

Interaction design

Tuur Vanhoutte

17 december 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	4
1.4.2 Ideal for dark themes	4
1.5 Hoisting	4
1.6 Scoped variables	4
1.6.1 Global scoped variables	5
1.6.2 Local scoped variables	5
1.7 Naming system	6
1.7.1 The two-level theming system	6
1.7.2 Global level	6
1.7.3 Local level	6
1.8 Herhalingsvragen	6
2 Forms	7
2.1 Form	7
2.1.1 Voorbeelden	7
2.1.2 Basic form syntax	7
2.2 Input types	7
2.2.1 HTML5 input types	8
2.2.2 Text-achtigen	8
2.2.3 Time-achtigen	8
2.2.4 Option-achtigen	9
2.3 Toggle or Checkbox?	10
2.3.1 Toggle switch = veredelde checkbox	10
2.3.2 Checkbox	10
2.3.3 Voorbeelden	10
2.3.4 Toggle Switch of Toggle Button?	12
2.4 Textarea	13
2.5 Select	13
2.6 Range	13
2.7 Hors catégorie	13
2.8 Attributes	13
2.8.1 Minimum attributes	14
2.8.2 Veelgebruikte attributes	14
2.8.3 Lege attributes	14
2.9 Labels	15
2.9.1 Label koppelen aan input - Manier 1	15

2.9.2 Label koppelen aan input - Manier 2	15
2.10 Buttons	16
2.10.1 Als input type	16
2.10.2 Als button element	16
2.11 States	16
2.11.1 :hover	16
2.11.2 :active	16
2.11.3 :focus	17
2.12 Validation	17
2.12.1 Client side validation	17
2.12.2 Server side validation	18
2.12.3 Voorbeelden	18
2.13 Extra (geen vragen op examen)	19
2.13.1 States	19
2.13.2 HTML5 form validation	19
2.13.3 Best practices	19
3 Affordances	19
3.1 Explicit affordance	20
3.1.1 Voorbeelden	20
3.2 Pattern affordance	20
3.2.1 Pattern metaphors	21
3.3 Hidden affordance	21
3.4 False affordance	21
3.5 Negative affordance	21
3.6 Overzicht	23
4 Micro Interactions	23
4.1 Wat?	23
4.2 Actie - reactie - feedback	23
4.2.1 Voorbeeld: Toggle switch	23
4.3 Versterkt door animatie	23
4.4 Waar gebruiken?	24
4.5 Voorbeelden	24
4.5.1 Iphone Mute	24
4.5.2 Pull to refresh	24
4.5.3 Nightmode	24
4.5.4 Facebook like	24
4.6 Combinaties van micro-interacties	25
4.7 Verandering aanduiden	25
4.8 Don't	25
4.9 Meer dan visuele feedback	25
4.10 Resources	25
5 API-calls, JavaScript basics & debugging	26
5.1 JavaScript	26
5.1.1 Manier van werken	26
5.1.2 Beschikbare structuren	26
5.2 API calls in JS	26
5.2.1 Data ophalen in JS	26
5.2.2 Async/await functie	27
5.2.3 Error handling	27

5.3	Debugging	27
5.3.1	Logging	27
5.3.2	Browser breakpoints	27
6	Animation	28
6.1	Transitions	28
6.2	Animations	28
6.3	Speed	28
6.4	Duration	28
6.5	Easing	29
6.5.1	Cubic-bezier curves	29
6.5.2	Ease-out	30
6.5.3	Ease	31
6.5.4	Ease-in-out	32
6.5.5	Ease-in-back en Ease-out-back	32
6.6	CSS transition	33
6.6.1	Transition-timing-function	34
6.6.2	Multiple transitions	34
6.7	CSS animation	35
6.7.1	Gelijkwaardige properties:	35
6.7.2	Hoe maak je een CSS animation?	35
6.7.3	Zelfstudie	35
6.8	High performance animations	35
6.9	JavaScript libraries	36
6.9.1	Gsap	36
6.9.2	Anime.js	36
6.9.3	Mo.js	36
6.9.4	Popmotion	36
6.9.5	Lottie + bodymovin	36
6.10	Oefening	36
7	Grid Layout	37
7.1	Basics	37
7.1.1	Grid container	37
7.1.2	Grid layout	37
7.2	Grid line	37
7.3	Grid track	38
7.4	Grid cell	38
7.5	Grid area	38
7.6	Properties	39
7.6.1	Properties voor de parent (grid container)	39
7.6.2	Properties voor de child (grid items)	39
7.7	Grid-gap	40
7.8	Grid-template-columns en grid-template-rows	40
7.8.1	Syntax	40
7.8.2	Automatische line-names	41
7.8.3	Track size special notations	41
7.8.4	grid-x-start, grid-x-end	41
7.9	Grid-template-areas	42
7.9.1	Voorbeeld	42
7.10	Grid-area	43
7.11	Voorbeelden	43

7.12 Zelfstudie	43
7.13 Samenvatting: wat moet je weten?	43
8 Data visualisatie	43
8.1 Waarom data visualisatie	43
8.2 Soort data	44
8.3 Wat wil je tonen?	44
8.4 Chart models	44
8.4.1 Barchart	45
8.4.2 Stacked Barchart	46
8.4.3 Linechart	48
8.4.4 Barchart of Linechart?	50
8.4.5 Piechart	50
8.4.6 Map	52
8.4.7 Scatter plot	53
8.4.8 Bubble chart	54
8.4.9 Radar	55
8.4.10 Interactie & animatie	56
8.5 Resources	57
8.6 Libraries	57
9 Accessibility	57

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

```
1 :root {  
2     --my-cool-color: HotPink;  
3 }
```

1.2.2 Cascading variable: applying the variable

Applying your custom property using the var() function

```
1 .element {  
2     color: var(--my-cool-color)  
3 }
```

1.3 Syntax

```
1 :root {  
2     --my-cool-color: HotPink;  
3 }  
4  
5 p {  
6     color: var(--my-cool-color);  
7 }  
8  
9 .foo {  
10    background-color: var(--my-cool-color);  
11 }
```

1.3.1 CSS Variables Are Case Sensitive

```
1 :root {  
2     --foo: HotPink;  
3     --FOO: #BADA55;  
4 }  
5 p {  
6     color: var(--foo);
```

```
7     }
8     .foo {
9         background-color: var(--FOO);
10    }
```

1.3.2 This is wrong

```
1 .test {
2     --side: margin-top;
3     var(--side): 20px
4 }
```

1.3.3 Use the calc() function to do math

```
1 :root {
2     --whitespace: 20px;
3     --whitespace-lg: var(--whitespace) * 2; /* won't work */
4     --whitespace-lg: calc(var(--whitespace) * 2); /* correct */
5 }
```

1.3.4 Kan ook shorthand values bevatten

```
1 :root {
2     --transition: all .1s ease-out;
3 }
4 a {
5     transition: var(--transition);
6 }
```

1.3.5 Kan ook bestaan uit andere variables.

```
1 :root {
2     --transition-property: all;
3     --transition-duration: .1s;
4     --transition-timing-function: ease-out;
5     --transition: var(--transition-property)
6             var(--transition-duration)
7             var(--transition-timing-function);
8 }
9
10 a {
11     transition: var(--transition);
12 }
```

1.3.6 Default values

```
1 .c-button {
2     border: 1px solid var(--button-color, HotPink);
```

```

3     background-color: transparent;
4 }
5 .c-button:hover {
6     background-color: var(--button-color, HotPink);
7 }
8 .c-button--beta {
9     --button-color: #BADA55;
10}

```

1.3.7 Default values with other variables

```

1 :root {
2     --color-pink: HotPink;
3     --color-badass: #BADA55;
4 }
5 .c-button {
6     border: 1px solid var(--button-color, var(--color-pink));
7 }
8 .c-button:hover {
9     background-color: var(--button-color, var(--color-pink));
10}
11 .c-button--beta {
12     --button-color: var(--color-badass);
13}

```

1.4 Cascade

CSS Variables Are Subject to the Cascade and Inheritance rules

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

```

1 :root {
2     --my-cool-color: HotPink;
3 }
4 /* Alle elementen binnen de root waar je
5 --my-cool-color variable toepast zullen de
6 value HotPink overerven. */
7
8 p {
9     color: var(--my-cool-color);
10}
11
12 .foo {
13     /* Elke paragraaf in het element met
14     de class ".foo" krijgt de kleur #BADA55.*/
15     --my-cool-color: #BADA55;
16 }

```

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

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

```
1  :root {  
2      --whitespace: 1em;  
3  }  
4  @media screen and (min-width: 768px) {  
5      :root {  
6          --whitespace: 2em;  
7      }  
8  }  
9  .c-card {  
10     padding: var(--whitespace);  
11 }
```

