

Outils de développement de sites web

Johann Chopin

Deuxième semestre de la 1ère années de la licence
Informatique et ingénierie du web.

❖ Outils de développement de sites web

Plan

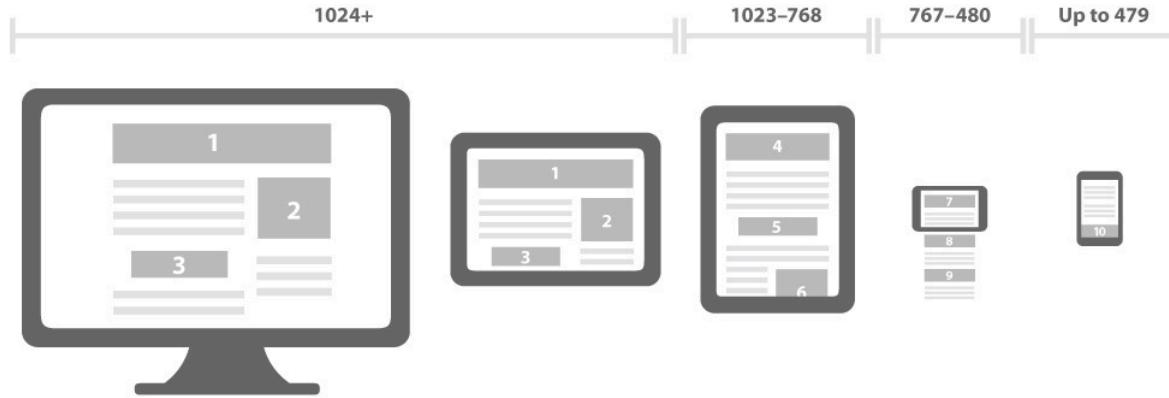
- 1. Récapitulatif des connaissances
- 2. Techniques d'agencement d'interface digitale
 - 2.1. Desktop first vs mobile first
 - 2.2. Static layout
 - 2.3. Fluid design
 - 2.4. Adaptive design
 - 2.5. Responsive design
- 3. Responsive Design
 - 3.1. Un peu d'histoire
 - 3.2. Redimensionner une interface Web
 - 3.3. Media Queries
 - 3.4. Flexbox
 - 3.5. CSS grid
 - 3.6. Flexbox vs CSS Grid
- 4. CSS frameworks
 - 4.1. Introduction
 - 4.1.1. Méthodologie BEM
 - 4.1.2. Qu'est-ce qu'un framework CSS?
 - 4.1.3. General Purpose VS Utility based frameworks
- 4.2. Bootstrap
 - 4.2.1. Installation
 - 4.2.2. Qu'est-ce que Bootstrap?
 - 4.2.3. Gestion du layout
 - 4.2.4. Gestion du contenu
 - 4.2.5. Les composants
 - 4.2.6. Les utilitaires
 - 4.2.7. Exercice
 - 4.2.8. Utilisation dans l'environnement Nodejs
 - 4.2.9. Gestion du theme
- 4.3. Tailwindcss
 - 4.3.1. Installation
 - 4.3.2. Styliser avec des classes utilitaires
 - 4.3.3. Responsive design
 - 4.3.4. Ajout de styles personnalisés
 - 4.3.5. Dark mode
 - 4.3.6. Exercice
- 5. En résumé
- 6. Ressources

Récapitulatif des connaissances

→ Cours **Introduction au web**:

- Connaître les principaux standards du web : **HTML5 et CSS3**
- Maîtriser la notion de séparation contenu/forme à travers la description d'un document sous forme arborescente et traduire ce modèle en un document HTML
- Réaliser la mise en forme en utilisant le langage **CSS3**
- Savoir développer des programmes en JavaScript
- Rendre le document dynamique et le manipuler via l'interface DOM et front-end JavaScript.

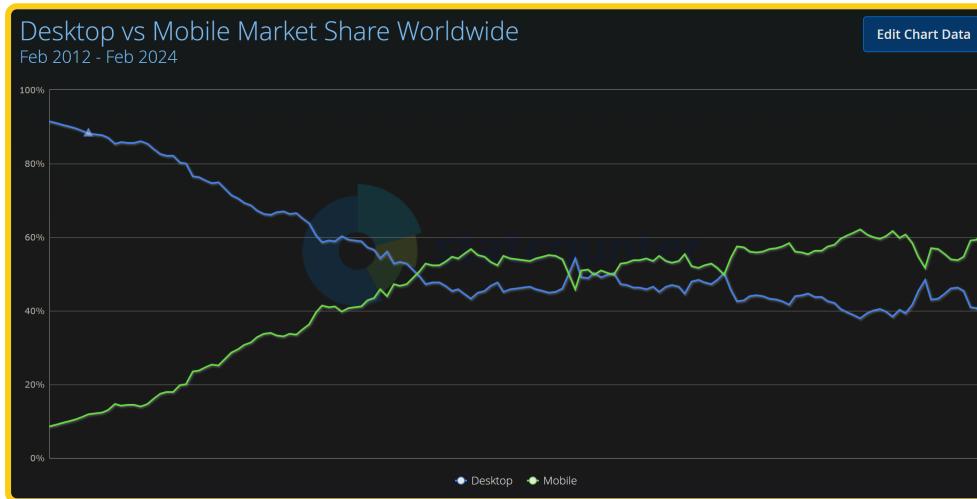
Techniques d'agencement d'interface digitale



Desktop first vs mobile first

Mobile: **63.92%**

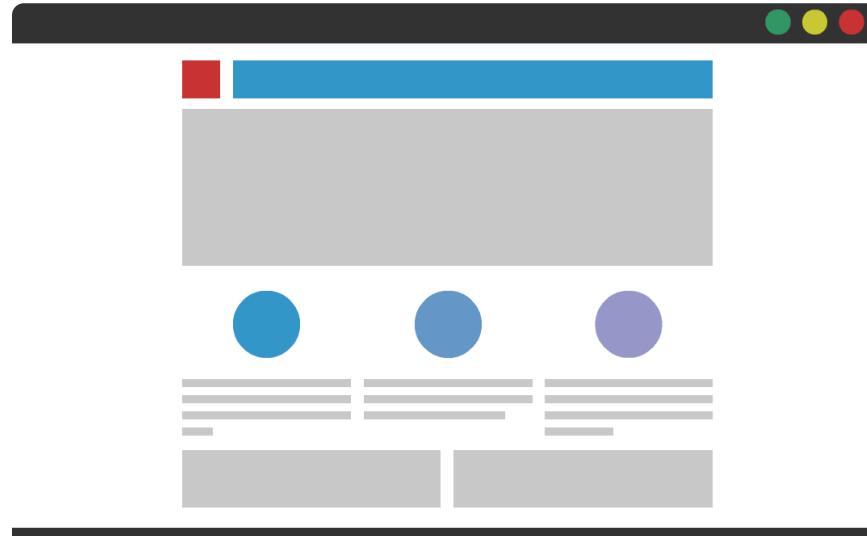
Desktop: **36.08%**



Source: GlobalStats statcounter

Static layout

Dans cet agencement, le site web est englobé dans un "conteneur" programmé de manière statique avec une largeur (et hauteur) fixe en pixels.



-  Contrôle total de l'apparence de l'interface.
-  Développement rapide
-  Dépendance totale aux dimensions initiales.



Fluid design

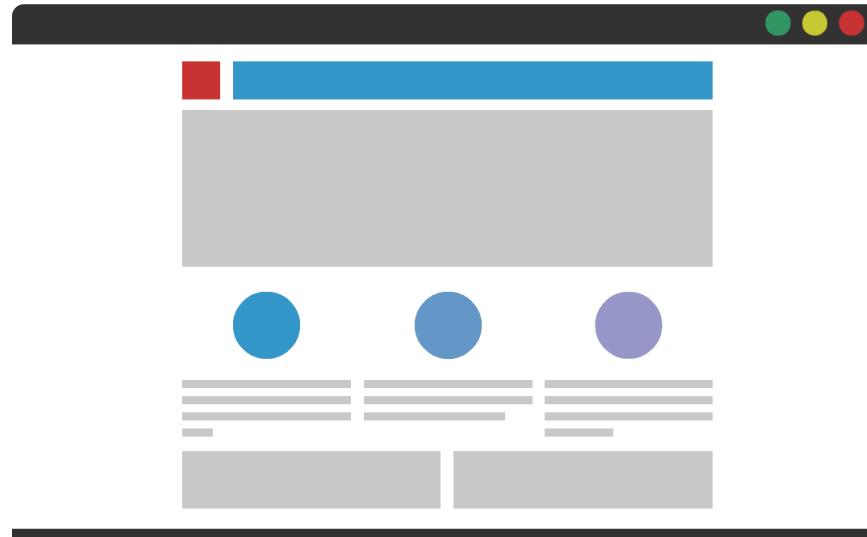
Avec un agencement fluide, les tailles ne sont plus spécifiés en valeur statique, mais en **valeur relative** (% , em , vw , calc() , ...).



