



# Développement Web et MultiMedia 1

## CHAPITRE 6

### LES FEUILLES DE STYLE EN CASCADE (CSS3)

# Qu'est-ce que CSS ?

- ▶ CSS : **Cascading Style Sheets**
- ▶ Permet la création de styles (règles de mise en forme) qui définissent comment les éléments HTML doivent être affichés (couleurs, polices, marges/padding, taille, positionnement, bordures ...)
- ▶ Permet la séparation du contenu de la mise en forme (contenu/structure dans le HTML et styles dans le CSS)

# Structure de base

- ▶ Une règle en CSS se compose d'un **sélecteur** et un bloc de déclaration.
  - ▶ Le sélecteur, c'est la balise HTML (body , h1 , p, etc.), l'identifiant (id) ou la classe (class) ;
  - ▶ Le bloc de déclaration contient une ou plusieurs déclarations séparées par des points-virgules
- ▶ Chaque **déclaration** est composée de :
  - ▶ La **propriété**, c'est l'attribut qu'on veut appliquer (font , background, etc.)
  - ▶ La **valeur** qui précise les caractéristiques de la propriété



# Structure de base

## ► Syntaxe

```
Sélecteur {  
    propriété1: valeur1;  
    propriété2: valeur2;  
}
```

## ► Exemple



The diagram illustrates the CSS syntax for an example. It shows the selector 'h1' in a pink circle, followed by an opening curly brace. The first property 'color' is in an orange box, followed by a colon, the value 'blue' in a yellow box, and a semicolon. The second property 'font-size' is in an orange box, followed by a colon, the value '12px' in a yellow box, and a closing curly brace. The entire example is enclosed in a green rounded rectangle.

```
h1 { color : blue ; font-size : 12px }
```

“

# INTÉGRATION DANS LE HTML

”

# Intégration dans le HTML

- ▶ Il y a 3 méthodes pour intégrer du CSS dans le HTML :
  - ▶ **Intégrés (en anglais : in-line styles)** directement dans les balises du fichier HTML via un attribut style (**méthode la moins recommandée**).
  - ▶ Dans l'en-tête **<head>** du fichier HTML ;
  - ▶ Dans un fichier **.css** externe (**méthode la plus recommandée**) ;

# Styles intégrés (*in-line styles*)

- ▶ Les styles **pour un seul élément** se trouvent dans la balise d'ouverture de l'élément à l'aide de **l'attribut style**.
  - ▶ Aucune séparation de contenu et de la mise en forme
  - ▶ Aucune réutilisation des styles
- ▶ Exemple

```
<html>

  <body>

    <h1 style="color:red ; text-align:center"> Un premier titre
  </h1>

    <h2> Un deuxième titre </h2>

  </body>

</html>
```

# Feuille de Style Interne

- ▶ Les styles se trouvent **dans le fichier HTML** entre des balises `<style>` et `</style>` qui doivent se trouver dans **la section <head>** du document HTML.
- ▶ (+) Tout ce dont on a besoin pour spécifier le contenu et son affichage est dans le même fichier
- ▶ (-) Ne sépare pas nettement (au niveau des fichiers) le contenu et la mise en forme
- ▶ (-) Les styles déclarés ainsi ne s'appliquent que sur le document HTML dans lequel ils se trouvent (ils ne peuvent pas s'appliquer à d'autres documents HTML)



# Feuille de Style Interne

- ▶ Exemple précédant avec des styles **CSS internes**

```
<html>
  <head>
    <style>
      h1 {color:red; text-align : center; }
    </style>
  </head>
  <body>
    <h1> Un premier titre </h1>
    <h2> Un deuxième titre </h2>
  </body>
</html>
```

# Fichier CSS externe

- ▶ Les styles se trouvent dans un fichier « .CSS » à part.
- ▶ Le fichier CSS est lié au fichier HTML via une balise **<link>** à ajouter dans l'entête **<head>** du document.
- ▶ Exemple précédant avec des styles **CSS externes**

fichier\_style.css

```
h1 {    color : red ;  
      text-align : center;}
```

exemple3.html

```
<html>  
  <head>  
    <link  rel="stylesheet"  
          type="text/css"  
          href="fichier_style.css" >  
  </head>  
  <body>  
    <h1> Un premier titre </h1>  
    <h2> Un deuxième titre </h2>  
  </body>  
</html>
```

# Fichier CSS externe

- ▶ La balise `<link>` avertit le navigateur qu'il faudra réaliser un lien.
- ▶ L'attribut `rel=stylesheet` précise qu'il y trouvera une feuille de style externe.
- ▶ L'attribut `type="text/css"` précise que l'information est du texte et du genre CSS.
- ▶ L'attribut classique de lien `href` donne le chemin d'accès au fichier CSS à lier.
- ▶ **NB :** Dans un même document HTML on peut inclure plusieurs fichiers CSS.

