Web Development I

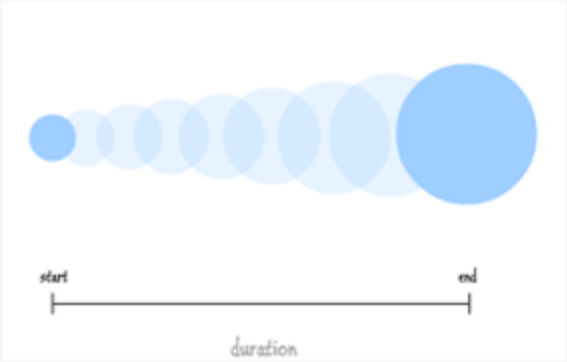
Les 10: Animaties

HO
GENT

Animaties

- Animatie is de illusie van beweging door het na elkaar afspelen van verschillende stilstaande beelden, zogenaamde *frames*.
  BRON: https://nl.wikipedia.org/wiki/Animatie_(media)
- Er zijn twee manieren om animaties toe te voegen aan een webpagina met behulp van CSS.
  CSS transitions en **CSS animations**.

CSS transitions



Theoriebestanden (voorbeelden)

- Open in Visual Studio Code
  de map **01-css-transitions**

HO GENT

1-inleiding-css-transitions-javascript.html

- Normaal zal als je de waarde wijzigt van een CSS property de webpagina onmiddellijk bijgewerkt worden met de nieuwe waarde.

- Met een CSS transition kan je ervoor zorgen dat als de waarde van een CSS property wijzigt, de overgang van de oude naar de nieuwe waarde geleidelijk verloopt over een zekere tijdsperiode. De browser zal hierbij automatisch de tussenliggende beelden (frames) tekenen en zo een animatie creëren.

HO GENT

1-inleiding-css-transitions-javascript.html

```
#my-div {
  ...
  width: 100px;
  /* CSS transition:
  om een transitie-effect te bekomen
  moeten we minimaal de transition-duration instellen */
  transition-duration: 2s;
}

// Als er op button1 geklikt wordt wijzigen we de
// breedte naar 300px
button1.onclick = () => myDiv.style.width = '300px';
```

2-inleiding-css-transitions-hover.html

- In plaats van de breedte van het div-element te wijzigen met JavaScript kunnen we de breedte ook wijzigen met CSS door gebruik te maken van de **:hover** pseudo class.
- Als we de breedte van het div-element wijzigen naar 300px als er gehoverd wordt over het element, dan zal deze wijziging opnieuw onmiddellijk gebeuren, tenzij we een CSS transition instellen.
  Op de volgende dia is er een CSS transition toegevoegd zodat de overgang naar de 300px geleidelijk zal verlopen en 2 seconden zal duren. De browser zal opnieuw automatisch de tussenliggende frames tekenen en zo een illusie van beweging creëren.

2-inleiding-css-transitions-hover.html

```
div {
  ...
  width: 100px;
  /* CSS transition:
  om een transitie-effect te bekomen
  moeten we minimaal de transition-duration instellen */
  transition-duration: 2s;
}

/* als er gehoverd wordt over het div-element
wijzigen we zijn breedte. */
div:hover {
  width: 300px;
}
```

CSS transition properties

- De CSS properties die je kan instellen voor een transition (overgang), met hun beginwaarden, zijn:

```
transition-property: all;
transition-duration: 0s;
transition-timing-function: ease;
transition-delay: 0s;
```

3-css-transition-properties.html

```
 div {
   ...
   width: 100px;
   background-color: lightgreen;
 }
 /* CSS transition */
 div {
   transition-property: width;
   transition-duration: 2s;
   transition-timing-function: linear;
   transition-delay: 1000ms;
 }
 div:hover {
   width: 300px;
   background-color: lightpink;
 }
```

transition-duration property

- Deze property definieert de duur van de transitie uitgedrukt in milliseconden of seconden, m.a.w hoe lang het duurt om de overgang van de oude naar de nieuwe waarde te maken.
- De **transition-duration** is standaard 0 seconden wat betekent dat de transitie onmiddellijk gebeurt en er geen animatie is. **Bijgevolg moeten we minstens de transition-duration instellen om de transitie te zien.** Het is tevens 'best practice' om ook de **transition-property** property in te stellen.