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

1.4.2 Ideal for dark themes

```
1  :root {  
2      --color: white;  
3      --background-color: black  
4  }  
5  @media (prefers-color-scheme: dark) {  
6      :root {  
7          --color: black;  
8          --background-color: white  
9      }  
10 }  
11 .html {  
12     background-color: var(--background-color);  
13     color: var(--color);  
14 }
```

<https://codepen.io/simoncoudeville-nmct/pen/eY0xbPp>

1.5 Hoisting

= Accessing a Variable First and Declaring Later

```
1  body{  
2      background: var(--bg-fill);  
3  }  
4  :root{  
5      --bg-fill: green;  
6  }
```

1.6 Scoped variables

Twee soorten:

- Global Scoped Variables

- Local Scoped Variables

1.6.1 Global scoped variables

```

1  :root {
2      --global-color: black;
3 }
```

- :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/0JLBKXq>

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.

```

1  .alert {
2      --alert-color: #222;
3      color: var(--alert-color);
4      border-color: var(--alert-color);
5  }
6  :root {
7      --global-fontSize: 16px;
8  }
9  .c-button {
10     --button-fontSize: var(--global-fontSize);
11     font-size: var(--button-fontSize);
12 }
```

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.

```
1 .c-alert {  
2     /* Component scoped variables are always defined by global variables */  
3     --c-alert--Padding: var(--global--spacer--md);  
4     --c-alert--primary--BackgroundColor: var(--global--primary-color);  
5     --c-alert__title--FontSize: var(--global--secondary-title--fontSize);  
6     /* --block--propertyCamelCase */  
7     padding: var(--c-alert--padding);  
8 }  
9  
10    /* --block--state--propertyCamelCase */  
11 .c-alert--primary {  
12     background-color: var(--c-alert--primary--backgroundColor);  
13 }  
14  
15    /* --block__element--propertyCamelCase */  
16 .c-alert__title {  
17     font-size: var(--c-alert__title--fontSize);  
18 }
```

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

```

1 <form action="">
2   <input type="">
3   ...
4   Submit
5 </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

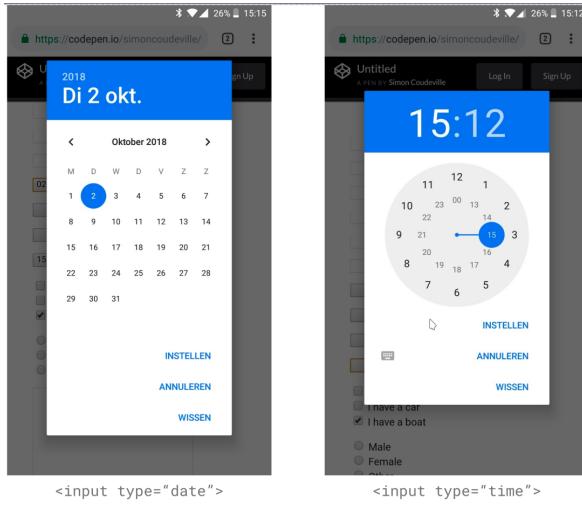
- Browser validatie
- Smartphone keyboard verandert
- Testen op je smartphone:
<https://mobiforge.com/design-development/html5-mobile-web-forms-and-input-types>
- Live demo: <https://codepen.io/simoncoudeville-nmct/pen/gQYqBY>

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



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 een 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/>

Weapon Type:

Archery

Modern Firearm

Muzzleloader

Please select exactly one

Don't use two radio buttons for a single binary choice:

Do you agree to the terms of service for this site?

I agree I don't agree

Use a check box instead:

I agree to the terms of service for this site.

Back Next

Figuur 2

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

2.3 Toggle or Checkbox?



Figuur 3: Links: toggle switch, rechts: checkbox

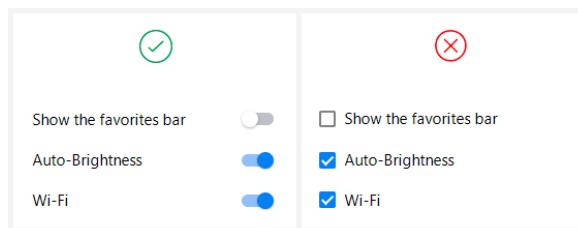
2.3.1 Toggle switch = veredelde checkbox

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

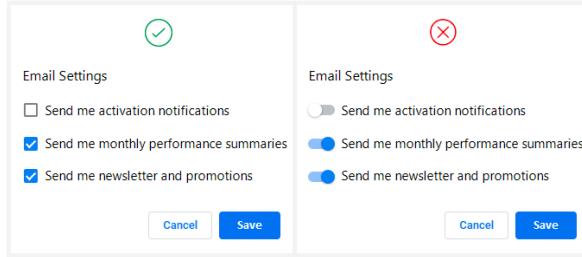
2.3.2 Checkbox

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

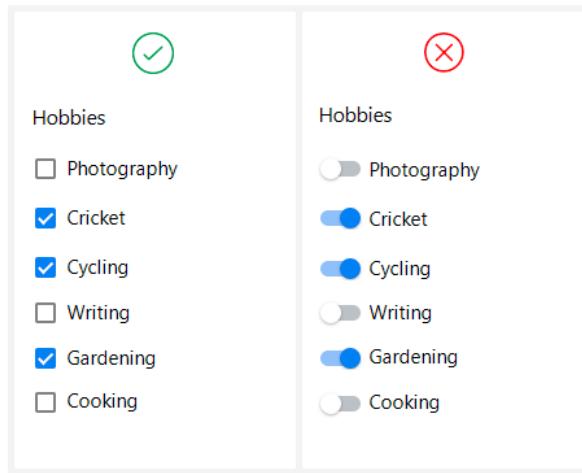
2.3.3 Voorbeelden



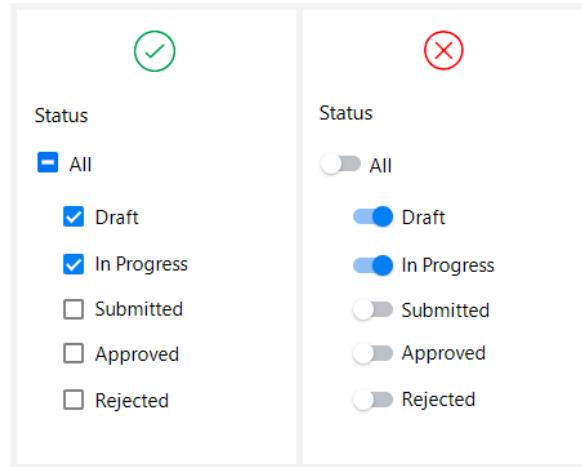
Figuur 4: De opties die een directe reactie vereisen kunnen het best geselecteerd worden met een toggle switch.



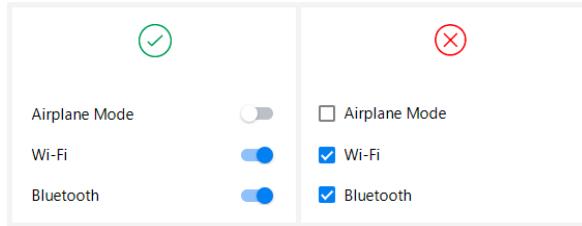
Figuur 5: Checkboxes hebben de voorkeur wanneer een expliciete actie vereist is om instellingen toe te passen.



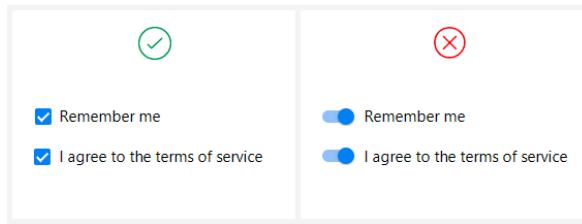
Figuur 6: Selecteren van meerdere opties in een lijst biedt betere ervaring met checkboxes.



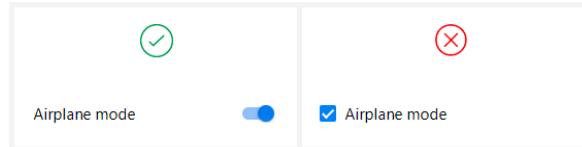
Figuur 7: Indeterminate state wordt het best voorgesteld met een checkbox. (zie 'All' aan de linkerkant)



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.

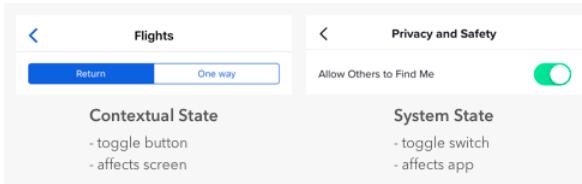
2.3.4 Toggle Switch of Toggle Button?

Toggle buttons = soort van veredelde radio button of meerdere buttons

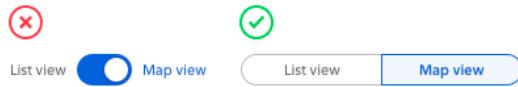
- Contextual state
- Heeft invloed op het huidige scherm
- Opposing options

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

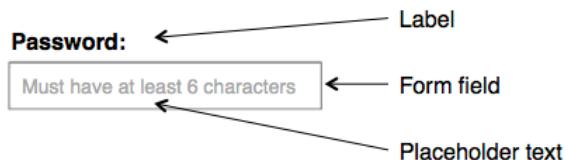
- = Eigenschappen van de input
- bv: type, value, id, name, ...
 - Alle attributes: https://www.w3schools.com/html/html_form_attributes.asp

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

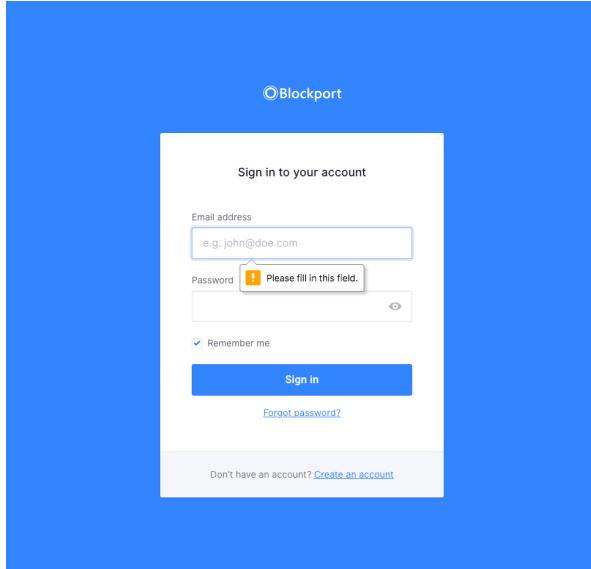
- **value**
 - Kan je gebruiken om een default value in te geven
 - Indien leeg wordt dit wat je hebt ingevuld
- **placeholder**
 - hint
 - voorbeeld van de wat de value moet zijn
 - Verdwijnt automatisch als je begint te typen
 - Placeholder nooit gebruiken als alternatief voor een label!



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/ugnly>

2.9.1 Label koppelen aan input - Manier 1

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