▶ Exemple :

```
<link rel="stylesheet" type="text/css" href="reset.css">  
<link rel="stylesheet" type="text/css" href="layout.css">  
<link rel="stylesheet" type="text/css" href="skin.css">
```

# Les commentaires en CSS

- ▶ Un commentaire CSS commence par `/*` et se termine par `*/`.
- ▶ Les commentaires peuvent également étendre sur plusieurs lignes.
- ▶ **Exemple**

```
p {      /* This is a single-line comment */
  text-align: center;

  /* This is a multiple-line
     comment */
}
```

“

# LES SÉLECTEURS

”

# Sélecteur d'élément

- Définir un style CSS pour un élément en se basant sur son nom.
- Ainsi, tous les éléments ayant le même nom suivent ce style.

- **Exemple**

```
<html>
  <head>
    <style>
      p { text-align: center ; color: red;}
    </style>
  </head>
  <body>
    <p> Un premier paragraphe </p>
    <p> Un deuxième paragraphe </p>
  </body>
</html>
```



Tous les éléments **<p>** suivent le même style CSS : ils seront alignés au centre, avec une couleur de texte en rouge

Un premier paragraphe  
Un deuxième paragraphe

# Sélecteur d' *id*

- ▶ But : utiliser l'attribut « **id** » d'un élément HTML pour lui attribuer un style CSS unique (un style pour un seul élément).
  - ▶ Définir tout d'abord un style CSS pour un sélecteur id selon la syntaxe suivante :

```
#nom_id {liste des déclarations}
```

- ▶ Seulement l'élément HTML (unique) ayant comme `id = nom_id` suit ce style

# Sélecteur d' *id*

## ► Exemple

```
<html>
  <head>
    <style>
      #para1 { text-align: center; color: red;}
    </style>
  </head>
  <body>
    <p id= "para1"> Un premier paragraphe </p>
    <p id="para2">  Un deuxième paragraphe </p>
  </body>
</html>
```

La règle de style ci-dessus sera appliquée à l'élément HTML avec id = "para1" seulement

Un premier paragraphe

Un deuxième paragraphe



# Sélecteur de *class*

- ▶ But : attribuer un style pour **certain**s éléments appartenant à une classe bien déterminées
  - ▶ Définir tout d'abord une **class** de style CSS selon la syntaxe suivante :

```
.nom_class{liste des déclarations}
```

- ▶ Tout élément contenant dans sa balise d'ouverture le nom de la class **<nom\_element class="nom\_class">** suit ce style
- ▶ Les éléments HTML peuvent se référer à plus d'une classe
- ▶ Il est possible de spécifier que seuls certains éléments HTML spécifiques devraient être affectés par une classe.

# Sélecteur de **class**: Exemple 1

```
<html>
<head>
  <style>
    .type { text-align: center; color: red;}
  </style>
</head>
<body>
  <h1 class= "type"> Titre </h1>
  <p class= "type"> Un premier paragraphe
</p>
  <p> Un deuxième paragraphe </p>
</body>
</html>
```

Tous les éléments HTML avec  
**class = "type"** seront rouges et  
alignés au centre

**Titre**

Un premier paragraphe

Un deuxième paragraphe

## Sélecteur de **class**: Exemple 2

```
<html>
<head>
  <style>
    .type { text-align: center; color: red;}
    .large { font-weight: bold;}
  </style>
</head>
<body>
  <p class= "large type"> Un premier paragraphe </p>
  <p class= "type"> Un deuxième paragraphe </p>
</body>
</html>
```

