



Le Responsive Design

Objectifs

- ✓ Optimiser et rendre compatible son site pour tous les supports (*mobile, tablette, ordinateur de bureau et portable*).

01

Qu'est-ce que le Responsive Design ?



Le Responsive Web Design (RWD) est une conception permettant à un site web d'adapter son affichage sur chaque type d'écran.

De nos jours, la consultation d'un site web peut se faire avec un ordinateur de bureau, un ordinateur portable, un smartphone, une tablette, une tv, etc.

Les tailles de tous ces écrans sont multiples et variées, c'est la raison pour laquelle il est important de s'assurer que l'affichage de son site web est bien correct et ajusté à chacun de ces écrans.

Pour obtenir une expérience utilisateur (UX) renforcée, il est nécessaire d'apporter du confort visuel à l'internaute sans scroll (barre de défilement horizontale) et sans que la manipulation d'un zoom soit nécessaire.

Lorsque nous avons un espace d'affichage réduit, la plupart du temps, l'intégration est épurée au maximum.

Le Responsive Web Design (RWD) est de plus en plus incontournable et demandé par les clients lors de la création d'un site web.

02

Comment faire pour adapter son site à chaque écran ?

Nous n'allons pas créer une version de site pour chaque écran (sinon ce serait assez long et cela change tout le temps à chaque nouvelle sortie de la part d'un constructeur de téléphone ou d'ordinateur), nous allons donc privilégier la création d'une interface de site web qui pourrait s'auto-adapter à chaque écran.

Depuis la mise à jour du langage CSS (CSS3), la technologie media queries voit son apparition pour traiter ce genre de cas.

Grâce aux médias queries, nous allons pouvoir définir des règles d'affichage en fonction des résolutions et de la taille des écrans. C'est une très bonne avancée puisque c'est ce que nous voulons faire !

Cas d'étude : vous avez 2 zones à afficher sur une page web.

- Sur écran d'ordinateur, vous disposer d'une surface d'écran très large, vous pourriez les afficher côte à côte.
- Sur smartphone, la surface est très réduite, vous pourrez donc faire le choix de les empiler les unes en dessous des autres pour profiter de la largeur maximum de l'écran.

Très grand écran (ordinateur de bureau)	Très petit écran (mobile - smartphone)
---	--



Pour acquérir et renforcer nos connaissances, il nous faudrait savoir à peu près quelle est la taille des différents écrans (majeur) ? Voici des éléments de réponses au prochain chapitre.

03 Taille en résolution des (principaux) écrans

Voici un tableau récapitulant la taille des principaux écrans :

Écran	Écran minimum	Écran réduit	Écran moyen	Grand Écran
Illustration				
Type	SmartPhone	Tablette	Ordinateur Portable	Ordinateur de bureau
Résolution d'écran taille en largeur (width)	< 768 px	>= 768 px < 992 px	>= 992 px < 1200 px	>= 1200 px

Même si nous n'avons pas besoin de bootstrap pour faire du responsive design, ces chiffres sont repris de ce framework css (qui est responsive nativement) car ils correspondent assez bien à la réalité.

Même si d'un appareil à l'autre il y a des différences, nous pouvons donc retenir qu'il y a 4 tailles d'écrans majeurs.

04 Type Fixe, Fluide ou Adaptatif

1. Un site web fixe est créé dans des mesures fixes souvent exprimées en pixels
La taille ne change pas selon les situations.
2. Un site web fluide est créé dans des mesures variables souvent exprimées en pourcentage.
Unité de mesure privilégié : pourcentage, em, vh, vw, etc.
3. Un site web adaptatif est créé dans des mesures fixes mais diffère selon la taille d'écran.
Exploité grâce aux médias queries !

Dans la plupart des cas, nous pouvons éliminer la 1ère solution consistant à créer un site web avec des mesures fixes.

En effet, l'utilisation des tablettes ou smartphone pour consulter des pages web n'est pas une "mode", cela continue d'augmenter chaque année selon les statistiques.

Nous avons donc 2 types de site responsive possible lorsqu'on démarre un projet : le fluide et/ou l'adaptatif !

Nous verrons en détail les avantages, inconvénients et différences de ces deux techniques lors de ce tutorial.



➤ **Le saviez-vous ?**

La hauteur d'un site dépend de son contenu.

La largeur est souvent de 960px car ce nombre est divisible par 1 2 3 4 5 6 8 10 12 15 16 20 24 30 32 40 48 60 64 80 96 120 160 192 240 320 480 960. Cela offre beaucoup de possibilité de mise en page !

05

FrameWork Css pour un site responsive

Plusieurs Framework CSS peuvent aider à la création d'un site web complètement responsive design, plusieurs d'entre eux sont régulièrement utilisés.

[Bootstrap CSS](#) est l'un des framework faisant l'objet d'un tutorial sur notre site web.

06

Différence entre 1 site web responsive design, 1 version mobile et 1 application mobile

En utilisant la technologie Responsive Design, le contenu du site web s'adapte automatiquement à la largeur et à la hauteur de l'écran qui le consulte.

En optant pour le développement d'une version mobile de votre site web, vous prévoyez un affichage classique pour les ordinateurs de bureau et un affichage différent et optimisé pour les mobiles (cela peut aussi permettre de personnaliser l'affichage des informations d'un support à l'autre).

Une application mobile s'apparente davantage au monde du logiciel (sauf pour les applications hybrides) disponible sur une plateforme de téléchargement, l'internaute choisit ensuite d'installer l'application et éventuellement de la garder sur son écran d'accueil pour faciliter ses prochaines consultations.

Avantages et Inconvénients des sites responsive design et version mobile

Avantages du Responsive Design

- une seule adresse URL pour le site web
- Meilleure indexation et référencement naturel (Google accorde de l'importance aux sites web prévoyant une adaptation pour les mobiles, depuis 2015)
- Moins de maintenance (une seule version globale adaptable à modifier pour la faire évoluer)
- Les zones, colonnes, images, textes, etc. se déplacent automatiquement

Avantages du site version mobile

- Forte personnalisation (il est possible d'avoir 2 versions différentes et complémentaires entre la version mobile et le site web)
- La version mobile est particulièrement adaptée aux petits écrans
- La version mobile peut être plus rapide à charger

Responsive Design : Mobile First !

Certains développeurs front considèrent que la construction d'une interface doit d'abord se faire pour les mobiles en tout premier lieu.

Devant la progression des surfs sur le web avec un mobile, cela permet de partir d'une version light qui ensuite sera déclinée avec parfois plus d'éléments pour les versions supérieures (tablette, ordinateur portable, ordinateur de bureau).

Exemples

07

Le ViewPort

L'attribut « viewport » permet de préciser au navigateur quelle taille il doit prendre pour afficher une page web.

`width=device-width` permet de régler la largeur de la page web sur la largeur de la fenêtre de l'appareil qui consulte actuellement la page

`initial-scale=1` permet de régler le niveau de zoom sur 100 % (par défaut).

Pour démarrer une structure de site web responsive, nous utiliserons les balises et attributs suivants :

```
responsive1.html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Mon Site Responsive</title >
    <link rel="stylesheet" href="responsive1.css">
    <meta name="viewport" content="width=device-width, initial-scale=1">
  </head>
  <body>
    ...
  </body>
</html>
```

08

Exemple sans responsive design (taille fixe)

Il n'y a pas si longtemps que cela (quelques années) les sites web étaient construits avec des mesures fixes et ne s'adaptait pas aux différents écrans car il n'y en avait souvent qu'un seul (l'écran d'ordinateur de bureau).

Voici un exemple Html simple SANS responsive design FIXE :

```
fixe1.html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" >
    <title>Mon Site SANS Responsive</title>
    <link rel="stylesheet" href="fixe1.css">
  </head>
  <body>
    <div class="zone-fixe"> Zone Fixe</div>
  </body>
</html>
```

La partie Css :

```
fixe1.css
.zone-fixe{
  width: 1000px;
```

```
background: blue;
color: white;
padding: 10px;
}
```

Explication du code :

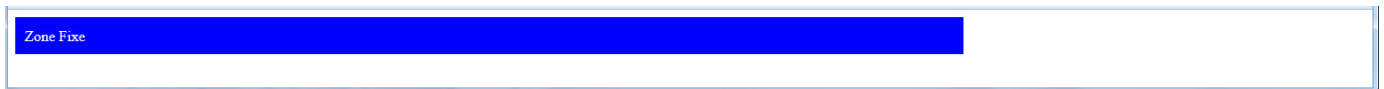
La propriété `width` permet d'indiquer la largeur, dans notre cas elle fera 1000 pixels, ce qui est une taille fixe.

Essayez donc de redimensionner votre fenêtre, vous verrez qu'elle fera toujours toujours 1000 pixels et sur les petits écrans (petites fenêtres en largeur) cela risque de créer une scrollbar (barre de défilement) horizontale, c'est ce que nous voulons absolument éviter ! c'est ce qu'on appelle un site web n'étant pas responsive et par conséquent fixe. A éviter !

La propriété `background` n'est présente que pour afficher et voir notre zone sur l'écran.

