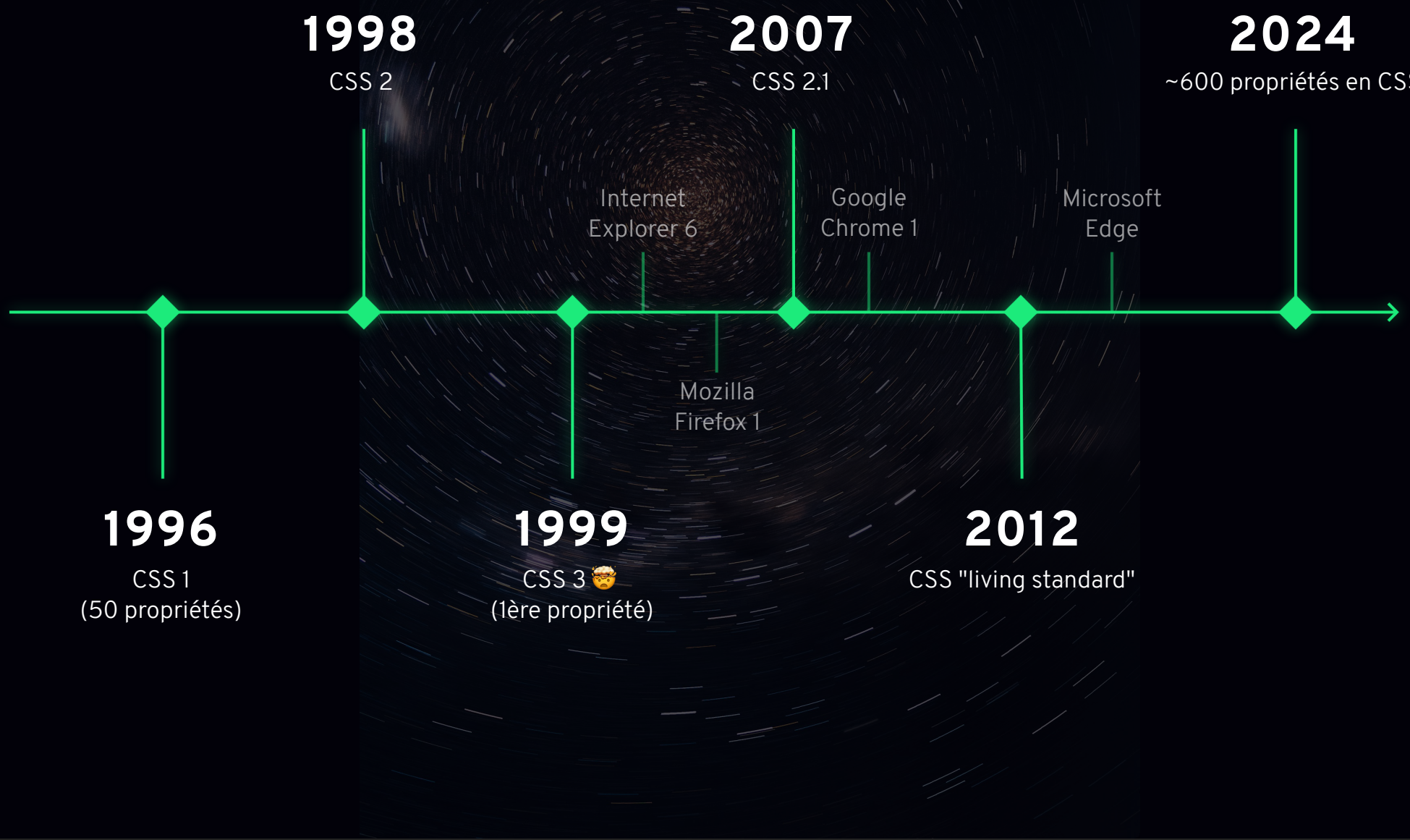
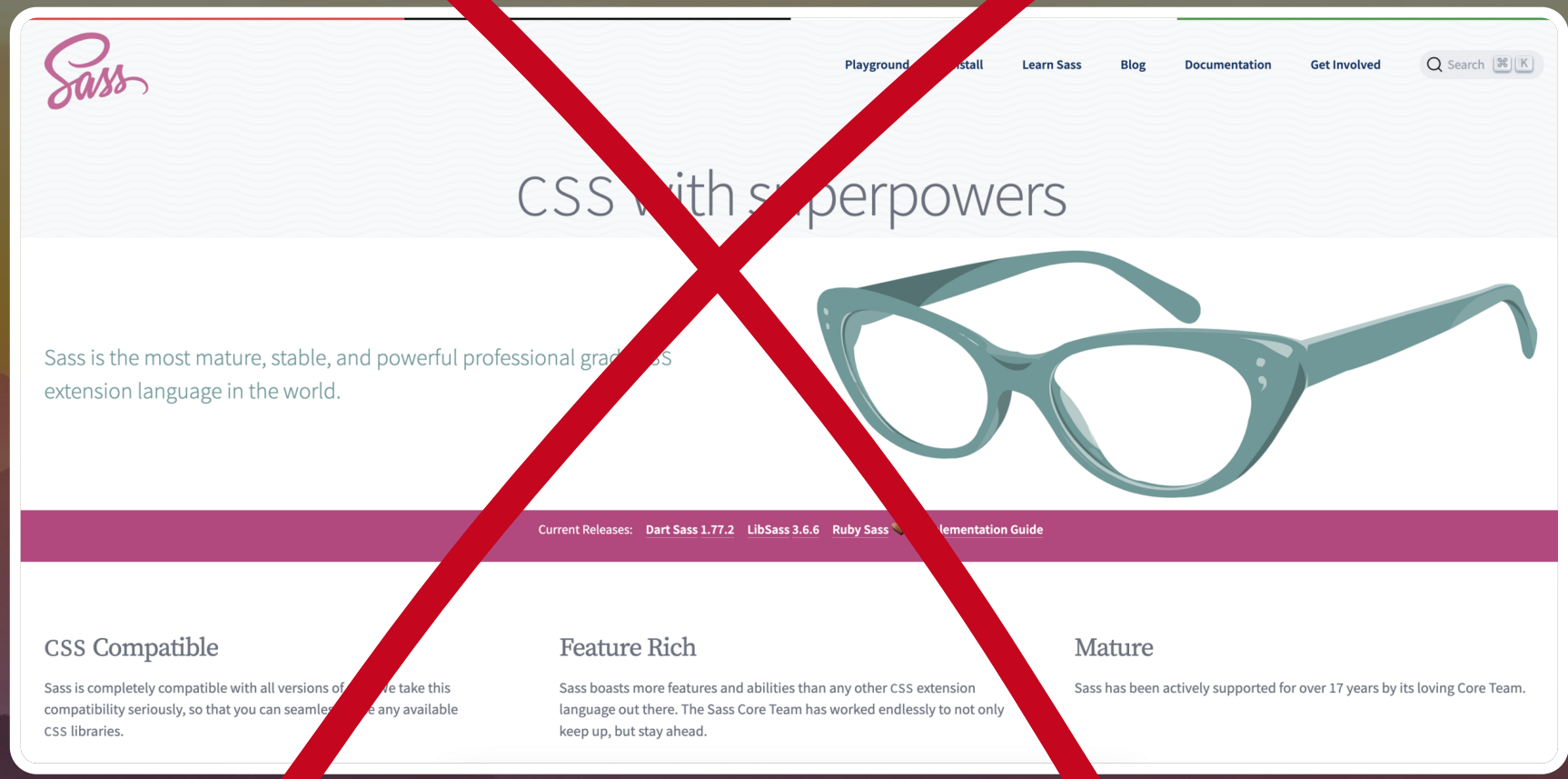
