Interaction design

Tuur Vanhoutte

12 oktober 2020

Inhoudsopgave

1 CSS Variables **1**
- 1.1 Wat? . . . 1
- 1.2 Opbouw & Syntax . . . 1
  - 1.2.1 Custom property: defining the variable . . . 1
  - 1.2.2 Cascading variable: applying the variable . . . 1
- 1.3 Syntax . . . 1
  - 1.3.1 CSS Variables Are Case Sensitive . . . 1
  - 1.3.2 This is wrong . . . 2
  - 1.3.3 Use the calc() function to do math . . . 2
  - 1.3.4 Kan ook shorthand values bevatten . . . 2
  - 1.3.5 Kan ook bestaan uit andere variables. . . . 2
  - 1.3.6 Default values . . . 2
  - 1.3.7 Default values with other variables . . . 3
- 1.4 Cascade . . . 3
  - 1.4.1 CSS variables can be made conditional with @media and other conditional rules 3
  - 1.4.2 Ideal for dark themes . . . 4
- 1.5 Hoisting . . . 4
- 1.6 Scoped variables . . . 4
  - 1.6.1 Global scoped variables . . . 4
  - 1.6.2 Local scoped variables . . . 5
- 1.7 Naming system . . . 5
  - 1.7.1 The two-level theming system . . . 5
  - 1.7.2 Global level . . . 5
  - 1.7.3 Local level . . . 5
- 1.8 Herhalingsvragen . . . 6

2 Forms **6**
- 2.1 Form . . . 6
  - 2.1.1 Voorbeelden . . . 6
  - 2.1.2 Basic form syntax . . . 7
- 2.2 Input types . . . 7
  - 2.2.1 HTML5 input types . . . 7
  - 2.2.2 Text-achtigen . . . 7
  - 2.2.3 Time-achtigen . . . 8
  - 2.2.4 Option-achtigen . . . 8
- 2.3 Checkbox or toggle? . . . 9
  - 2.3.1 Checkbox . . . 9
  - 2.3.2 Toggle switch = veredelde checkbox . . . 9
  - 2.3.3 Voorbeelden . . . 10
  - 2.3.4 Toggle Switch of Toggle Button? . . . 11
- 2.4 Textarea . . . 12
- 2.5 Select . . . 12
- 2.6 Range . . . 13
- 2.7 Hors catégorie . . . 13
- 2.8 Attributes . . . 13
  - 2.8.1 Minimum attributes . . . 13
  - 2.8.2 Veelgebruikte attributes . . . 13
  - 2.8.3 Lege attributes . . . 14
- 2.9 Labels . . . 14
  - 2.9.1 Label koppelen aan input - Manier 1 . . . 15

2.9.2 Label koppelen aan input - Manier 2 . . . 15
2.10 Buttons . . . 15
2.10.1 Als input type . . . 15
2.10.2 Als button element . . . 15
2.11 States . . . 16
2.11.1 :hover . . . 16
2.11.2 :active . . . 16
2.11.3 :focus . . . 16
2.12 Validation . . . 17
2.12.1 Client side validation . . . 17
2.12.2 Server side validation . . . 17
2.12.3 Voorbeelden . . . 17
2.13 Extra (geen vragen op examen) . . . 18
2.13.1 States . . . 18
2.13.2 HTML5 form validation . . . 18
2.13.3 Best practices . . . 19

3 Affordances **19**
3.1 Explicit affordance . . . 19
3.1.1 Voorbeelden . . . 20
3.2 Pattern affordance . . . 20
3.2.1 Pattern metaphors . . . 20
3.3 Hidden affordance . . . 20
3.4 False affordance . . . 21
3.5 Negative affordance . . . 21
3.6 Overzicht . . . 22

4 Micro Interactions **22**
4.1 Wat? . . . 22
4.2 Actie - reactie - feedback . . . 22
4.2.1 Voorbeeld: Toggle switch . . . 22
4.3 Versterkt door animatie . . . 22
4.4 Waar gebruiken? . . . 23
4.5 Voorbeelden . . . 23
4.5.1 Iphone Mute . . . 23
4.5.2 Pull to refresh . . . 23
4.5.3 Nightmode . . . 23
4.5.4 Facebook like . . . 23
4.6 Combinaties van micro-interacties . . . 24
4.7 Verandering aanduiden . . . 24
4.8 Don't . . . 24
4.9 Meer dan visuele feedback . . . 24
4.10 Resources . . . 24

5 API-calls, JavaScript basics & debugging **25**
5.1 JavaScript . . . 25
5.1.1 Manier van werken . . . 25
5.1.2 Beschikbare structuren . . . 25
5.2 API calls in JS . . . 25
5.2.1 Data ophalen in JS . . . 25
5.2.2 Async/await functie . . . 26
5.2.3 Error handling . . . 26

5.3 Debugging 26
5.3.1 Logging 26
5.3.2 Browser breakpoints 26

1 CSS Variables

1.1 Wat?

- CSS custom properties for cascading variables.
- CSS Variables = Custom Properties.
- Relatief nieuwe manier om veelgebruikte values om te zetten naar Variables.
- To keep consistency, set global variables for everything except layout values.