```

1 <label for="input_id">label text</label>
2 <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.

```

1 <label>
2   label text
```

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

2.10 Buttons

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

2.10.1 Als input type

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

2.10.2 Als button element

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

Gebruik het button element!

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

2.11 States

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

Deze volgorde in de CSS is zeer belangrijk!

<https://codepen.io/simoncoudeville-nmct/pen/oNxJbe>

2.11.1 :hover

CSS declarations worden geactiveerd...

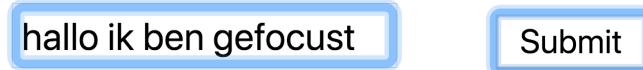
- Op pc: wanneer een gebruiker de muis over een element beweegt.
- Op mobiele toestellen: als een gebruiker een element "indruwt" en "loslaat" er geen specifieke focus of active styles zijn gedeclareerd of als :hover na :focus of :active komt.

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 "indruwt".
- 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 foutberichten
 - Zet foutberichten 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

.....|

Je wachtwoord is niet sterk genoeg. Probeer het langer te maken of voegen symbolen toe zoals !, #, or %.

✗ Sterkte van je wachtwoord: Zwak
 ✓ Mag niet je naam of e-mailadres bevatten
 ✓ Ten minste 8 tekens
 ✗ Bevat een getal of symbol

Figuur 16

Registreer je via Facebook of Google

E-mailadres

E-mail is vereist.

Voornaam

Voornaam is verplicht.

Achternaam

Achternaam is verplicht.

Creëer een Wachtwoord

Wachtwoord is vereist.

Verjaardag

Om je aan te melden moet je minstens 18 jaar zijn. Anderen zien je geboortedatum niet.

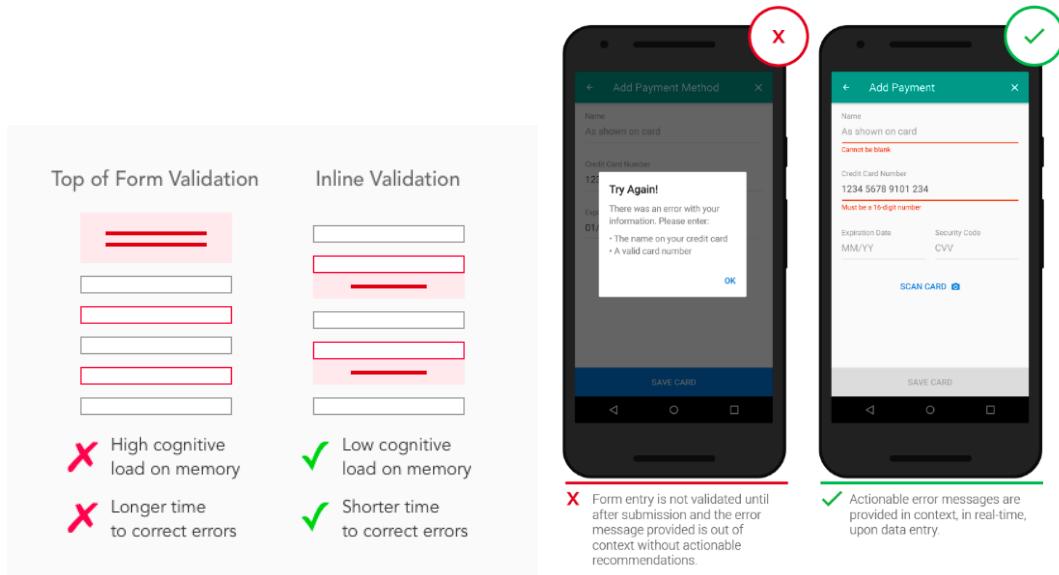
Maand Dag Jaar

Selecteer je geboortedatum om door te gaan.

We sturen je via e-mail marketingacties, speciale aanbiedingen, inspiratie en updates van ons beleid.

Registreer

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

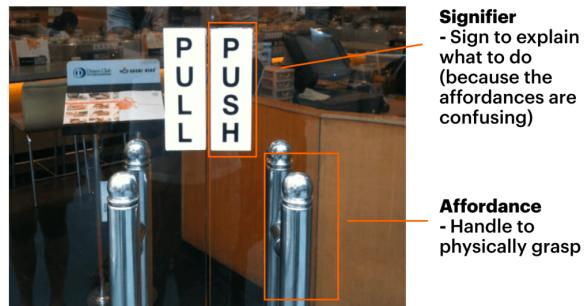
https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Form_validation

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 or pushable is



Figuur 19: Affordance en signifier

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

Vormen:

- Explicit affordance
- Pattern affordance
 - Pattern metaphor
- 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 subtile 3d-vorm

3.2 Pattern affordance

- Implicit
- 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

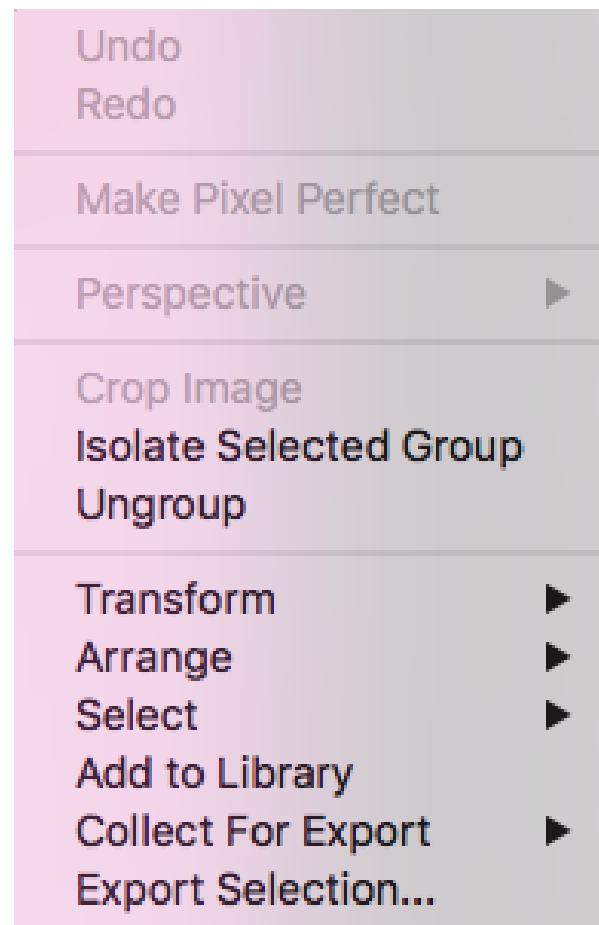
- Verborgen acties
- Standaard wordt de affordance van een element pas onthuld als de gebruiker actie heeft ondernomen
- Om al complexe interfaces minder te clutteren
- Om minder belangrijke acties minder aandacht te geven
- bv:
 - Reveal on hover
 - Swipen van homescreen