  Voorbeeld: `transition-duration: 2s;`

transition-property property

- Bij het instellen van een transition op een element kan je via deze property instellen voor welke CSS property(ies) er een transitie-effect moet toegepast worden.

  De beginwaarde voor deze property is `all`, zodat er standaard een transitie-effect toegepast wordt op alle CSS properties.

Voorbeeld: `transition-property: width;`

transition-timing-function property

- Bij animaties is het belangrijk om de snelheid waarmee een waarde wijzigt te kunnen controleren. Bijvoorbeeld traag in het begin en geleidelijk sneller. Zo kunnen we een meer natuurlijke beweging creëren. Hiervoor dient de **transition-timing-function** property. De waarde voor deze property is een <easing-function> die de versnellingscurve definieert.
- De standaardwaarde is `ease` en definieert een meer natuurlijke beweging dan de waarde `linear`. Om verschillende waarden met elkaar te vergelijken kan je gebruik maken van Example Easing function comparison op MDN.

Voorbeeld: `transition-timing-function: linear;`

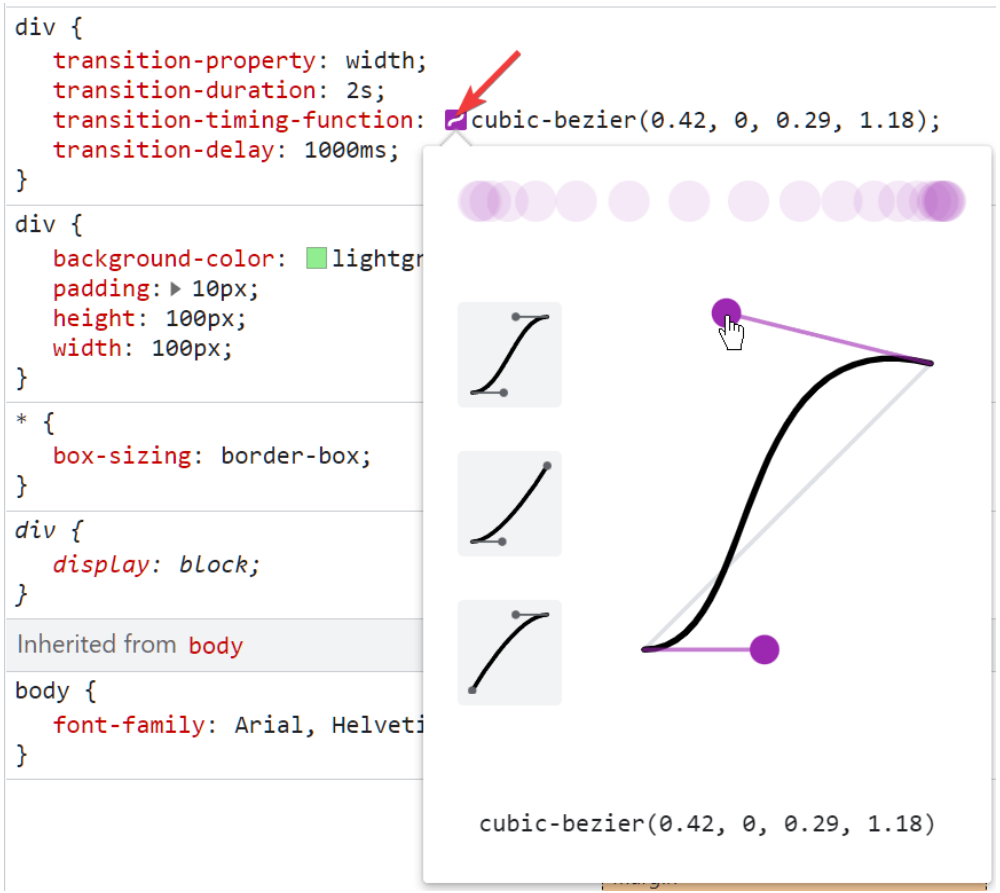
Easing functions

- Meer geavanceerde timing functions kunnen gemaakt worden aan de hand van een cubic-bezier function

```
transition-timing-function: cubic-bezier(0.99, 0.21, 0, 0.85);
```

- Je kan een cubic-bezier function visualiseren en eventueel aanpassen met de Chrome of Firefox Dev Tools (zie schermafbeelding op de volgende dia en onderstaande procedure)
  - Stel een waarde voor de **transition-timing-function** in, zoals **ease**.
  - Open de **Chrome Dev Tools** en selecteer het element met de transition.
  - Klik op het icoontje bij de **transition-timing-function** om de **cubic bezier editor** te openen.
  - Kies een andere waarde of gebruik de hendels om de curve aan te passen

Cubic bezier editor in de Chrome Dev Tools



HO GENT

transition-delay property

- Bepaalt hoe lang er moet gewacht worden met het starten van het transitie-effect nadat de waarde van een property gewijzigd is.
- De standaardwaarde is `0s` en betekent dat het transitie-effect onmiddellijk start nadat de property gewijzigd is.