1.2 Opbouw & Syntax

1.2.1 Custom property: defining the variable

Custom property: value

```
--my-cool-color: HotPink;
```

1.2.2 Cascading variable: applying the variable

Applying your custom property using the var() function

```
var(--my-cool-color)
```

1.3 Syntax

```
:root {
    --my-cool-color: HotPink;
}

p {
    color: var(--my-cool-color);
}

.foo {
    background-color: var(--my-cool-color);
}
```

1.3.1 CSS Variables Are Case Sensitive

```
:root {
    --foo: HotPink;
    --FOO: #BADA55;
}
p {
    color: var(--foo);
}
.foo {
    background-color: var(--FOO);
}
```

1.3.2 This is wrong

```
.test {
    --side: margin-top;
    var(--side): 20px
}
```

1.3.3 Use the calc() function to do math

```
:root {
    --whitespace: 20px;
    --whitespace-lg: var(--whitespace) * 2; /* won't work */
    --whitespace-lg: calc(var(--whitespace) * 2); /* correct */
}
```

1.3.4 Kan ook shorthand values bevatten

```
:root {
    --transition: all .1s ease-out;
}
a {
    transition: var(--transition);
}
```

1.3.5 Kan ook bestaan uit andere variables.

```
:root {
    --transition-property: all;
    --transition-duration: .1s;
    --transition-timing-function: ease-out;
    --transition: var(--transtion-property)
        var(--transition-duration)
        var(--transition-timing-function);
}

a {
    transition: var(--transition);
}
```

1.3.6 Default values

```
.c-button {
    border: 1px solid var(--button-color, HotPink);
    background-color: transparent;
}
.c-button:hover {
    background-color: var(--button-color, HotPink);
}
.c-button--beta {
    --button-color: #BADA55;
}
```

1.3.7 Default values with other variables

```
:root {
    --color-pink: HotPink;
    --color-badass: #BADA55;
}
.c-button {
    border: 1px solid var(--button-color, var(--color-pink));
}
.c-button:hover {
    background-color: var(--button-color, var(--color-pink));
}
.c-button--beta {
    --button-color: var(--color-badass);
```

1.4 Cascade

CSS Variables Are Subject to the Cascade and Inheritance rules

https://codepen.io/simoncoudeville-nmct/pen/BaBMGPZ

```
:root {
    --my-cool-color: HotPink;
}
/* Alle elementen binnen de root waar je --
my-cool-color variable toepast zullen de
value HotPink overerven. */

p {
    color: var(--my-cool-color);
}

.foo {
/* Elke paragraaf in het element met
de class ".foo" krijgt de kleur #BADA55.*/

--my-cool-color: #BADA55;
}
```

https://codepen.io/simoncoudeville-nmct/pen/aboMBop

1.4.1 CSS variables can be made conditional with @media and other conditional rules

```
:root {
    --whitespace: 1em;
}
@media screen and (min-width: 768px) {
    :root {
        --whitespace: 2em;
    }
}
.c-card {
    padding: var(--whitespace);
}
```

https://codepen.io/simoncoudeville-nmct/pen/JjXaQwB

1.4.2 Ideal for dark themes

```
:root {
    --color: white;
    --background-color: black
}
@media (prefers-color-scheme: dark) {
    :root {
    --color: black;
    --background-color: white
    }
}
.html {
    background-color: var(--background-color);
    color: var(--color);
}
```

https://codepen.io/simoncoudeville-nmct/pen/eYOxbPp

1.5 Hoisting

= Accessing a Variable First and Declaring Later

```
body{
    background: var(--bg-fill);
}
:root{
    --bg-fill: green;
}
```

1.6 Scoped variables

Twee soorten:

- Global Scoped Variables
- Local Scoped Variables

1.6.1 Global scoped variables

```
:root {
    --global-color: black;
}
```

- :root is a CSS pseudo-class selector used to select the element that represents the root of the document.
- :root is hetzelfde als html maar is specifieker
- :root = html
- :root > html

https://codepen.io/simoncoudeville-nmct/pen/OJLBKXq

Global variables kan je overal hergebruiken en overschrijven. Ideaal dus voor values die veel hergebruikt worden:

- colors
- whitespace
- border-radius
- transitions
- ...

1.6.2 Local scoped variables

- Local scoped variables worden gedeclareerd binnen een specifieke selector.
- Local scoped variables hebben access tot global scoped variables.
- Ideaal voor components.

```
.alert {
    --alert-color: #222;
    color: var(--alert-color);
    border-color: var(--alert-color);
}
:root {
    --global-fontSize: 16px;
}
.c-button {
    --button-fontSize: var(--global-fontSize);
    font-size: var(--button-fontSize);
}
```

1.7 Naming system

1.7.1 The two-level theming system

Systeem om global variables te gebruiken in local variables.

1.7.2 Global level

De hoofdreden om global variables te hebben is consistentie.

They are prefixed with the word global and follow this formula:

--global--concept--modifier--state--propertyCamelCase

- a concept is something like a spacer, main-title or text
- a state is something like hover, or expanded
- a modifier is something like sm, or lg
- and a propertyCamelCase is something like backgroundColor or fontSize

1.7.3 Local level

They follow this formula:

--block__element--modifier--state--propertyCamelCase

- The block__element--modifier selector name is something like alert__actions or alert–primary

- a state is something like hover or active
- The value of component scoped variables is always defined by a global variable.

```
.c-alert {
    /* Component scoped variables are always defined by global variables
       */
    --c-alert--Padding: var(--global--spacer--md);
    --c-alert--primary--BackgroundColor: var(--global--primary-color);
    --c-alert__title--FontSize: var(--global--secondary-title--fontSize);
    /* --block--propertyCamelCase */
    padding: var(--c-alert--padding);
}

/* --block--state--propertyCamelCase */
.c-alert--primary {
    background-color: var(--c-alert--primary--backgroundColor);
}

/* --block__element--propertyCamelCase */
.c-alert__title {
    font-size: var(--c-alert__title--fontSize);
}
```

1.8 Herhalingsvragen

- Hoe worden CSS variables nog genoemd?
- Hoe worden CSS variables opgebouwd?
- Wat is CSS Hoisting?
- Wat is het verschil tussen global en local variables?

