Intro au Responsive

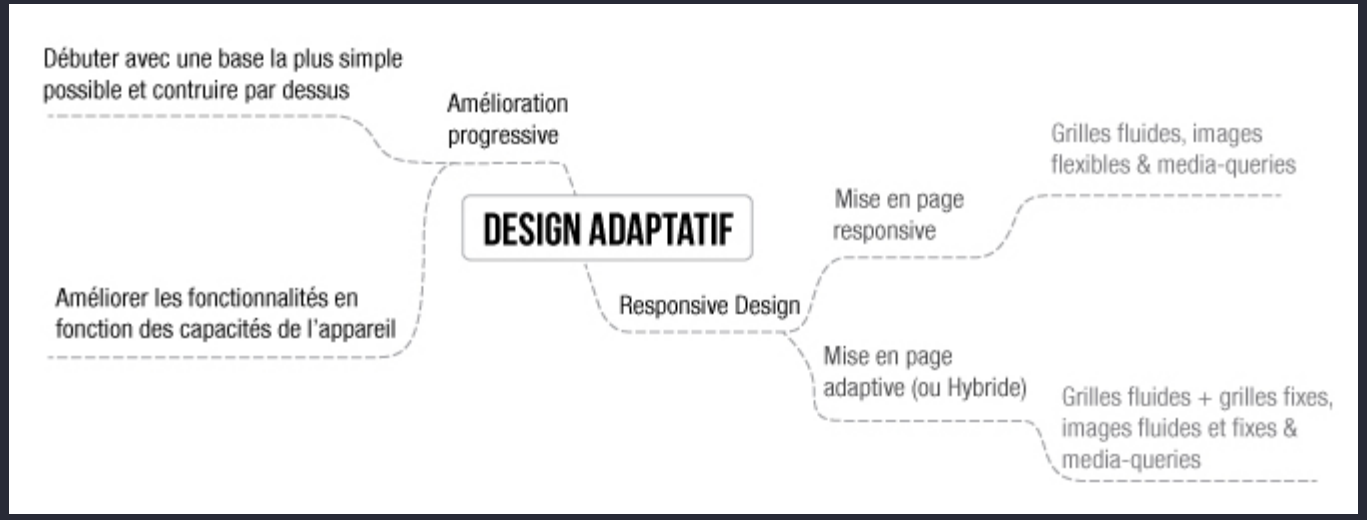
Le **responsive webdesign** est un sous-ensemble du concept d'**adaptative webdesign**

Responsive

Mise en page : images flexibles, grilles fluides, media queries

Adaptative

UX design : apporter la meilleure expérience utilisateur à la plus large audience possible


Débuter avec une base la plus simple possible et contruire par dessus
Amélioration progressive
DESIGN ADAPTATIF
Améliorer les fonctionnalités en fonction des capacités de l'appareil
Responsive Design
Mise en page responsive
Grilles fluides, images flexibles & media-queries
Mise en page adaptive (ou Hybride)
Grilles fluides + grilles fixes, images fluides et fixes & media-queries

Amélioration progressive / Progressive enhancement

- Concept d'accessibilité et de sémantique
- Séparer le fond de la forme
- Proposer au fur et à mesure un enrichissement de la mise en forme, en fonction des capacités des périphériques d'affichage