Voorbeeld: `transition-delay: 800ms;`

Meerdere transities instellen

- Als we meerdere CSS properties wijzigen dan kunnen we voor elke CSS property apart de transition properties instellen. Standaard zullen alle transities dezelfde waarden gebruiken.

```
div {
  transition-property: opacity, height;
  transition-duration: 3s, 5s;
}
```

De Transition shorthand property

- Zie MDN-transition shorthand property.
- Het voorbeeld op de vorige dia kan dus korter geschreven worden als:

```
div {
  transition: opacity 3s, height 5s;
}
```

Transitions op element en element:hover

- Wanneer je een transition instelt op de element selector, dan wordt bij het hoveren de transitie zowel voorwaarts (bij het binnengaan) als achterwaarts (bij het verlaten) toegepast.
- Wanneer je de transition instelt op de element:hover selector, dan wordt de transitie enkel voorwaarts (bij het binnengaan) toegepast. Bij het verlaten keert het element onmiddellijk terug naar zijn begintoestand.
- Het is ook mogelijk om verschillende ‘transition-*’ properties in te stellen voor de voorwaartse en de achterwaartse transitie.
  De ‘transition-*’ properties voor de voorwaartse transitie stel je dan in op element:hover en deze voor de achterwaartse op het element zelf.

4-css-transition-forward-backward.html

```
div {
  ...
  width: 100px;
  /* Achterwaartse transitie */
  transition: width 200ms;
}

div:hover {
  /* Voorwaartse transitie */
  transition: width 2s;
  width: 300px;
}
```

Animatable CSS properties

- Je kan *transition* niet bij elke CSS property gebruiken. Een lijst met alle CSS properties die kunnen geanimeerd worden vind je op ‘MDN - animatable CSS properties’.

Opmerkingen

- Om waarden te wijzigen zonder gebruik te maken van JavaScript hebben we tot nu toe steeds **:hover** gebruikt, maar je kan hiervoor ook andere pseudo classes gebruiken zoals **:focus** en **:checked** (zie Les 9: Oefening Hamburger menu)

CSS 2D transform

CSS 2D transform

- Een property die veel gebruikt wordt bij CSS transitions (en CSS animations) is de **transform** property (en de bijhorende property **transform-origin**).
- Met de **transform** property kan je een element schalen (scale), roteren (rotate), scheef maken (skew) en verplaatsen (translate).

Theoriebestanden (voorbeelden)

- Open in Visual Studio Code
  de map **02-transform**

HO GENT

Inleiding transform property

HO GENT

1-transform-inleiding.html

- Voeg de volgende CSS-code toe:

```
/* Het tweede div-element 300px naar
rechts verschuiven en 30 graden roteren. */
div:nth-of-type(2) {
  transform: translateX(300px) rotate(30deg);
}
```

CSS 2D transform

- transform:
  - De mogelijke waarden zijn:
    - **none** (is de initiële waarde en betekent dat er geen transformatie is).
    - één of meerdere van de CSS transform functions.
  - voorbeeld: `transform: rotate(20deg) translateX(50px);`
- transform-origin:
  - Stelt de oorsprong in. De initiële waarde is **transform-origin: center center;** (wat overeenkomt met **transform-origin: 50% 50%;**) m.a.w. de oorsprong ligt standaard in het middelpunt van het element.
  - voorbeeld: `transform-origin: 0% 0%; /* x-offset y-offset */`