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.	Als 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-navigate>
- <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)

```
1 let anObject = {
2     fork: 'Fork'
3 }
4
5 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)

```
1 const serverendpoint = "https://api.example.com/v1/endpoint"
2
```

```

3 // GET
4 fetch(serverEndPoint)
5   .then(function (response) {
6     console.log(response);
7     return response.json();
8   })
9   .then(function (json) {
10    // Callback
11    console.log(JSON.stringify(json));
12 });

```

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

```

1 const getAPINewWay = async function() {
2   const get = await fetch(serverEndPoint, { headers: headers });
3   const joke = await get.json();
4   console.log({ joke });
5 };

```

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

```

1 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

```

1 console.log({myVariable, anotherOne});
2 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.

6 Animation

6.1 Transitions

- Geanimeerde verandering tussen 2 states
- User interaction
- Belangrijk onderdeel van User Interface design
- Micro interactions: hover, clicks, taps, gestures, ...
- <https://codepen.io/rachelcope/pen/raGwPq>
- <https://tympanus.net/Development/Animocons/>

6.2 Animations

- Een getimed animatie, heeft een start en een einde
- Start zonder dat de user iets doet
- De user kan er ook niets aan doen
- Eventueel pauzeren of stoppen
- <https://codepen.io/chrisgannon/pen/EjVyXN>
- <https://codepen.io/sol0mka/pen/og0YJj>

6.3 Speed

Wordt bepaald door:

- Duration
- Easing

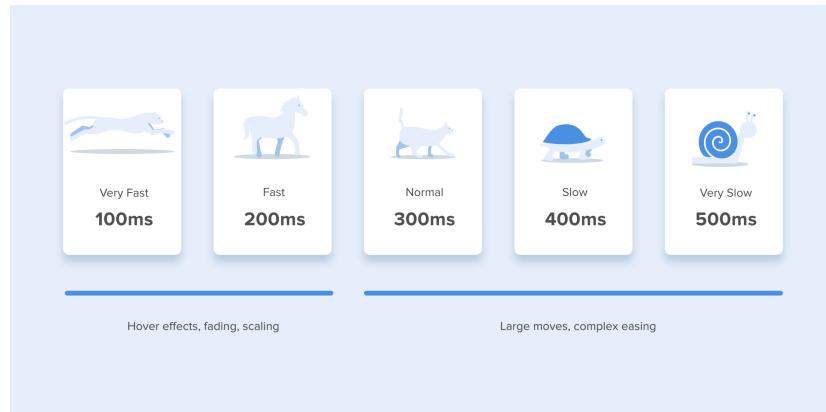
6.4 Duration

- Hoe lang duurt de perfecte transition?
- Niet te lang en niet te kort
- Tussen 200ms en 500ms voor interface elementen
- Onder de 100ms zie je nauwelijks
- Boven de 1s is te traag en hinderlijk
- Afhankelijk van:
 - Grootte object
 - Afstand die ze afleggen
 - Complexiteit
 - Schermgrootte

Tips

- Selection controls have a duration of 100ms:

- <https://material.io/design/motion/speed.html#duration>
- Animated elements that traverse a large portion of the screen have the longest durations
- <https://uxdesign.cc/the-ultimate-guide-to-proper-use-of-animation-in-ux-10bd98614fa9>



Figuur 23

6.5 Easing

- Progressie van de beweging over tijd (duration).
- Specificeert de versnelling of vertraging.
- Duration blijft hetzelfde.
- In plaats van een constant tempo (linear).
- Maakt de animatie natuurlijker.
- Geeft de animatie karakter.
- Alsof elementen beïnvloed worden door natuurlijke krachten zoals wrijving en zwaartekracht.
- 1 van de essentiële principes van animatie:
- <https://vimeo.com/93206523>

Tips

- Transitions without easing look stiff and mechanical
- <https://material.io/design/motion/speed.html#easing>
- <https://uxdesign.cc/the-ultimate-guide-to-proper-use-of-animation-in-ux-10bd98614fa9>

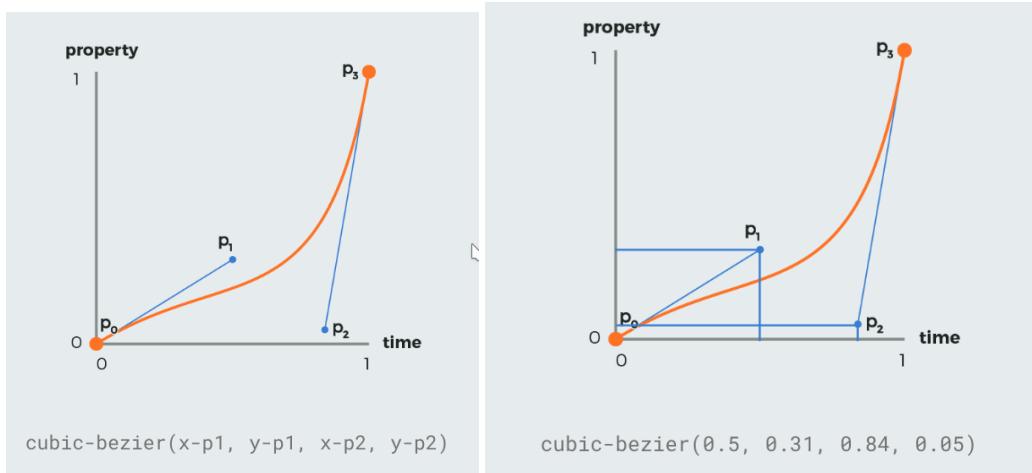
6.5.1 Cubic-bezier curves

- Visualiseren de progressie van de beweging over tijd (duration).
- Timing functions
- 4 punten:
 - p0 en p3 = begin en einde

- p_1 en p_2 = control points
- $\text{cubic-bezier}(x-p1, y-p1, x-p2, y-p2)$

Voorbeelden

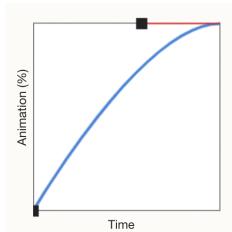
<https://callmenick.com/dev/level-up-animations-cubic-bezier/>



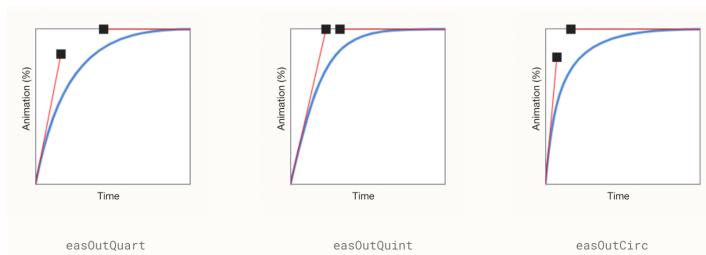
Figuur 24: Cubic-bezier function

6.5.2 Ease-out

- Decelerate easing
- Begint snel, vertraagt naar het einde
- Ideaal voor inkomende elementen

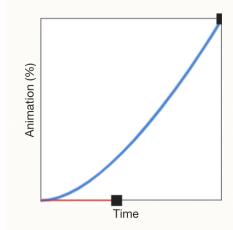


Figuur 25: Ease-out

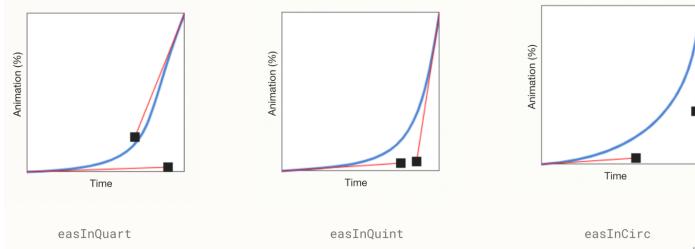


Figuur 26: Ease-out extra functions

- Accelerate easing
- Begint traag
- Versnelt naar het einde
- Ideaal voor uitgaande elementen



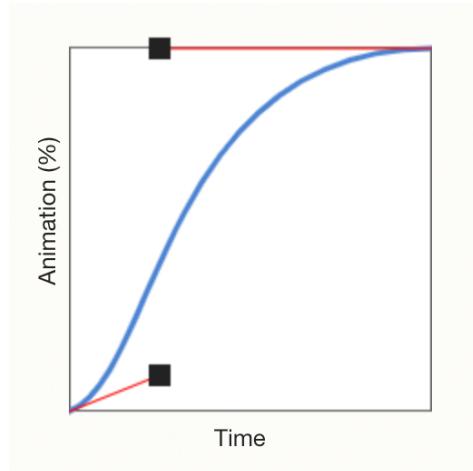
Figuur 27: Ease-in



Figuur 28: Ease-in extra functions

6.5.3 Ease

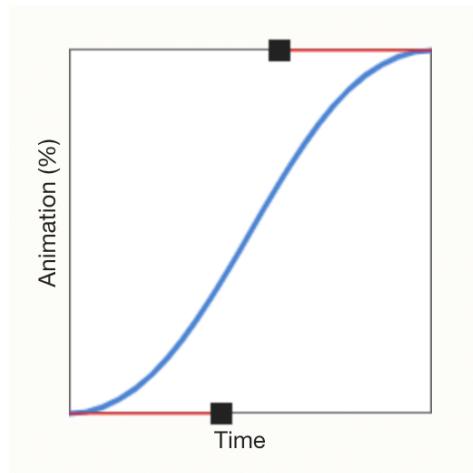
- Meestvoorkomende soort easing
- Easing zowel op begin als einde
- Legt de nadruk op het einde van de animatie
- Geeft meer tijd aan de vertraging dan aan de versnelling.



Figuur 29

6.5.4 Ease-in-out

- Begin en einde versnellen en vertragen even snel
- Tennis effect
- Heeft een specifiek karakter
- Ook niet voor elke situatie

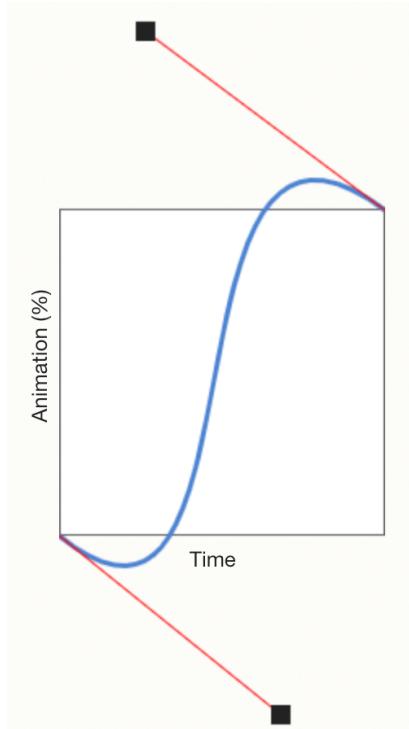


Figuur 30: Ease-in-out

6.5.5 Ease-in-back en Ease-out-back

- **Bounce effect**
- Maakt het nog levendiger (anticipation)
- Niet voor elke situatie
- Enkel voor beweging (niet voor color changes)

- Negatieve y-waarde



Figuur 31: Ease-in/out-back

6.6 CSS transition

- Automatische animatie tussen 2 verschillende property values.
- Keert automatisch terug als de animatie nog niet helemaal gedaan is
- Altijd op de initiële state declareren.
- <https://codepen.io/simoncoudeville-nmct/pen/bQNMQK>

Voorbeelden