`https://codepen.io/simoncoudeville-nmct/pen/vYBbbXz?editors=1100`

2 Forms

2.1 Form

- HTML forms are a very powerful tool for interacting with users
- Groot deel van een digitale interface
- In veel vormen en maten

2.1.1 Voorbeelden

- Nieuwe repo maken op github
- Profiel aanmaken
- Route kiezen DeLijn
- ...

2.1.2 Basic form syntax

```
<form action="">
    <input type="">
    ...
    Submit
</form>
```

Maak geen forms zonder (submit) button! Zonder (submit) button kan je niet op enter duwen

2.2 Input types

- Alle input types: `https://www.w3schools.com/tags/att_input_type.asp`
- Dit zijn de belangrijkste types die je moet kennen:
  - Text-achtigen
  - Time-achtigen
  - Option-achtigen
  - Textarea
  - Select
  - Range
  - Hors catégorie

2.2.1 HTML5 input types

- Browser validatie
- Smartphone keyboard verandert
- Testen op je smartphone:

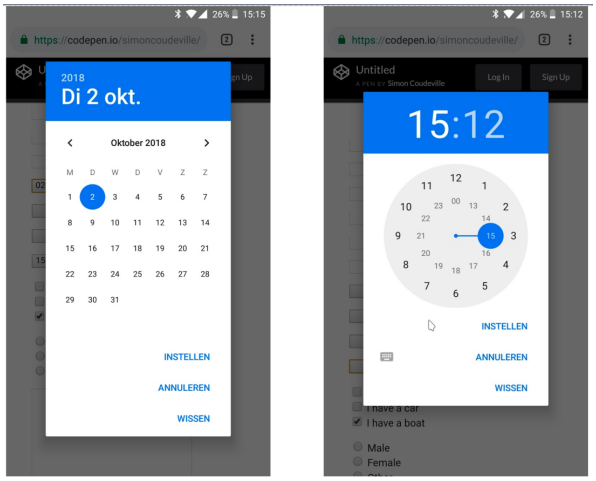
  `https://mobiforge.com/design-development/html5-mobile-web-forms-and-input-types`
- Live demo: `https://codepen.io/simoncoudeville-nmct/pen/gQYqBY`

2.2.2 Text-achtigen

- Zien er visueel ongeveer hetzelfde uit
- Op smartphones: keyboard veranderd op basis van welk type de input is
- **text** - basis input type voor HTML5 input types
- **password** - vervangt de letters door bolletjes
- **email** - valideert de browser enkel als je een correct e-mail adres invult
- **number**
  - Aanvaardt enkel nummers
  - Heeft pijltjes om te vermeerderen of te verminderen
- **tel** - telefoonnummers

2.2.3 Time-achtigen

- Zien er visueel ongeveer hetzelfde uit als de text-achtigen
- date - toont een native date picker
- week - toont een variant van de native date picker
- month - toont een variant van de native date picker
  - Vooral date zal je kunnen gebruiken.
- number - aanvaardt enkel nummers
- tel - handig voor mobile



`<input type="date">` `<input type="time">`

Figuur 1: <input type="date"> en <input type="time"> op smartphones

2.2.4 Option-achtigen

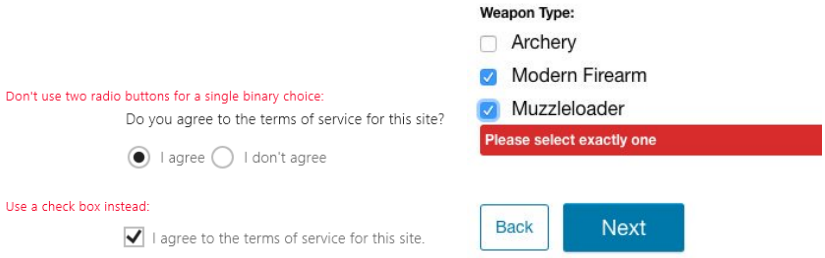
checkbox

- Meerdere keuzes mogelijk uit een aantal keuzes
- Of kan alleen bestaan

radio(button)

- Slechts 1 keuze mogelijk uit een aantal keuzes
- Kan niet alleen bestaan, altijd in en group
- Gekoppeld aan elkaar door het name attribuut
- Niet voor enkele binaire keuzes
- Kan je ook customizen met CSS

Geschiedenis van de radio-button: `https://www.jitbit.com/radio-button/`



Figuur 2

Laat een checkbox er nooit uitzien als een radio button en omgekeerd!

2.3 Checkbox or toggle?

Figuur 3: Links: toggle switch, rechts: checkbox

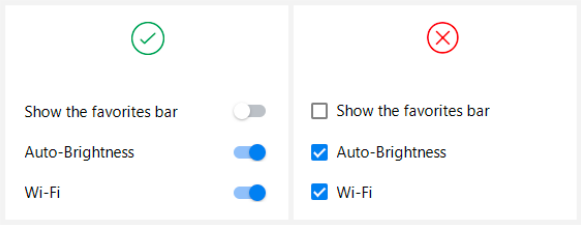
2.3.1 Checkbox

- Checked of niet
- Heeft nog confirmatie nodig
- Meerdere opties die bij elkaar horen
- Checken van sub options (intermediate state)
- Enkele ja/nee optie

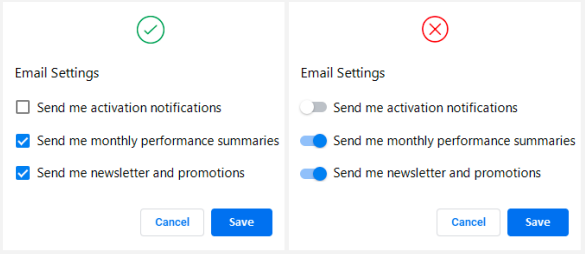
2.3.2 Toggle switch = veredelde checkbox