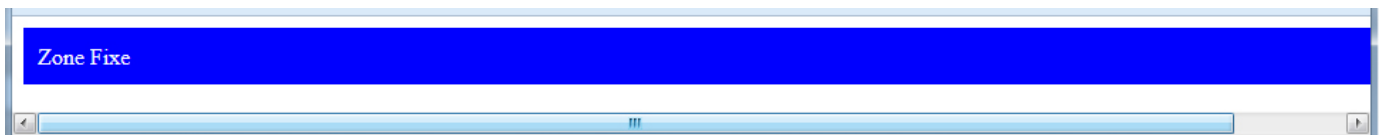
Résultat

Taille grand écran (ordinateur de bureau) :



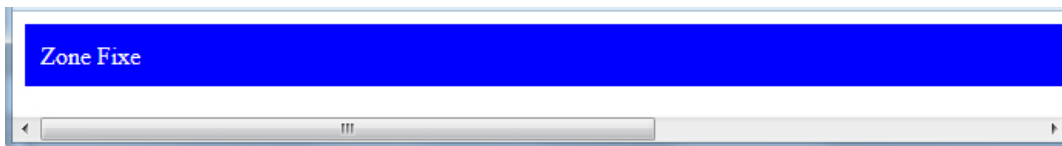
Rien à signaler de particulier

Taille écran moyen (ordinateur portable) :



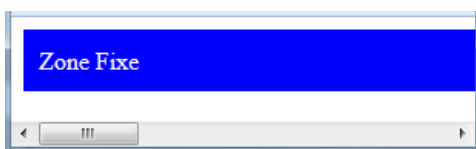
Nous voyons l'apparition d'une scrollbar horizontale et cela est très déconseillé !

Taille écran réduit (tablette) :



La scrollbar horizontale s'agrandit car la taille est fixe !

Taille petit écran (mobile) :



Sur un mobile, la scrollbar est très importante et donc cela risque d'être difficile pour le mobinaute de lire le texte qui pourrait être écrit complètement à droite (et même si son téléphone mobile fait apparaître la page sur un seul affichage (de manière condensée) cela risque d'être très petit et le mobinaute sera obligé de zommer !



➤ Informations

Les largeurs complément fixes sont à proscrire car elles ne sont plus adaptées aux modes de consommation et aux habitudes de surf des internautes !

Avantages	Inconvénients
Moins prise de tête (normal : aucun responsive)	Plus du tout adapté à notre époque !

Voici un exemple Html simple de responsive design fluide :

```
fluide1.html
<!Doctype html>
<html>
  <head>
    <meta charset="utf-8" >
    <title>Mon Site Responsive Fluid</title>
    <link rel="stylesheet" href="fluide1.css" />
    <meta name="viewport" content="width=device-width, initial-scale=1">
  </head>
  <body>
    <div class="zone-fluid">
      Zone Fluid
    </div>
  </body>
</html>
```

La partie Css :

```
fluide1.css
.zone-fluid{
background: red;
width: 70%;
}
```

Explication du code :

La propriété `width` permet d'indiquer la largeur, dans notre cas elle fera 70 %.

70 % de quoi ? 70 % de la taille de la fenêtre, essayez donc de redimensionner votre fenêtre, vous verrez qu'elle sera toujours à bonne largeur, l'utilisation des pourcentages c'est ce qu'on appelle du responsive fluide (une fois que cela est généralisé à tout le site web).

La propriété `background` n'est présente que pour afficher et voir notre zone sur l'écran.

Résultat

Taille grand écran (ordinateur de bureau) :



La largeur fait bien 70 % de 1200 pixels (soit environ 840 pixels).

Taille écran moyen (ordinateur portable) :



La largeur fait bien 70 % de 992 pixels (soit environ 695 pixels). ET SURTOUT : nous n'avons pas de scrollbar.

Taille écran réduit (tablette) :



La largeur fait bien 70 % de 768 pixels (soit environ 538 pixels). Sans Scrollbar.

Taille petit écran (mobile) :

Zone Fluid

La largeur fait bien 70 % de 380 pixels (soit environ 266 pixels). Et même dans le cas du plus petit écran nous n'avons pas de scrollbar car nous faisons toujours 70 % de quelque chose, nous serons donc toujours moins large de 30 % !



➤ Informations

Les pourcentages constituent une bonne option à privilégier mais nous ne construisons pas la totalité d'un site web avec des pourcentages, nous en reparlerons plus bas dans ce tuto.

Avantages	Inconvénients
Largeur toujours adaptée	Pas de contrôle sur la manière dont s'affichent les éléments dans une situation précise. C'est toujours la même.

10

Exemple simple responsive design Adaptatif

Pour cet exemple, nous retrouvons l'utilisation des pixels. je sais qu'on a dit que l'utilisation des pixels seulement n'était pas une bonne solution, mais là, nous les utiliserons avec les medias queries

Voici un exemple Html simple de responsive design adaptatif :

```
adaptatif1.html

<!doctype html>
<html>
  <head>
    <meta charset="utf-8" >
    <title>Mon Site Responsive Adaptatif</title>
    <link rel="stylesheet" href="adaptatif1.css" />
    <meta name="viewport" content="width=device-width, initial-scale=1">
  </head>
  <body>
    <div class="zone-adaptative">
      Zone adaptative
    </div>
  </body>
</html>
```

La partie Css :

```
adaptatif1.css

.zone-adaptative{
background: orange;
}
@media (min-width: 768px) {
  .zone-adaptative {
    width: 750px;
  }
}
@media (min-width: 992px) {
  .zone-adaptative {
    width: 970px;
  }
}
@media (min-width: 1200px) {
  .zone-adaptative {
    width: 1170px;
  }
}
```

Explication du code :

La propriété `width` permet d'indiquer la largeur, dans notre cas `.zone-adaptative` n'a aucune largeur particulière (et donc par défaut : la totalité en largeur de la fenêtre).

La règle `@media` permet d'indiquer et d'adapter la largeur de notre `.zone-adaptative` selon la taille et résolution d'écran.

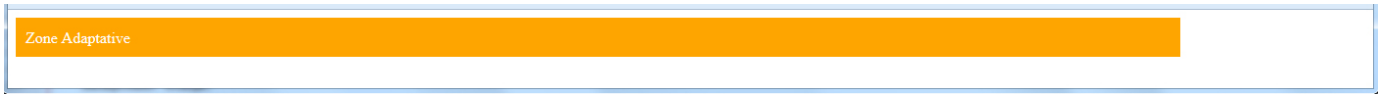
C'est ce qu'on appelle du RESPONSIVE ADAPTATIF ! il s'agit d'un comportement différent exprimé en fonction de diverses situations.

La suite des explications se trouve dans les résultats :

La propriété `background` n'est présente que pour afficher et voir notre zone sur l'écran.

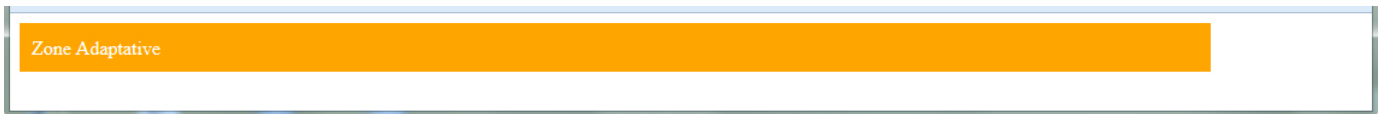
Résultat

Taille grand écran (ordinateur de bureau) :



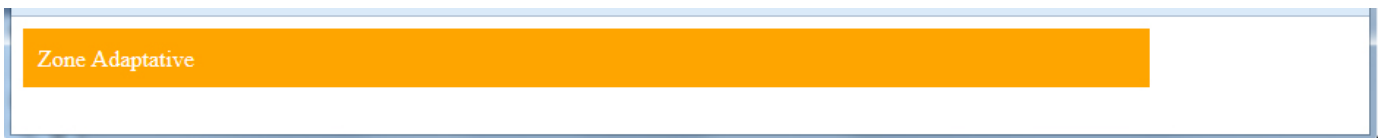
La largeur de notre zone fait bien 1170 pixels. La largeur de l'écran est supérieure ou au minimum sur 1200 pixels (`@media (min-width: 1200px)`), nous avons donc la place nécessaire pour 1170px.

Taille écran moyen (ordinateur portable) :



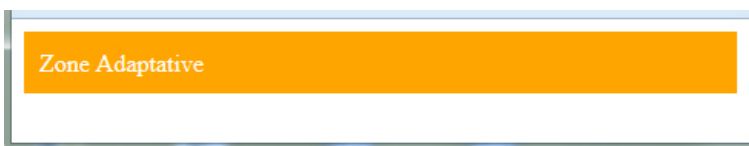
La largeur de notre zone fait bien 970 pixels. La largeur de l'écran est supérieure ou au minimum sur 992 pixels (`@media (min-width: 992px)`), nous avons donc la place nécessaire pour 970px.

Taille écran réduit (tablette) :



La largeur de notre zone fait bien 750 pixels. La largeur de l'écran est supérieure ou au minimum sur 768 pixels (`@media (min-width: 768px)`), nous avons donc la place nécessaire pour 750px.

Taille petit écran (mobile) :



La largeur n'est pas précisée. Par conséquent lorsque la largeur n'est pas précisée, elle fait toute la largeur de l'écran ! et pas 1 pixel de plus, du coup on prend la totalité de la place dont on dispose et on n'est sûr de ne pas déborder et de ne pas avoir de scrollbar horizontale (barre de défilement).