```

1 .transition {
2   // Welke property er moet geanimeerd worden (default: all, optioneel)
3   transition-property: scale;
4
5   // duur van de animatie, in miliseconden of seconden
6   transition-duration: 300ms;
7
8   // easing (optioneel)
9   transition-timing-function: ease-in;
10
11  // vertraging vooraleer de animatie start
12  transition-delay: 1s;
13
14  // transition property: shorthand voor:

```

```

15     transition: [transition-property] [transition-duration]
16         [transition-timing-function] [transition-delay];
17 }
```

Basic voorbeeld:

https://www.w3schools.com/cssref/tryit.asp?filename=trycss3_transition-property

6.6.1 Transition-timing-function

- Easing
- Standard easing names: (zijn ook cubic-bezier functions)
 - ease (default)
 - linear
 - ease-in
 - ease-out
 - ease-in-out
- Custom cubic-bezier (x-p1, y-p1, x-p2, y-p2)
- <https://matthewlein.com/tools/ceaser>

Transition-timing-function step values

- **step-start**: the transition jumps instantly to the final state
- **step-end**: the transition stays at the initial state until the end, when it instantly jumps to the final state
- **step(4, end)**: by using steps() with an integer, you can define a specific number of steps before reaching the end. The state of the element will not vary gradually, but rather **jump** from state to state in separate instants
- <https://cssreference.io/property/transition-timing-function/>
- Wordt niet vaak gebruikt

6.6.2 Multiple transitions

```

1 .transition {
2   transition: width 1s ease-in, height 2s ease-out, background-color 3s ease-in-out, transform 4s
3
4   // of:
5
6   transition-property: width, height, background-color, transform;
7   transition-duration: 1s, 2s, 3s, 4s;
8   transition-timing-function: ease-in, ease-out, ease-in-out, linear;
9 }
```

<https://codepen.io/simoncoudeville-nmct/pen/rQ0mGB>

6.7 CSS animation

- Syntax om een getimed animatie in CSS te schrijven.
- Niet gebruiken voor transitions:
 - Animations keren niet vanzelf terug: <https://codepen.io/simoncoudeville-nmct/pen/YRyQeJ>

6.7.1 Gelijkaardige properties:

- animation-duration
- animation-timing-function: werkt met dezelfde names en cubic-beziers.
 - gebeurt per keyframe!

<https://jaketrent.com/post/css-animation-timing-function-per-keyframe-segment/>

6.7.2 Hoe maak je een CSS animation?

- Geef de animatie een naam
- Definieer de animation in @keyframes
- Roep de animation-naam op in een element en bepaal hoe het wordt geanimeerd.
- Basic voorbeeld: <https://codepen.io/simoncoudeville-nmct/pen/Zmbqvd>

6.7.3 Zelfstudie

- <https://robots.thoughtbot.com/css-animation-for-beginners>
- <https://cssreference.io/property/animation-delay/>
- <https://codepen.io/collection/nbEZgX/>

6.8 High performance animations

- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties
- Welke properties kan je best gebruiken om te animeren?
- <https://www.html5rocks.com/en/tutorials/speed/high-performance-animations/>
- Als het kan:
 - opacity
 - transform: translate
 - transform: rotate
 - transform: scale
- Gebruik de will-change property om de browser te waarschuwen wat geanimeerd zal worden

<https://developers.google.com/web/fundamentals/design-and-ux/animations/animations-and-performance>
<https://medium.com/outsystems-experts/how-to-achieve-60-fps-animations-with-css3-db7b98610108>

6.9 JavaScript libraries

- Niet alles is mogelijk met CSS
- Complexe easing functions
- Timeline control
- Transform properties apart animeren
- Complex sequencing
- Performantie
- Browser compatibility
- ...

6.9.1 Gsap

- <https://greensock.com/gsap/>
- <https://greensock.com/showcase>

6.9.2 Anime.js

- <https://animejs.com/>

6.9.3 Mo.js

- <https://github.com/mojs/mojs>
- <https://tympanus.net/Development/Animocons/>

6.9.4 Popmotion

- <https://popmotion.io/>

<https://www.npmtrends.com/animejs-vs-gsap-vs-mo-js-vs-popmotion>

6.9.5 Lottie + bodymovin

- Render After Effects animations natively on Web, Android and iOS, and React Native
- <https://github.com/airbnb/lottie-web>
- <http://airbnb.io/lottie/#/>
- <https://codepen.io/airnan/>
- Volgend semester in Web/App

6.10 Oefening

- <https://codepen.io/simoncoudeville-nmct/pen/RqwzLw>
- Probeer de ball te laten springen zoals in het voorbeeld.
- Met high performance animation properties en de will-change property

7 Grid Layout

7.1 Basics

- Flexbox = simpele 1-dimensionale layouts
- CSS Grid = complexe 2-dimensionale layouts
 - Columns & Rows
 - Global layouts, dashboards, ...
 - Krachtig en uitgebreid
- Browser support: sinds 2017
- CSS tricks: <https://css-tricks.com/snippets/css/complete-guide-grid/#prop-grid-auto-columns-rows>
- Playground: <https://codepen.io/simoncoudeville-nmct/pen/JeMPPZ>

7.1.1 Grid container

= parent

- Bepaalt de layout
- <https://codepen.io/simoncoudeville-nmct/pen/WNNPqQg>

7.1.2 Grid layout

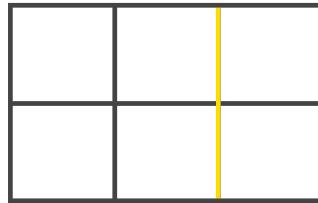
- Volgen de layout
- Default source order van de children maar laat toe om de order zowel verticaal als horizontaal te veranderen.