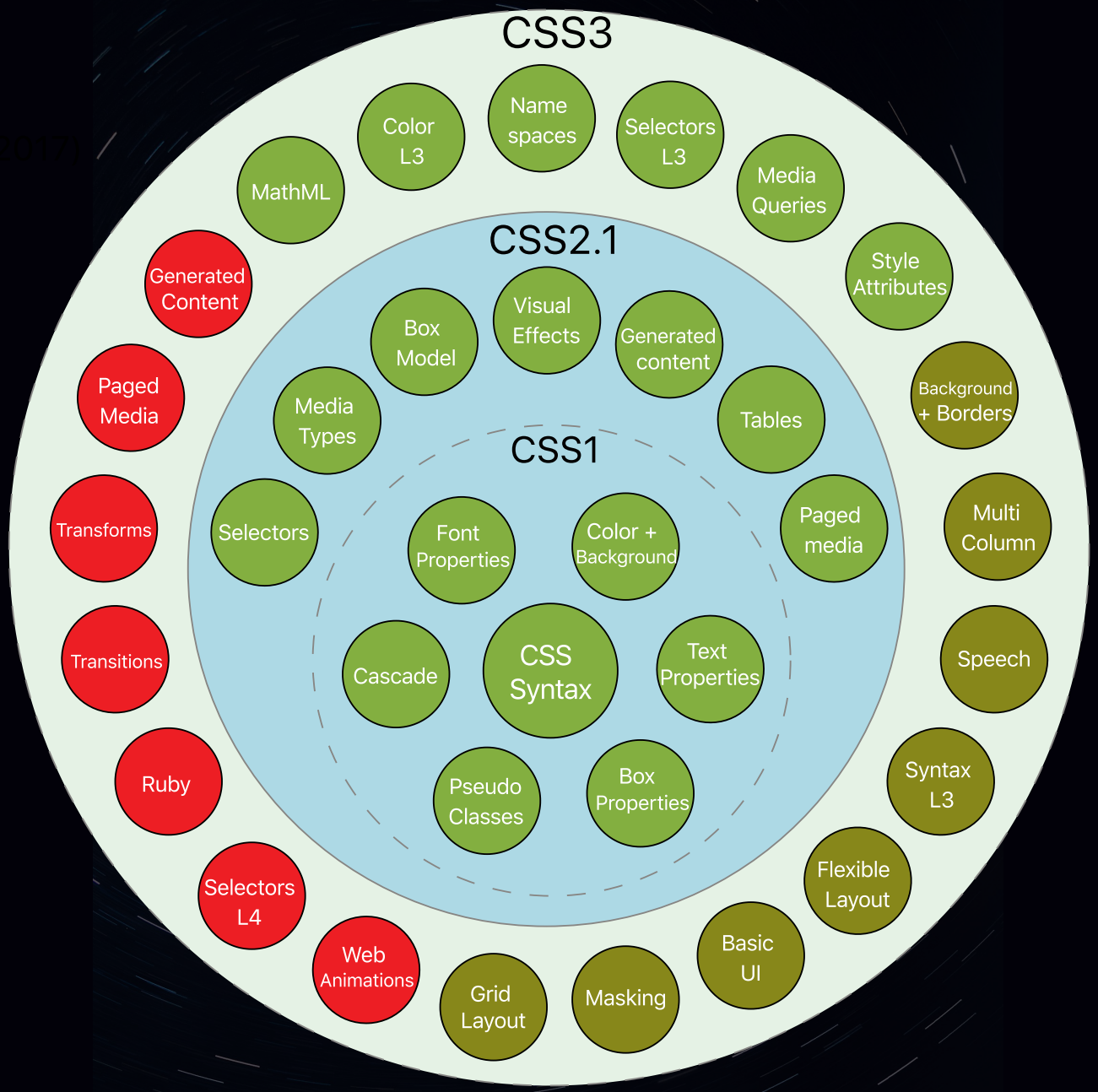
CSS MODERNES