Compatibilité minimale

Se mettre dans la pire situation

Penser à la majorité des cas

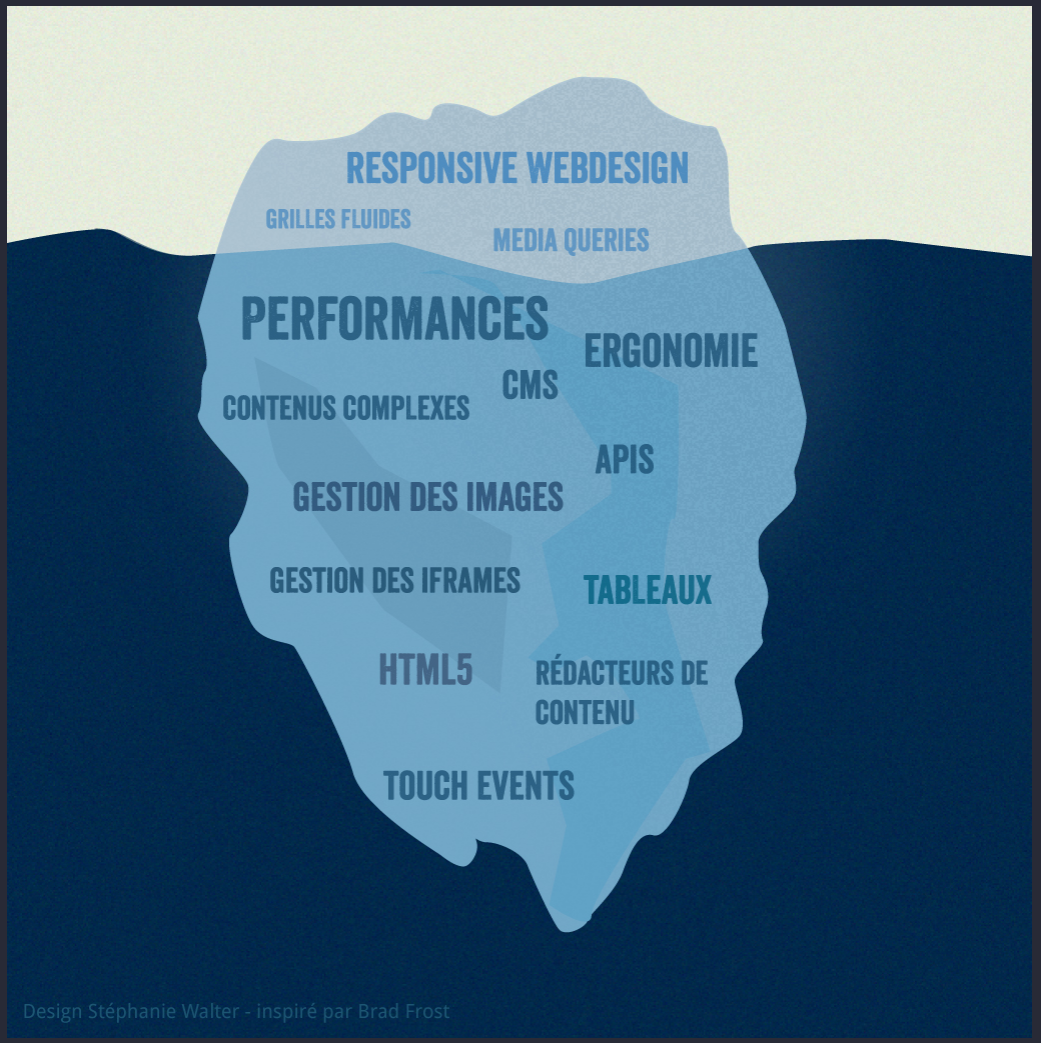
Device utilisé, rendu navigateur, support CSS/JS, accessibilité, rapidité réseau, etc.

Enrichi ensuite

Chargement asynchrone (JS, images, contenus, etc.)

Stratégie offline, cache

Effets visuels avancés / animations


RESPONSIVE WEBDESIGN
GRILLES FLUIDES
MEDIA QUERIES
PERFORMANCES
ERGONOMIE
CMS
CONTENUS COMPLEXES
APIS
GESTION DES IMAGES
GESTION DES IFRAMES
TABLEAUX
HTML5
RÉDACTEURS DE CONTENU
TOUCH EVENTS
Design Stéphanie Walter - inspiré par Brad Frost

Media-Queries

Les médias queries sont des *instructions conditionnelles* qui permettent d'appliquer des styles CSS **en fonction de la largeur de l'écran ou de la résolution du périphérique de sortie**

Syntaxe

Une requête média se compose d'un **type de média** optionnel et d'une ou plusieurs expressions de **caractéristiques de média**

Plusieurs requêtes peuvent être combinées entre elles grâce à des **opérateurs logiques**

Types de medias

Un type de média définit une catégorie générale d'appareil

Il en existe 4 :

- `screen` : appareils dotés d'un écran
- `print` : documents en aperçu avant impression
- `speech` : outils de synthèse vocale
- `all` : tous les appareils

Caractéristiques de média

Elles décrivent certaines caractéristiques spécifiques de l'agent utilisateur, de l'appareil d'affichage ou de l'environnement

En voici quelques-unes couramment utilisées :

En voici quelques-unes couramment utilisées :

- `min-width` / `max-width`: Largeur de la viewport*
- `min-height` / `max-height`: Hauteur de la viewport*
- `aspect-ratio`: Rapport largeur/hauteur de la viewport*
- `orientation`: Orientation la viewport*
- `resolution`: Densité de pixel pour l'appareil d'affichage
- `prefers-color-scheme`: Préférence de theme de l'utilisateur

** zone d'affichage*

Depuis récemment, on peut simplifier `min-width` et `min-height` :

- `width`: Largeur de la viewport*
- `height`: Hauteur de la viewport*
- `aspect-ratio`: Rapport largeur/hauteur de la viewport*
- `orientation`: Orientation la viewport*
- `resolution`: Densité de pixel pour l'appareil d'affichage
- `prefers-color-scheme`: Préférence de theme de l'utilisateur

** zone d'affichage*

Exemple de disposition

Un menu en *flexbox* qui va changer de direction

```
nav {
  display: flex;
  flex-direction: row;
}

@media screen and (max-width: 480px) {
  nav {
    flex-direction: column;
  }
}
```

Exemple de disposition

Un menu en *flexbox* qui va changer de direction

```css
nav {
  display: flex;
  flex-direction: row;
}

@media (width <= 480px) {
  nav {
    flex-direction: column;
  }
}
```

Exemple d'orientation

Une navbar qui sera positionnée en *sticky* en fonction de l'orientation

```
nav {
  display: flex;
  justify-content: : space-evenly;
}

@media (orientation: portrait) {
  nav {
    position: sticky;
    top: 0;
  }
}
```