```
1  <!-- Container is de grid container -->
2  <div class="container">
3      <div class="item item-1"></div>
4      <div class="item item-2"></div>
5      <div class="item item-3"></div>
6  </div>
7
8  <!-- Item elementen zijn grid-items. Subitems niet -->
9  <div class="container">
10     <div class="item"></div>
11     <div class="item">
12         <p class="sub-item"></p>
13     </div>
14     <div class="item"></div>
15 </div>
```

7.2 Grid line

- De scheidingslijnen die de structuur van het grid bepalen
- Vertical: column grid line

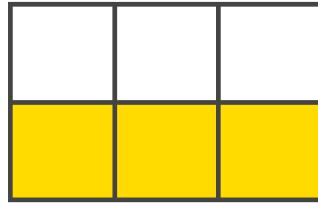
- Horizontal: row grid line



Figuur 32: Column grid line

7.3 Grid track

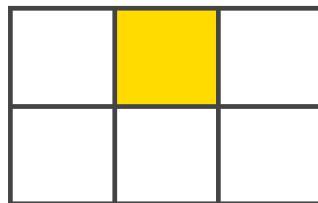
- De ruimte tussen 2 opeenvolgende grid lines
- Vertical: columns
- Horizontal: rows



Figuur 33: Grid track tussen de 2de en de 3de row grid lines

7.4 Grid cell

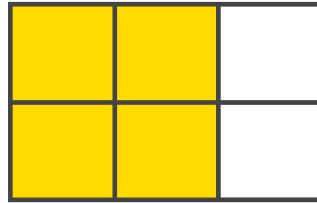
- De ruimte tussen 2 opeenvolgende row & column grid lines



Figuur 34: Grid cell tussen row grid lines 1 & 2, en column grid lines 2 & 3.

7.5 Grid area

- De ruimte tussen 2 row & column grid lines
- Kan meerdere grid cells bevatten



Figuur 35: Grid area tussen row grid lines 1 & 3, en column grid lines 1 & 3.

7.6 Properties

7.6.1 Properties voor de parent (grid container)

- display: grid | inline-grid
- **Grid tracks, lines, en areas definieren:**
 - grid-template-columns
 - grid-template-rows
 - grid-template-areas
 - grid-template (shorthand)
- **Gap (gutter)**
 - grid-column-gap
 - grid-row-gap
 - grid-gap (shorthand)
- **Alignment**
 - justify-items
 - align-items
 - place-items
 - justify-content
 - align-content
 - place-content
- **Auto-generated grid tracks**
 - grid-auto-columns
 - grid-auto-rows
 - grid-auto-flow
- grid: (shorthand)

7.6.2 Properties voor de child (grid items)

- **Locatie binnen de container**
 - grid-column-start

- grid-column-end
- grid-row-start
- grid-row-end
- grid-column
- grid-row
- grid-area
- **Alignment:**
 - justify-self
 - align-self
 - place-self

7.7 Grid-gap

- = De gutter tussen opeenvolgende items (niet links of rechts, boven of onder)
- grid-column-gap: 4px;
- grid-row-gap: 8px;
- grid-gap: <grid-row-gap> <grid-column-gap>

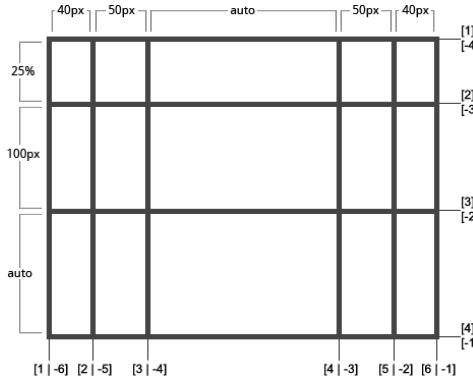
7.8 Grid-template-columns en grid-template-rows

- Definieren de columns en rows
- Meerdere values gescheiden door spaties
- De values staan voor de track size
- De spaties staan voor de grid lines

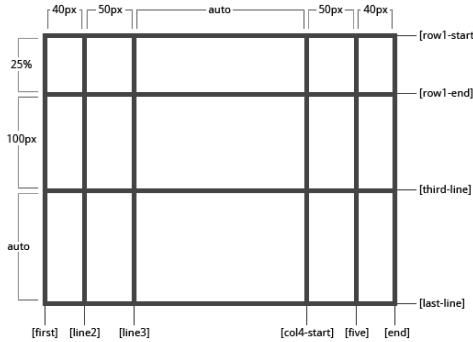
7.8.1 Syntax

- **track-size:** auto, px, vh, vw, % of een fraction van de overgebleven ruimte (**fr**)
- **line-name:** automatische positieve en negatieve nummers. Maar je kan die zelf een naam geven om bij een item op te roepen

7.8.2 Automatische line-names



Figuur 36: **grid-template-columns**: 40px 50px auto 50px 40px;
grid-template-rows: 25% 100px auto;



Figuur 37: **grid-template-columns**: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];
grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line];

7.8.3 Track size special notations

- 1fr: fraction unit = 1 keer de overgebleven ruimte
- auto: neemt de breedste/hoogste content als basis voor de breedte/hoogte
- repeat(12, 1fr): maakt 12 kolommen aan van 1 fraction unit
- minmax(260px, 1fr): maakt 1 column of row aan van minimum 260px en maximum 1 fraction unit
- auto-fit en auto-fill keywords: responsive grid system zonder media queries!
- Combinaties: grid-template-columns: minmax(150px, 1fr) auto 1fr;
 - Dit maakt 1 column aan van minimum 150px en maximum 1fr, 1 column die automatisch de breedte van de content krijgt en 1 column die altijd 1 fraction unit is.

7.8.4 grid-x-start, grid-x-end

Waarbij x == column of row

- Bepaalt de locatie van een grid items binnen het grid door te refereren naar grid lines
- Values:
 - <line>: positief of negatief nummer of de zelfgekozen grid line name
 - span <number> of <name>: hoeveel grid tracks moet het element innemen
- Bijvoorbeeld:
 - grid-column-start: 2;
 - grid-column-end: span 2;
 - grid-column: 2 / span 2;

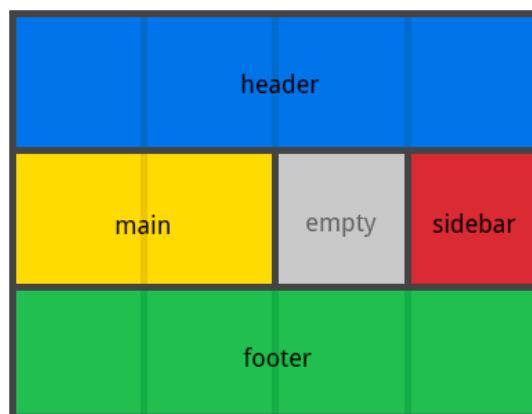
7.9 Grid-template-areas

- Laat toe om namen te geven aan grid areas
- Visuele representatie van je grid
- Values
 - **<grid-area-name>** = een zelfgekozen naam
 - . = een lege grid cell
 - **none** = geen grid lines

7.9.1 Voorbeeld