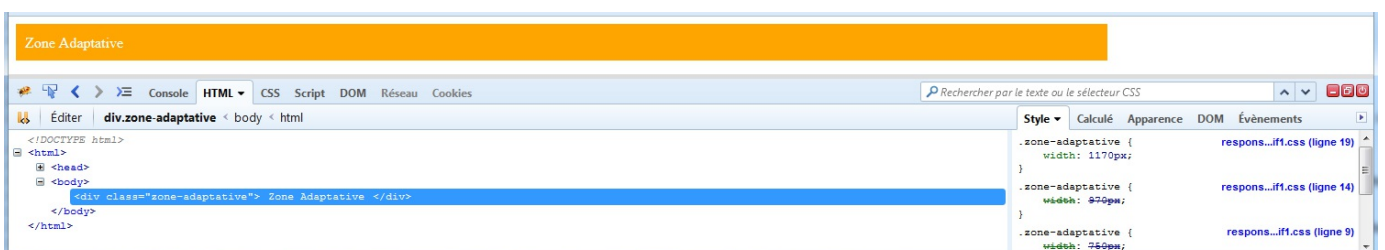


➤ Bon à savoir

Noter que nous ne réglons pas forcément la hauteur (`height`) car cela dépend du contenu à l'intérieur de chaque zone.

Observations

FireBug



Comme vous pouvez le constater plusieurs largeurs (width) sont transmises au navigateur, et selon la largeur de la fenêtre il barre (annule) celles qui ne correspondent pas à nos médias queries.



➤ En Conclusion

Cet exemple est une bonne option à privilégier, c'est un avantage d'avoir un contrôle précis (au pixel près) sur le code, mais cela peut parfois être long et contraignant de réécrire le comportement d'une multitude de zones, nous l'avons fait pour une zone mais imaginez le code que nous devrions écrire et les tests que nous devrions réaliser si nous avons 50 zones par page et un nombre conséquent de pages ? Cela peut forcer à réécrire toute la partie css du site 4 fois. Nous allons donc retenir cette option pour la suite en la combinant à l'utilisation des pourcentages (adaptatif + fluide !)

Avantages	Inconvénients
Contrôle précis dans chaque situation	Beaucoup de code à réécrire

11

Exemple responsive design Fluide et Adaptatif

Alors, qu'est-ce qui serait le mieux entre le fluide et l'adaptatif d'après vous ? les deux comportent des avantages et des inconvénients, mais pourquoi choisir ?

Pour monter en puissance et en efficacité, l'idéal serait plutôt de combiner les deux !!

Un site web responsive design peut être à la fois fluide et adaptatif !

Fluide = utilisation des pourcentages.

Adaptatif = Différence par type d'écran (règles medias queries)

Voici un exemple Html simple de responsive design fluide et adaptatif :

```
responsive-design.html

<!doctype html>
<html>
  <head>
    <meta charset="utf-8" >
    <title>Mon Site Responsive Fluide et Adaptatif</title>
    <link rel="stylesheet" href="responsive-design.css" />
    <meta name="viewport" content="width=device-width, initial-scale=1">
  </head>
  <body>
    <div class="zone-1">
      <div class="zone-2"> Zone 2 </div>
      <div class="zone-3"> Zone 3 </div>
      <div class="clear"></div>
    </div>
  </body>
</html>
```

La partie Css :

```
responsive-design.css

.zone-1{
background: brown;
color: white;
border: 3px solid #000;
}
.zone-2{
width: 100%;
background: pink;
color: white;
}
.zone-3{
width: 100%;
background: gray;
color: white;
}
.clear{ clear: both; }
```

```

@media (min-width: 768px) {
  .zone-1 { width: 750px; }
}
@media (min-width: 992px) {
  .zone-1 { width: 970px; }
  .zone-2, .zone-3 { width: 50%; float: left; }
}
@media (min-width: 1200px) {
  .zone-1 { width: 1170px; }
  .zone-2, .zone-3 { width: 50%; float: left; }
}

```

Explication du code :

Nous avons une zone `.zone-1` qui fait office de conteneur et qui contient donc les deux autres `.zone-2` et `.zone-3`

La propriété `width` de la zone `.zone-1` n'a aucune largeur particulière (et donc par défaut : la totalité en largeur de la fenêtre).

La propriété `width` de la zone `.zone-2` est fixée à 100% de son parent (le parent est `zone-1` qui fait la totalité de la fenêtre). idem pour la zone `.zone-3`

Il est donc normal que `zone-2` et `zone-3` s'affichent l'une en dessous de l'autre.

SAUF QUE, nous avons prévu des cas différents (`@media`) en fonction des situations :

- Taille grand écran (ordinateur de bureau - `@media (min-width: 1200px)`)

La largeur de notre `zone-1` est fixée à 1170 pixels.

`.zone-2` et `.zone-3` sont redimensionnées à 50% de leur parent (le parent est `zone-1` à 1170px) et sont mis en position `float: left;`. Par conséquent `zone-2` et `zone-3` se retrouvent côte à côte.

- Taille écran moyen (ordinateur portable - `@media (min-width: 992px)`)

La largeur de notre `zone-1` est fixée à 970 pixels.

`.zone-2` et `.zone-3` sont redimensionnées à 50% de leur parent (le parent est `zone-1` à 970px) et sont mis en position `float: left;`. Par conséquent `zone-2` et `zone-3` se retrouvent côte à côte.

- Taille écran réduit (tablette - `@media (min-width: 768px)`)

La largeur de notre `zone-1` est fixée à 750 pixels.

`.zone-2` et `.zone-3` ne sont pas en position `float: left;` et prennent 100% de largeur, du coup sur écran réduit elles passeront l'une en dessous de l'autre pour profiter de la pleine largeur dont ils disposent.

- Taille petit écran (mobile / smartphone)

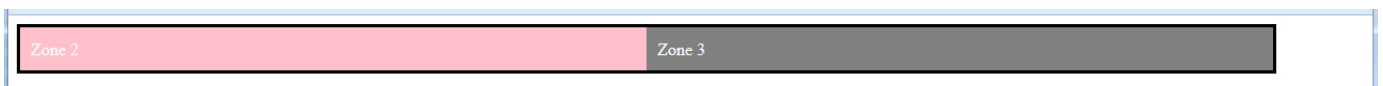
Il n'y a pas de règle pour le mobile, c'est donc le style par défaut (en dehors des `@media`) qui s'applique.

`.zone-2` et `.zone-3` ne sont pas en position `float: left;` et prennent 100% de largeur, du coup sur petit écran elles passeront l'une en dessous de l'autre pour profiter de la pleine largeur dont ils disposent.

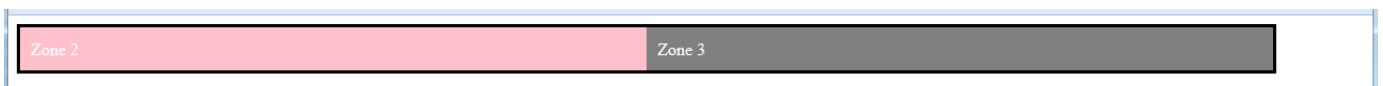
Cet exemple démontre à la fois une utilisation des pourcentages (sur le principe du responsive design fluid) et aussi l'utilisation des media queries (`@media` sur le principe du responsive design adaptatif).

Résultat

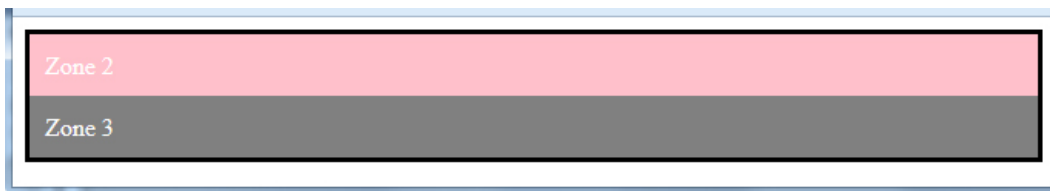
Taille grand écran (ordinateur de bureau) :



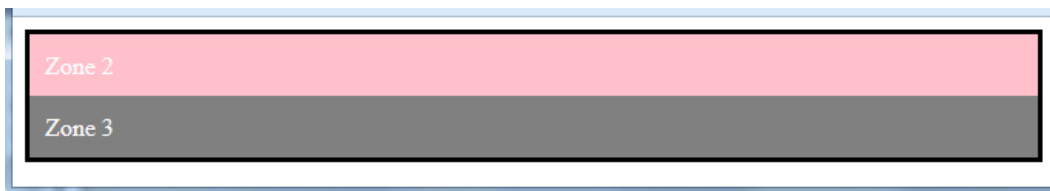
Taille écran moyen (ordinateur portable) :



Taille écran réduit (tablette) :



Taille petit écran (mobile) :



Sur mobile, nous n'avons pas vraiment la place de mettre les zones côte à côte et préférons les empiler l'une en dessous de l'autre.



➤ En Conclusion

C'est la solution à suivre ! Pour construire votre site web avec un responsive design précis vous aurez besoin de l'adaptatif (avec les règles @media) mais pour ne pas réécrire votre site web 4 fois vous aurez besoin du fluide avec les pourcentages ! vive le responsive adaptatif et le responsive fluide !