- Aan of af zetten
- Instant response zonder confirmatie
- Afzonderlijke features of settings
- Enkele aan/af beslissing

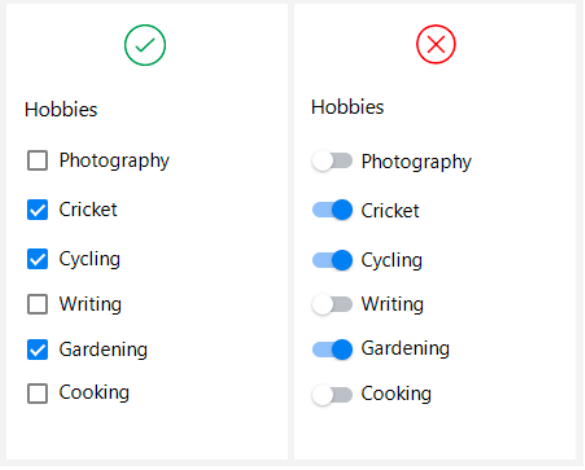
2.3.3 Voorbeelden



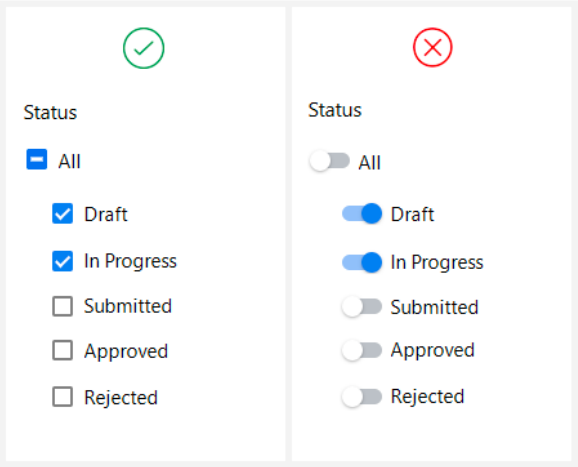
Figuur 4: De opties die een directe reactie vereisen kunnen het best geselecteerd worden met een toggle switch.



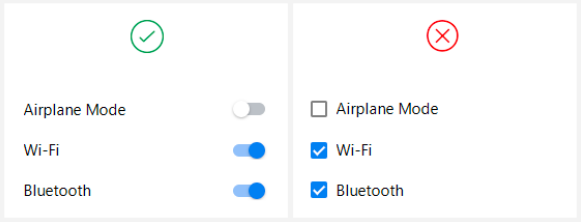
Figuur 5: Checkboxes hebben de voorkeur wanneer een expliciete actie vereist is om instellingen toe te passen.



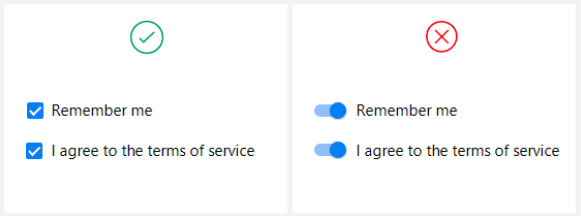
Figuur 6: Selecteren van meerdere opties in een lijst biedt betere ervaring met checkboxes.



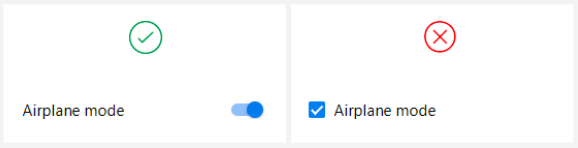
Figuur 7: Indeterminate state wordt het best voorgesteld met een checkbox. (zie 'All' aan de linkerkant)



Figuur 8: Afzonderlijke features of settings zijn dan weer logischer met toggle switches.



Figuur 9: Een enkele ja/nee optie is logischer met een checkbox.



Figuur 10: Een enkele aan/af beslissing is logischer met een toggle switch.

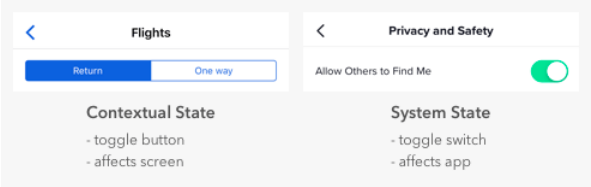
2.3.4 Toggle Switch of Toggle Button?

Toggle buttons = soort van veredelde radio button of meerdere buttons

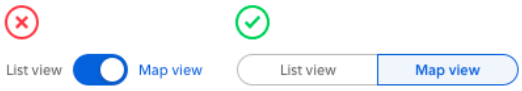
- Contextual state
- Heeft invloed op het huidige scherm
- Opposing options

Toggle switch = veredelde checkbox

- System state
- Heeft invloed op de volledige app
- Binary options



Figuur 11



Figuur 12: Switches are for binary options, not opposing options.

2.4 Textarea

- <textarea>
- Geen input type, apart element
- Multi-line text input control
- De gebruiker kan optioneel de grootte aanpassen van het tekstvak

2.5 Select

- <select>
- Geen input type, apart element
- dropdown list
- De <option> tags binnen het <select> element definieren de beschikbare options in de lijst
- **Native select** behouden als je designt: geen eigen element proberen te maken ⇒ gebruiksvriendelijker op smartphone.