```

1 .container {
2   grid-template-columns: 50px 50px 50px 50px;
3   grid-template-rows: auto;
4   grid-template-areas:
5     "header header header header"
6     "main main . sidebar"
7     "footer footer footer footer";
8 }
```



Figuur 38: Element met de .container class

7.10 Grid-area

- Laat toe om een grid item op een **voorgedefinieerde** grid area te plaatsen
- (Nog) een kortere manier dus om de volgorde te bepalen

7.11 Voorbeelden

- CSS grid playground: <https://codepen.io/simoncoudeville-nmct/pen/JeMPPZ>
- Responsive dashboard grid: <https://codepen.io/simoncoudeville-nmct/pen/aQmENm>

7.12 Zelfstudie

- <https://css-tricks.com/snippets/css/complete-guide-grid/>
- <https://dev.to/perborgen/want-to-learn-css-grid-here-s-my-free-full-length-course-31ln>
- <https://css-tricks.com/look-ma-no-media-queries-responsive-layouts-using-css-grid/>
- Auto-fill vs auto-fit:
 - <https://css-tricks.com/auto-sizing-columns-css-grid-auto-fill-vs-auto-fit/>
- Minmax function:
 - <https://bitsofco.de/how-the-minmax-function-works/>

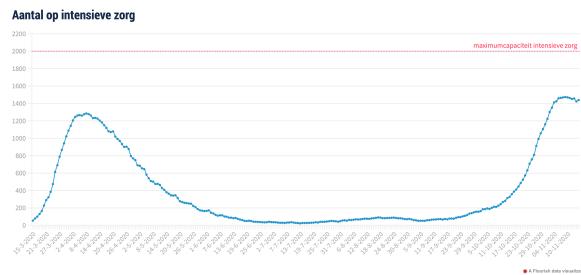
7.13 Samenvatting: wat moet je weten?

- Waarom zou je CSS grid gebruiken ipv flexbox?
- Wat is een grid-area/-line/-cell?
- Met welke property bepaal je witruimte tussen 2 grid cells
- Waar staat fr voor? Wat is 1fr?
- Hoe maak je een grid met 12 kolommen die altijd even breed is?
- Auto-fill vs auto-fit?

8 Data visualisatie

8.1 Waarom data visualisatie

= om getallen in verhalen te veranderen



Figuur 39: Actueel voorbeeld van datavisualisatie

8.2 Soort data

- Time-based data
- Categorized data
- Geographic distribution
- Multi-dimension data (meerdere variabelen per categorie)

8.3 Wat wil je tonen?

- Vergelijkingen
- Proporities
- Tijd
- Geografie
- Relaties
- Correlatie
- Onderverdelingen

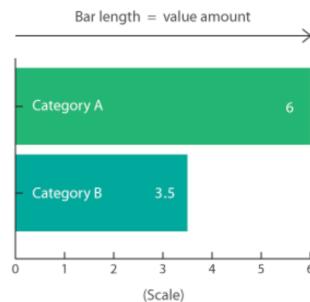
Verschillende manieren om dezelfde data te tonen

8.4 Chart models

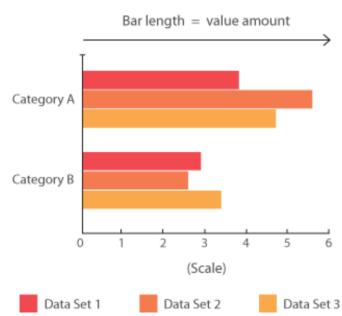
- Barchart & stacked barchart
- Linechart
- Piechart en donut chart
- Map
- Scatter plot
- Bubble chart
- Radar
- ...
- Combinaties
- Varianten

8.4.1 Barchart

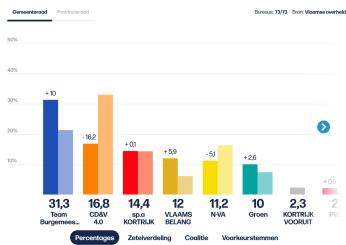
- Horizontale of verticale balken waarvan de lengte de waarde aantoon.
- **Gebruiken voor:**
 - Tijd exploraties
 - Vergelijken van categorieën
 - Correlaties aantonen
- **Soort data:**
 - Time based data
 - Categorized data
- **Niet gebruiken om:**
 - Waarden met verschillende units te vergelijken



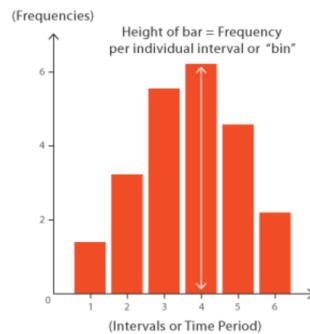
Figuur 40: Gecategorizeerde data



Figuur 41: Gecategorizeerde data



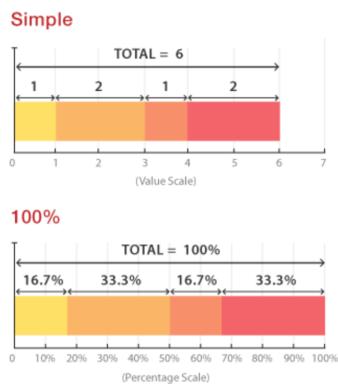
Figuur 42: Gecategoriseerde data



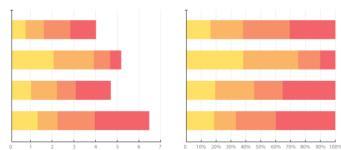
Figuur 43: Histogram: voor tijdsgebaseerde data

8.4.2 Stacked Barchart

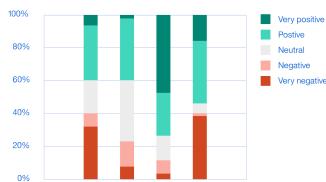
- Elke balk is verdeeld in meerdere categorieën.
- **Gebruiken voor:**
 - Tijd analyses
 - Vergelijken van categorieën
 - Correlaties aantonen
 - Onderverdelingen tonen
- **Soort data:**
 - Time based data
 - Categorized data
- **Niet gebruiken wanneer:**
 - De focus ligt op het vergelijken van de grootte van de individuele categorie
 - De totale som van de elementen in de bar niet relevant is



Figuur 44



Figuur 45



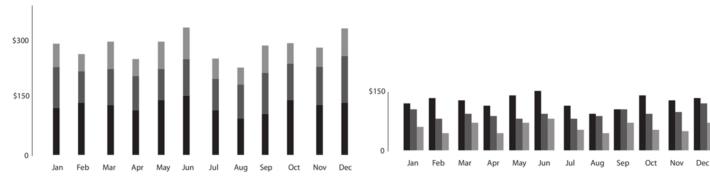
Figuur 46



Figuur 47: Google Drive storage capacity



Figuur 48: Verkiezingsuitslag 2020

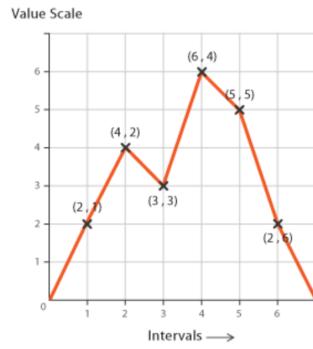


Figuur 49: Verkoopcijfers van een bedrijf met 3 locaties per maand. Links: stacked, rechts, gewone barchart

8.4.3 Linechart

- Serie van data punten geconnecteerd door lijn segmenten.
- **Gebruiken voor:**
 - Tijd analyses (doorlopend)
 - Vergelijken
 - Correlaties aantonen
- **Soort data:**

- Time based data



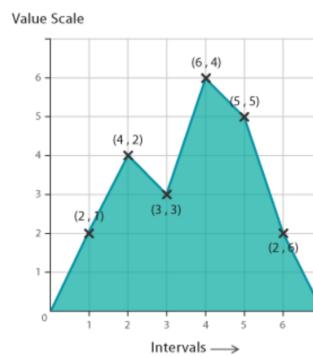
Figuur 50



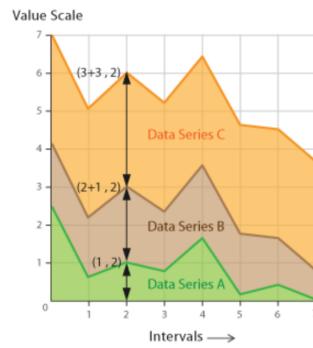
Figuur 51: Grouped linechart



Figuur 52: Corona



Figuur 53: Area graph

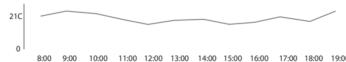