Le bénéfice peut paraître faible sur cet exemple mais si vous travaillez de cette façon en pourcentage tout en prévoyant les cas spécifique @media, à coup sûr cela vous soulagera dans la construction d'un site web tout entier.

12

La règle @Media

Comme vous le savez, en fonction de la largeur de la fenêtre, la règle « @media » permet de préciser au navigateur quelle propriété CSS il doit adopter et prendre en compte pour l'affichage sur une page web.

Pour cela, le code CSS suivant est à retenir :

```
@media (min-width: 768px) { ... } /* règle pour les affichages inférieur a 768px de large */  
@media (min-width: 992px) { ... } /* règle pour les affichages inférieur a 992px de large */  
@media (min-width: 1200px) { ... } /* règle pour les affichages inférieur a 1200px de large */
```

Nous voyons ci-dessus l'utilisation du MIN-WIDTH (pour largeur minimum)

Nous aurions pu également le voir dans l'autre sens MAX-WIDTH (pour largeur maximum)

```
@media (max-width: 768px) { ... } /* règle pour les affichages supérieur a 768px de large */  
@media (max-width: 992px) { ... } /* règle pour les affichages supérieur a 992px de large */  
@media (max-width: 1200px) { ... } /* règle pour les affichages supérieur a 1200px de large */
```

Ou éventuellement mettre des conditions associées, exemple :

```
@media (min-width: 768px) and (max-width: 992px) { ... } /* règle pour les affichages supérieure a 768px et inférieur a 992px de large */
```

Il est également possible de préciser l'orientation de l'appareil :

```
@media (orientation: portrait) { ... } /* règle pour les affichages en mode portrait */
```

N'oubliez pas : il est possible de travailler avec les unités de mesures en POURCENTAGE et en EM pour éviter de devoir réécrire tous les comportements de son site dans des media queries.

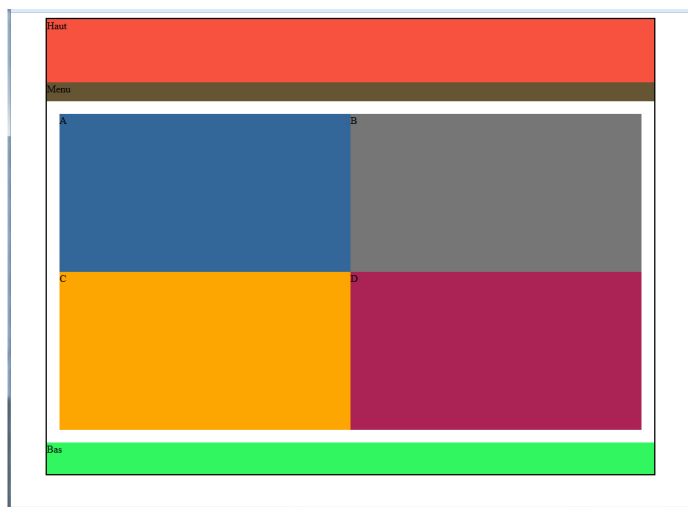
En effet, l'idéal est d'utiliser les POURCENTAGES et EM pour que la taille de son site soit adaptable et de conserver l'utilisation des medias queries pour les comportements spécifiques liés à des contraintes matérielles.

Comment savoir si un site web est responsive design ?

Pour cela il suffit de redimensionner la fenêtre du navigateur et de la réduire afin de voir si le contenu s'adapte.

Si le site n'est pas responsive design, vous verrez une barre de défilement horizontale s'afficher.

Exemple en plein écran :



Exemple en écran réduit :

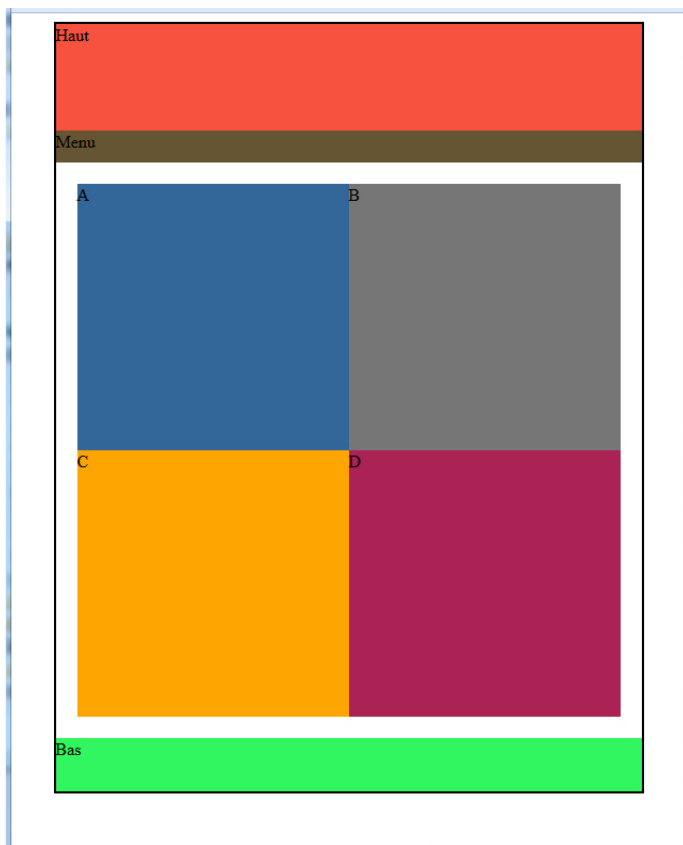


➤ **Attention**

Cela est à bannir !

Les zones restent à la même échelle en terme de taille (largeur) et une scrollbar (barre de défilement horizontale) fait son apparition !

Voici le comportement engendré lorsqu'une page web est responsive design :



Les zones se réduisent en largeur, tout tient sur la largeur de la page, l'objectif est donc atteint !

Comment savoir si un site web possède une version mobile ?

Dans la plupart des cas, le site web vous propose la consultation de leur version mobile (parfois il vous redirige dessus sans vous demander de confirmation).

Le site web détecte que vous utilisez un smartphone (via l'utilisation du langage JavaScript).

Créer sa page web responsive design

13

Une structure Fixe (non responsive)

Nous allons étudier la structure d'un site web fixe.

Exemple Html Fixe :

site-fixe.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mon Site Fixe SANS Responsive</title>
    <link rel="stylesheet" href="site-fixe.css" />
  </head>
  <body>
    <div id="conteneur">
      <header>
        <p>Haut</p>
      </header>
      <nav>
        <p>Menu</p>
      </nav>
      <section>
        <div class="a"><p>A</p></div>
        <div class="b"><p>B</p></div>
      </section>
    </div>
  </body>
</html>
```

```

<div class="clear"></div>
<div class="c"><p>C</p></div>
<div class="d"><p>D</p></div>
<div class="clear"></div>
</section>
<footer>
<p>Bas</p>
</footer>
</div>
</body>
</html>

```

Nous créons une div (div pour division) permettant d'englober les autres éléments du site dans une même zone.

Nous faisons appel à la balise header et footer respectivement pour le haut et bas de site.

La balise nav permet de créer une zone de navigation.

La balise section permettra dans notre cas de prévoir une partie centrale pour le contenu

Pour cet exemple, nous avons déclaré d'autres zones dans la partie section : a, b, c et d.

Exemple Css Fixe :

```

site-fixe.css

#conteneur{
border: 2px solid;
margin: 0 auto;
width: 960px;
}
header{
background: #f6523f;
height: 100px;
}
p{ margin: 0; }
nav{
background: #665533;
height: 30px;
}
section{
background: #f23f98;
margin: 20px;
}
footer{
background: #32f65f;
height: 50px;
}
.a{
float: left;
width: 460px;
height: 250px;
background: #336699;
}
.b{
float: left;
width: 460px;
height: 250px;
background: #767676;
}
.c{
float: left;
width: 460px;
height: 250px;
background: #fda500;
}
.d{
float: left;
width: 460px;
height: 250px;
background: #aa2355;
}
.clear{ clear: both; }

```

Certaines parties de CSS auraient pu être regroupées mais comme il s'agit d'un entraînement, nous avons choisi de décomposer le code

Décryptons le code CSS Fixe :

Cadre explication (background de fond) :

#conteneur permet de sélectionner la zone (div) ayant pour id conteneur (pour contenir toute la page web).

border: 2px solid; permet de dessiner une bordure de 2 pixels tout autour.

`margin: 0 auto;` permet de centrer la page web horizontalement.

`width: 960px;` permet de donner une largeur fixe.

`header` permet de sélectionner la zone du haut (balise `header`).

`background: #f6523f;` permet d'appliquer une couleur de fond.

`height: 100px;` permet de donner une hauteur fixe.

`p` permet de sélectionner tous les paragraphes.

`margin: 0;` permet de "casser" l'héritage de marges.

`nav` permet de sélectionner la zone de navigation (menu).

`background: #665533;` permet d'appliquer une couleur de fond.

`height: 30px;` permet de donner une hauteur fixe.

`section` permet de sélectionner la zone ayant pour balise `<section>`.

`background: #f23f98;` permet d'appliquer une couleur de fond.

`margin: 20px;` permet de donner une marge tout autour de la zone.

`footer` permet de sélectionner la zone du bas ayant pour balise `<footer>`.

`background: #32f65f;` permet d'appliquer une couleur de fond.

`height: 50px;` permet de donner une hauteur fixe.

`.a, .b, .c, .d` permet de sélectionner la zone ayant pour classe css "a".

`float: left;` donne un effet flottant

`width: 460px;` donne une largeur fixe

`height: 250px;` donne une hauteur fixe

`background: #336699;` donne une couleur de fond

`.clear` permet de sélectionner la zone ayant pour classe css "clear".

`clear: both;` permet de "stopper" l'effet flottant (initié par le `float: left;`).



Le Résultat (Fixe)



14 Une structure Adaptative

Vous l'aurez compris, pour obtenir un site responsive Design, il faut s'en préoccuper dès l'origine de la création du projet.

Exemple Html Adaptatif :

