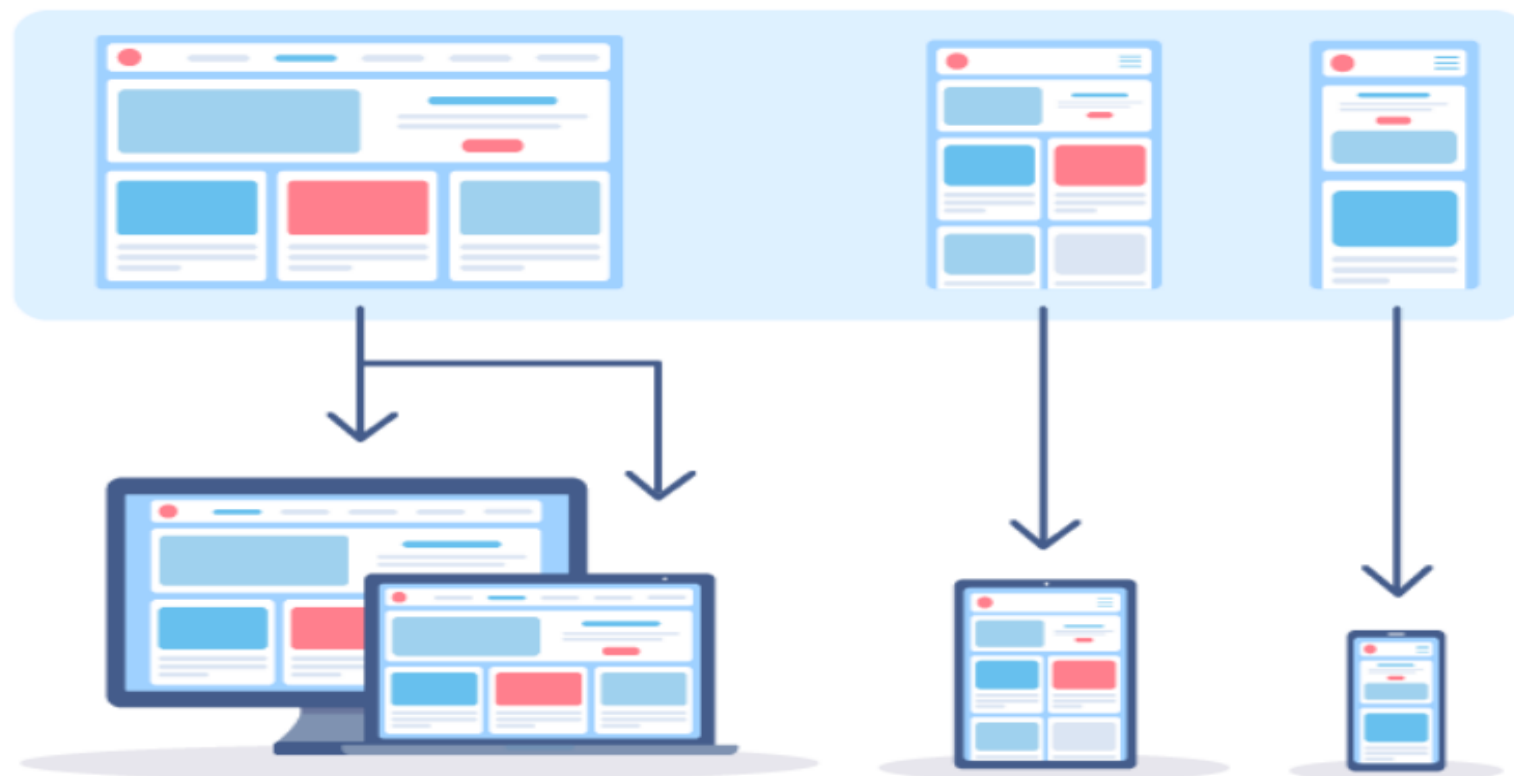
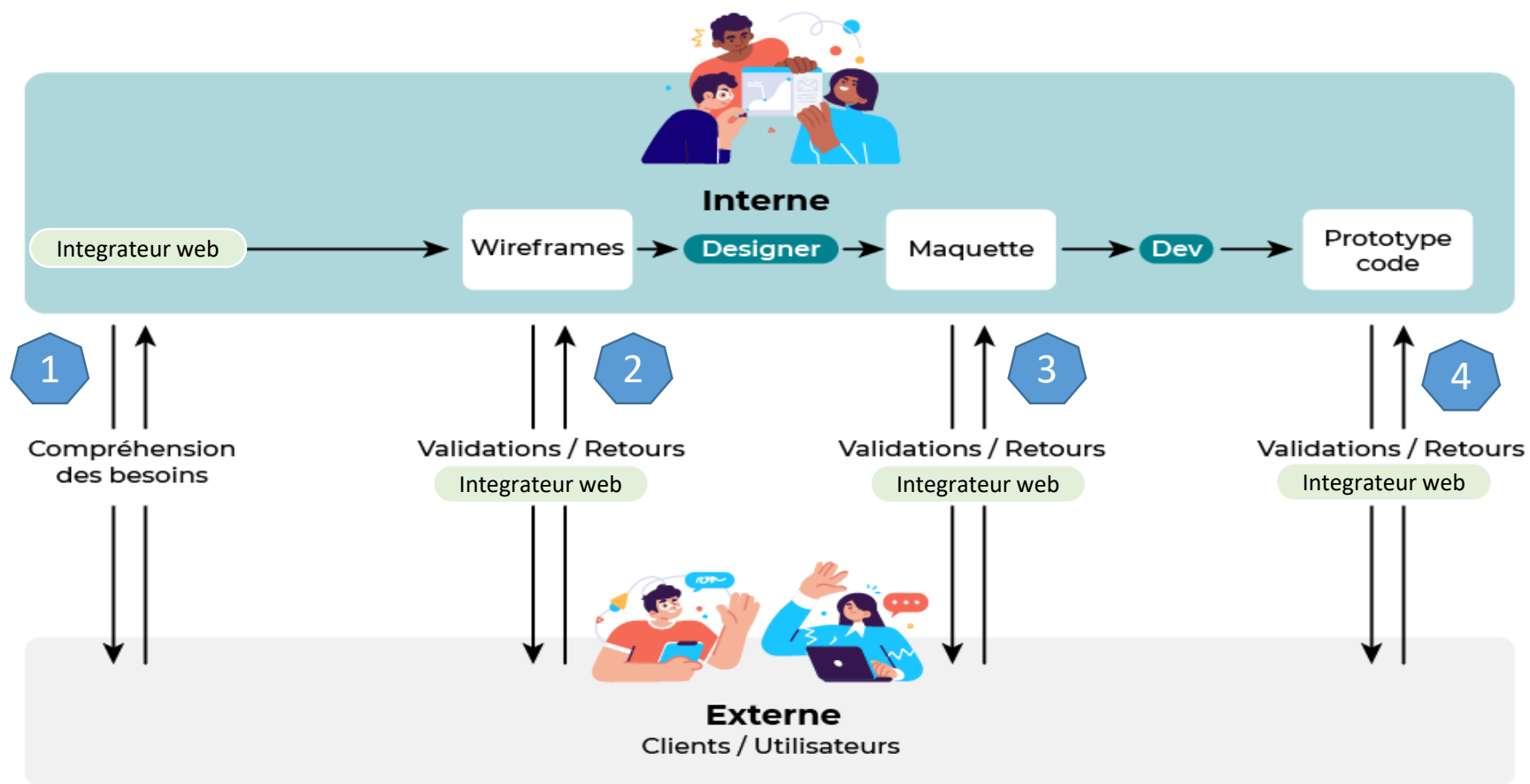