2.6 Range

- Slider control
- Voor nummers
- Default range van 0 tot 100
  - restrictions zijn mogelijk met max, min en step attributes

2.7 Hors catégorie

- **file** - file upload
- **hidden** - voor developers
- **color** - toont een color picker

2.8 Attributes

= Eigenschappen van de input

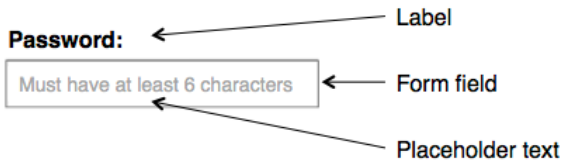
- bv: type, value, id, name, ...
- Alle attributes: `https://www.w3schools.com/html/html_form_attributes.asp`

2.8.1 Minimum attributes

- **type** - defineert het type input (duh!)
- **name**
  - Voor developers
  - Zorgt er voor dat je weet wat je waar ingevuld hebt
  - Ook om radio buttons aan elkaar te koppelen
  - **Dus altijd een name voorzien!**

2.8.2 Veelgebruikte attributes

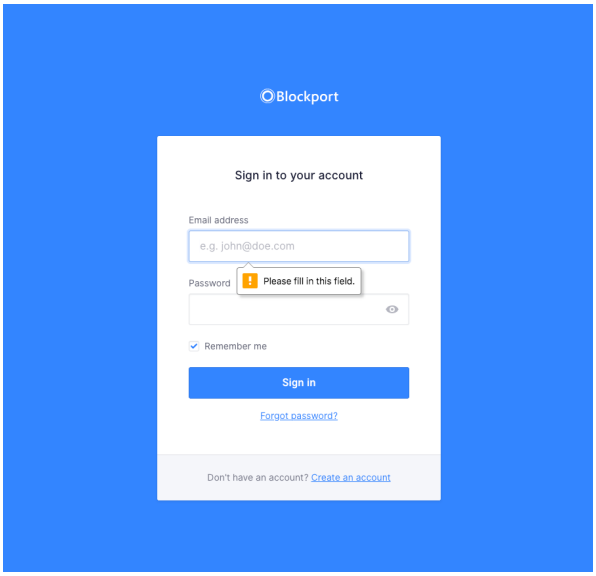
- **value**
  - Kan je gebruiken om een default value in te geven
  - Indien leeg wordt dit wat je hebt ingevuld
- **placeholder**
  - hint
  - voorbeeld van de wat de value moet zijn
  - Verdwijnt automatisch als je begint te typen
  - Placeholder nooit gebruiken als alternatief voor een label!



Figuur 13

2.8.3 Lege attributes

- Hebben geen value
- Zijn waar of onwaar
- CSS pseudo classes
- Veel gebruikt:
  - **Checked** - radio options & checkboxes
  - **Required** - voor browservalidatie
  - **Disabled** - kan je niet aanpassen en wordt ook niet gesubmit
  - **Readonly** - kan je niet aanpassen maar wordt wel gesubmit
  - **Autofocus** - focust automatisch op het input type



Figuur 14: required

2.9 Labels

- Gekoppeld aan een input
- Usability improvement: toggles the input
- Elke input moet een label hebben!

- Niet vervangen door placeholder!
  - In een lang ingevuld formulier weet je op den duur niet meer wat je waar moet invullen
  - Alternatief: floating label pattern:
  - `https://dribbble.com/shots/3429471-Floating-label-input-field`
  - `https://codepen.io/soulrider911/pen/ugnyl`

2.9.1 Label koppelen aan input - Manier 1

- Met for en id
- Label text is apart aanspreekbaar met CSS.

```
<label for="input_id">label text</label>
<input type="text" name="whatever" id="input_id">
```

2.9.2 Label koppelen aan input - Manier 2

- Geen for en id nodig (maar wel altijd aangeraden)
- Label text is niet aanspreekbaar met CSS.

```
    <label>
        label text
        <input type="text" name="whatever" id="input_id">
    </label>
```

2.10 Buttons

Elk formulier moet een button hebben! (Binnen de form tag)

2.10.1 Als input type

```
<input type="submit" value="verzenden">
<input type="button" value="verzenden">
```

2.10.2 Als button element

```
<button>verzenden</button>
```

Gebruik het button element!

```
<button class="c-button">
    <span class="c-button__label">Verzenden</span>
    <svg class="c-button__symbol>...</svg>
</button>
```

2.11 States

1. :hover
2. :active
3. :focus

Deze volgorde in de CSS is zeer belangrijk!

`https://codepen.io/simoncoudeville-nmct/pen/oNxJbea`

2.11.1 :hover

CSS declarations worden geactiveerd...

- Op pc: wanneer een gebruiker de muis over een element beweegt.
- Op mobiele toestellen: als een gebruiker een element ïnduwtën "loslaatën er geen specifieke focus of active styles zijn gedeclareerd of als :hover na :focus of :active komt.

2.11.2 :active

- CSS declarations worden geactiveerd wanneer een gebruiker met de muis klikt.
- CSS declarations worden geactiveerd op mobiele toestellen als een gebruiker een element - induwt".
- Extra feedback feedback

2.11.3 :focus

- Focus toont duidelijk welk element op de pagina keyboard events kan ontvangen
- Het element dat gefocust is heeft een duidelijke focus ring of outline of een andere visuele clue die de designer voorzien heeft.
- Welk element gefocust is kan je bedienen met het keyboard via tab of shift tab
- De volgorde is de tab order
- Interactieve HTML elementen zoals input, buttons, links zijn impliciet focusbaar. Ze worden automatisch aan de tab order toegevoegd.
- Paragrafen, divs, images enz... zijn niet focusbaar

Figuur 15

- Voorbeeld van een form waar je niet op kan klikken, te bedienen met de tab-toets
- `http://udacity.github.io/ud891/lesson2-focus/01-basic-form/`
- Probeer eens een ticket te boeken voor...
  - een round trip

  - van Sydney naar Melbourne
  - van 12 oktober tot 23 oktober 2018
  - aan het venster
  - en je wil geen promotionele aanbiedingen
- Met tab, de pijltjes, spatie voor checkbox, . . .