-  Si la taille de l'écran change, la proportion des éléments restera la même.
-  L'agencement peut s'adapter en fonction des paramètres de l'utilisateur.
-  Les conceptions de largeur fluide ne sont pas non plus à l'abri des cas extrêmes.
-  Les images et textes sont difficiles à mettre à l'échelle.

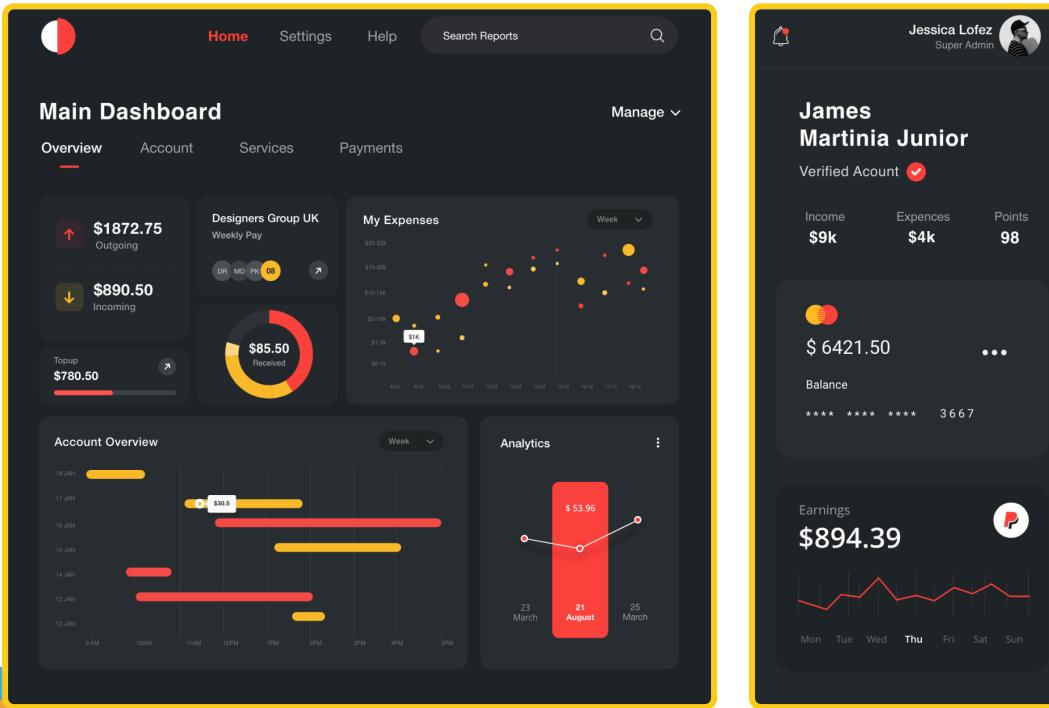
Adaptive design

L'agencement adaptatif signifie qu'il existe plusieurs versions de la mise en page qui sont affichées en fonction de la taille de l'écran de l'utilisateur.



Techniques d'agencement d'interface digitale / Adaptive design

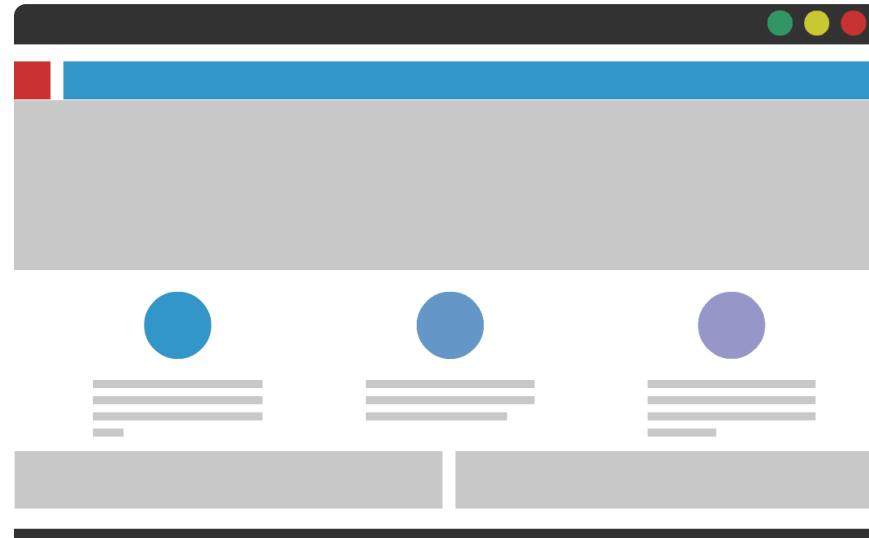
- Certitude que l'utilisateur bénéficiera d'une expérience optimale.
- Le serveur envoie exactement le contenu qui doit être chargé -> chargement rapide de la page.
- Chaque agencement doit être conçu et développé individuellement.



Responsive design

L'agencement responsive prend le meilleur des deux mondes de l'agencement fluide et adaptatif.

Il existe plusieurs points de rupture, qui divisent toutes les tailles d'écran possibles en tranches. L'interface a une présentation légèrement (ou complètement) différente selon la taille de l'écran sur lequel elle est visualisée.



-  SEO (Search Engine Optimization) friendly
-  Maintenance faible car il n'existe qu'une version de l'interface pour toutes les tailles d'écran.
-  Contenu du site moins optimisé et donc potentiellement plus lent.
-  Développement plus long avec une courbe d'apprentissage élevée.

 De nos jours, les mises en page responsive sont quasiment un standard pour toutes interfaces web.

Responsive Design

Le Responsive Web Design, ou RWD, est une approche de la conception qui prend en compte l'ensemble des appareils et leur taille, permettant une **adaptation automatique** à l'écran.

Le contenu peut donc être visualisé sur une tablette, un téléphone, une télévision ou même une montre.



Un peu d'histoire

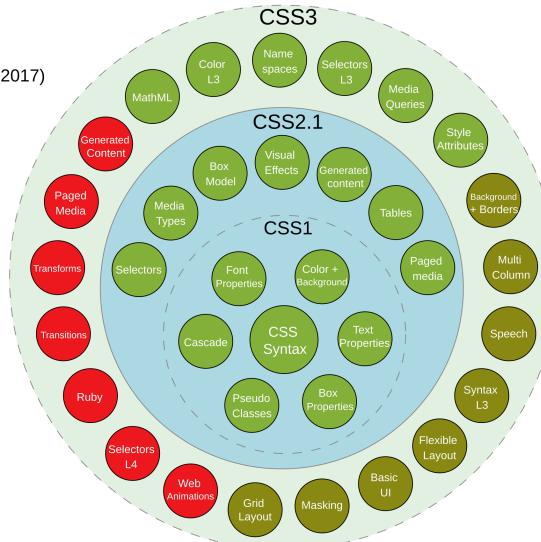
En 2010, Ethan Marcotte, initie le terme *Responsive Web Design* dans un article du même nom.

À l'époque, la recommandation était d'utiliser les CSS **float** pour la mise en page et les **media queries** pour connaître la largeur du navigateur, en créant des mises en page pour différents **breakpoints**.

CSS3

Taxonomy & Status (September 2017)

- W3C Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Obsolete or inactive



CSS3 a énormément évolué depuis et mets à disposition une multitude de fonctionnalités intégrées aux plateformes web qui facilitent la conception de sites réactifs:

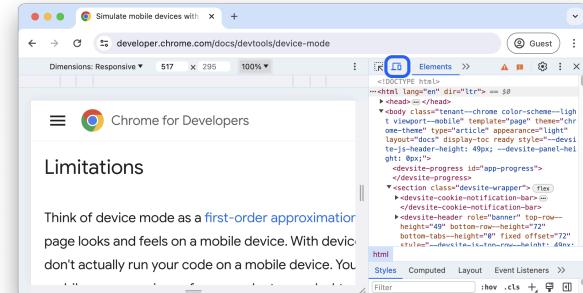
- Les Media Queries
- Le modèle de boîte CSS (flexbox)
- CSS grid

Redimensionner une interface Web

La majorité des navigateurs proposent un outils permettant de développer de manière responsive.