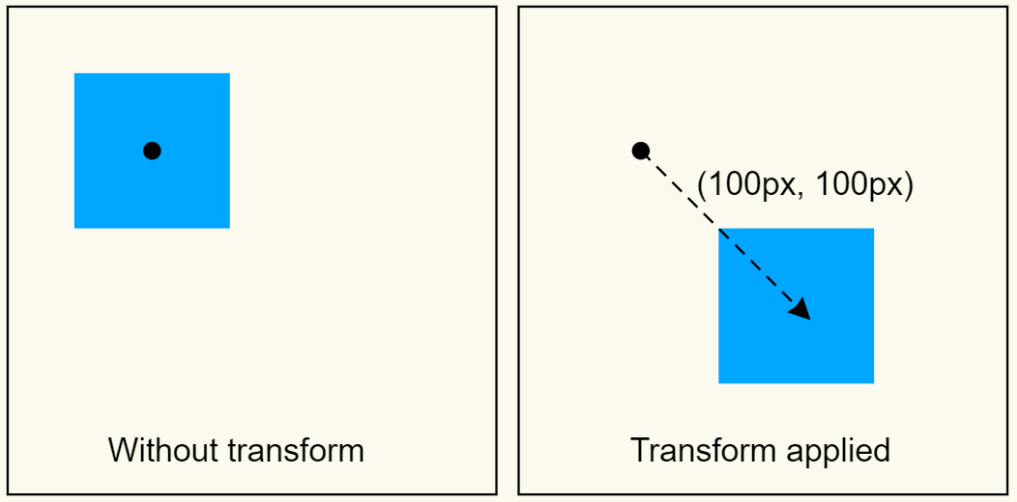
Enkele CSS 2D transform functions

2D transform-function	Beschrijving	Voorbeeld
rotate()	roteren	`transform: rotate(45deg);`
scale()	schalen	`transform: scale(1.2);`
translateX()	verplaatsen in de X-richting	`transform: translateX(50%);`
translateY()	verplaatsen in de Y-richting	`transform: translateY(200px);`
translate()	verplaatsen in de X- en Y-richting	`transform: translate(100px,200px);`
skew()	scheef maken	`transform: skew(10deg, 10deg);`

translate()

Verplaatst het element 100 pixels in de X- en de Y-richting. De X-richting is horizontaal naar rechts en de Y-richting is verticaal naar beneden.

```
div {transform: translate(100px, 100px);}
```



BRON: https://www.w3.org/TR/css-transforms-1/#example-76e5e4e0

HO GENT

Voorbeelden transform

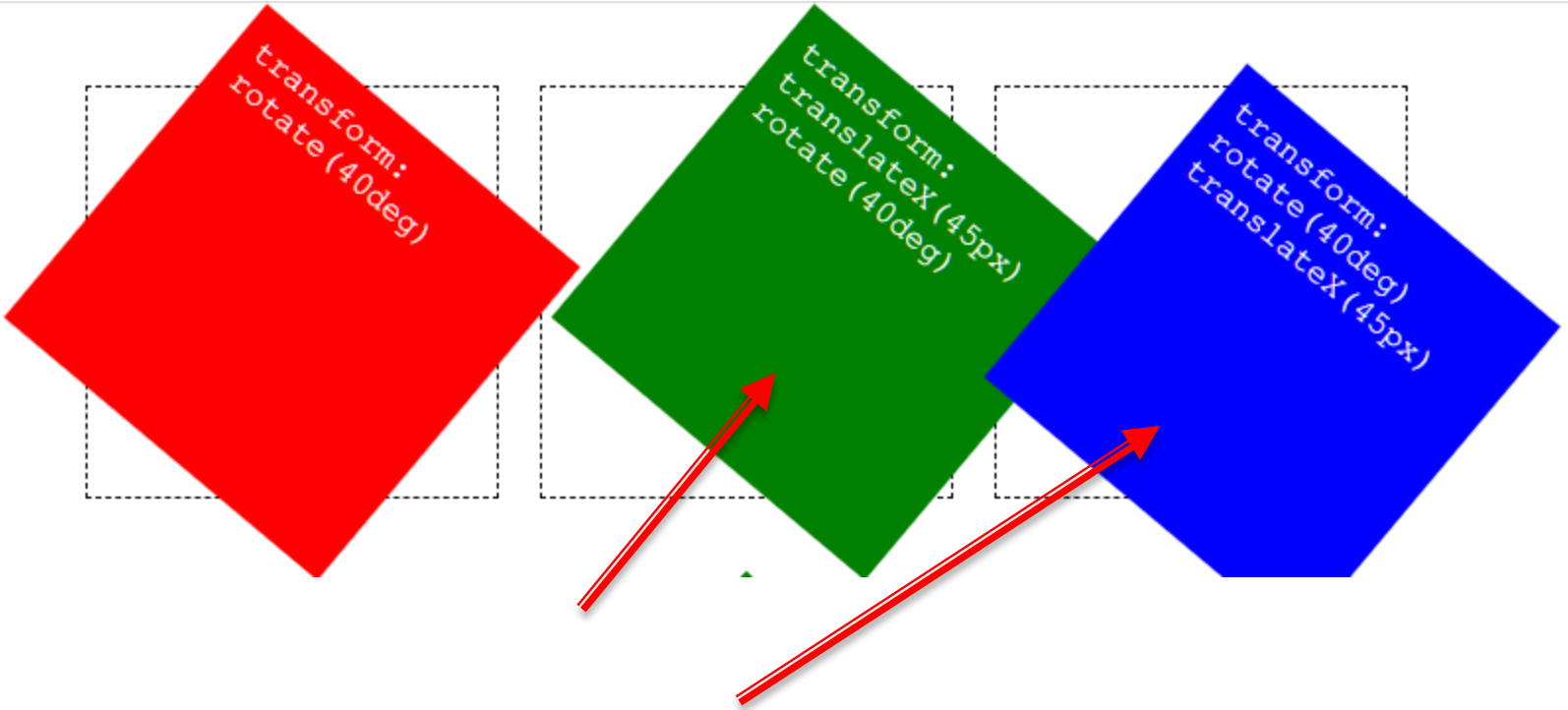
- Zie theorie-repo:
  - 2-transform-rotate-translate.html
  - 3-transform-rotate-hover.html
  - 4-transform-translateX-hover.html