```

site-adaptatif.html
<!Doctype html>
<html>
  <head>
    <title>Mon Site Responsive</title>
    <link rel="stylesheet" href="site-adaptatif.css" />
  </head>
  <body>
    <div id="conteneur">
      <header>
        <p>Haut</p>
      </header>
      <nav>
        <p>Menu</p>
      </nav>
      <section>
        <div class="a"><p>A</p></div>
        <div class="b"><p>B</p></div>
      </section>
    </div>
  </body>
</html>

```



```

<div class="clear"></div>
<div class="c"><p>C</p></div>
<div class="d"><p>D</p></div>
<div class="clear"></div>
</section>
<footer>
<p>Bas</p>
</footer>
</div>
</body>
</html>

```

Nous créons une div (div pour division) permettant d'englober les autres éléments du site dans une même zone.

Nous faisons appel à la balise header et footer respectivement pour le haut et bas de site.

La balise nav permet de créer une zone de navigation.

La balise section permettra dans notre cas de prévoir une partie centrale pour le contenu

Pour cet exemple, nous avons déclaré d'autres zones dans la partie section : a, b, c et d.

Exemple Css adaptatif :

```

site-adaptatif.css

#conteneur{
border: 2px solid;
margin: 0 auto;
width: 960px;
}
header{
background: #f6523f;
height: 100px;
}
p{ margin: 0; }
nav{
background: #665533;
height: 30px;
}
section{
background: #f23f98;
margin: 20px;
}
footer{
background: #32f65f;
height: 50px;
}
.a{
float: left;
width: 460px;
height: 250px;
background: #336699;
}
.b{
float: left;
width: 460px;
height: 250px;
background: #767676;
}
.c{
float: left;
width: 460px;
height: 250px;
background: #fda500;
}
.d{
float: left;
width: 460px;
height: 250px;
background: #aa2355;
}
.clear{ clear: both; }
/***** PARTIE RESPONSIVE ADAPTATIVE *****/
@media screen and (max-width: 1050px){
#conteneur{ width: 750px; }
.a, .b, .c, .d{ width: 355px; }
}
@media screen and (max-width: 780px){
#conteneur{ width: 550px; }
.a, .b, .c, .d{ width: 255px; }
}
@media screen and (max-width: 580px){
#conteneur{ width: 100%; }
.a, .b, .c, .d{ width: 100%; height: 125px; }
section{ margin: 0; }
}
/***** PARTIE RESPONSIVE ADAPTATIVE *****/

```

Certaines parties de CSS auraient pu être regroupées mais comme il s'agit d'un entraînement, nous avons choisi de décomposer le code

Décryptons le code CSS adaptatif :

explications:

`#conteneur` permet de sélectionner la zone (div) ayant pour id conteneur (pour contenir toute la page web).

`border: 2px solid;` permet de dessiner une bordure de 2 pixels tout autour.

`margin: 0 auto;` permet de centrer la page web horizontalement.

`width: 960px;` permet de donner une largeur fixe.

`header` permet de sélectionner la zone du haut (balise header).

`background: #f6523f;` permet d'appliquer une couleur de fond.

`height: 100px;` permet de donner une hauteur fixe.

`p` permet de sélectionner tous les paragraphes.

`margin: 0;` permet de "casser" l'héritage de marges.

`nav` permet de sélectionner la zone de navigation (menu).

`background: #665533;` permet d'appliquer une couleur de fond.

`height: 30px;` permet de donner une hauteur fixe.

`section` permet de sélectionner la zone ayant pour balise <section>.

`background: #f23f98;` permet d'appliquer une couleur de fond.

`margin: 20px;` permet de donner une marge tout autour de la zone.

`footer` permet de sélectionner la zone du bas ayant pour balise <footer>.

`background: #32f65f;` permet d'appliquer une couleur de fond.

`height: 50px;` permet de donner une hauteur fixe.

`.a, .b, .c, .d` permet de sélectionner la zone ayant pour classe css "a".

`float: left;` donne un effet flottant

`width: 460px;` donne une largeur fixe

`height: 250px;` donne une hauteur fixe

`background: #336699;` donne une couleur de fond

.clear permet de selectionner la zone ayant pour classe css "clear".

clear: both; permet de "stopper" l'effet flottant (initié par le float: left;).

Voici la partie responsive adaptative :

@media permet de prévoir un comportement différent en fonction d'une résolution d'écran.

@media screen and (max-width: 1050px){#conteneur{ width: 750px; } .a, .b, .c, .d{ width: 355px; } } entre 780px et 1050px, le #conteneur fera 750px de large et les zones .a, .b, .c, .d feront 355px. de large

@media permet de prévoir un comportement différent en fonction d'une résolution d'écran.

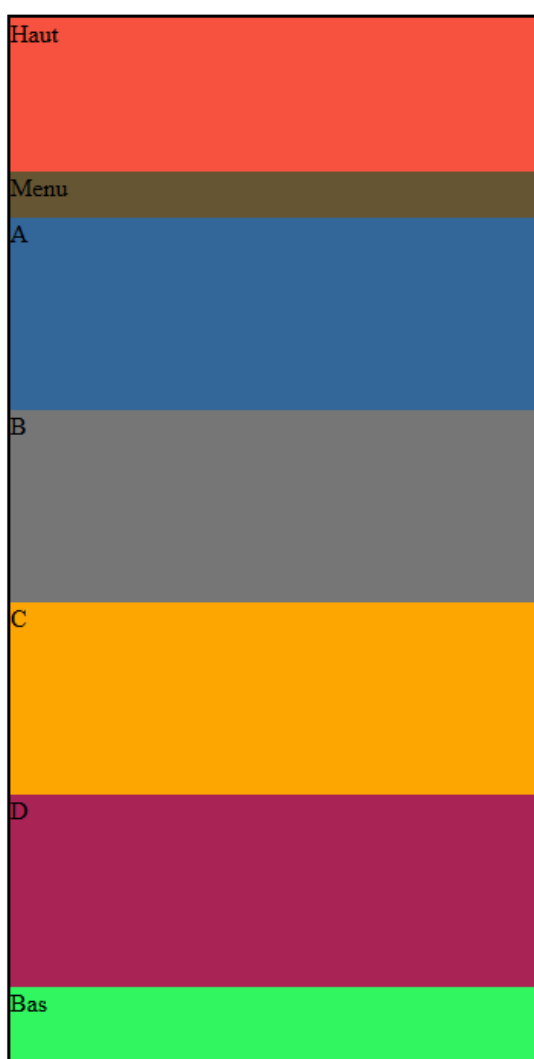
@media screen and (max-width: 1050px){#conteneur{ width: 750px; } .a, .b, .c, .d{ width: 355px; } } entre 780px et 1050px, le #conteneur fera 750px de large et les zones .a, .b, .c, .d feront 355px. de large

@media permet de prévoir un comportement différent en fonction d'une résolution d'écran.

@media screen and (max-width: 780px){ #conteneur{ width: 550px; } .a, .b, .c, .d{ width: 255px; } } entre 0px et 780px, le #conteneur fera 550px et les zones .a, .b, .c, .d feront 255px.

@media permet de prévoir un comportement différent en fonction d'une résolution d'écran.

@media screen and (max-width: 580px){ #conteneur{ width: 100%; } .a, .b, .c, .d{ width: 100%; height: 125px; }section{ margin: 0; } } entre 0px et 580px, le #conteneur fera 100% (580px ou -) et les zones .a, .b, .c, .d feront 100% (580px ou -)



Le Résultat (adaptatif)

Si vous avez copié/collé le code, n'hésitez pas à redimensionner votre fenêtre pour tester l'élasticité du site

15

Une structure Fluide

Exemple Html Fluide :

```
site-fluide.html
<!DOCTYPE html>
<html>
  <head>
    <title>Mon Site Responsive</title>
    <link rel="stylesheet" href="site-fluide.css" />
  </head>
  <body>
    <div id="conteneur">
      <header>
        <p>Haut</p>
      </header>
      <nav>
        <p>Menu</p>
      </nav>
      <section>
        <div class="a"><p>A</p></div>
        <div class="b"><p>B</p></div>
        <div class="clear"></div>
        <div class="c"><p>C</p></div>
        <div class="d"><p>D</p></div>
        <div class="clear"></div>
      </section>
      <footer>
        <p>Bas</p>
      </footer>
    </div>
  </body>
</html>
```

Nous créons une div (div pour division) permettant d'englober les autres éléments du site dans une même zone.

Nous faisons appel à la balise header et footer respectivement pour le haut et bas de site.

La balise nav permet de créer une zone de navigation.

La balise section permettra dans notre cas de prévoir une partie centrale pour le contenu

Pour cet exemple, nous avons déclaré d'autres zones dans la partie section : a, b, c et d.

Exemple Css fluide :