LES NOUVEAUTÉS QUI CHANGENT LA VIE


1998
CSS 2
2007
CSS 2.1
2024
~600 propriétés en CSS
Internet Explorer 6
Google Chrome 1
Microsoft Edge
Mozilla Firefox 1
1996
CSS 1
(50 propriétés)
1999
CSS 3 🤯
(1ère propriété)
2012
CSS "living standard"

CSS != SCSS


Sass
Playground
Learn Sass
Blog
Documentation
Get Involved
Search
CSS with superpowers
Sass is the most mature, stable, and powerful professional grade CSS extension language in the world.
Current Releases:
Dart Sass 1.77.2
LibSass 3.6.6
Ruby Sass
Implementation Guide
CSS Compatible
Sass is completely compatible with all versions of CSS. We take this compatibility seriously, so that you can seamlessly use any available CSS libraries.
Feature Rich
Sass boasts more features and abilities than any other CSS extension language out there. The Sass Core Team has worked endlessly to not only keep up, but stay ahead.
Mature
Sass has been actively supported for over 17 years by its loving Core Team.



source : Wikipedia

FONDAMENTAUX

variables
nesting
@supports
display: content

```css
:root {
  --color-text: hotpink;
  --spacing-md: 12px;
}

/* Button component */
.button {
  color: var(--color-text);
  padding: var(--spacing-md);
}
```

Ici nous déclarons les variables dans :root *(= tout le document)* et les associons à une valeur.