2-transform-rotate-translate.html

Merk op dat `transform: translateX(45px) rotate(40deg);`
niet hetzelfde is als `transform: rotate(40deg) translateX(45px);`

Zie het voorbeeld op de volgende slide.

2-transform-rotate-translate.html



HO GENT

Opmerkingen

- We beperken ons in deze cursus tot 2D transformaties, maar er zijn ook 3D transformaties mogelijk. Voor wie meer info wenst over 3D transform zie: https://3dtransforms.desandro.com/
- **Transform** werkt standaard niet bij inline-elementen zoals het a-element of het span-element, maar wel bij een img-element.
  (Technisch: transform werkt niet bij ‘atomic inline elements’)

CSS animations

Theoriebestanden (voorbeelden)

- Open in Visual Studio Code
  de map **03-css-animations**

HO GENT

CSS animations

- Bij een **CSS transition** konden we enkel een begin- en eindtoestand opgeven. Bij **CSS animations** zullen we meer toestanden zgn. keyframes kunnen opgeven en we zullen bijvoorbeeld ook de animatie een aantal keren kunnen herhalen.
- In animatie-terminologie is een **keyframe**, een frame waar je veranderingen in de animatie definieert.
- Een CSS animation kan automatisch starten bij het laden van de webpagina. Dit is zelfs de default.
  Je kan ze ook starten via JavaScript of met behulp van :hover.

CSS animation

- **STAP 1 (definiëren animatie)**
  Definieer de animatie door middel van keyframes.
  `@keyframes` at-rule
  De browser zal bij het afspelen van de animatie de tussenliggende waarden tussen de opeenvolgende keyframes automatisch genereren.
- **STAP 2 (afspelen van de animatie op een bepaald element)**
  properties `animation-name`, `animation-duration`, `animation-timing-function`, … . shorthand property `animation`

1-css-animation-ball-example

```
div {
  animation-name: my-animation;
  animation-duration: 2s;
}

@keyframes my-animation {
  0% {transform: translate(0,0);}
  50% {transform: translate(100px, 0);}
  100% {transform: translate(100px, 100px);}
}
```

CSS animation

- Het @keyframe statement.
- De shorthand property animation:
- De 8 long hand properties met hun beginwaarden:

```
animation-name: none;
animation-duration: 0s;
animation-delay: 0s;
animation-fill-mode: none;
animation-iteration-count: 1;
animation-timing-function: ease;
animation-direction: normal;
animation-play-state: running;
```

animation-name

- De naam van de @keyframes animation die moet afgespeeld worden.

animation-duration

- De lengte van de animatie.
- Analoog als bij transition-duration kan je hier waarden opgeven zoals 1s voor 1 seconde of 300ms voor driehonderd milliseconden.

animation-delay

- Is vergelijkbaar met **transition-delay**. We kunnen deze property gebruiken om een wachttijd op te geven vooraleer de animatie start. Dit is vooral nuttig in situaties waarbij er verschillende animaties tegelijkertijd worden afgespeeld.
- Als de animatie een aantal keren herhaald wordt, wordt de wachttijd niet bij elke herhaling toegepast, maar enkel bij de start van de animatie (om de animatie te pauzeren zie het @keyframes voorbeeld verderop).