**Un premier paragraphe**

Un deuxième paragraphe

# Les sélecteurs avancés

Sélecteur	Exemple
<b>A,B { } : regroupement de sélecteur</b>	<b>h1, h2 { text-align: center; color: red; }</b> Attribuer le même style CSS aux titres h1 et h2
<b>* { } : sélecteur universel</b>	Attribuer un style CSS pour toutes les balises sans exception
<b>A B { } : une balise contenue dans une autre</b>	<b>h3 i { color : red ; }</b> Sélectionne toutes les balises <i> situées à l'intérieur d'une balise <h3>
<b>A + B { } : une balise qui en suit une autre</b>	<b>h3 + p { }</b> Sélectionne la première balise <p> située après un titre <h3>.
<b>A[attribut] { } : une balise qui possède un attribut</b>	<b>a[title] { }</b> Sélectionne tous les liens <a> qui possèdent un attribut title. Exemple : <a href="http://site.com" title="Infobulle">

# Les sélecteurs avancés: Exemple

```
<html>
<head>
  <style>
    h1, h3 {color:red;}
    p strong {border:2px dashed blue;}
    p[title] {color:green;}
    * {font-family: broadway;}
  </style>
</head>
<body>
  <h1> Titre 1 </h1>
  <h3> Titre 2 </h3>
  <p> Un premier <strong>paragraphe</strong></p>
  <h3>Titre 3 </h3>
  <p title="informations"> Un deuxième paragraphe </p>
  <h2> Titre <strong>3</strong></h2>
</body>
</html>
```

**Titre 1**

**Titre 2**

**Un premier paragraphe**

**Titre 3**

**Un deuxième paragraphe**

**Titre 3**

“

# HÉRITAGE EN CSS


”

# Notion d'héritage

- ▶ L'héritage signifie que tout élément HTML **enfant** va hériter, « **en cascades** », des styles de **ses parents**.
- ▶ C'est par ailleurs de là que vient le nom du CSS : Cascading StyleSheets, ou Feuilles de Style en Cascades.
- ▶ **Exemple** : tous les éléments à l'intérieur de l'élément body sont des enfants de cet élément.
  - ▶ Si l'on applique un style à l'élément body, ses enfants en hériteront automatiquement.
- ▶ S'il y a un conflit, c'est-à-dire si un élément reçoit plusieurs fois une **même propriété** avec des **valeurs différentes**, le style prioritaire est **le plus proche de l'élément**.

# Exemple 1

```
<html>
<head>
  <meta charset="utf-8">
  <style>
    body {color: purple ;}
  </style>
</head>
<body>
  <h1>Un titre de niveau 1 </h1>
  <p>Un paragraphe avec<strong>texte important</strong> </p>
  <p> Un autre paragraphe </p>
</body></html>
```



les éléments h1, p et strong ont hérité des styles de leur parent **body**. Ainsi, nos différents textes s'affichent en violet.

**Un titre de niveau 1**

Un paragraphe avec **texte important**

Un autre paragraphe



## Exemple 2

```
<html>
<head>
  <meta charset="utf-8">
  <style>
    body    {color:  purple ;}
    p       {color:blue;}
    #para   {color: red;}
    strong  {color:green}
  </style>
</head>
<body>
  <h1> Un titre de niveau 1 </h1>
  <p> Un paragraphe avec <strong> texte important </strong>
    </p>
  <p> Un autre paragraphe </p>
  <p id="para"> Un troisième paragraphe </p>
</body>
</html>
```

**Un titre de niveau 1**

Un paragraphe avec **texte important**

Un autre paragraphe

Un troisième paragraphe

## Exemple 2: Explication

- ▶ Notez que nous appliquons à chaque fois la même propriété **color** à nos différents éléments avec des valeurs différentes.
  - ▶ Il va donc y avoir **conflit**.
- ▶ On applique une couleur violette à notre élément body.
  - ▶ Ainsi, tous les éléments contenus dans body vont hériter de cette couleur sauf si une autre couleur est définie entre temps.
- ▶ Pour les éléments de type p, auxquels on attribue une couleur bleue: *nos paragraphes doivent-ils hériter des styles définis avec le sélecteur body ou de ceux définis avec le sélecteur p ?*
  - ▶ Le sélecteur p cible de manière plus précise les paragraphes que le sélecteur body, et le style défini dans le sélecteur p est donc plus proche de nos paragraphes que celui défini dans body; c'est donc bien celui-ci qui sera appliqué.
  - ▶ Par défaut, tous nos paragraphes seront donc bleus

## Exemple 2: Explication

- Pour le paragraphe comportant ***l'id="para3"***. Les éléments possédant cet attribut vont avoir la couleur rouge.
- Pour le texte dans ***l'élément strong***, celui-ci récupère le style le plus proche de lui, c'est-à-dire celui qui lui est appliqué directement via le sélecteur strong. Donc, il est vert.