Pour Chromium et Firefox:

- Ouvrez l'inspecteur: Click droit sur la page > Inspect
- Utilisez le **device toolbar**



Pour Safari:

- Activer le mode développeur: Safari > Settings > Advanced > **Show features for developers**
- Depuis Safari allez dans le menu Develop > Enter Responsive Design Mode

À vous de jouer: Rendez-vous sur le site de l'ISFATES <https://www.dfhi-isfates.eu/de>, puis redimensionnez son interface pour la consulter au format mobile, tablette, ordinateur et enfin fluide.

Media Queries

Les media queries nous permettent d'effectuer une série de tests (largeur de l'écran, résolution, préférences de l'utilisateur) et d'appliquer du CSS de manière sélective pour styliser la page en fonction des besoins de l'utilisateur.

```
@media screen and (min-width: 80rem) {  
  .container {  
    margin: 1em 2em;  
  }  
}
```

À vous de jouer: Dans le repo [Q isfates-l1-outils-dev-sites-web/exercises](#), ouvrez dans votre IDE l'exercice **media-query**.

Rendez cette interface responsive en respectant les instructions suivantes:

- **screen width > 60rem:**

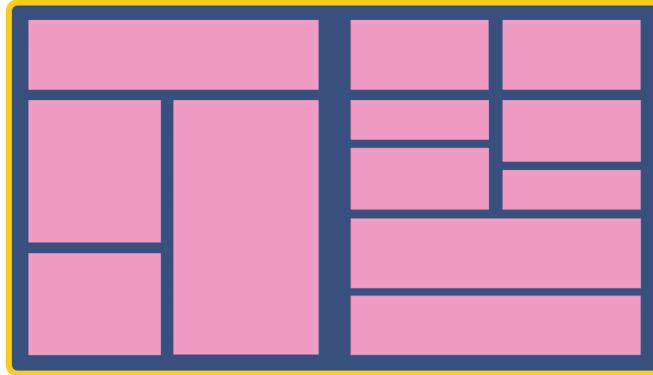
- Le contenu doit être centré en respectant une marge de 10% de la largeur de l'écran.
- Les blocks `nav` , `article` et `aside` doivent être alignés horizontalement.

- **screen width ≤ 60rem:**

- Le contenu ne doit prendre toute la largeur possible.
- Le blocks `aside` disparait de l'écran.
- Les blocks `nav` et `article` sont alignés verticalement et prennent toute la largeur disponible.



Les media queries peuvent être utiles pour réaliser un layout responsive, mais elles ne sont pas obligatoires. Ainsi, lorsqu'il s'agit d'aligner un contenu de manière plus complexe, des outils plus spécifiques sont nécessaires.



Flexbox

Le module de disposition des boîtes flexibles CSS (CSS Flexible Box Layout) est un module de CSS qui définit un **modèle de boîtes** optimisé pour la conception des interfaces utilisateurs.

En utilisant le modèle des boîtes flexibles, les éléments d'une conteneur flexible peuvent être disposés dans n'importe quelle direction et étendre ou réduire leurs dimensions pour éviter de dépasser en dehors de l'élément parent.

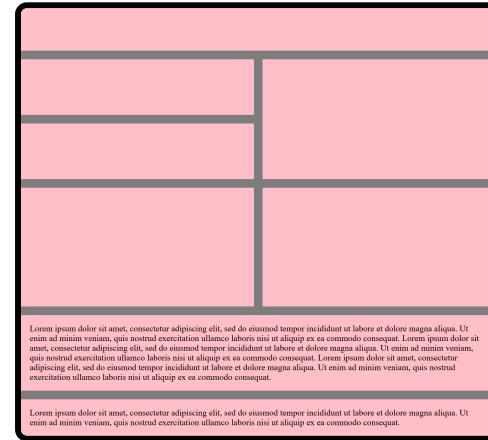
 [Flexbox Cheatsheet by CSS-TRICKS](#)

À vous de jouer: Dans le repo [osfates-l1-outils-dev-sites-web/exercises](#), ouvrez dans votre IDE l'exercice **flexbox-layout**. Réalisez, à l'aide des propriétés **flexbox**, cette interface responsive:

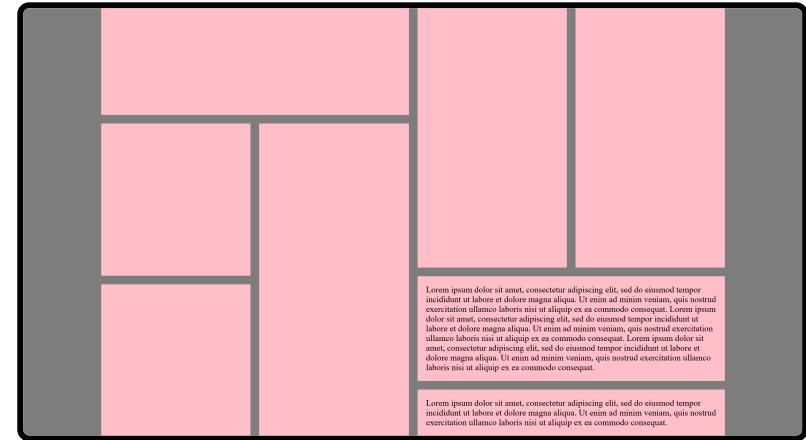
screen width \leq 30rem



screen width \leq 60rem



screen width $>$ 60rem



💡 Pour l'espacement entre les boîtes, regardez voir la propriété `gap`, `flex-grow` et `flex-basis` vont vous aider pour définir les tailles des boîtes. `flex-wrap` s'avère utile pour les retours à la ligne. Utilisez des classes CSS!

CSS grid

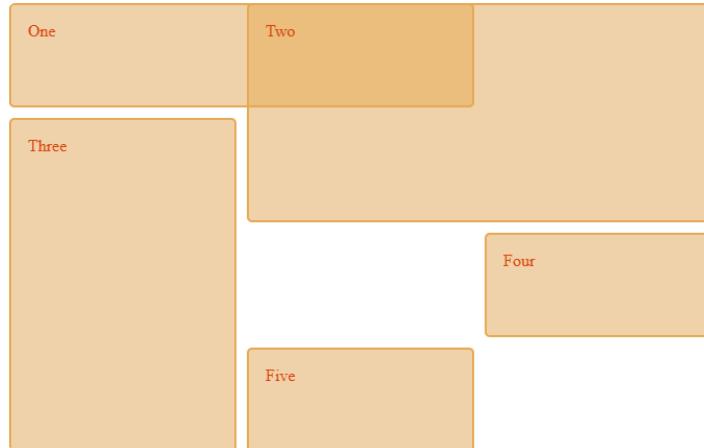
Grid est le tout premier module CSS créé spécifiquement pour résoudre les problèmes de mise en page que nous contournons tous depuis que nous développons des sites web.

Comme des tableaux, les grilles CSS permettent au développeur d'**aligner les éléments en colonnes et en lignes**.



Responsive Design / CSS grid

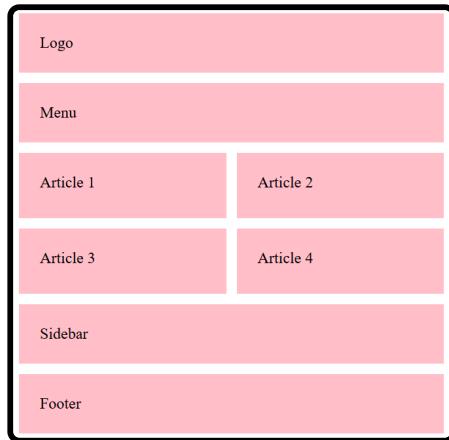
```
<div class="wrapper">  
  <div class="one">One</div>  
  <div class="two">Two</div>  
  <div class="three">Three</div>  
  <div class="four">Four</div>  
  <div class="five">Five</div>  
</div>
```