Voorbeeld: `animation-delay: 2s;`

animation-fill-mode

- Standaard zal een animatie nadat deze beëindigd is terugkeren naar haar begintoestand.
  Maar je kan deze bevriezen op haar eindpunt door gebruiken te maken van de **animation-fill-mode** property

Voorbeeld: `animation-fill-mode: forwards;`

animation-iteration-count

- Aantal keren dat de animatie afgespeeld wordt. Standaard wordt ze één keer afgespeeld. Je kan als waarde een getal opgeven of de waarde **infinite**, als je de animatie continu wilt afspelen.

Voorbeelden:

```
animation-iteration-count: 3;
animation-iteration-count: infinite;
```

animation-direction

- Met deze property controleer je de richting van de animatie.
- CSS animations beginnen normaal bij 0% en eindigen bij 100%.
- De mogelijke waarden voor animation-direction zijn:

  `normal` (initial value), `reverse`, `alternate`, `alternate-reverse`
- Bij `alternate` speelt de animatie van 0% naar 100% bij de eerste iteratie en bij de tweede terug naar 0%, enz.

Voorbeeld: `animation-direction: alternate;`

animation-play-state

- Met deze property kan je een animatie pauzeren of hervatten. De default is `running`.

Voorbeeld:

```
div:hover {
  animation-play-state: paused;
}
```

animation-timing-function

- Bij een CSS animation kan je bij elk keyframe een timing function opgeven (in het @keyframes statement) en je kan hiervoor dezelfde waarden gebruiken als bij de transition-timing-function.
- Ook als je slechts één enkele animation-timing-function opgeeft bij het element, zal de timing function toegepast worden op elk keyframe afzonderlijk en niet op de volledige animatie. Dit betekent dat in het voorbeeld op de volgende dia de animatie snel start dan vertraagt bij 25% om daarna weer te versnellen en terug te vertragen bij 50% en weer te versnellen enz. Het is daarom dat we de timing function voor een animation meestal op **linear** plaatsen.

2-css-animation-timing-function.html

```
div {
  ...
  animation-timing-function: linear;
}

@keyframes to-right {
  ...
  75% {
    ...
    animation-timing-function: ease-out;
  }
  ...
}
```

3-css-animation-arrow-bounce.html

```
.bounce {
  animation: 2s my-bounce 2; /* duration name iteration-count */
}

@keyframes my-bounce {
  /* de animatie pauzeert van 0% tot 20% */
  0%, 20% {transform: translateY(0);}
  40% {transform: translateY(-30px);}
  50% {transform: translateY(0);}
  60% {transform: translateY(-15px);}
  /* de animatie pauzeert van 80% tot 100% */
  80%, 100% {transform: translateY(0);}
}
```

@keyframes at-rule

HO GENT

Opmerkingen

- Je kan een animatie ook starten met :hover.

```
a:hover {
  animation: bounce 400ms;
}
```

- Gebruik `overflow: hidden;` om ongewenste schuifbalken bij het afspelen van animaties te verbergen.
- Je kan een volledig element transparant maken met de animatable CSS property opacity.

Variabelen

Variabelen

- CSS ondersteunt het gebruik van **variabelen**.
- Hiervoor declareer je een nieuw property wiens naam begint met --:

```
--primary-color
```

- De waarde van deze variabele kan je opvragen met:

```
var(--primary-color)
```

Variabelen

- Variabelen declareer je net zoals alle andere properties in een CSS rule:

```
:root {
  --primary-color: rgb(10, 10, 10)
}
```

- De `:root` pseudo-class is een goede plaats om globale variabelen te definiëren.