```
site-fluide.css
#conteneur{
border: 2px solid;
margin: 0 auto;
width: 80%;
max-width: 1200px;
font-size: 1.2em;
}
header{
background: #6523f;
height: 15vh;
}
p{ margin: 0; }
nav{
background: #665533;
height: 5vh;
}
section{
background: #f23f98;
margin: 2%;
}
footer{
background: #32f65;
height: 7.5vh;
}
.a{
float: left;
```

```
width: 50%;
height: 28vh;
background: #336699;
}
.b{
float: left;
width: 50%;
height: 28vh;
background: #767676;
}
.c{
float: left;
width: 50%;
height: 28vh;
background: #fda500;
}
.d{
float: left;
width: 50%;
height: 28vh;
background: #aa2355;
}
.clear{ clear: both; }
```

Certaines parties de CSS auraient pu être regroupées mais comme il s'agit d'un entraînement, nous avons choisi de décomposer le code

Décryptons le code CSS fluide :

Cadre explication (background de fond) :

#conteneur permet de sélectionner la zone (div) ayant pour id conteneur (pour contenir toute la page web).

border: 2px solid; permet de dessiner une bordure de 2 pixels tout autour.

margin: 0 auto; permet de centrer la page web horizontalement.

width: 80%; permet de donner une largeur de 80% au conteneur (80 % de la taille de l'écran, sauf dans le cas d'une max-width, celui ci sera donc toujours adapté).

max-width: 1200px; La largeur maximum est fixée à 1200px. Par conséquent le conteneur fera 80% de 1200px ou 80% d'une largeur inférieure (selon la taille de l'écran / fenêtre)

font-size: 1.2vw; La taille du texte sera de 1,2 [ViewPortWidth](#)

header permet de sélectionner la zone du haut (balise header).

background: #f6523f; permet d'appliquer une couleur de fond.

height: 15vh; permet de donner une hauteur de 15 [ViewPortHeight](#) (15% en hauteur).

p permet de sélectionner tous les paragraphes.

margin: 0; permet de "casser" l'héritage de marges.

nav permet de sélectionner la zone de navigation (menu).

background: #665533; permet d'appliquer une couleur de fond.

height: 5vh; permet de donner une hauteur de 5 [ViewPortHeight](#) (5% en hauteur)

section permet de sélectionner la zone ayant pour balise <section>.

background: #f23f98; permet d'appliquer une couleur de fond.

margin: 2%; permet de donner une marge tout autour de la zone de 2% (les pourcentages permettent d'imposer une marge variable selon la taille de

l'écran / fenêtre).

`footer` permet de sélectionner la zone du bas ayant pour balise `<footer>`.

`background: #32f65f;` permet d'appliquer une couleur de fond.

`height: 7.5vh;` permet de donner une hauteur de 7.5 [ViewportHeight](#) (7.5% en hauteur)

`.a, .b, .c, .d` permet de sélectionner la zone ayant pour classe css "a".

`float: left;` donne un effet flottant

`width: 50%;` donne une largeur variable : 50% de la taille de l'écran / fenêtre en largeur

`height: 28vh;` donne une hauteur variable : 50% de la taille de l'écran / fenêtre en hauteur

`background: #336699;` donne une couleur de fond

`.clear` permet de sélectionner la zone ayant pour classe css "clear".

`clear: both;` permet de "stopper" l'effet flottant (initié par le `float: left;`).

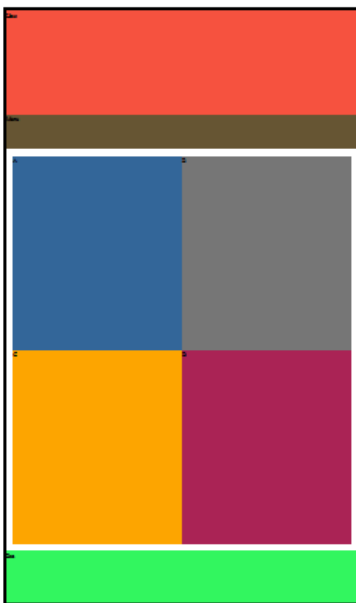
Avec les tailles variables (pourcentage et viewport), nous n'avons pas utilisé la partie css adaptative mais les deux techniques pourraient très bien être cumulées pour les besoins d'un projet.



➤ Informations

vw : « Viewport Width », correspond à l'unité relative à la largeur de votre écran

vh : « Viewport Height » correspond à l'unité relative à la hauteur de votre écran



Le Résultat (fluide)

Si vous avez copier/coller le code, n'hésitez pas à redimensionner votre fenêtre pour tester l'élasticité du site

Voici les différentes unités de mesures que nous pouvons rencontrer :

- **px** - `p{ font-size: 15px; }` - Taille de texte exprimée en pixels
PX « pixel » est une unité de mesure absolue qui apparait toujours de la même manière sur les différents écrans / résolutions.
- **em** - `p{ font-size: 1em; }` - Taille de texte exprimée en EM
EM est une unité de longueur relative à la police héritée. `font-size: 0.7 em;` correspond à 70% de la taille normale de la police.
- **mm** - `p{ font-size: 5mm; }` - Taille de texte exprimée en millimètres
- **cm** - `p{ font-size: 5cm; }` - Taille de texte exprimée en centimètres
- **%** - `div{ width: 70%; }` - Largeur d'une zone exprimée en pourcentage
Les pourcentages sont relatifs à la taille de la fenêtre.
- **vh** - `div{ height: 70vh; }` - Hauteur d'une zone exprimée en VH
VH « Viewport Height », fait référence à l'unité relative de la hauteur d'écran.
- **vw** - `div{ width: 70vw; }` - Largeur d'une zone exprimée en VH
VW : « Viewport Width », fait référence à l'unité relative de la largeur d'écran.
- **vmin** - `div{ width: 70vmin; }` - Largeur minimum d'une zone exprimée en VMIN (VM pour IE9+)
VMIN « Viewport Minimum » - fait référence à l'unité relative de la largeur d'écran (ou d'un bloc parent s'il y en a un). Nous utilisons la propriété VM pour IE9 et +.
- **vmax** - `div{ width: 200vmax; }` - Largeur maximum d'une zone exprimée en VMAX (VM pour IE9+)
VMAX « Viewport Maximum » - fait référence à l'unité relative de la largeur d'écran (ou d'un bloc parent s'il y en a un).

D'autres unités de mesures existent tel que : in, pt, pc, ex, ch, rem.

Site web responsive design

Voici une base responsive, n'hésitez pas à l'utiliser pour vos projets :

Voici le code Html :

base-responsive.html

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
```

```

<meta http-equiv="X-UA-Compatible" content="IE=edge">
<title>Base Template Responsive 2 colonnes Full Width</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<meta name="description" lang="fr" content="DESCRIPTION DU SITE">
<meta name="author" content="AUTEUR">
<meta name="robots" content="index, follow">

<!-- ICONES -->
<link rel="favicon-icon" href="img/favicon.png">
<link rel="shortcut icon" href="img/favicon.ico">

<!-- CSS -->
<link rel="stylesheet" href="css/base-responsive.css">

<!-- JS -->
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
</head>
<body>
<header>
<div class="container">
<div class="header-logo">
<h1 class="header-nom-site">Nom Du Site</h1>
</div>
<div class="header-slogan">
<p>Le Slogan du Site</p>
</div>
<div class="header-droite">
<span>Zone Haut Droite</span>
</div>
<div class="clear"></div>
</div>
</header>
<nav>
<div class="container">
<ul>
<li><a href="">Accueil</a></li>
<li><a href="">Qui Sommes nous ?</a></li>
<li><a href="">Contact</a></li>
</ul>
</div>
</nav>
<section>
<div class="container">
<main>
<h1>Titre de page</h1>
<hr>

<!-- Titre et niveaux -->
<h1>Titre niveau 1</h1>
<h2>Titre niveau 2</h2>
<h2>Titre niveau 3</h2>
<h2>Titre niveau 4</h2>
<h5>Titre niveau 5</h5>
<h6>Titre niveau 6</h6>
<hr>

<!-- Image Responsive -->
<h2>Responsive images</h2>
<p>Afin que vos images soient responsive, vous pouvez utiliser la classe <code>.img-responsive</code>.</p>
<p></p>
<hr>

<!-- Paragraphe -->
<h2>Paragraphe</h2>
<p>Le Lorem Ipsum est simplement du faux texte employé dans la <a href="#">composition et la mise en page avant impression</a>.
Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des
morceaux de texte pour réaliser un livre spécimen de polices de texte.</p>
<p> Il n'a pas fait que survivre cinq siècles, mais s'est aussi adapté à la bureautique informatique, sans que son contenu n'en soit
modifié. Il a été popularisé dans les années 1960 grâce à la vente de feuilles Letraset contenant des passages du Lorem Ipsum</p>
<hr>

<!-- Liste -->
<h2>Liste Non Ordonnée</h2>
<p>Voici un exemple de liste non ordonnée.</p>
<ul>
<li>Element A</li>
<li>Element B</li>
<li>Element C</li>
<li>Element D</li>
<li>Element E</li>
</ul>
<hr>

<h2>Liste Ordonnée</h2>
<p>Voici un exemple de liste ordonnée.</p>
<ol>
<li>Element 1</li>
<li>Element 2</li>
<li>Element 3</li>
<li>Element 4</li>
<li>Element 5</li>
</ol>
<hr>

<h2>Liste non mise en forme</h2>
<p>Pour une liste non mise en forme, vous pouvez utiliser la classe <code>list-unstyled</code>.</p>
<ul class="list-unstyled">
<li>Element</li>
<li>Element</li>
<li>Element</li>
<li>Element</li>
<li>Element</li>
</ul>
<hr>