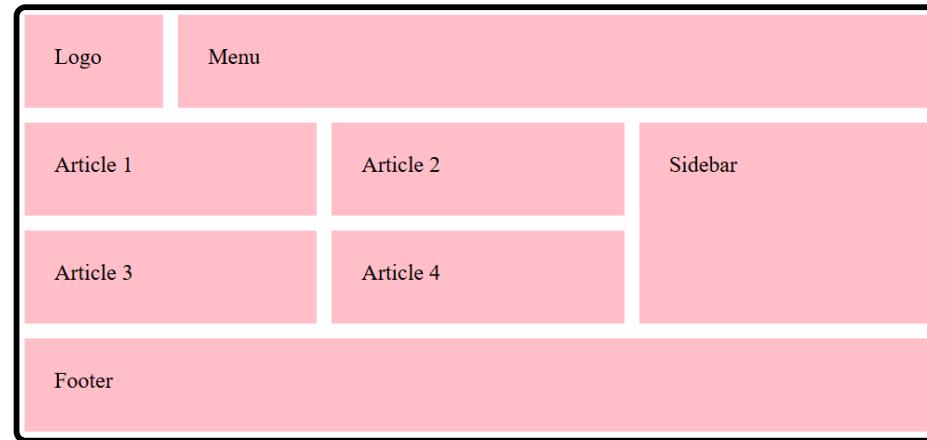
```
.wrapper {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
  grid-auto-rows: minmax(100px, auto);  
}  
.one {  
  grid-column: 1 / 3;  
  grid-row: 1;  
}  
.two {  
  grid-column: 2 / 4;  
  grid-row: 1 / 3;  
}  
.three {  
  grid-column: 1;  
  grid-row: 2 / 5;  
}  
.four {  
  grid-column: 3;  
  grid-row: 3;  
}  
.five {  
  grid-column: 2;  
  grid-row: 4;  
}
```

À vous de jouer: Dans le repo [osfates-l1-outils-dev-sites-web/exercises](#), ouvrez dans votre IDE l'exercice **grid-layout**. Réalisez, à l'aide d'une **CSS grid**, cette interface responsive en *mobile first*:

screen width < 60rem



screen width $\geq 60\text{rem}$

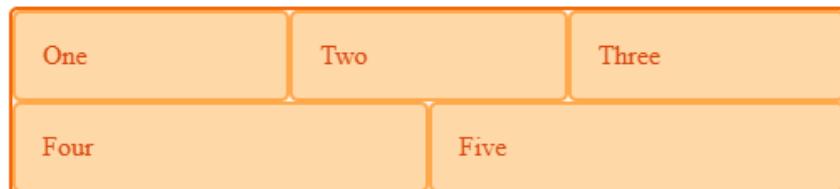


💡 Pour l'espacement entre les boîtes, regardez voir la propriété `gap`, `grid-template-areas` et `grid-area` vont vous aider pour définir les emplacements du contenu.

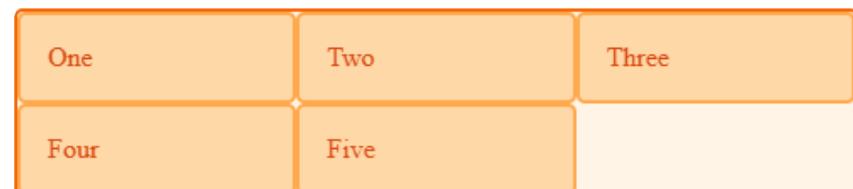
Flexbox vs CSS Grid

- **Flexbox** a été conçu pour une mise en page en **une seule dimension**, que ce soit une ligne ou une colonne.
- **CSS Grid** a été conçue pour une mise en page **bidimensionnelle**, lignes et colonnes à la fois.

```
.wrapper {  
  display: flex;  
  flex-wrap: wrap;  
}  
  
.wrapper > div {  
  flex: 1 1 150px;  
}
```



```
.wrapper {  
  display: grid;  
  grid-template-columns: repeat(3, 1fr);  
}
```

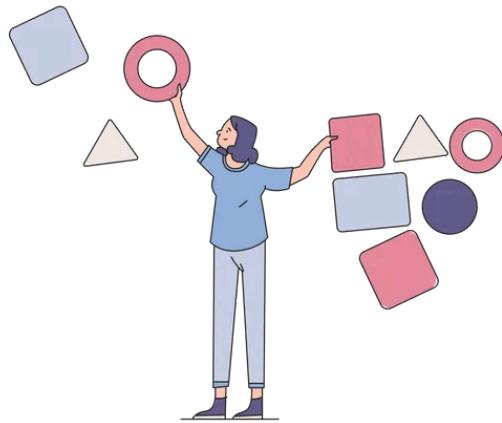


Une question simple à se poser lorsqu'il s'agit de choisir entre CSS grid et flexbox est la suivante:

- Si je dois uniquement contrôler la mise en page **par ligne ou par colonne**: utilisez **flexbox**.
- Si je dois contrôler la mise en page **par ligne et par colonne**: utilisez **CSS grid**.

💡 Les deux techniques sont complémentaires.

CSS frameworks



Introduction

"HTML et CSS sont tout autant faciles à écrire, qu'extrêmement complexes à maintenir."

— Un développeur en 2007

Lorsqu'il s'agit de projets importants et complexes, la manière dont vous organisez votre code est la clé de l'efficacité. Elle influe sur:

- l'efficacité à écrire du code
- la quantité de code à écrire
- le temps de chargement de votre projet sur un navigateur

Nous avons tous écrit du code CSS non maintenable...

```
.main-content #myArticle:not(.toc) p:last-child {  
    color: yellow;  
}  
  
#myForm .btn-save { background: red; }  
.discard-changes-button { background: green; }
```

Cela se transforme rapidement en amat de:

- Règles obscures
- Mauvais noms de classe
- CSS incohérent

Travailler en équipe de cette manière peut être très chaotique...

Très rapidement, la communauté CSS a développé différentes méthodologies pour écrire des styles **simplifiés, modulaires** et plus **flexibles**.

- **OOCSS:** Séparation du contenant et du contenu à l'aide d'*objets* CSS.
- **SMACSS:** Guide de style pour écrire votre CSS avec cinq catégories de règles CSS.
- **Atomic:** Décomposition des styles en éléments atomiques, ou indivisibles.
- **BEM:** Approche du développement web component-based.

Méthodologie BEM

BEM vise à normaliser le nom des classes CSS en les classant par **Block**, **Element** et **Modifier**.

- **Block:** Entité autonome qui a une signification en soi.
 - header , container , menu , checkbox , input
- **Element:** Partie d'un bloc qui n'a pas de signification autonome et qui est sémantiquement liée à son block.
 - menu item , list item , checkbox caption , header title
- **Modifier:** Indicateur placé sur un bloc ou un élément pour en modifier l'apparence ou le comportement.
 - disabled , highlighted , checked , fixed

Syntaxe de classe:

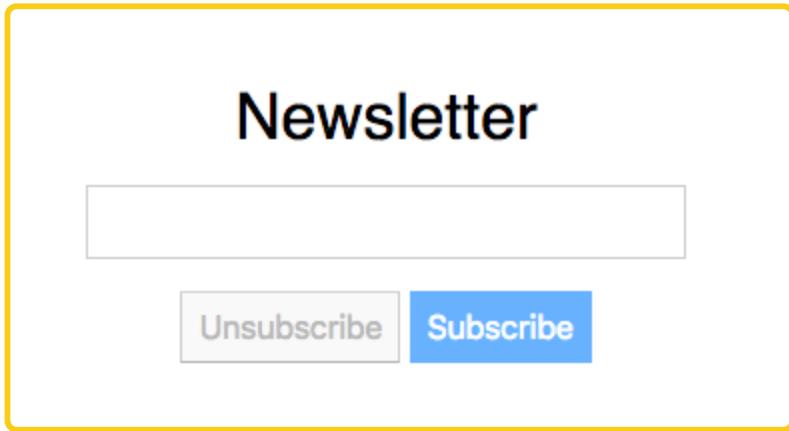
```
.block__element--modifier {}
```

```
/* Block component */
.btn {}

/* Element that depends upon the block */
.btn__price {}

/* Modifier that changes the style of the block */
.btn--orange {}
.btn__price--big {}
```

À nous de jouer: Nous devons développer le CSS pour la newsletter suivante. Quelles classes CSS devons-nous ajouter conformément à la méthodologie BEM?



```
<div class="newsletter">
  <h1>Newsletter</h1>
  <input type="email">
  <button>Unsubscribe</button>
  <button>Subscribe</button>
</div>
```

Qu'est-ce qu'un framework CSS?

Un framework CSS est une **bibliothèque préconçue** de styles CSS. Il offre aux développeurs une approche structurée pour créer des sites web réactifs, cohérents et visuellement attrayants sans avoir à partir de zéro.