Variabelen

- Variabelen volgen de regels van de **cascade** en **inheritance**.
- Je kan dus ook lokale variabelen declareren:

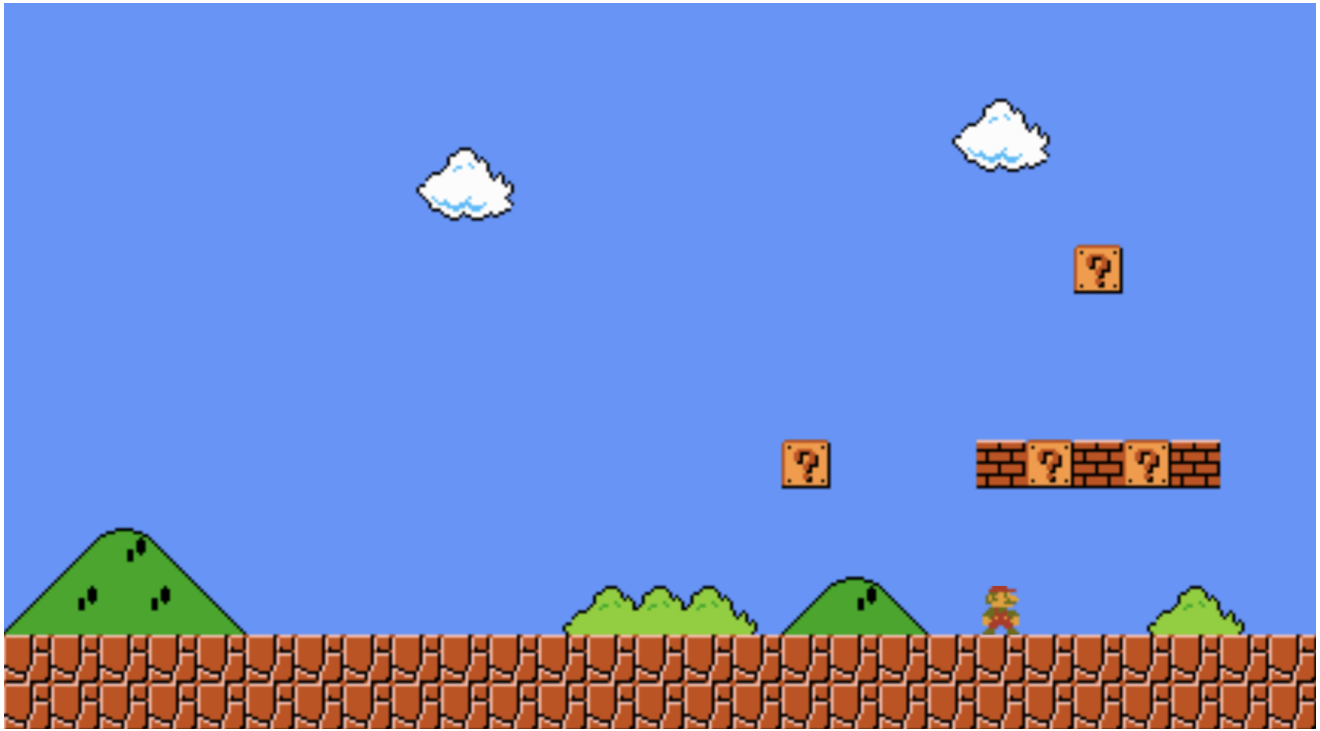
```
#container {
  --max-width: 800px;
}
```

- Deze variabele is beschikbaar binnen het element met id `#container`, en zijn kinderen.

Variabelen

- Variabelen zijn handig voor waarden die vaak herhaald worden:
  - Kleuren
  - Lettertypes
  - Groottes
  - Parameters van animaties
  - ...

Voorbeeld “04-variables”



HO GENT

Voorbeeld “04-variables”

- Dit voorbeeld maakt gebruik van volgende variabelen om de animatie te configureren:

```
:root {
  /* The size of a block that makes up the world of Mario */
  --block-size: 16px;
  /* How far Mario will walk (in blocks) */
  --steps: 18;
  /* How long it takes to walk a single block */
  --step-time: 0.3s;
  /* How long Mario should wait before starting his walk */
  --delay: 2s;
}
```

Voorbeeld “04-variables”

- Probeer zelf de gevraagde animatie uit te werken op basis van deze variabelen, of bekijk de oplossing om het resultaat te zien.
- Een geavanceerde versie van dit voorbeeld (waarbij Mario ook echt stapt) is te vinden in volgende repository: https://github.com/svanimpe/css-sprite-sheet-animation