```



```

<h2>Liste en ligne</h2>
<p>Pour une liste sur une seule ligne, vous pouvez utiliser la classe list-inline.</p>
<ul class="list-inline">
  <li>One</li>
  <li>Two</li>
  <li>Three</li>
  <li>Four</li>
</ul>
<hr>

<h2>Bouton</h2>
<p>Vous pouvez appliquer la classe css btn pour avoir un style prédéfinie sur vos boutons.</p>
<p><a href="#" class="btn">Submit</a></p>
<hr>

<!-- Tables -->
<h2>Tables</h2>
<p>Exemple d'un tableau de données</p>
<table>
  <thead>
    <tr>
      <th>#</th>
      <th>Prénom</th>
      <th>Nom</th>
      <th>Pseudo</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th>1</th>
      <td>Julien</td>
      <td>Cottet</td>
      <td>j.cottet</td>
    </tr>
    <tr>
      <th>2</th>
      <td>Amandine</td>
      <td>Thoyer</td>
      <td>lamandine</td>
    </tr>
    <tr>
      <th>3</th>
      <td>Thomas</td>
      <td>Winter</td>
      <td>twinter</td>
    </tr>
  </tbody>
</table>
</main>
<aside>
<h2>Sidebar</h2>
<p>Contrairement à une opinion répandue, <a href="#">le Lorem Ipsum</a> n'est pas simplement du texte aléatoire. Il trouve ses racines dans une oeuvre de la littérature latine classique datant de 45 av. J.-C., le rendant vieux de 2000 ans. Un professeur du Hampden-Sydney College, en Virginie, s'est intéressé à un des mots latins les plus obscurs, consecetur, extrait d'un passage du Lorem Ipsum, et en étudiant tous les usages de ce mot dans la littérature classique, découvrit la source incontestable du Lorem Ipsum. Il provient en fait des sections 1.10.32 et 1.10.33 du "De Finibus Bonorum et Malorum" (Des Suprêmes Biens et des Suprêmes Maux) de Cicéron. Cet ouvrage, très populaire pendant la Renaissance, est un traité sur la théorie de l'éthique. Les premières lignes du Lorem Ipsum, "Lorem ipsum dolor sit amet...", proviennent de la section 1.10.32.</p>
</aside>
<div class="clear"></div>
</div>
</section>
<footer>
  <div class="container">
    © Copyright 2016
  </div>
</footer>
</body>
</html>

```

Et, voici le code CSS :

base-responsive.css

```

/***** GENERAL *****/
body{
  margin: 0;
  padding: 0;
  color: #333;
  font: 1em/1.2 Helvetica, Arial, Geneva, sans-serif;
}
p{ line-height: 1.5; margin: 0; }
a:link { color: #337ab7; text-decoration: none; }
a:visited { color: #002366; }
a:focus { color: #000; }
a:hover { text-decoration: underline; }
a:active { color: #f43c80; }
hr{
  margin: 1em 0 2em 0;
  border: 0;
  border-top: 1px solid #ddd;
}
table{
  border-collapse: collapse;
  border-top: 1px solid #ddd;
  width: 100%;
  max-width: 100%;
  margin-bottom: 20px;
}

```

```

}
th, td{
padding: 0.5em 1em;
vertical-align: top;
text-align: left;
border-bottom: 1px solid #ddd;
}
.container{
max-width: 70em;
margin: 0 auto;
}
.clear{ clear: both; }
h1, h2, h2, h2, h5, h3 {
font-weight: 500;
margin: 0 0 0.5em;
}
.img-responsive { max-width: 100%; }
/***** GENERAL *****/
/***** HAUT *****/
header {
padding: 1em 3em;
min-height: 30px;
background: #dedede;
}
.header-slogan , .header-droite { padding: 0.3em 0 0 0; }
.header-logo{
float: left;
margin-bottom: 1em;
margin-right: 5%;
width: 30%;
/* background: red; */
}
h1.header-nom-site { margin: 0; font-size: 1.5em; }
.header-slogan
{
float: left;
width: 30%;
margin-bottom: 1em;
margin-right: 5%;
text-align: center;
/* background: blue; */
}
.header-droite
{
float: left;
width: 30%;
text-align: right;
/* background: green; */
}
/***** HAUT *****/
/***** MENU *****/
nav{
background: #555;
text-align: center;
min-height: 30px;
}
nav ul{
margin: 0;
padding: 0;
list-style: none;
}
nav ul li{
display: inline;
margin: 0;
}
nav ul li a{
display:inline-block;
text-decoration: none;
padding: 0 2em;
color: #fff !important;
min-height: 30px;
line-height: 30px;
}
nav ul a:Hover{
background: #337ab7;
text-decoration: none;
}
/***** MENU *****/
/***** CONTENU *****/
section{ padding: 1em;}
main
{
float: left;
width: 65%;
margin-right: 5%;
margin-bottom: 1em;
}
aside
{
float: left;
width: 30%;
margin-bottom: 1em;
}
/***** CONTENU *****/
/***** BAS *****/
footer
{
color: #fff;
background: #555;
padding: 1em 1.25em;
}
/***** BAS *****/

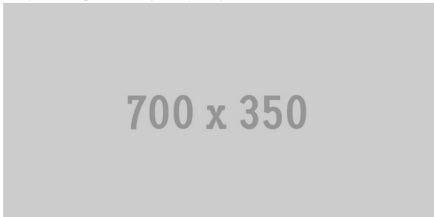
```

```

/***** RESPONSIVE TABLETTE *****/
@media (max-width: 60em)
{ /* une seule colonne (one column page) */
  body:before{ background: #fda500; content: "responsive version tablette"; position: absolute; color: #fff; padding: 2px; font-size: 12px; }
  main{ width: 100%; }
  aside{ width: 100%; }
}
/***** RESPONSIVE TABLETTE *****/
/***** RESPONSIVE SMARTPHONE *****/
@media (max-width: 36em)
{ /* width 100% pour les éléments du header et du menu (full width) */
  body:before{ background: #337ab7; content: "responsive version smartphone"; position: absolute; color: #fff; padding: 2px; font-size: 12px; }
  .header-logo{ width: 100%; text-align: left; }
  .header-slogan{ width: 100%; text-align: left; }
  .header-droite{ width: 100%; text-align: left; }
  nav ul li a{ display: block; border-bottom: 1px solid #c0c0c0; text-align: left; }
}
/***** RESPONSIVE SMARTPHONE *****/
/***** REPRISE FACULTATIVE *****/
.list-unstyled{
  padding-left: 0;
  list-style: none;
}
.list-inline{
  padding-left: 0;
  margin-left: -5px;
  list-style: none;
}
.list-inline > li{
  display: inline-block;
  padding-right: 5px;
  padding-left: 5px;
}
}
.btn{
  color: #fff !important;
  background-color: royalblue;
  border-color: #222;
  display: inline-block;
  padding: .5em 1em;
  margin-bottom: 0;
  font-weight: 400;
  line-height: 1.2;
  text-align: center;
  white-space: nowrap;
  vertical-align: middle;
  cursor: pointer;
  border: 1px solid transparent;
  border-radius: .2em;
  text-decoration: none;
}
.btn:hover{
  color: #fff !important;
  background-color: #111;
}
.btn:focus{
  color: #fff !important;
  background-color: #fda500;
}
.btn:active{
  color: #fff !important;
  background-color: #143c80;
}
/***** REPRISE FACULTATIVE *****/

```

Le Résultat (responsive) Desktop écran d'ordinateur :

Nom Du Site	Le Slogan du Site	Zone Haut Droite
	Accueil	Qui Sommes nous ? Contact
Titre de page Titre niveau 1 Titre niveau 2 Titre niveau 3 Titre niveau 4 Titre niveau 5 Titre niveau 6		Sidebar Contrairement à une opinion répandue, le Lorem Ipsum n'est pas simplement du texte aléatoire. Il trouve ses racines dans une oeuvre de la littérature latine classique datant de 45 av. J.-C., le <i>rendant vieux</i> de 2000 ans. Un professeur du Hampden-Sydney College, en Virginie, s'est intéressé à un des mots latins les plus obscurs, <i>consectetur</i> , extrait d'un passage du Lorem Ipsum, et en étudiant tous les usages de ce mot dans la littérature classique, découvrit la source incontestable du Lorem Ipsum. Il provient en fait des sections 1.10.32 et 1.10.33 du "De Finibus Bonorum et Malorum" (Des Suprêmes Biens et des Suprêmes Maux) de Cicéron. Cet ouvrage, très populaire pendant la Renaissance, est un traité sur la théorie de l'éthique. Les premières lignes du Lorem Ipsum, "Lorem ipsum dolor sit amet...", proviennent de la section 1.10.32.
Responsive images Afin que vos images soient responsive, vous pouvez utiliser la classe <code>.img-responsive</code> .		
		