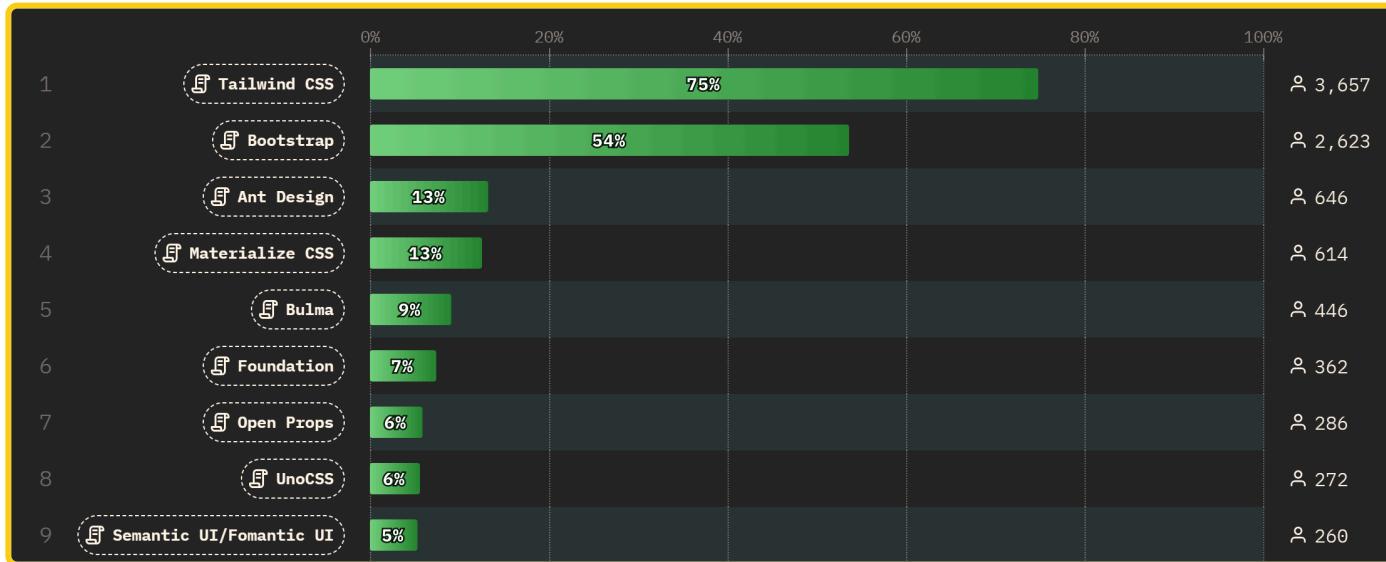
Ces frameworks contiennent des **classes** et des **règles** CSS pré-écrites qui simplifient le processus de stylisation des pages web, réduisent le temps de développement et **facilitent le maintien d'un design-system** cohérent pour l'ensemble d'un projet.

Fonctionnalités majeures:

- Système de grille
- Gestion d'un thème (couleurs, polices, espacements...)
- Conception responsive
- Utilitaires et classes d'aide
- Composants préétablis

Il existe 2 grandes familles:

- General Purpose frameworks
- Utility based frameworks



Frameworks utilisés par un échantillon de 10.000 développeurs. Chiffres du 2024 State of CSS.

General Purpose VS Utility based frameworks

General Purpose framework

Fournit des composants pré-conçus et réutilisables, en favorisant:

- **Développement rapide:** Assemblez rapidement des interfaces réactives.
- **Cohérence:** Assure une uniformité de l'application grâce à des composants standard.
- **Facilité d'utilisation:** Idéal pour les développeurs qui préfèrent des composants prêts à l'emploi.

```
<button  
  class="btn btn-primary"  
>Bootstrap Button</button>
```

Utility based framework

Met à disposition des classes utilitaires pour composer des designs personnalisés permettant:

- **Contrôle granulaire:** Stylez les éléments avec précision sans écraser les styles.
- **Efficacité:** Réduit la nécessité d'écrire du CSS personnalisées, ce qui accélère le développement.
- **Flexibilité:** Idéal pour créer des designs uniques et personnalisés.

```
<button  
  class="bg-blue-500 hover:bg-blue-700 text-white font-bold"  
>Tailwind Button</button>
```

General Purpose framework



Bootstrap

Star 172k



Bulma

Star 50k



Material Design

Star 17k

Utility based framework



tailwindcss

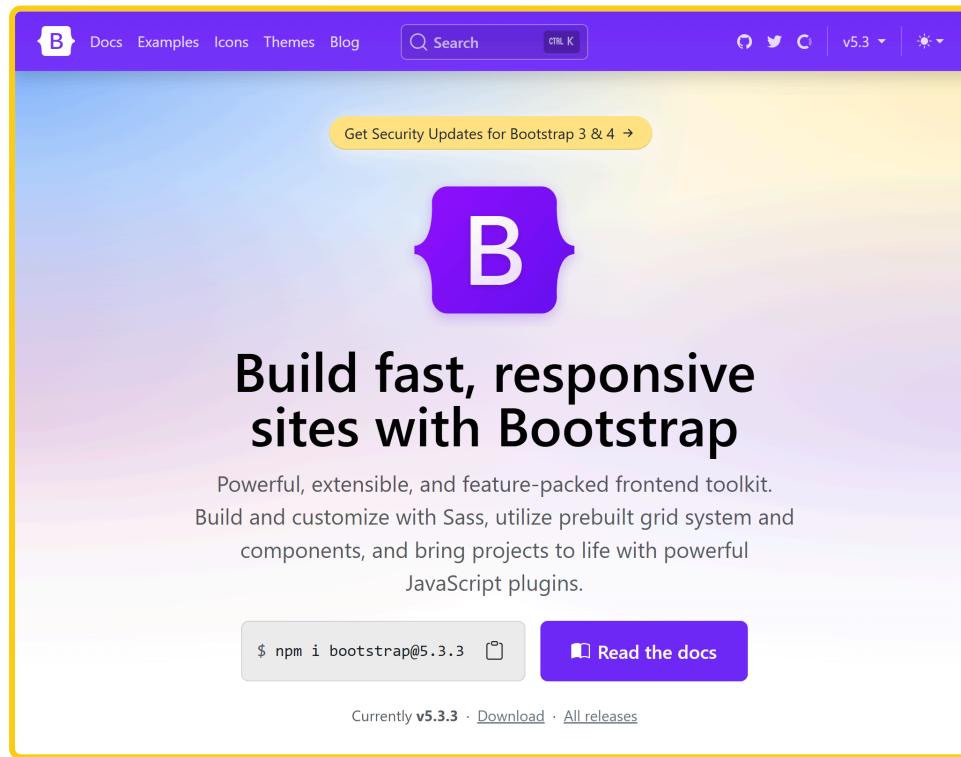
Star 86k



Open Props

Star 5k

Bootstrap



<https://getbootstrap.com/>

Installation

- Bootstrap 5 (sortie en 2021) est la version la plus récente de Bootstrap (sortie en 2013).
- La principale différence entre Bootstrap 5 et Bootstrap 3&4 est que la librairie **jQuery** n'est plus utilisée.

Il est possible d'utiliser Bootstrap sans étape de build via **CDN (Content Delivery Network)**:

```
<!doctype html>
<html>
  <head>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-...
  </head>
  <body>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY...
  </body>
</html>
```

Dans le repo [exercises](#), vous trouverez un template nommé **bootstrap-starter**.

Qu'est-ce que Bootstrap?

Bootstrap n'est rien de plus qu'une collection de fichiers CSS et JavaScript:

```
bootstrap/
└── css/
    ├── bootstrap-grid.css
    ├── bootstrap-grid.css.map
    ├── bootstrap-grid.min.css
    ├── bootstrap-grid.min.css.map
    ├── bootstrap-grid rtl.css
    ├── bootstrap-grid rtl.css.map
    ├── bootstrap-grid rtl.min.css
    ├── bootstrap-grid rtl.min.css.map
    ├── bootstrap-reboot.css
    ├── bootstrap-reboot.css.map
    ├── bootstrap-reboot.min.css
    ├── bootstrap-reboot.min.css.map
    └── bootstrap-reboot rtl.css
```



Bootstrap est construit avec **SASS** (Syntactically Awesome Style Sheets), un langage de script préprocesseur interprété ou compilé en CSS.

```
// SCSS (.scss)
nav {
  ul {
    margin: 0;
    padding: 0;
  }
  li { display: inline-block; }
  a {
    display: block;
    text-decoration: none;
  }
}
```

```
// SASS (.sass)
nav
  ul
    margin: 0
    padding: 0

    li
      display: inline-block

    a
      display: block
      text-decoration: none
```

```
/* CSS (.css) */
nav ul {
  margin: 0;
  padding: 0;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  text-decoration: none;
}
```

```
@use 'sass:color';

$colors: (primary: blue, secondary: green, danger: red);
$font-stack: 'Arial, sans-serif';

@mixin button-style($bg-color) {
  background: $bg-color;
  color: white;
  border-radius: 5px;
  font-family: $font-stack;
  cursor: pointer;
}

@each $name, $color in $colors {
  .btn-#{$name} {
    @include button-style($color);
    &:hover {
      background: color.adjust($color,$lightness: -20%);
    }
  }
}
```