Comment?





L'objectif est d'étudier le web Design et ceci à travers les techniques utilisées pour la conception et la structuration des interfaces web et les principes pour qu'un site soit de qualité.



- Les techniques pour la conception et la structuration des interfaces web
- Quelques principes à suivre pour qu'un site soit de qualité tels que :
  - Ergonomie
  - Utilisabilité
  - Disponibilité
  - Accessibilité

Donc il est crucial de s'interroger sur la finalité du site car il doit répondre à un besoin et non pas à une envie. Ainsi pour éviter tout problème on va étudier quelques étapes nécessaires pour la création d'un site telles que



## Initiation et audit

Il s'agit d'une phase de **collecte et de mise en forme des informations** : communiquées par le **client** (objectifs, utilités, moyens, cibles, fonctionnalités,...) **collectées sur internet** (activité du secteur, stratégie des concurrents, benchmark,...).  
Cet audit débouche sur la rédaction de documents synthétisant et formalisant les données collectées.



## Spécifications fonctionnelles et techniques

Le **cahier des spécifications fonctionnelles et techniques** est un document qui permettra de présenter avec un maximum de détails et de thématiques abordées l'ébauche du futur site Internet.

Et on en trouve:

- Problématique du projet : contexte du projet, utilité(s), cible(s), micro-utilités, positionnement,...
- Accès au site
- Architecture du site
- Arborescence du site et schéma des liens
- Ergonomie
- Contenus, fonctionnalités et interactions
- Spécificités techniques



## Charte graphique

En fonction des spécifications, des éléments et contraintes ergonomiques, des fonctionnalités, est déclinée une charte graphique sous forme d'un ou de plusieurs "images" du site.

Elle est remaniée jusqu'à ce qu'elle satisfasse le client.

C'est sa validation qui conditionne l'étape suivante



## Intégration et développement

Lors de la construction des pages, il est nécessaire de raisonner à chaque instant sur les choix effectués pour un garantir un maximum d'évolutivité et de pérennité au site , apporter au site la portabilité la plus large, et faciliter l'utilisabilité du site et son indexation par les moteurs de recherche.





**Le web design désigne la conception de l'interface web** : l'architecture interactionnelle, l'organisation des pages, l'arborescence et la navigation dans le site web .

Il s'agit d'une phase essentielle dans la conception d'un tel site.