Ces variables sont appliquées sur ce bouton à l'aide de var().

VARIABLES CSS

Leur vrai nom : "Custom Properties"

support

Les "variables" CSS (custom properties) sont des propriétés personnalisables, manipulables et réutilisables.

UN CAS D'USAGE CLASSIQUE ?

```css
:root {
  --gap: 16px;
}

.card {
  display: grid;
  gap: var(--gap);
}
@media (min-width: 576px) {
  .card {
    --gap: 32px;
  }
}
```

modifier des gouttières dynamiquement

VARIABLES CSS

Leur vrai nom : "Custom Properties"

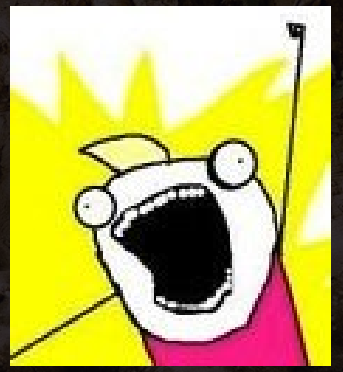
```css
:root {
  /* Transitions et animations */
  --transition-duration: 250ms;

  /* Niveaux de z-index */
  --z-under-page-level: -1;
  --z-above-page-level: 1;
  --z-header-level: 1000;
  --z-above-header-level: 2000;
  --z-above-all-level: 3000;

  /* Espacements */
  --spacing-0: 0;
  --spacing-1: 1px;
  --spacing-2: 0.125rem;
  --spacing-4: 0.25rem;
  --spacing-8: 0.5rem;
  --spacing-12: 0.75rem;
  --spacing-16: 1rem;
  --spacing-20: 1.25rem;
  --spacing-24: 1.5rem;
  --spacing-32: 2rem;
  --spacing-40: 2.5rem;
  --spacing-48: 3rem;
  --spacing-64: 4rem;
```

VARIABLES CSS

Leur vrai nom : "Custom Properties"

support

Toutes les valeurs de nos CSS se réfèrent à des variables. (Presque) aucune valeur "en dur".



VARIABLES ALL THE THINGS !

sans nesting

```css
.link {
  display: flex;
  flex-wrap: wrap;
}

.link:hover, .link:focus {
  color: hotpink;
}

@media (min-width: 576px) {
  .link {
    flex-direction: column;
  }
}

.link > * {
  margin-left: 1rem;
}
```

La syntaxe "classique" de CSS engendre fréquemment des répétitions de sélecteurs qui compliquent la maintenabilité du projet dans la durée.

NESTING

Imbrication de sélecteurs CSS

support

La syntaxe imbriquée (nesting) rassemble les règles CSS dans un seul sélecteur et évite un grand nombre de répétitions inutiles.

sans nesting

```css
.link {
  display: flex;
  flex-wrap: wrap;
}

.link:hover, .link:focus {
  color: hotpink;
}

@media (min-width: 576px) {
  .link {
    flex-direction: column;
  }
}

.link > * {
  margin-left: 1rem;
}
```

avec nesting

```scss
.link {
  display: flex;
  flex-wrap: wrap;

  &:hover, &:focus {
    color: hotpink;
  }

  @media (min-width: 576px) {
    flex-direction: column;
  }

  & > * {
    margin-left: 1rem;
  }
}
```

NESTING

Imbrication de sélecteurs CSS

support 

ATTENTION AU NOMBRE D'IMBRICATIONS !

```css
.title {
  color: #000; /* je suis noir par défaut */
}
@supports (backdrop-filter: initial) {
  .title {
    backdrop-filter: blur(10px) brightness(60%);
    color: #fff;
  }
}
```

Si backdrop-filter n'est pas reconnu par le navigateur, la couleur de texte reste noire.