SASS apporte de nombreuses features au langage:

- Variables (\$...)
- Syntaxe d'emboîtement (Nesting)
- Import (@use)
- Fonctions (@mixin)
- Héritage (@extend)
- Opérateurs mathématiques et boucles

↗ <https://sass-lang.com/playground/>

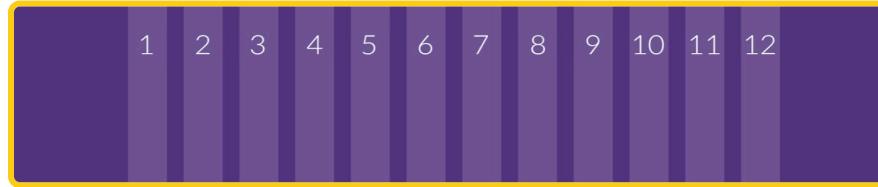
Gestion du layout

Concepts fondamentaux:

- Les **breakpoints** sont les éléments constitutifs du responsive design. Utilisez-les pour contrôler quand votre mise en page peut être adaptée à viewport ou à une taille d'appareil particulière.
- Utilisez les **media queries** pour architecturer votre CSS par breakpoints.
- **Mobile first, responsive design is the goal.** Le CSS de Bootstrap vise à appliquer le strict minimum de styles pour qu'une mise en page fonctionne au plus petit breakpoints, puis il superpose des styles pour ajuster cette conception aux appareils plus grands.

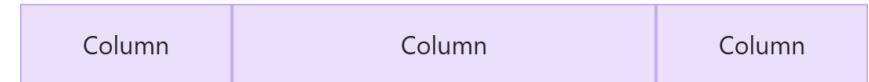
Breakpoint	Infixe de classe	Dimensions
Extra small	<i>None</i>	<576px
Small	sm	≥576px
Medium	md	≥768px
Large	lg	≥992px
Extra large	xl	≥1200px
Extra extra large	xxl	≥1400px

Bootstrap's grid system:



- Le système de grille de Bootstrap utilise une série de `containers`, `rows`, and `columns` pour mettre en page et aligner le contenu.
- Il est construit avec **flexbox** et est entièrement responsive.

```
<div class="container">
  <div class="row">
    <div class="col">Column</div>
    <div class="col-6">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```



💡 Il est possible d'emboîter une grille `.row` dans une `.col`.

Il est possible d'utiliser les **breakpoints au niveau des classes** pour facilement réaliser des layouts responsives:

La syntaxe est la suivante: `.col-{breakpoint}-{size}`

```
<div class="container">
  <div class="row">
    <div class="col-12 col-md-6"> ... </div>
    <div class="col-12 col-md-3"> ... </div>
    <div class="col-12 col-md-3"> ... </div>
  </div>
</div>
```

.col-12 .col-md-6

.col-12 .col-md-3

.col-12 .col-md-3

.col-12 .col-md-6

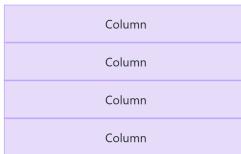
.col-12 .col-md-3

.col-12 .col-md-3

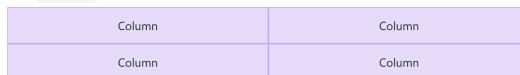
Utilisez les classes responsive `.row-cols-*` pour définir rapidement le nombre de colonnes qui correspondent le mieux à votre contenu et à votre mise en page:

```
<div class="container">
  <div class="row row-cols-1 row-cols-md-2 row-cols-lg-4">
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
    <div class="col">Column</div>
  </div>
</div>
```

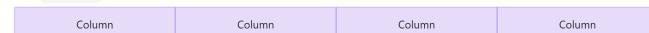
Default



\geq md



\geq lg



Gestion du contenu

Bootstrap propose un système de classes facilitant la gestion de la typographie:

```
<h1>h1. Bootstrap heading</h1>
<h2>h2. Bootstrap heading</h2>
<h3>h3. Bootstrap heading</h3>
<p class="h4">h4. Bootstrap heading</p>
<p class="h5">h5. Bootstrap heading</p>
<p class="h6">h6. Bootstrap heading</p>
```

```
<p class="fs-1">.fs-1 text</p>
<p class="fs-2">.fs-2 text</p>
<p class="fs-3">.fs-3 text</p>
<p class="fs-4">.fs-4 text</p>
<p class="fs-5">.fs-5 text</p>
<p class="fs-6">.fs-6 text</p>
```

h1. Bootstrap heading
h2. Bootstrap heading
h3. Bootstrap heading
h4. Bootstrap heading
h5. Bootstrap heading
h6. Bootstrap heading

.fs-1 text
.fs-2 text
.fs-3 text
.fs-4 text
.fs-5 text
.fs-6 text

Les composants

L'une des principales raisons du succès de Bootstrap est sa **profusion de composants**:

- Alerts
- Card
- Modal
- Navbar
- Tooltips
- <https://getbootstrap.com/docs/5.3/components>

Les composants de Bootstrap sont en grande partie construits à l'aide d'une nomenclature de **base-modifier**.

La syntaxe est la suivante: `.{base}-{modifier}`

```
<button class="btn">Base button</button>
<button class="btn btn-primary">Primary</button>

<nav class="navbar navbar-expand-lg"> ... </nav>
```

Cependant, la syntaxe est vite simplifiée pour offrir une meilleure DX:

```
<ul class="nav">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="#">Link</a>
  </li>
</ul>
```

Les utilitaires

Pour des besoins plus spécifique, Bootstrap propose de nombreuse classes utilitaires (utilities):

- Gestion des couleurs: `.bg-primary` , `.text-danger`
- Gestion des borders: `.border` , `.border-top` , `.border-primary` , `.rounded`
- Gestion du display: `.d-block` , `.d-none` , `.d-md-flex`
- Gestion de flexbox: `.justify-content-center` , `.align-items-end` , `flex-grow-1`
- Gestion de l'opacité: `.opacity-0` , `.opacity-75`
- Gestion des espacements: `.m-0` , `.px-5` , `.mt-auto` , `.gap-3` , `.row-gap-3` , `.column-gap-3`
- Gestion du texte: `.text-center` , `.text-lg-start` , `.text nowrap` , `.text-uppercase` , `.fw-bold`
- Gestion du contexte d'empilement: `.z-n1` , `.z-0` → `.z-3`
- <https://getbootstrap.com/docs/5.3/utilities>

Exercice

À vous de jouer: Réalisez l'interface suivante en utilisant uniquement les composants, le système de grille est les utilities de Bootstrap:

Oh my Interface!

This is a title

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Main call to action Secondary action

View Edit 9 mins

View Edit 9 mins

View Edit 9 mins

View Edit 9 mins

Instructions:

- Basez-vous sur le template d'exercice **bootstrap-starter**
- L'interface devra être responsive et mobile first
- Construisez une grille de composants `card`
 - Ils devront s'aligner sur une colonne
 - En taille `md`, affichez 2 `card` par ligne
 - En taille `lg` 3 et en `4`
- Une `navbar` sera visible avec le titre de l'interface
- À partir de la taille `xl`, tout le contenu de l'interface devra être centré
- La section de présentation devra être centré à partir de la taille `md`

Utilisation dans l'environnement Nodejs

Node.js est un environnement d'exécution JavaScript gratuit, open-source et multiplateforme qui permet d'**exécuter du code JavaScript en dehors d'un navigateur web**.

Cela permet de créer des serveurs, des applications web, des outils de ligne de commande et des scripts.

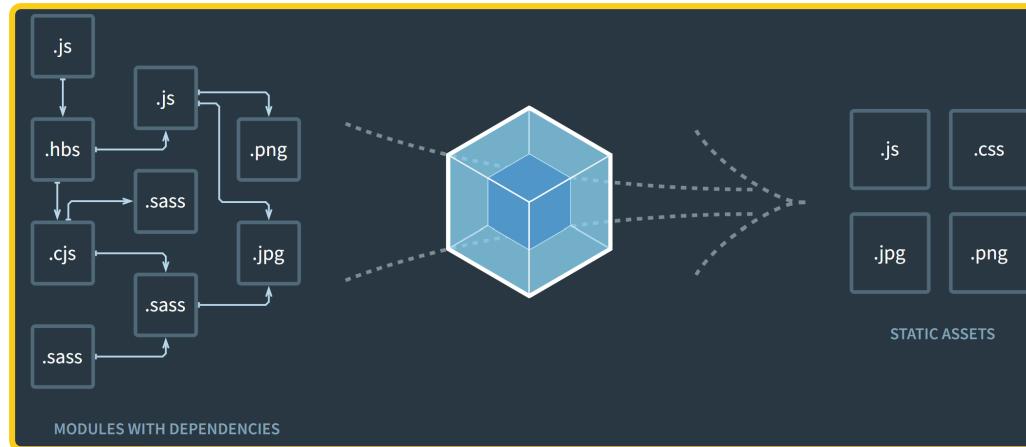


Illustration du fonctionnement d'un bundler JavaScript.



Le registre gratuit **npm** (Node Package Manager) est devenu le centre du partage de code JavaScript et, avec plus de **deux millions de packages**, le plus grand registre de logiciels au monde.