2.12 Validation

- Voorkom dat gebruikers fouten maken
- Als ze dan toch fouten maken en kunnen submitten:
  - Verzorg duidelijke foutboodschappen
  - Zet foutboodschappen inline bij hun input

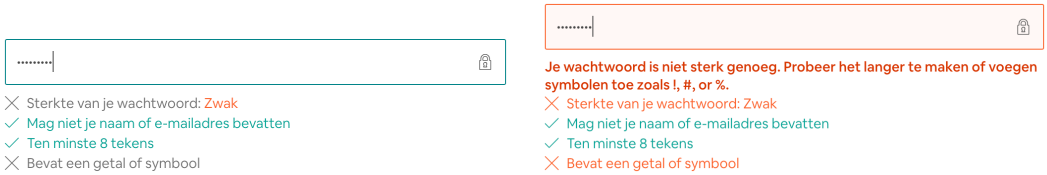
2.12.1 Client side validation

- Voor dat data wordt doorgestuurd naar de server
- Instant response
- HTML5 validation
-
  - Required, valid & invalid pseudo classes om instant te tonen of het juist is of niet.
  - `https://codepen.io/chriscoyier/pen/JXgKjb`
- + Javascript validation omdat je html kan aanpassen in developers tools

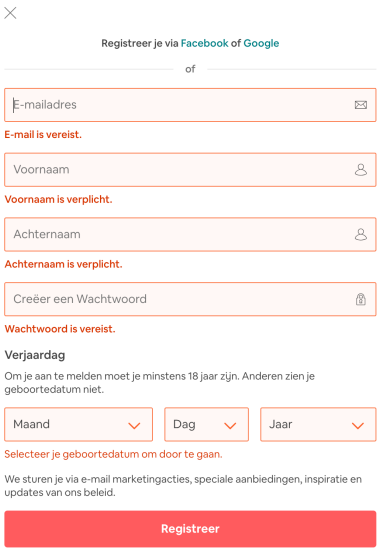
2.12.2 Server side validation

- Voordat het opgeslaan wordt in de database
- Laatste

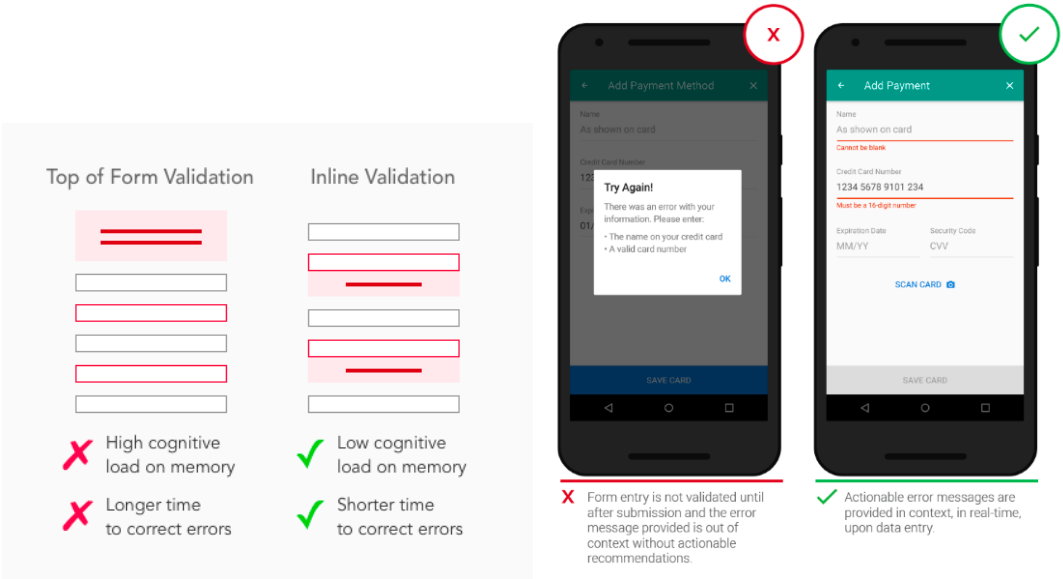
2.12.3 Voorbeelden



Figuur 16



Figuur 17



Figuur 18: Waar validation gebruiken?

2.13 Extra (geen vragen op examen)

2.13.1 States

`https://zellwk.com/blog/style-hover-focus-active-states/`

2.13.2 HTML5 form validation

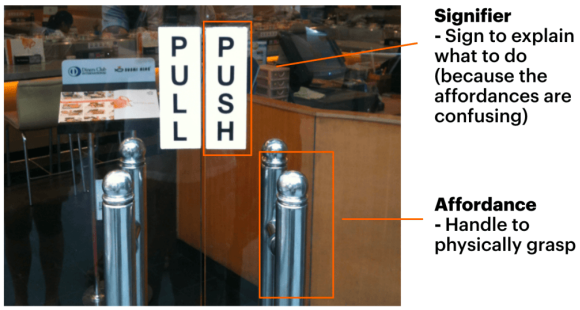
`https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation`

2.13.3 Best practices

- https://www.smashingmagazine.com/2018/08/best-practices-for-mobile-form-design
- https://uxplanet.org/10-rules-for-efficient-form-design-e13dc1fb0e03
- https://uxmovement.com/mobile/stop-misusing-toggle-switches
- https://www.bram.us/2019/01/18/building-better-forms-by-not-taking-away-affordances

3 Affordances

- Een 'affordance' is een aanwijzing dat een object kan worden gebruikt.
- Wat zorgt ervoor dat een interactief component clickable, swipable, pull of pushable is



