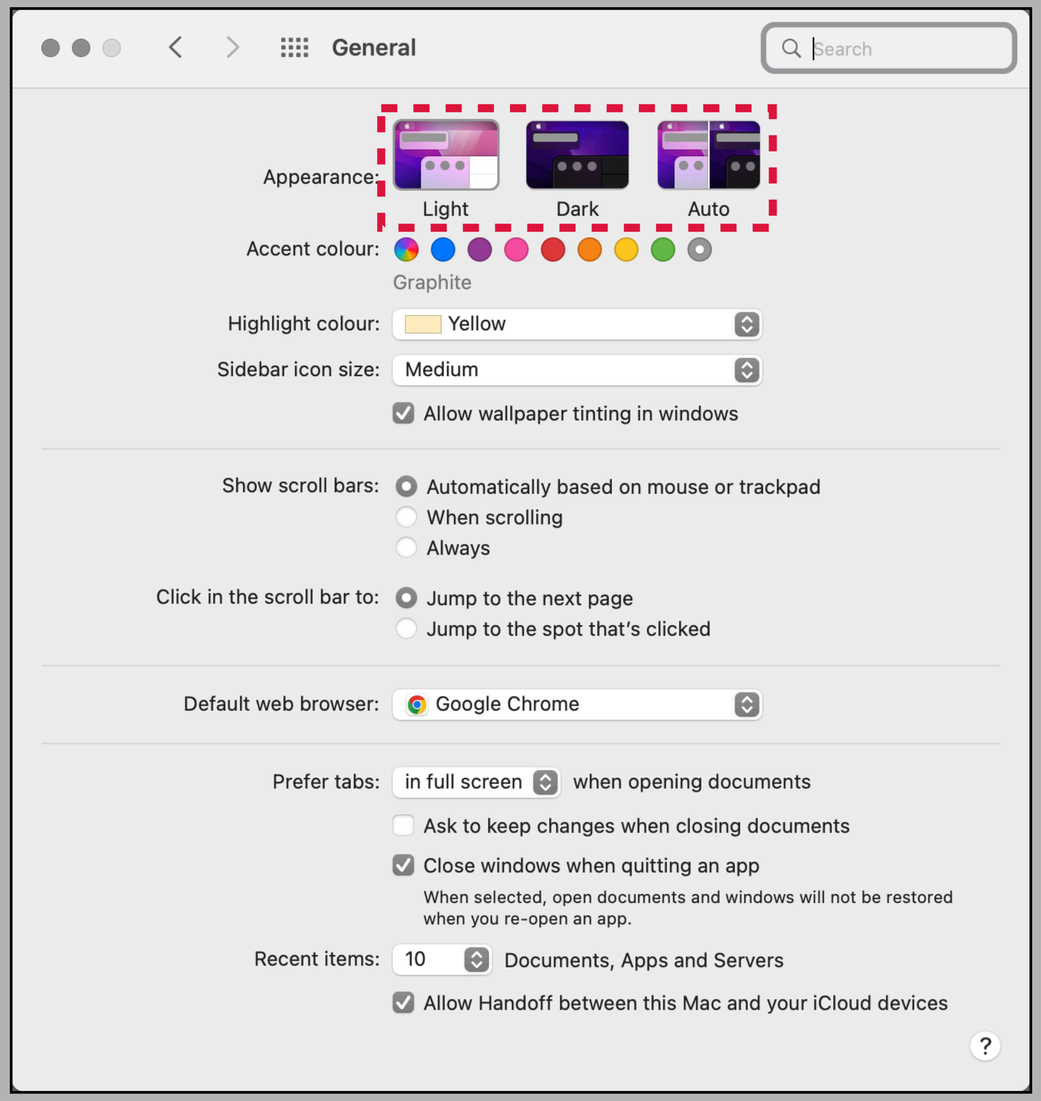
Dark/Light - Windows

Settings → Personalisation → Colours



Dark/Light - macOS

System preferences → general



auto, light en dark met :has() en light-dark()

Om **auto, light en dark mode** te implementeren was altijd best veel JS nodig.

Met de relatief nieuwe **:has() en light-dark()** features kan dat met veel minder code en **alleen met CSS**.