La règle @supports() s'assure du support navigateur pour une propriété donnée et n'applique les styles que si la condition est respectée.

@SUPPORTS

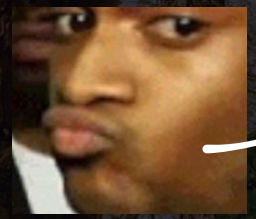
Détection du support navigateur

support

```html
<div class="row">
  <div class="col-6">
    <div>1</div>
    <div>2</div>
    <div>3</div>
  </div>
</div>
```

+

```css
.row {
  display: flex;
}
/* ou bien */
.row {
  display: grid;
}
```



marche pô

Je veux appliquer Flexbox ou Grid sur "row" et avoir une action sur les div 1, 2 et 3...
(toute ressemblance avec Bootstrap n'est pas fortuite)

+

```css
.col-6 {
  display: contents;
}
```

hop, je ne suis plus là !
(uniquement en CSS)

+

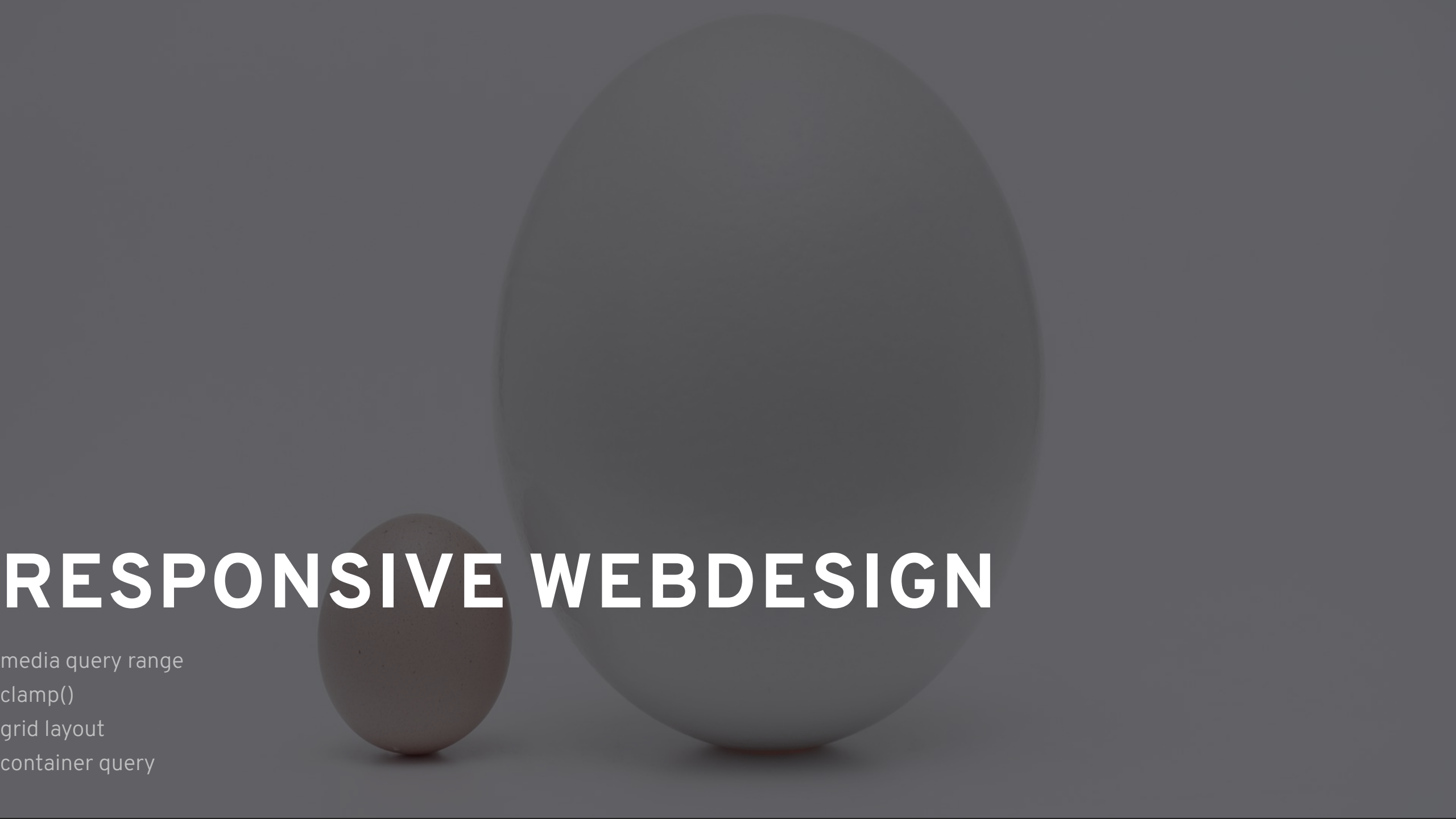
```css
.row {
  display: flex;
}
/* ou bien */
.row {
  display: grid;
}
```