Exemple de thème

Un site qui sera affiché en *darkmode* suivant la préférence de l'utilisateur sur son système (Windows / macOS / Linux)

```css
body {
  background-color: white;
  color: black;
}

@media (prefers-color-scheme: dark) {
  body {
    background-color: black;
    color: white;
  }
}
```

Il existe de nombreuses combinaisons de règles Media queries, en utilisant les opérateurs logiques

```
@media screen and (max-width: 650px) {}

@media (min-width: 700px) and (orientation: landscape) {}

@media print and (aspect-ratio: 16/9) {}
```

La page de la documentation MDN à ce sujet est très complète :

🔗 Utiliser les Media queries

Connaître les caractéristiques de la plupart des devices mobiles :

🔗 screensiz.es

Un outil intéressant à garder dans ses favoris

Optimiser les images

Des images optimisées sont essentielles pour une bonne expérience utilisateur, pour des raisons de **performance**, mais aussi de **confort visuel**

Pour optimiser le poids, des outils existent comme

🔗 compressor.io

Mais au dela de l'optimisation, quelle utilité par exemple pour une image de fond d'avoir une résolution de 3000x1500 alors que la fenêtre du navigateur Web ne fait que 1200px de large ?

De plus, des pages HTML peuvent demander à charger de nombreuses images, dont la plupart ne seront pas visibles immédiatement par l'utilisateur (qui ne voit que le haut de la page lors du chargement)

Sans compter la complexité de gestion de la qualité en fonction des différents écrans et densités de pixels (Rétina, OLED, …) sur tous les devices (ordinateurs, tablettes, smartphones, …)

Aujourd'hui, HTML propose des solutions natives et faciles à mettre-en-place pour gérer ces problématiques

Lazy loading

```
<img src="image.jpg" loading="lazy">
```

Cet attribut indique au navigateur de ne pas charger l'image tant qu'elle n'est pas visible dans la viewport

`srcset`

Cet attribut va permettre de définir une image adaptée au terminal de consultation en ciblant à la fois la **taille de l'écran et la densité de pixels**

Il utilise des *descripteurs de source* :

- `w` : descripteur de largeur
- `x` : descripteur de densité de pixels

Admettons que nous disposons d'une même image en 2 tailles différentes : 300px et 600px

```
<img srcset="image-300.jpg 300w, image-600.jpg 600w">
```

```
<img srcset="image-300.jpg 300w, image-600.jpg 600w">
```

Ici, on indique explicitement au navigateur que l'on dispose de deux images (300px et 600px grâce au descripteur `w`) et que l'on souhaite qu'il choisisse la plus adaptée en fonction de la taille de la viewport (qu'il connaît)

C'est ensuite au navigateur de faire le choix de la meilleure image à charger

`sizes`

Cet attribut **vient en complément de `srcset`**, est n'est valable que si on a choisi le descripteur `w`

Il permet d'indiquer au navigateur différentes tailles de viewport pour lesquelles on dispose d'une image adaptée

```
<img srcset="/image-300.jpg 300w, /image-600.jpg 600w"
     sizes="(max-width: 400px) 200px, (max-width: 800px) 100vw, 50vw">
```

Dans cet exemple, le navigateur est informé que :

Pour une taille de viewport inférieure ou égale à 400px, l'image devra s'afficher en 200px de large. Il prendra donc la valeur la plus proche, à savoir : image-300.jpg. Cependant, si la densité de pixels vaut 3 (= écrans super Rétina), alors il multipliera les 200px attendus par 3, et prendra donc l'image de 600px de large pour une meilleure qualité d'affichage.

```
<img srcset="/image-300.jpg 300w, /image-600.jpg 600w"
     sizes="(max-width: 400px) 200px, (max-width: 800px) 100vw, 50vw">
```

Dans cet exemple, le navigateur est informé que :

> *Pour une taille de viewport inférieure ou égale à 800px, l'image devra s'afficher à 100% de la largeur de la viewport (c'est-à-dire 800px). Il choisira donc systématiquement image-600.jpg.*

```
<img srcset="/image-300.jpg 300w, /image-600.jpg 600w"
     sizes="(max-width: 400px) 200px, (max-width: 800px) 100vw, 50vw">
```

Dans cet exemple, le navigateur est informé que :

> *Si aucune des conditions précédentes n'est remplie, l'image devra s'afficher à 50% de la largeur de la viewport. C'est-à-dire que si on dispose d'une viewport faisant 820px, l'image s'affichera à 410px de large (= 50% de la viewport). Le navigateur choisira donc l'image de 300px de large car c'est la valeur la plus proche de 410px (pour un écran de 1x).*

Il est important de comprendre que toutes ces possibilités ont pour but de donner un maximum d'information au navigateur pour qu'il puisse charger les images de la meilleure façon possible

Mais le navigateur est libre de choisir la méthode qu'il souhaite et vous ne serez pas forcément sûr de voir les images charger de la manière que vous avez définie

L'article suivant reprend en détails les notions vues dans ce chapitre :

🔗 Optimal Images in HTML.