```
npm install bootstrap  
npm install -D sass
```

À vous de jouer: Dans le repo [isfates-l1-outils-dev-sites-web/exercises](#), ouvrez dans votre IDE le template **bootstrap-vite** et suivez les instructions du fichier `README.md`.

Gestion du theme

En faisant un import de Bootstrap via `scss`, il nous est facilement possible de s'approprier le theme:

```
// style.scss
$primary: green;

$grid-breakpoints: (
  xs: 0,
  sm: 576px,
  md: 768px,
  lg: 992px,
  xl: 1200px,
  xxl: 1400px
);

@import "bootstrap/scss/bootstrap";
```

Vous pouvez consulter toutes les variables Bootstrap sous dans le fichier `SCSS` :

```
node_modules/bootstrap/scss/_variables.scss
```

À vous de jouer: Modifiez le theme de Bootstrap de sorte à:

- augmenter toutes les espacements (`spacer`) de 2 fois la taille originale
- définir tous les `border-radius` à `0`
- augmenter toutes les tailles de police (`font-size`) de 1.5 fois la taille originale

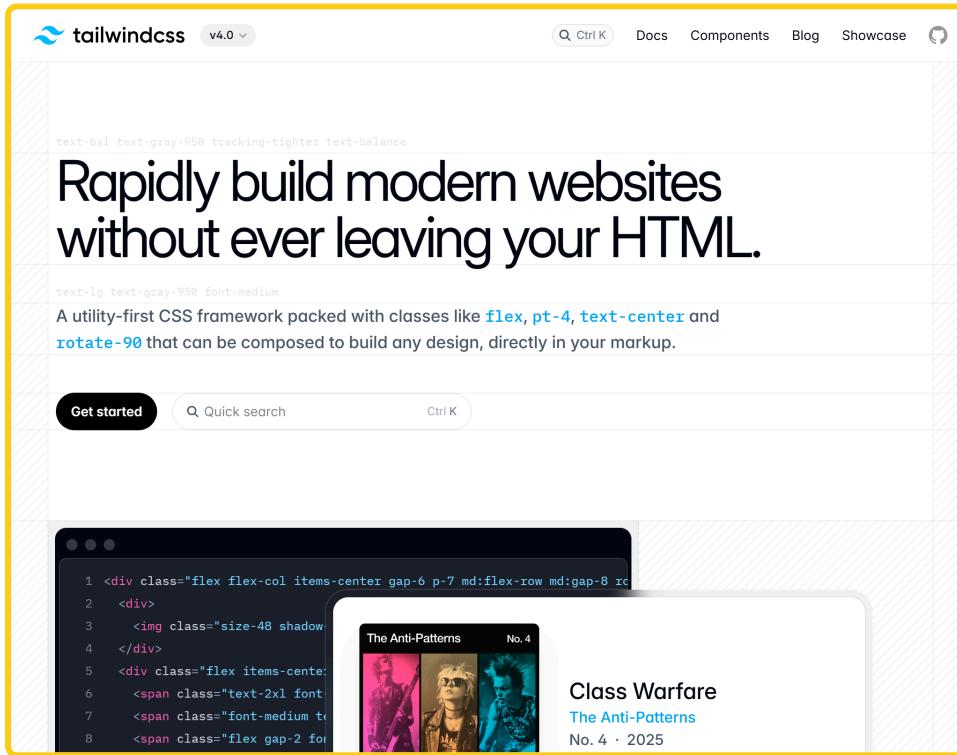
Correction:

```
// styles.scss

@import "bootstrap/scss/bootstrap";

//#
```

Tailwindcss



<https://tailwindcss.com/>

Installation

Tailwind CSS **analyse tous vos fichiers HTML**, composants JavaScript et autres templates à la recherche de noms de classes, **génère les styles correspondants** et les écrit dans un fichier CSS statique.

Cela permet de s'assurer que votre **CSS est aussi petit que possible**, et c'est également ce qui rend possible des fonctionnalités telles que les valeurs arbitraires.

- `.text-[22px]`
- `.top-[117px]`

💡 Là où d'autres bibliothèques peuvent inclure un grand nombre de codes supplémentaires, tailwind css ne livrera que ce qui est réellement utilisé.

Tailwindcss est communément utilisé dans l'écosystème **Nodejs**, et s'installe via son package manager **NPM**:

```
npm install tailwindcss @tailwindcss/cli
```

Dans le cadre de ce cours, nous utiliserons sa version CDN:

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <script src="https://unpkg.com/@tailwindcss/browser@4"></script>
  </head>
  <body> ... </body>
</html>
```

Utilisez le template **tailwind-starter** dans le repository [exercises](#).

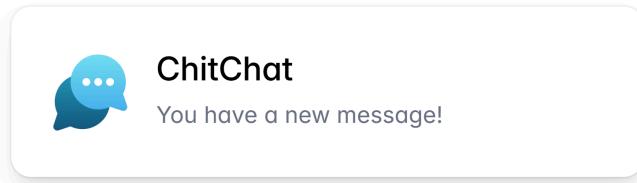
Dans ce cours nous utiliserons la version 4 de tailwind. Celle-ci apporte 2 grands changements à la version précédente:

- **Configuration CSS-first:** la configuration de votre projet passe du JavaScript (`tailwind.config.js`) à du CSS pure.
- **Thème utilisant des variables CSS:** Tailwind CSS v4.0 prend tous design tokens et les rend disponibles comme variables CSS par défaut.

```
@import "tailwindcss";  
  
@theme {  
  --font-display: "Satoshi", "sans-serif";  
  --breakpoint-3xl: 1920px;  
}
```

Styliser avec des classes utilitaires

Avec Tailwind, vous donnez du style en combinant de nombreuses classes de présentation à usage unique (*utility classes*) directement dans votre balisage:



```
<div class="mx-auto flex max-w-sm items-center gap-x-4 rounded-lg bg-white p-6 shadow-lg outline outline-black/5">
  
  <div>
    <div class="text-xl font-medium text-black">ChitChat</div>
    <p class="text-gray-500">You have a new message!</p>
  </div>
</div>
```

Cette façon de concevoir les choses va à l'encontre de nombreuses bonnes pratiques traditionnelles. Elle possède néanmoins de nombreux avantages:

- **La réalisation est plus rapide:** ne perdez pas de temps à trouver des noms de classes, à prendre des décisions concernant les sélecteurs ou à passer d'un fichier HTML à un fichier CSS.
- **Les changements sont plus sûrs:** l'ajout ou la suppression d'une classe d'utilité à un élément n'affecte jamais que cet élément.
- **Votre CSS cesse de croître:** les classes utilitaires sont réutilisables, votre CSS ne continue pas à croître linéairement avec chaque nouvelle fonctionnalité que vous ajoutez à un projet.
- **La maintenance des anciens projets est facilité:** changer quelque chose signifie simplement trouver cet élément dans votre projet et changer les classes.

Comme Bootstrap, TailwindCSS utilise des **classes avec des variants** pour appliquer des styles ciblés.