DISPLAY: CONTENTS

Effacement d'une boîte en CSS

support

Un élément en display: contents ne génère plus de boîte. Ses enfants héritent directement des propriétés de son parent, un peu comme si on faisait "sauter un étage" en CSS.



RESPONSIVE WEBDESIGN

media query range
clamp()
grid layout
container query

#old

```css
@media (min-width: 576px) {
  ...
}
@media (max-width: 768px) {
  ...
}
@media (min-width: 320px) and (max-width: 768px) {
  ...
}
```

#new

```css
@media (width >= 576px) {
  ...
}
@media (width <= 768px) {
  ...
}
@media (320px <= width <= 768px) {
  ...
}
```

et en plus on peut
l'imbriquer (nesting) !

```css
.card {
  display: flex;
  flex-direction: column;

  @media (width >= 576px) {
    flex-direction: row;
  }
}
```

MEDIA QUERIES RANGE

La nouvelle syntaxe intuitive

La syntaxe moderne (range) des Media Queries est bien plus intuitive que la syntaxe historique

support

version Media Queries

```css
h1 {
  font-size: 2rem;

  @media (width >= 480px) {
    font-size: 3rem;
  }

  @media (width >= 768px) {
    font-size: 4rem;
  }

  @media (width >= 1280px) {
    font-size: 5rem;
  }
}
```

```css
:root {
  --spacing-m: 1rem;

  @media (width >= 480px) {
    --spacing-m: 1.5rem;
  }

  @media (width >= 768px) {
    --spacing-m: 2rem;
  }

  @media (width >= 1280px) {
    --spacing-m: 3rem;
  }
}
```

version clamp

```css
h1 {
  font-size: clamp(2rem, 0.8261rem + 5.2174vw, 5rem);
}
:root {
  --spacing-m: clamp(1rem, 0.2174rem + 3.4783vw, 3rem);
}
```

Merci à https://utopia.fyi/clamp/calculator et https://alsacreations.github.io/elastic/ !

CLAMP()

Valeur fluide entre une borne inférieure et supérieure

clamp() accepte 3 valeurs : une valeur minimale, la valeur souhaitée et une valeur maximale.