Figuur 19: Affordance en signifier

https://uxdesign.cc/what-is-an-affordance-6b60f2de79f2

Vormen:

- Explicit affordance
- Pattern affordance
  - Pattern metaphore
- Hidden affordance
- False affordance
- Negative affordance

3.1 Explicit affordance

- Een duidelijke affordance
- Door taal, bv: klik hier om . . . ⇒ signifiers
- Door hoe het er fysiek uitziet
- Makkelijk discoverable

3.1.1 Voorbeelden

- ‘Click to play the video’
- Een grote knop met felle achtergrondkleur en subtiele 3d-vorm

3.2 Pattern affordance

- Impliciet
- Meest voorkomende type
- Maakt het mogelijk om in een complexe interface snel de interactieve onderdelen aan de gebruiker duidelijk te maken
- bv:
  - Navigation bar
  - Links
  - Logo
  - Link in bovenaan rechts
  - Toggle switch

3.2.1 Pattern metaphors

- Metaforische affordance
- Real-world object als metafoor
- Icons
- Pattern metaforen
- Oppassen met heel herkenbare metaforen anders te designen.

3.3 Hidden affordance

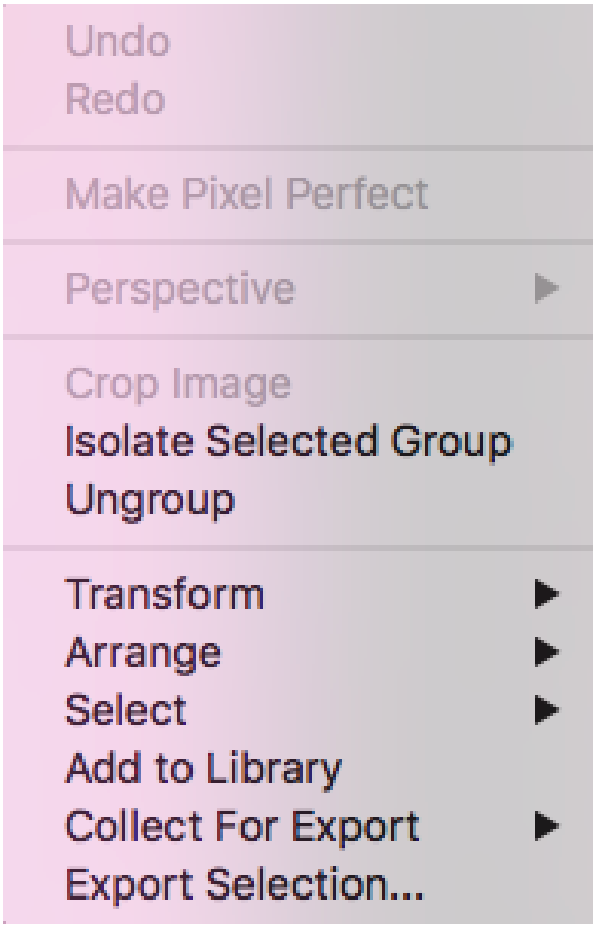
- Verborgen acties
- Standaard wordt de affordance van een element pas onthuld als de gebruiker actie heeft ondernomen
- Om al complexe interfaces minder te clutteren
- Om minder belangrijke acties minder aandacht te geven
- bv:
  - Reveal on hover
  - Swipen van homescreen

3.4 False affordance

- Een woord dat onderlijnd is maar geen link is
- Een groene button die iets verwijderd
- Dark patterns?
- Een grijs woord waar op het eerste zicht geen actie achter zit maar toch een link of button is
- Om minder belangrijke acties minder aandacht te geven

3.5 Negative affordance

- Soms is het nodig om aan te geven dat een UI-element op dit moment geen mogelijkheden biedt
- Grijze elementen
- Buttons die disabled zijn



Figuur 20: De grijze elementen zijn duidelijk niet klikbaar

3.6 Overzicht

Type	Pro's	Cons	Use case
Explicit	Ondubbelzinnig, laag risico dat de gebruiker de affordance mist	Slordige, cluttered interface	niet intuïtieve interacties
Pattern	Brengt de affordance snel en duidelijk over	Vertrouwd op eerdere ervaring met vergelijkbare interfaces	Als patterns stevig zijn gevestigd
Hidden	Clean interface	Mogelijkheden worden mogelijk niet ontdekt	Minder belangrijke interacties
FALSE	None	Alles	Vermijden
Negative	Helpt frustratie te voorkomen door aan te geven dat een element geen affordance heeft	Niet nodig voor elementen die geen affordance hebben.	Alse een UI-element op dit moment geen mogelijkheden biedt.

Figuur 21: Overzicht affordance

4 Micro Interactions

4.1 Wat?

- Micro-interactions geven gebruikers onmiddellijke visuele feedback na het uitvoeren van een actie
- Ze wekken het vertrouwen op dat een uitgevoerde actie heeft plaatsgevonden en dat er als gevolg daarvan iets is gebeurd.
- Voegen een klein beetje 'plezier' toe aan de UI
- Een klein interactief onderdeel van functionaliteit dat precies 1 iets doet
- **Een hover effect is geen micro-interaction!**

4.2 Actie - reactie - feedback

Altijd 3 stappen: actie, reactie en feedback

4.2.1 Voorbeeld: Toggle switch

- Actie: gebruiker duwt op de toggle switch om een setting af te zetten
- Reactie: setting wordt afgezet
- Feedback: de toggle switch beweegt naar de andere kant, de achtergrondkleur verandert