Tablette :

responsive version tablette

Nom Du Site

Le Slogan du Site

Zone Haut Droite

Accueil

Qui Sommes nous ?

Contact

Titre de page

Titre niveau 1

Titre niveau 2

Titre niveau 3

Titre niveau 4

Titre niveau 5

Titre niveau 6

Responsive images

Afin que vos images soient responsive, vous pouvez utiliser la classe `.img-responsive`.



700 x 350

Smartphone :

responsive version smartphone

Nom Du Site

Le Slogan du Site

Zone Haut Droite

Accueil

Qui Sommes nous ?

Contact

Titre de page

Titre niveau 1

Titre niveau 2

Titre niveau 3

Titre niveau 4

Titre niveau 5

Titre niveau 6

Responsive images

Afin que vos images soient responsive, vous pouvez utiliser la classe `.img-responsive`.



700 x 350

Tout d'abord, au niveau de l'arborescence, un dossier `/base-responsive/` est créé.

Le fichier `base-responsive.html` est créé à la racine de ce même dossier

2 sous dossiers sont créés : `/img/` et `/css/`

Voici un lien vers l'icone [favicon.ico](#) et [favicon.png](#) à enregistrer et à placer dans le dossier `/img/` du projet.

Le code JavaScript ci-dessous permet l'interprétation des balises html5 (header, footer, main, aside, nav, etc.) par les anciens navigateurs :

```
<script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
```

Ensuite, ce n'est pas un conteneur qui contient tout le site mais bien un conteneur à l'intérieur de chaque zone importante (header, nav, section, footer) pour donner un effet pleine largeur (full width).

```
<header>
  <div class="container">
    ...
  </div>
</header>
<nav>
  <div class="container">
    ...
  </div>
</nav>
<section>
  <div class="container">
    ...
  </div>
</section>
<footer>
  <div class="container">
    ...
  </div>
</footer>
```

C'est la raison pour laquelle le container n'a pas été nommé avec un ID mais une classe `.container` afin que l'on puisse la rappeler plusieurs fois.

1 zone compose le menu du site et 3 zones prennent position dans la partie du haut de site `header-logo`, `header-slogan` et `header-droite` :

```
<header>
  <div class="container">
    <div class="header-logo">
      <h1 class="header-nom-site">Nom Du Site</h1>
    </div>
    <div class="header-slogan">
      <p>Le Slogan du Site</p>
    </div>
    <div class="header-droite">
      <span>Zone Haut Droite</span>
    </div>
    <div class="clear"></div>
  </div>
</header>
<nav>
  <div class="container">
    <ul>
      <li><a href="">Accueil</a></li>
      <li><a href="">Qui Sommes nous ?</a></li>
      <li><a href="">Contact</a></li>
    </ul>
  </div>
</nav>
```

Le code CSS de base pour ces différentes zones est le suivant :

```
/****** HAUT *****/
header {
padding: 1em 3em;
min-height: 30px;
background: #dedede;
}
.header-slogan , .header-droite { padding: 0.3em 0 0 0; }
.header-logo{
float: left;
margin-bottom: 1em;
margin-right: 5%;
width: 30%;
}
h1.header-nom-site { margin: 0; font-size: 1.5em; }
.header-slogan
{
float: left;
width: 30%;
margin-bottom: 1em;
margin-right: 5%;
text-align: center;
}
.header-droite
{
float: left;
width: 30%;
text-align: right;
}
/****** HAUT *****/
/****** MENU *****/
nav{
background: #555;
text-align: center;
min-height: 30px;
}
nav ul{
margin: 0;
padding: 0;
list-style: none;
}
nav ul li{
display: inline;
margin: 0;
}
nav ul li a{
display:inline-block;
text-decoration: none;
padding: 0 2em;
color: #fff !important;
min-height: 30px;
line-height: 30px;
}
nav ul a:hover{
background: #337ab7;
text-decoration: none;
}
/****** MENU *****/
```

Vous pouvez apercevoir que chaque zone se voit dotée d'une largeur de 30 % de l'écran, ainsi lorsqu'une connexion aura lieu avec un plus petit écran, le site sera élastique et s'adaptera automatiquement (pourcentage % = responsive fluide).

Concernant le haut de site, nous n'avons pas défini de règle différente ou complémentaire dans la version responsive inférieure a 60 em (pour écran intermédiaire type tablette).

En revanche, pour la version responsive inférieure à 36 em (pour petit écran type smartphone), nous avons défini le code suivant :

```
@media (max-width: 36em)
{
body:before{ background: #337ab7; content: "responsive version smartphone"; position: absolute; color: #fff; padding: 2px; font-size: 12px; }
.header-logo{ width: 100%; text-align: left; }
.header-slogan{ width: 100%; text-align: left; }
.header-droite{ width: 100%; text-align: left; }
nav ul li a{ display: block; border-bottom: 1px solid #c0c0c0; text-align: left; }
}
```

Pour une meilleure visibilité (par le mobinaute), ce code permet d'étirer les zones du haut à 100% (sur toute la largeur) et ainsi ne plus bénéficier de l'effet côte à côte `float: left;`.

De la même manière, les liens du menu prendront toute la place en largeur.

La 1ère ligne `body:before` permet de garder une indication lors des tests (par l'intermédiaire de la propriété `content` qui nous permet d'intégrer du texte dans le code Html), il ne faudra pas oublier de la retirer lors de la mise en ligne (en production).

2 zones composent le milieu du site *main* et *aside* :

```
<section>
  <div class="container">
    <main>
      <h1>Titre de page</h1>
      <hr>

      <!-- Titre et niveaux -->
      <h1>Titre niveau 1</h1>
      <h2>Titre niveau 2</h2>
      <h2>Titre niveau 3</h2>
      <h2>Titre niveau 4</h2>
      <h5>Titre niveau 5</h5>
      <h6>Titre niveau 6</h6>
      <hr>

      <!-- Image Responsive -->
      <h2>Responsive images</h2>
      <p>Afin que vos images soient responsive, vous pouvez utiliser la classe <code>.img-responsive</code>.</p>
      <p></p>
      <hr>

      <!-- Paragraphe -->
      <h2>Paragraphe</h2>
      <p>Le Lorem Ipsum est simplement du faux texte employé dans la <a href="#">composition et la mise en page avant impression</a>. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.</p>
      <p> Il n'a pas fait que survivre cinq siècles, mais s'est aussi adapté à la bureautique informatique, sans que son contenu n'en soit modifié. Il a été popularisé dans les années 1960 grâce à la vente de feuilles Letraset contenant des passages du Lorem Ipsum.</p>
      <hr>
    </main>
    <aside>
      <h2>Sidebar</h2>
      <p>Contrairement à une opinion répandue, <a href="#">le Lorem Ipsum</a> n'est pas simplement du texte aléatoire. Il trouve ses racines dans une oeuvre de la littérature latine classique datant de 45 av. J.-C., le rendant vieux de 2000 ans. Un professeur du Hampden-Sydney College, en Virginie, s'est intéressé à un des mots latins les plus obscurs, consetetur, extrait d'un passage du Lorem Ipsum, et en étudiant tous les usages de ce mot dans la littérature classique, découvrit la source incontestable du Lorem Ipsum. Il provient en fait des sections 1.10.32 et 1.10.33 du "De Finibus Bonorum et Malorum" (Des Suprêmes Biens et des Suprêmes Maux) de Cicéron. Cet ouvrage, très populaire pendant la Renaissance, est un traité sur la théorie de l'éthique. Les premières lignes du Lorem Ipsum, "Lorem ipsum dolor sit amet...", proviennent de la section 1.10.32.</p>
    </aside>
  <div class="clear"></div>
</div>
</section>
```

Voici le code CSS pour la partie contenu :

```
/****** CONTENU *****/
section{ padding: 1em;}
main
{
  float: left;
  width: 65%;
  margin-right: 5%;
  margin-bottom: 1em;
}
aside
{
  float: left;
  width: 30%;
  margin-bottom: 1em;
}
/****** CONTENU *****/
```

La zone `main` se voit dotée d'une largeur de 65 % de l'écran et `aside` d'une largeur de 30% de l'écran, avec les 5% de marges cela fait bien 100%, ainsi lorsqu'une connexion aura lieu avec un plus petit écran, le site sera élastique et s'adaptera automatiquement (pourcentage % = responsive fluide).

Le milieu de site est sur 2 colonnes et sera réduit à 1 colonne (l'une en dessous de l'autre) dans la version responsive inférieure à 60 em pour une taille d'écran moyen (type tablette).

```
/****** RESPONSIVE TABLETTE *****/
@media (max-width: 60em)
{ /* une seule colonne (one column page) */
  body:before{ background: #fda500; content: "responsive version tablette"; position: absolute; color: #fff; padding: 2px; font-size: 12px; }
  main{ width: 100%; }
  aside{ width: 100%; }
}
```

Par conséquent, pour la version responsive inférieure à 36 em (petit écran type smartphone), il ne sera pas nécessaire d'y retoucher (puisque cela aura été fait dans la version type tablette).

Le bas du site n'est pas concerné par la partie responsive.

© Copyright *Aucune reproduction, même partielle (textes, documents, images, etc.), ne peut être faite sans l'accord de son auteur.*

Liam TARDIEU
EVOGUE.fr - EPROJET.fr