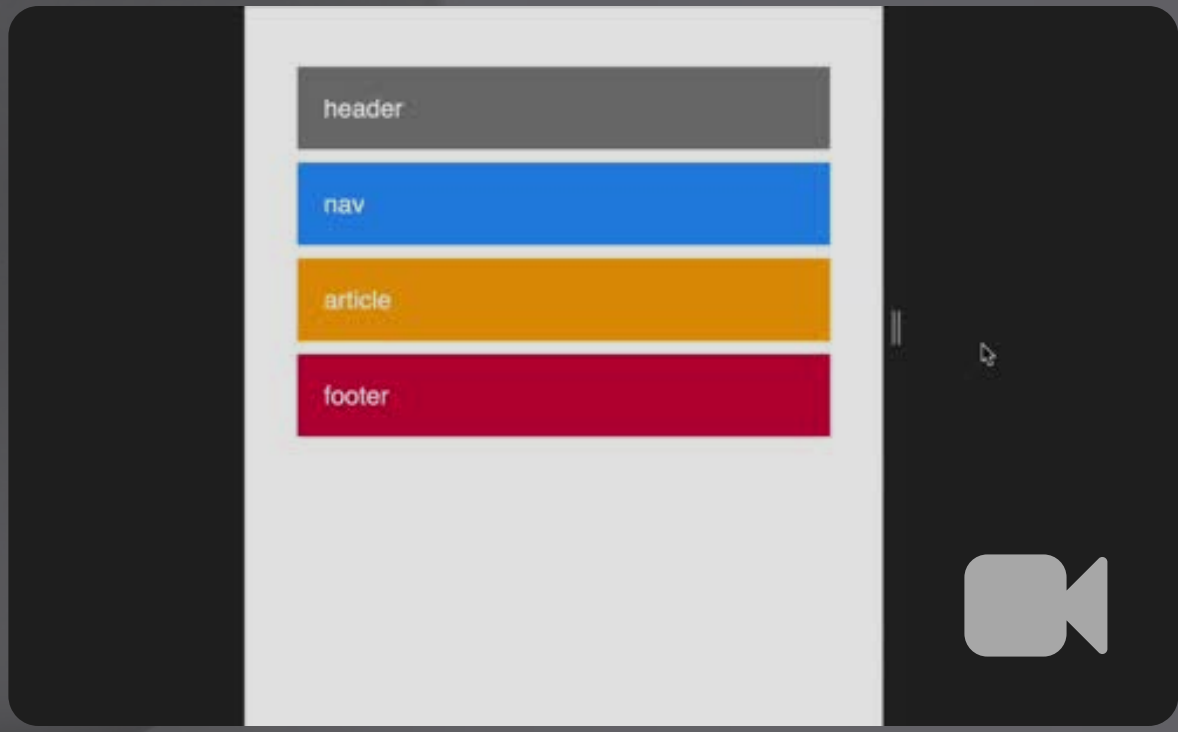
support

affichage sur écran réduit :
une seule colonne

```css
.container {
  display: grid;
  gap: 10px;

  @media (width > 576px) {
    grid-template-columns: 200px 1fr;

    & header,
    & footer {
      grid-column: span 2;
    }
  }
}
```

affichage sur écran plus large : deux colonnes

```html
<div class="container">
  <header>header</header>
  <nav>nav</nav>
  <article>article</article>
  <footer>footer</footer>
</div>
```



Grid Layout est le modèle de positionnement responsive en CSS par excellence. Tout est y réalisable : placement, fluidité, alignements, etc.

GRID LAYOUT

Le positionnement moderne en CSS

support

```css
.container {
  display: grid;
  place-content: center;
}
```

propriété raccourcie de justify-content et align-content

```html
<div class="container">
  <h1 class="title"></h1>
</div>
```

Centrer verticalement et horizontalement avec Grid Layout ?

Trop facile !

GRID LAYOUT

Le positionnement moderne en CSS

support

```css
.card-container {
  container-type: inline-size;
}

@container (inline-size >= 400px) {
  .card {
    /* styles à appliquer */
  }
}
```

Je fais une requête sur la largeur (inline-size) de .card-container

Si la condition de largeur est remplie, alors j'applique plein de jolis styles CSS

```html
<div class="card-container">
  <article class="card">
    <img>
    <h2></h2>
    <p></p>
    <button></button>
  </article>
</div>
```

Les Container Queries permettent d'interroger la taille d'un conteneur ancêtre et d'adapter conditionnellement des styles CSS sur un élement descendant.

CONTAINER QUERIES

Styles conditionnés par la taille d'un conteneur

support

INTERACTIVITÉ

scroll-snap


```css
.card-group {
  overflow-x: auto;
  scroll-snap-type: x mandatory;
}
.card {
  scroll-snap-align: center;
}
```

adhérence des enfants sur l'axe x

alignement de chaque card sur la "grille"
(ici, elle sera centrée)

SCROLL-SNAP

Technique de scroll avec des points d'accroche

support

Scroll-snap crée des positions d'accroche lors du défilement, permettant de garantir la position sur laquelle on arrive après avoir fait défiler du contenu.

EXERCICES !

RESPONSIVE

BADGE

SCROLL

Disclaimer : ces exercices contiennent peut-être des consignes non vues en formation

(en clair, faut vous débrouiller un peu)