In de HTML

```
<label>
  <input type="radio" name="theme" value="auto" checked> auto
</label>
<label>
    <input type="radio" name="theme" value=“light"> licht
</label>
<label>
  <input type="radio" name=“theme" value=“dark"> donker
</label>
```

Drie radio buttons voor auto (default), light en dark

In de CSS

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

  --bg: light-dark(Pink, MediumVioletRed);
  --clr: light-dark(Black, White);
}

html:has([value="light"]:checked) {
    color-scheme: light;
}

html:has([value="dark"]:checked) {
    color-scheme: dark;
}

html {
   color: var(--clr);
   background-color: var(--bg);
}
```

auto, light en dark met :has() en light-dark()

Om **auto, light en dark mode** te implementeren was altijd best veel JS nodig.

Met de relatief nieuwe **:has() en light-dark()** features kan dat met veel minder code en **alleen met CSS**.

In de HTML

```
<label>
  <input type="radio" name="theme" value="auto" checked> auto
</label>
<label>
    <input type="radio" name="theme" value=“light"> licht
</label>
<label>
  <input type="radio" name=“theme" value=“dark"> donker
</label>
```

In de CSS

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

  --bg: light-dark(Pink, MediumVioletRed);
  --clr: light-dark(Black, White);
}

html:has([value="light"]:checked) {
    color-scheme: light;
}

html:has([value="dark"]:checked) {
    color-scheme: dark;
}

html {
   color: var(--clr);
   background-color: var(--bg);
}
```

Naast de standaard light modus van de browser ook de dark modus activeren.

De browser gebruikt dan de modus (light of dark) die actief is in het besturingssysteem.

auto, light en dark met :has() en light-dark()

Om **auto, light en dark mode** te implementeren was altijd best veel JS nodig.

Met de relatief nieuwe **:has() en light-dark()** features kan dat met veel minder code en **alleen met CSS**.

In de HTML

```
<label>
  <input type="radio" name="theme" value="auto" checked> auto
</label>
<label>
    <input type="radio" name="theme" value=“light"> licht
</label>
<label>
  <input type="radio" name=“theme" value=“dark"> donker
</label>
```

In de CSS

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

  --bg: light-dark(Pink, MediumVioletRed);
  --clr: light-dark(Black, White);
}

html:has([value="light"]:checked) {
    color-scheme: light;
}

html:has([value="dark"]:checked) {
    color-scheme: dark;
}

html {
   color: var(--clr);
   background-color: var(--bg);
}
```

Custom properties definiëren met de light-dark() functie.

De custom properties hebben dan een light en dark variant die worden gebruikt als de light of dark modus van de browser actief is.

auto, light en dark met :has() en light-dark()

Om **auto, light en dark mode** te implementeren was altijd best veel JS nodig.

Met de relatief nieuwe **:has() en light-dark()** features kan dat met veel minder code en **alleen met CSS**.

In de HTML

```
<label>
  <input type="radio" name="theme" value="auto" checked> auto
</label>
<label>
    <input type="radio" name="theme" value=“light"> licht
</label>
<label>
  <input type="radio" name=“theme" value=“dark"> donker
</label>
```

In de CSS

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

  --bg: light-dark(Pink, MediumVioletRed);
  --clr: light-dark(Black, White);
}

html:has([value="light"]:checked) {
    color-scheme: light;
}

html:has([value="dark"]:checked) {
    color-scheme: dark;
}

html {
   color: var(--clr);
   background-color: var(--bg);
}
```

De custom properties gebruiken zoals altijd.

auto, light en dark met :has() en light-dark()

Om **auto, light en dark mode** te implementeren was altijd best veel JS nodig.

Met de relatief nieuwe **:has() en light-dark()** features kan dat met veel minder code en **alleen met CSS**.

In de HTML

```
<label>
  <input type="radio" name="theme" value="auto" checked> auto
</label>
<label>
    <input type="radio" name="theme" value=“light"> licht
</label>
<label>
  <input type="radio" name=“theme" value=“dark"> donker
</label>
```

In de CSS

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

  --bg: light-dark(Pink, MediumVioletRed);
  --clr: light-dark(Black, White);
}

:root:has([value=“light"]:checked) {
    color-scheme: light;
}

:root:has([value=“dark"]:checked) {
    color-scheme: dark;
}

html {
   color: var(--clr);
   background-color: var(--bg);
}
```

Als de radio button met de value light gecheckt is, mag de browser alleen de light modus gebruiken.

Als de radio button met de value dark gecheckt is, mag de browser alleen de dark modus gebruiken.

FEED DATA

Identifies what makes an influencer