“

# LES ÉLÉMENTS DIV ET SPAN

”

# La balise `<div>`

- ▶ La balise `<div>` permet de définir un "bloc" contenant du code HTML pouvant lui-même comprendre d'autres balises.
- ▶ Ce bloc pourra :
  - ▶ Posséder un style spécifique
  - ▶ Il est très commun d'attribuer un attribut `class` à un élément `div` afin de pouvoir le cibler plus facilement.
  - ▶ Être positionné à l'endroit de son choix dans la page
- ▶ `<div>` possède les attributs *margin*, *padding*, *width*, *height*.
- ▶ **NB** : Les navigateurs placent toujours un saut de ligne avant et après un élément `<div>`.

# Exemple

```
<html>
<head>
  <style>
    .class-div { background-color : #88BB11;
                  font-weight:bold;}
  </style>
</head>

<body>
  <h1> Elément Div </h1>
  <div class="class-div">
    <h2> Sous titre </h2>
    <p> Un paragraphe avec texte important </p>
    <p> Un autre paragraphe </p>
  </div>
</body>
</html>
```

## Elément Div

### Sous titre

Un paragraphe avec texte important

Un autre paragraphe

# La balise `<span>`

- ▶ La balise `<span>` permet d'appliquer des styles à des parties du texte d'un paragraphe.
- ▶ Elle ne provoque pas de saut de ligne avant ou après et elle occupe juste l'espace horizontal nécessaire.

# Exemple

```
<html>
<head>
  <style>
    .green {background-color : #88BB11;}
    .blue {background-color : #1188BB;}
  </style>
</head>
```

```
<body>
  <h1> Eléments span </h1>
  <p> Un paragraphe avec <span class="green">texte vert
    </span> </p>
  <p> Un paragraphe avec <span class="blue"> texte bleu
    </span></p>
</body>
</html>
```

## Eléments span

Un paragraphe avec texte vert

Un paragraphe avec texte bleu



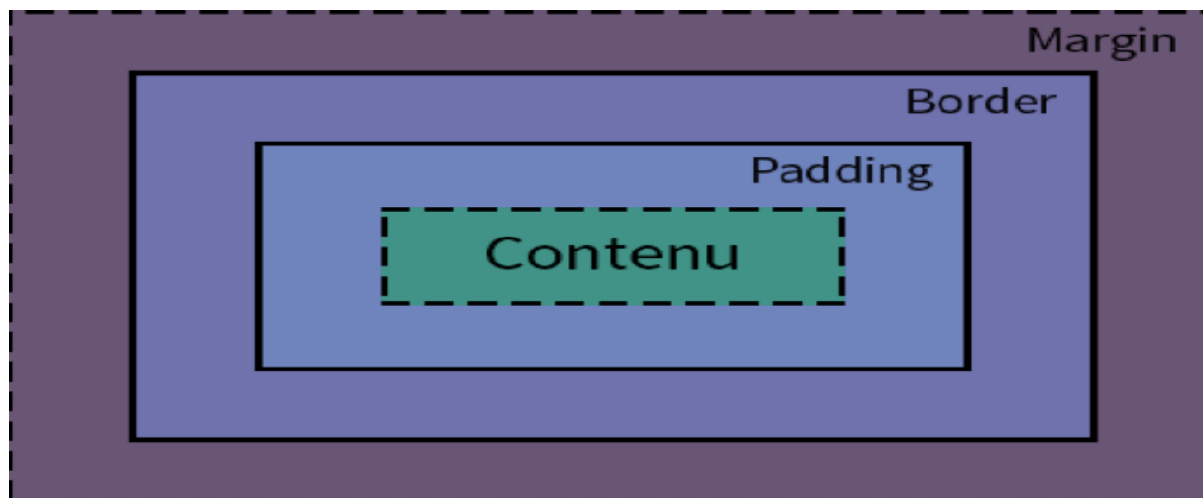
“

# POSITIONNEMENT EN CSS

”