4.3 Versterkt door animatie

- Reactie en feedback is duidelijker

- Transition van de ene state naar de andere
- Maakt het verschil

4.4 Waar gebruiken?

- Show system status
- Highlight changes
- Context behouden
- Calls to action
- Visualiseren van input
- Make tutorials come alive
- Foutmeldingen
- Succesmelding, voltooide acties
- Het tonen van veranderingen

4.5 Voorbeelden

4.5.1 Iphone Mute

- Actie: user wil geluid op mute zetten op telefoon
- Reactie: geluid staat op stil, meldingen, bellen, etc
- Feedback: toestel vibreert

4.5.2 Pull to refresh

- Actie: gebruiker trekt de pagina naar beneden
- Reactie: de pagina refresht
- Feedback: een refresh-icoontje verschijnt bovenaan de pagina

4.5.3 Nightmode

- Actie: het wordt donker
- Reactie: Modus wordt aangepast
- Feedback: kleuren veranderen, helderheid wordt aangepast

4.5.4 Facebook like

- Onderverdeling feedback
  - Uitgebreid: comment
  - Beknopt: Like
- Microinteraction die legendarisch is geworden

4.6 Combinaties van micro-interacties

- Meerdere micro-interactions maken een geheel en tonen wat er volgt op elkaar, en wat de actie ten gevolg heeft.

4.7 Verandering aanduiden

Duidelijke feedback bij:

- Voltooiing van todo
- Verbergen van todo

4.8 Don't

- Niet overdrijven
- Draagt de interactie bij aan het geheel
- Is de animatie niet storend
- Geen micro-interaction maken als er geen/weinig interactie is van de gebruiker

4.9 Meer dan visuele feedback

- Geluid is een belangrijk deel van interactie
- Rekening houden met mute-state
- Met mate & subtiel

4.10 Resources

- http://microinteractions.com/downloads/Microinteractions_Full_Color_Edition_excerpt.pdf
- https://vimeo.com/91559869
- http://microinteractions.com/what-is-a-microinteraction/
- https://dribbble.com/shots/2306244-Hoverboard-shop
- https://dribbble.com/shots/3167358-Microinteractions-for-to-do-list-app
- http://www.uiparade.com/portfolio/simple-toggle/
- https://uxplanet.org/best-practices-for-microinteractions-9456211aeed0
- https://techcrunch.com/2015/08/07/google-maps-new-night-mode-feature-makes-it-easier-to-navigat
- https://www.webdesignerdepot.com/2015/07/7-secrets-for-enhancing-ux-with-micro-interactions/
- http://zurb.com/blog/you-re-thinking-too-big-create-impact-wit
- https://dribbble.com/shots/2764222-Event-App-Concept
- https://dribbble.com/shots/1797373-Pull-Down-To-Refresh
- https://material.io/guidelines/motion/material-motion.html
- https://uxplanet.org/best-practices-for-microinteractions-9456211aeed0

- https://www.facebook.com/360Connext/photos/a.514490188584653.119917.217283351638673/649673881732949/?type=3&theater
- https://dribbble.com/shots/2174325-Alcatel-Watch-IA-Diagram/attachments/400188

5 API-calls, JavaScript basics & debugging

5.1 JavaScript

- Enige client side manier om op het web te werken
- Is geëvolueerd tot mogelijke full-stack oplossing
- Single thread, never blocking ⇒ asynchrone manier van werken

5.1.1 Manier van werken

1. DOM element in een variabele stoppen
2. Eventlistener (interactie), trigger
3. In de eventlistener functie: inhoud schrijven, actie ondernemen

```
// #1 ophalen van DOM element
let button = document.querySelector( '.js-button' );

// #2 Toevoegen interactie en dan iets mee doen
button.addEventListener( 'click', function( e ) {
      console.log( e );
      // #3 Schrijven van inhoud, actie uitvoeren
      aDomElement.innerHTML = 'Button has been clicked.';
});
```

Figuur 22: Manier van werken

5.1.2 Beschikbare structuren

- Objecten (data gestructureerd met naam bijhouden)
- Arrays (behouden een element met een bepaalde positie/nummer)

```
let anObject = {
    fork: 'Fork'
}

let anArray = ['Fork', 'Spoon'];
```

5.2 API calls in JS

5.2.1 Data ophalen in JS

- Via de fetch API
- Fetch API returns een promise (belofte)

```
const serverendpoint = "https://api.example.com/v1/endpoint"

// GET
fetch(serverEndPoint)
```

```
    .then(function (response) {
        console.log(response);
        return response.json();
    })
    .then(function (json) {
        // Callback
        console.log(JSON.stringify(json));
    });
```

5.2.2 Async/await functie

- In een async functie kunnen we een promise afwachten
- Vanaf dat moment kan je weer met een callback werken indien gewenst
- In plaats van '.then' gebruik je 'await', met 'async' in de functiedeclaration

```
const getAPINewWay = async function() {
    const get = await fetch(serverEndPoint, { headers: headers });
    const joke = await get.json();
    console.log({ joke });
};
```

Kan nog beter: we kunnen de promise pas afwachten als de tweede '**then**' voltooid is

```
const data = await fetchData(serverEndPoint);
```

5.2.3 Error handling

Op deze laatste manier van schrijven is de .catch niet beschikbaar om te gebruiken. Meest eenvoudige manier om op te lossen: alles in een try-catch block zetten. Later maken we sowieso een class voor data-access, authentication, etc...

5.3 Debugging

5.3.1 Logging

```
console.log({myVariable, anotherOne});
console.table([arrItems]); // voor arrays kan je console.table() gebruiken
```

5.3.2 Browser breakpoints

In de browser kan je breakpoints zetten om de code op een bepaalde lijn te stoppen, en lijn per lijn te overlopen. Het is ook mogelijk om in de browser (snel) elementen op te vragen en te wijzigen.