Figuur 54: Stacked area graph

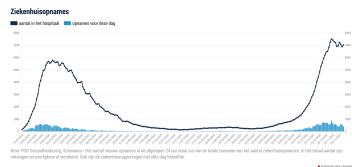
8.4.4 Barchart of Linechart?



Figuur 55: Verkoopcijfers van een bedrijf per maan



Figuur 56: Temperatuur per uur



Figuur 57: Combinatie: verschillende data voorstellen op dezelfde grafiek

8.4.5 Piechart

- Cirkelvormige grafiek verdeeld in segmenten die proporties voorstellen.

- **Gebruiken voor:**

- Vergelijken van categorieën
- Onderverdelingen tonen

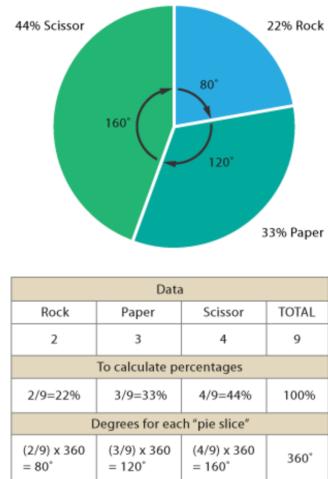
- **Soort data:**

- Categorized data

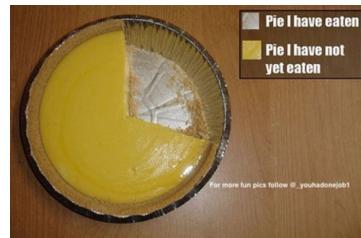
- **Niet gebruiken wanneer:**

- Er meer dan 6 categorieën zijn

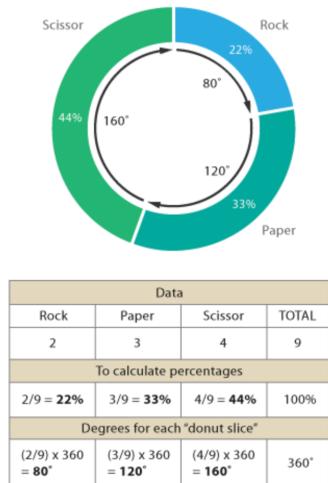
- Enkel te gebruiken als er een totaal is.



Figuur 58



Figuur 59: Most accurate piechart



Figuur 60: Donut chart: gat in het midden van de pie. Je kan de lengte van de bogen lezen.

8.4.6 Map

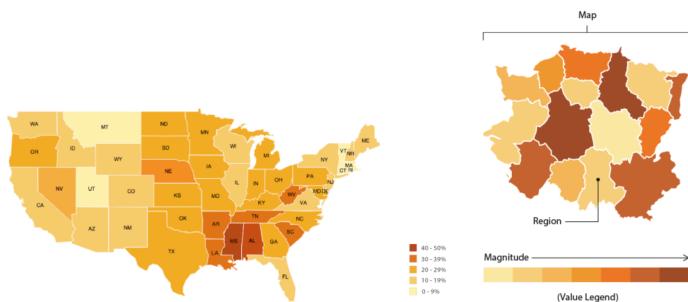
- Cirkelvormige grafiek verdeeld in segmenten die proporties voorstellen.

- **Gebruiken voor:**

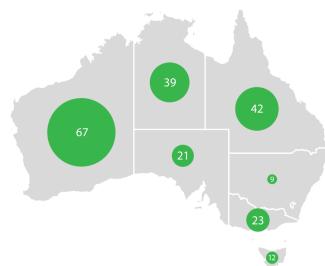
- Geografische verspreiding
- Tijd analyses
- Vergelijken

- **Soort data:**

- Geographic distribution



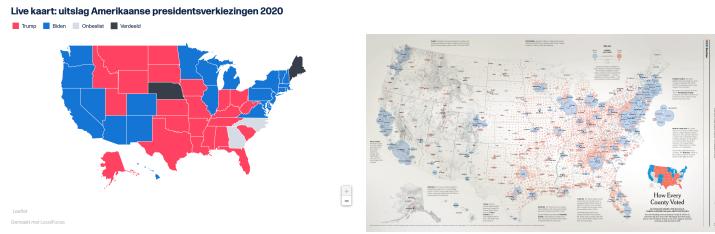
Figuur 61



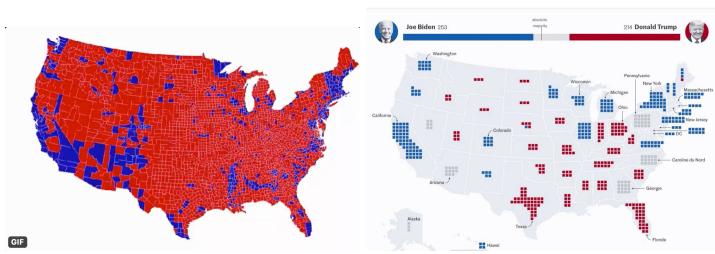
Figuur 62



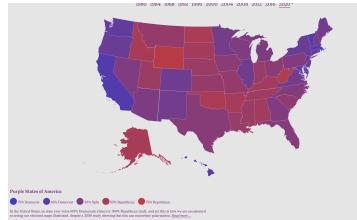
Figuur 63



Figuur 64



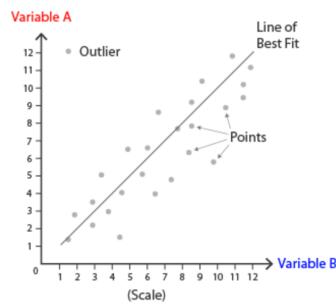
Figuur 65



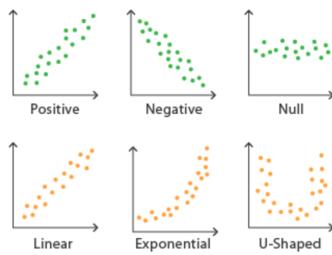
Figuur 66

8.4.7 Scatter plot

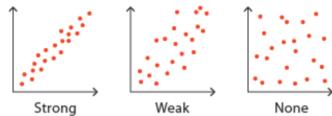
- Een grafiek met uitgezette punten die de relatie tussen twee sets gegevens weergeven.
- **Gebruiken voor:**
 - Grote hoeveelheid data
 - Vergelijken tussen 2 values
 - Correlatie tussen 2 values
 - Tijd analyses
- **Soort data::**
 - categorized data
 - multi-dimension data



Figuur 67



Figuur 68: Types correlatie



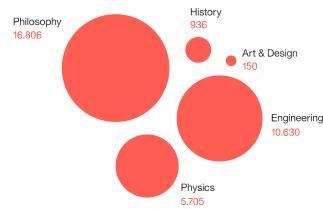
Figuur 69: Correlatiesterkte

8.4.8 Bubble chart

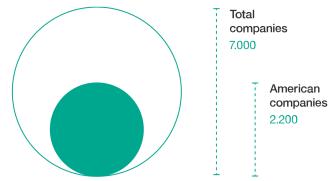
- Multivariabele chart met meerdere variabelen
- Uitgebreide versie van de scatter plot
- Kan zonder axis
- **Gebruiken voor:**
 - Alternatief voor bar chart
 - Vergelijken van categorieën door properties
 - Onderverdelingen duidelijk maken
- **Soort data:**
 - categorized data
 - Multi-dimensional data



Figuur 70



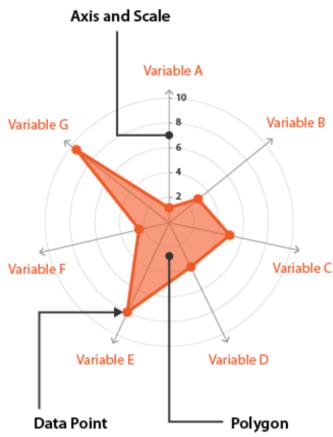
Figuur 71: Aantal studenten per faculteit



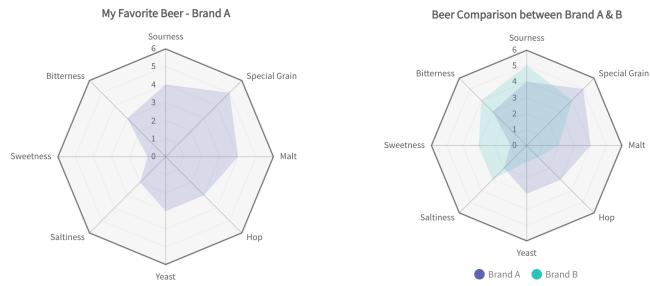
Figuur 72: Aantal fabricage bedrijven

8.4.9 Radar

- Diagram dat wordt gebruikt om gelijktijdig waarden van meerdere indicatoren weer te geven.
- **Gebruiken voor:**
 - het vergelijken van meerdere kwantitatieve variabelen
 - correlaties
- **Soort data:**
 - categorized data



Figuur 73



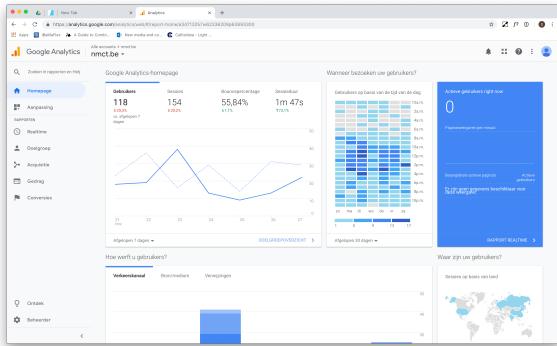
Figuur 74

8.4.10 Interactie & animatie

- Groot deel van data visualisatie
- Help om vergelijkingen nog duidelijker te maken
- Versterkt het verhaal

Voorbeelden:

- [https://www.gapminder.org/tools/#\\$state\\$time\\$value=1891; ;&chart-type=bubbles](https://www.gapminder.org/tools/#$state$time$value=1891; ;&chart-type=bubbles)
- <https://www.vrt.be/vrtnws/nl/2019/05/27/bekijk-hier-alle-verkiezingsresultaten/#/8/9/2000/kaart>



Figuur 75: Google Analytics

8.5 Resources

- <https://www.reddit.com/r/dataisbeautiful>
- <https://www.gapminder.org/answers/how-did-the-world-population-change/>
- <https://www.gapminder.org/answers/how-does-income-relate-to-life-expectancy/>
- <https://datavizcatalogue.com/search.html>
- <https://www.ibm.com/design/v1/language/experience/data-visualization/>

8.6 Libraries

- <https://thenextweb.com/dd/2015/06/12/20-best-javascript-chart-libraries/>

9 Accessibility

- <https://www.boia.org/blog/what-are-the-four-major-categories-of-accessibility>
- <https://www.24a11y.com/2019/pixels-vs-relative-units-in-css-why-its-still-a-big-deal>
- <https://www.smashingmagazine.com/2019/02/buttons-user-interfaces/>
- <https://css-tricks.com/a-complete-guide-to-links-and-buttons/>