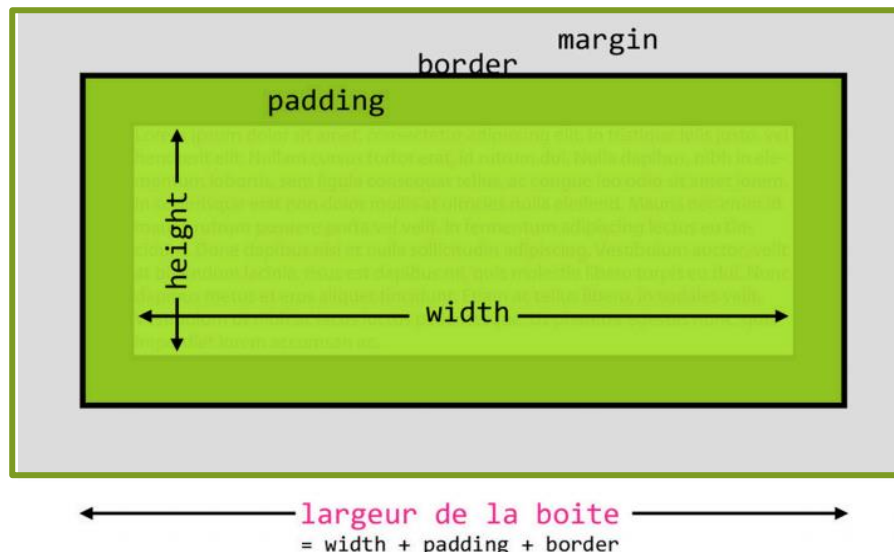
# Le modèle de boîte

- ▶ Tout élément HTML = une boîte
- ▶ La première boîte représente le contenu du document
- ▶ Autour du contenu, la deuxième boîte contient la marge intérieure de l'élément La troisième boîte contient la bordure de l'élément
- ▶ La dernière boîte va contenir la marge extérieure de l'élément



# Le modèle de boîte

- ▶ Tout élément HTML = une boîte
- ▶ La propriété **width** définit la largeur totale d'un élément ?
  - ▶ Par défaut, la propriété **width** définit la largeur du **contenu**.
  - ▶ La largeur totale d'un élément se définit comme suit :  
**Largeur = width + padding + border**



# Exemple

```
<html> <head>
  <style>
    .p1 {
      /* font: bleu-vert */      background-color:#088;
      /* largeur de l'élément */  width: 200px;
      /* marge intérieure */     padding : 30px;
      /* bordure: vert */        border: 20px solid #0C0;
      /* marge extérieure */     margin: 50px;
    }
    .p2 {
      /* font: pink */           background-color:pink;
      /* largeur de l'élément */  width: 400px;
      /* marge intérieure */     padding : 50px;
      /* bordure: vert */        border: 30px solid blue;
      /* marge extérieure */     margin: 50px;
    }
  </style>
</head>
```

# Example

Para 1 spacing

```
<body >  
  <p class= "p1">  
    Para 1 spacing </p>  
  <p class= "p2">  
    Para 2 spacing </p>  
</body>  
</html>
```

Para 2 spacing

# Unités de mesure CSS

- ▶ CSS offre différentes unités pour exprimer les dimensions.
- ▶ Il n'y a pas de restriction à utiliser telle unité à tel ou tel endroit: Si une propriété accepte une valeur en px ('margin: 5px') elle accepte également une valeur en pouces ou en centimètres ('margin: 1.2in; margin: 0.5cm') et vice-versa.

	<b>Recommandé</b>	<b>Usage occasionnel</b>	<b>Non recommandé</b>
<b>Écran</b>	em, px, %	ex	pt, cm, mm, in, pc
<b>Imprimante</b>	em, cm, mm, in, pt, pc, %	px, ex	

- ▶ Source: <https://www.w3.org/Style/Examples/007/units.fr.html>

# Unités de mesure CSS

Unité	Valeurs		Exemple	Description
<b>%</b>	Relatives	entière	25%	Pourcentage par rapport à une référence (taille de police d'une boîte bloc, fenêtre, calque, cellule, etc.)
<b>px</b>		entière	200px	Pixel (un point sur l'écran de l'ordinateur)
<b>em</b>		réelle	2.5em	L'unité est la largeur de la lettre M (majuscule)
<b>ex</b>		réelle	0.5ex	L'unité est la hauteur de la lettre x (minuscule)
<b>pt</b>	absolues	entière	14pt	Point typo (1 pt = 1/72 inch, 1/12 pica)
<b>pc</b>		réelle	12pc	Pica (12 points, 1/6 pouce)
<b>cm</b>		entière	10cm	Centimètre
<b>mm</b>		entière	5mm	Millimètre
<b>in</b>		réelle	3in	Inch (1 inch = 1 pouce = 2.54 cm)