Pour styliser un élément sur des états tels que le survol ou le focus, **préfixez** n'importe quel classe **avec l'état** que vous souhaitez cibler:

```
<button class="bg-sky-500 hover:bg-sky-700 ...>Save changes</button>
```

- **Pseudo-classes:** `:hover` , `:focus` , `:first-child` et `:required`
- **Pseudo-éléments:** `::before` , `::after` , `::placeholder` et `::selection`
- **Sélecteurs d'attributs:** `[dir=<< rtl >>]` et `[open]`
- **Sélecteurs enfant:** `& > *` et `& *`
- <https://tailwindcss.com/docs/hover-focus-and-other-states>

Responsive design

Chaque classe utilitaire de Tailwind peut être appliquée de manière conditionnelle à différents **breakpoints**, ce qui facilite la construction d'interfaces réactives complexes sans jamais quitter votre HTML.

```

```

Breakpoint	Préfixe de classe	Dimensions
Small	sm	$\geq 40\text{rem}$
Medium	md	$\geq 48\text{rem}$
Large	lg	$\geq 64\text{rem}$
Extra large	xl	$\geq 80\text{rem}$
Extra extra large	2xl	$\geq 96\text{rem}$

Ces tailles sont customisable dans le thème de TailwindCSS:

```
@theme {  
  --breakpoint-xs: 30rem;  
  --breakpoint-2xl: 100rem;  
}
```

Pour appliquer un utilitaire uniquement sur **une plage de breakpoints spécifique**, empilez un variant responsive comme `md` avec un variant `max-*` :

```
<div class="md:max-xl:flex">  
  ...  
</div>
```

Ajout de styles personnalisés

Une des plus grandes limitations lorsque l'on travaille avec un framework est de comprendre ce que l'on est censé faire lorsque que le framework ne le gère pas.

"No matter what you're building you never feel like you're fighting the framework."

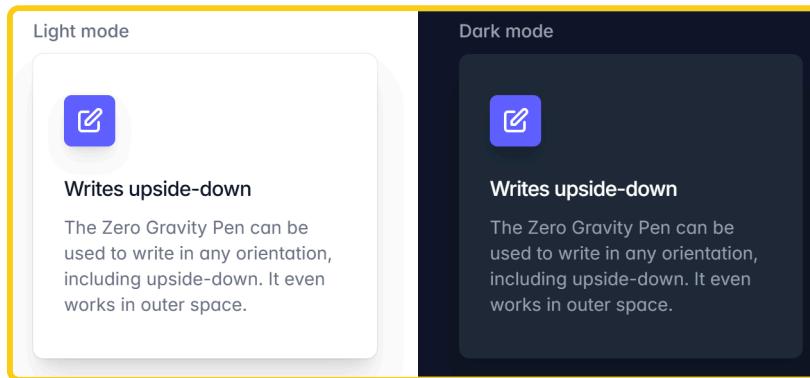
— <https://tailwindcss.com/docs/adding-custom-styles>

TailwindCSS analysant votre `html`, il est possible d'utiliser des valeurs arbitraires:

```
<div class="top-[117px] lg:top-[344px]>
  <div class="grid grid-cols-[1fr_500px_2fr]>
    <!-- ... -->
  </div>
  <div class="bg-[url('/what_a_rush.png')]">
    <!-- ... -->
  </div>
</div>
```

Dark mode

Tailwind inclut un variant `dark` qui permet de styliser votre site différemment lorsque le mode sombre est activé.



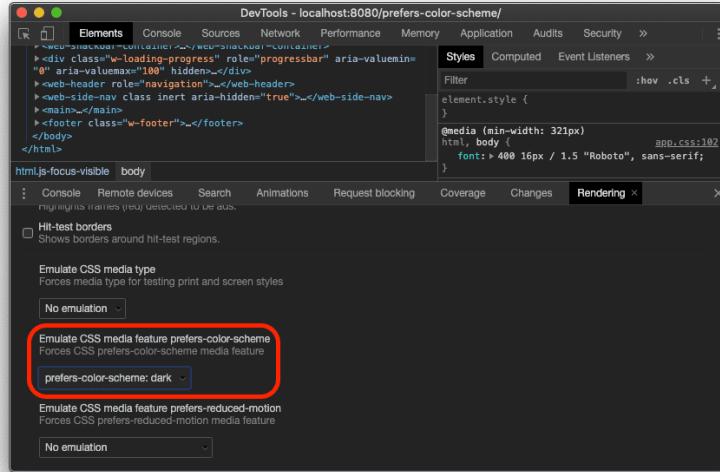
```
<div class="bg-white dark:bg-gray-800 text-gray-900 dark:text-white ...">  
  <!-- ... -->  
</div>
```

Par défaut, cette fonction utilise la fonctionnalité de média CSS native `prefers-color-scheme` :

```
:root {  
  --text-color: black;  
}  
  
h1 {  
  color: var(--text-color);  
}  
  
@media (prefers-color-scheme: dark) {  
  :root {  
    --text-color: white;  
  }  
}
```

Il existe 2 solutions pour tester le dark mode:

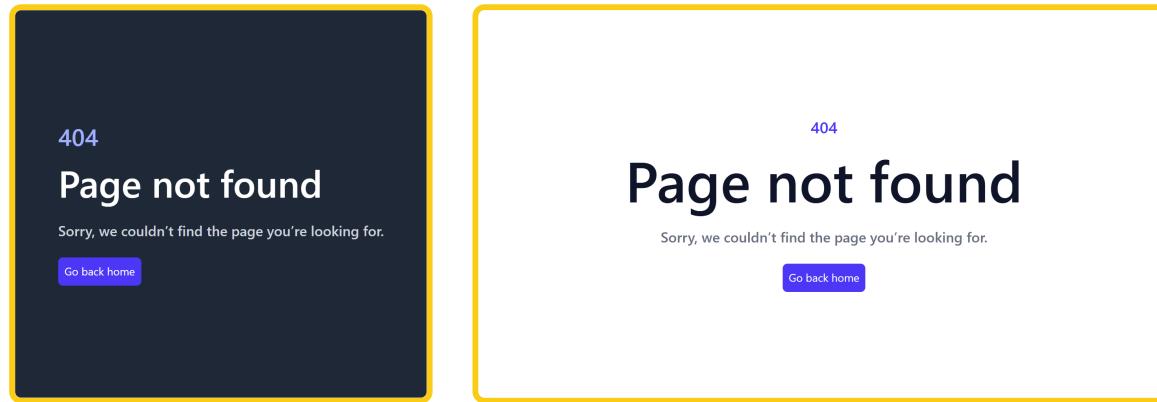
1. Utiliser le mode dark dans votre inspecteur:  > More Tools > Rendering



2. Utiliser la classe `dark` : `<body class="dark"> ... </body>`.

Exercice

À vous de jouer: Réalisez l'interface suivante en utilisant uniquement les classes de TailwindCSS. Ouvrez pour cela l'exercice `tailwind-404-page` :



Instructions:

- Basez-vous sur le template d'exercice **tailwind-404-page**
- L'interface devra être lisible en dark mode
- La couleur primaire à utiliser se nomme `indigo-600`
- Le contenu de la page sera centré horizontalement et verticalement
- Le texte sera aligné sur la gauche. Passé le breakpoint `sm`, il devra être centré.
- La taille du texte `404` sera réduite passé le breakpoint `sm`
- Utilisez la documentation <https://tailwindcss.com/docs> et sa **barre de recherche**

En résumé

- **Techniques de mise en page**
 - Approches modernes: Static, Fluid, Adaptive, Responsive
 - Media Queries: personnalisation selon les écrans (taille, type de media, ...)
 - Flexbox & CSS Grid: modules avancés pour des mises en page dynamiques
- **Frameworks CSS**
 - Gain de temps & structure
 - Bootstrap: Composants réutilisables, grille flexible
 - TailwindCSS: Classes utilitaires pour un design sur-mesure
- **Tendances et évolutions du développement web**
 - **CSS moderne**: évolution avec des nouvelles fonctionnalités (grid , flexbox , sass ,...)
 - **Mobile-first**: l'optimisation mobile est devenue une priorité
 - **Design system**: structurer les styles avec des bibliothèques cohérentes

Ressources

Ce cours a été élaboré à partir des ressources suivantes :

- <https://product-alpaca.medium.com/so-what-exactly-is-the-difference-between-fixed-fluid-adaptive-and-responsive-layouts-and-why-3773272d8481>
- <https://ux.stackexchange.com/questions/21/for-websites-is-it-better-to-have-a-variable-width-layout-or-a-fixed-width-layo>
- <https://www.geeksforgeeks.org/advantages-and-disadvantages-of-responsive-web-design/>
- <https://alistapart.com/article/responsive-web-design/>
- <https://developer.mozilla.org/en-US/docs/Web/CSS>
- <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- <https://css-tricks.com/understanding-flex-grow-flex-shrink-and-flex-basis/>
- <https://getbem.com/>
- <https://github.com/troxler/awesome-css-frameworks>
- <https://strapi.io/blog/bootstrap-vs-tailwind-css-a-comparison-of-top-css-frameworks>
- <https://sass-lang.com/>
- <https://tailwindcss.com/docs>
- <https://nodejs.org/en>
- <https://www.npmjs.com/>