# La notion de flux

- ▶ Le navigateur affiche les éléments HTML dans leur ordre d'apparition
  - ▶ Le navigateur procède verticalement, du début à la fin du document
- ▶ Les éléments de type **block** sont affichés en succession verticale
  - ▶ p, div, li, section, ...
  - ▶ **Par défaut, un élément block occupe l'intégralité de la largeur de son parent**
- ▶ Les éléments de type **inline** sont affichés en succession horizontale
  - ▶ span, a, em, ...
  - ▶ **Par défaut, un élément inline a la largeur de son contenu**



# Positionner un bloc en CSS

- ▶ La propriété **position** permet de modifier le placement d'un élément dans le flux
  - ▶ 4 valeurs : absolute, relative, fixed, static
- ▶ **position: static**
  - ▶ comportement par défaut
  - ▶ Avec cette option, le concepteur ne peut pas contrôler le positionnement et la visibilité de l'élément

# Positionner un bloc en CSS

- ▶ En utilisant des positions absolues/relatives
- ▶ **position: relative** positionnement par rapport à l'élément précédent
- ▶ **position: absolute** positionnement par rapport au coin supérieur gauche du conteneur
- ▶ **top: Npx;** distance depuis le bord supérieur
- ▶ **left: Npx** distance depuis le bord gauche
- ▶ **position: fixed;** positionnement pour fixer le bloc dans la fenêtre et le garder visible lors d'un défilement de la page
- ▶ Autres propriétés possibles après position:
  - ▶ rendre le bloc visible ou invisible  
**visibility: visible|hidden** (ne s'applique qu'aux blocs)
  - ▶ superposer des blocs **z-index: 1|2|3...**

# float

- ▶ On peut retirer un élément du flux pour faire en sorte que le contenu qui le suit «s'écoule» autour de lui.
- ▶ On utilise pour cela la propriété **float**
  - ▶ **float:left** -> l'élément flotte à gauche
  - ▶ **float:right** -> l'élément flotte à droite

# Exemple 1

```
<html>
<head><style>
  div.conteneur {
    background-color: teal;
    height: 200px; width: 200px;
    position: absolute; top: 20px; left:40px;}
  div.bloc1 {
    background-color: yellow;
    height: 50px; width: 50px;
    position: absolute; top: 50px; left:100px;}
</style></head>
<body>
  <div class="conteneur" id="conteneur">bla bla bla bla...
  <div class="bloc1" id="bloc1">bloc 1</div>
</div>
</body></html>
```

bla bla bla bla bla bla bla bla  
bla bla bla bla

bloc 1

## Exemple 2

```
<html>
<head><style>
  div.conteneur {
    background-color: teal;
    height: 200px; width: 200px;
    position: absolute; top: 20px; left: 40px; }
  div.bloc1 {
    background-color: yellow;
    height: 50px; width: 50px;
    position: relative; display: inline; }
</style></head>
<body>
  <div class="conteneur" id="conteneur">bla bla bla bla...
  <div class="bloc1" id="bloc1">bloc 1</div>
</div>
</body></html>
```

bla bla bla bla bla bla bla bla  
bla bla bla bla bloc 1

# Exemple 3

```
<html>
<head><style>
  div.bloc1 {
    background-color: teal;
    height: 100px; width: 100px;
    position: absolute; top: 40px; left: 40px; z-index: 2; }
  div.bloc2 {
    background-color: yellow;
    height: 100px; width: 100px;
    position: absolute; top: 20px; left: 20px; z-index: 1; }
</style></head>
<body>
  <div class="bloc1" id="bloc1">bloc 1</div>
  <div class="bloc2" id="bloc2">bloc 2</div>
</body></html>
```

bloc 2

bloc 1



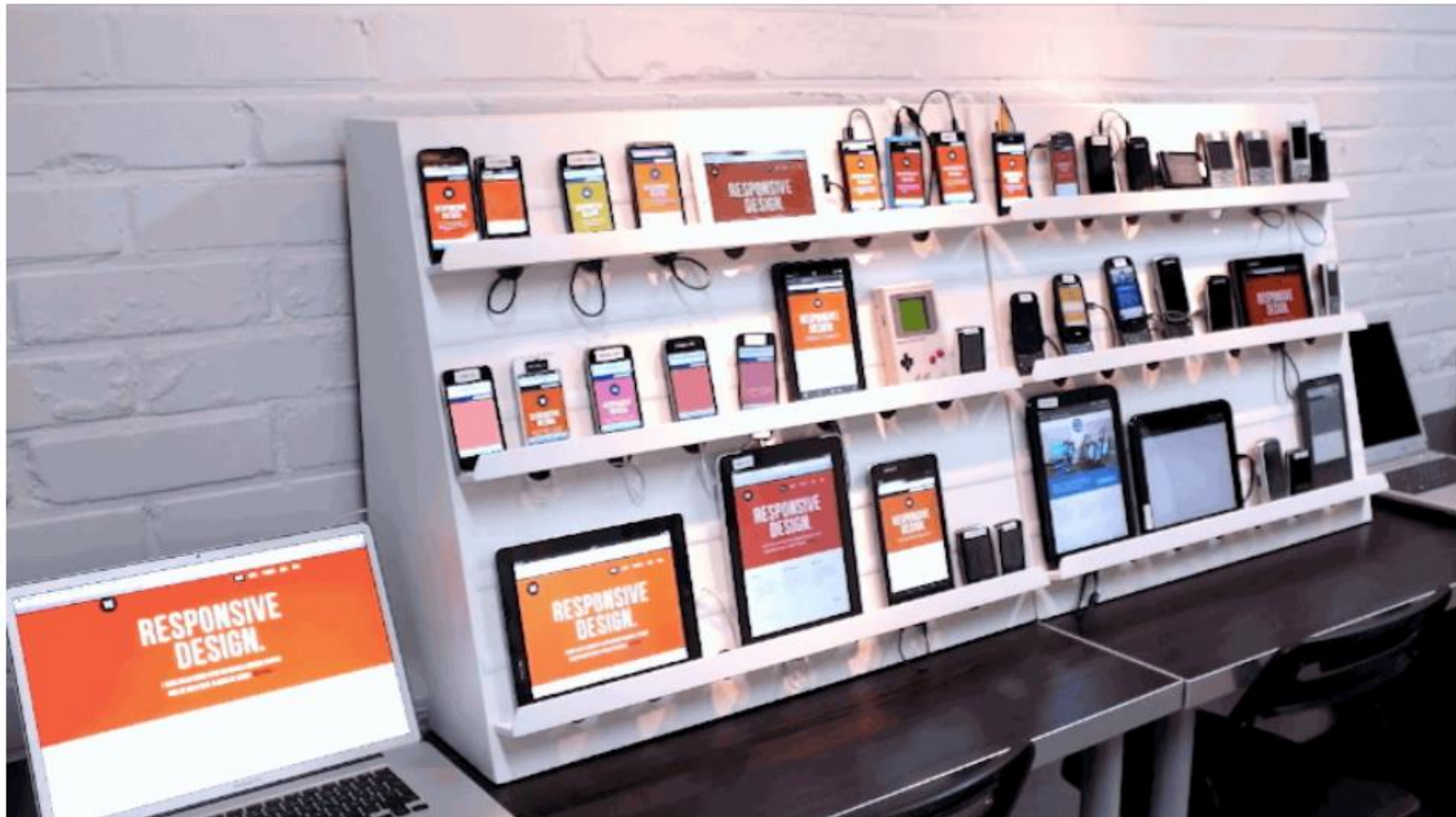
“

# RESPONSIVE WEB DESIGN

”

# Qu'est ce que le responsive

- Adaptation à différents supports :





# Mobile First

Desktop first :



Mobile first :



# Principe

- ▶ Adaptation du layout au dispositif d'affichage en s'appuyant sur :
  - ▶ des grilles fluides
    - ▶ utilisation de tailles relatives : % ou em plutôt que px ou cm.
- ▶ des images de taille flexible
  - ▶ utilisation de tailles relatives : % ou em plutôt que px ou cm.
- ▶ des media queries
  - ▶ Des règles CSS différentes selon le dispositif
  - ▶ souvent la largeur (width) de l'écran ou de la fenêtre.



Questions?