**La conception d'un design web** tient compte des contraintes spécifiques du support Internet, notamment en termes d'ergonomie, d'utilisabilité et d'accessibilité.

**Le web design réclame donc des compétences variées** : en *programmation*, en *ergonomie* et en *interactivité*, ainsi qu'une bonne *connaissance des contraintes techniques* liées à ce domaine : **diversité des terminaux web** et de leurs affichages, accessibilité, spécificités des **différents langages** et **processus**, *portabilité*, *respect des recommandations du W3C*.



## Objectifs

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

## 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.



## 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.

**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) ?



## 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	$\geq 768$ px < 992 px	$\geq 992$ px < 1200 px	$\geq 1200$ px

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.



## 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 privilégiée : pourcentage, em, vh, vw, rem, 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 !

## 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 frameworks



## 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).



## 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>
```



## 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 :

La partie Css :

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>
```

fixe1.css

```
.zone-fixe{
  width: 1000px;
  background: blue;
  color: white;
  padding: 10px;
}
```

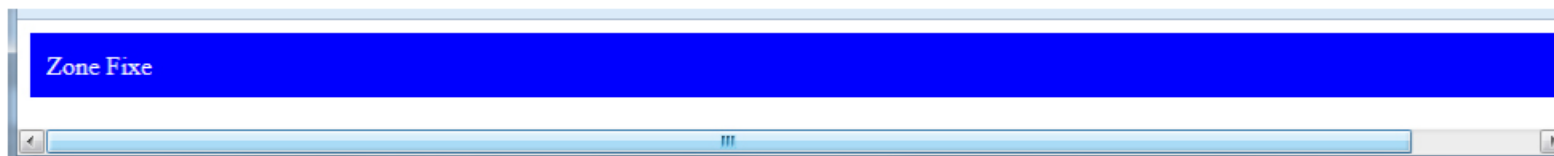


## Résultat

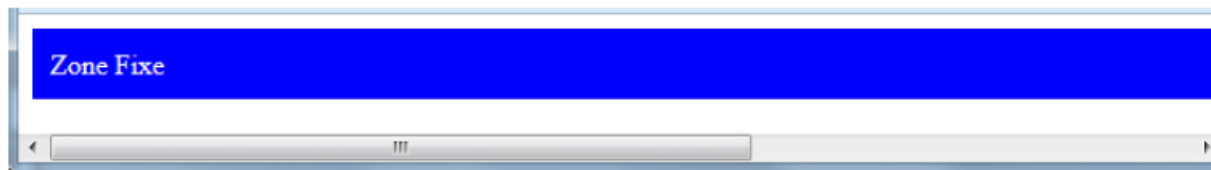
**Taille grand écran** (ordinateur de bureau) :



**Taille écran moyen** (ordinateur portable) :



**Taille écran réduit** (tablette) :



**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 apparaitre 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 !



## Exemple simple responsive design fluide

Voici un exemple Html simple de responsive design fluide :

La partie Css :

```
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>
```

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).



## Résultat

**Taille grand écran (ordinateur de bureau) :**

Zone Fluid

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

**Taille écran moyen (ordinateur portable) :**

Zone Fluid

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

**Taille écran réduit (tablette) :**

Zone Fluid

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 % !

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 % !



## 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.*

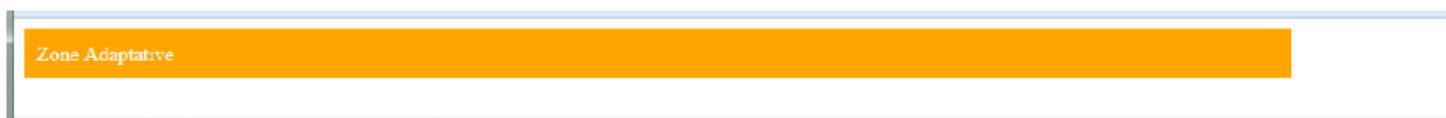
### Résultat

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).



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 defilement).



#### ➤ Bon à savoir

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

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

### Observations

FireBug

The screenshot shows the FireBug developer tool interface. At the top, there's a preview of an orange bar labeled 'Zone Adaptative'. Below it, the 'HTML' tab is selected, showing the following code:

```
<!DOCTYPE html>
<html>
  <head>
  <body>
    <div class="zone-adaptative"> Zone Adaptative </div>
  </body>
</html>
```

The 'Style' tab is also open, showing three CSS rules for the '.zone-adaptative' class:

- respons...if1.css (ligne 19): width: 1170px;
- respons...if1.css (ligne 14): width: 800px;
- respons...if1.css (ligne 9): width: 750px;



## Exemple responsive design Fluide et Adaptatif

Alors, qu'est-ce qui serait le mieux entre le fluid 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>
```



# 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; }
}
```

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).



### 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.

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.



## Résultat

Taille grand écran (ordinateur de bureau) :

Zone 2

Zone 3

Taille écran moyen (ordinateur portable) :

Zone 2

Zone 3

Taille écran réduit (tablette) :

Zone 2

Zone 3

Taille petit écran (mobile) :

Zone 2

Zone 3

*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 !



## 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 :

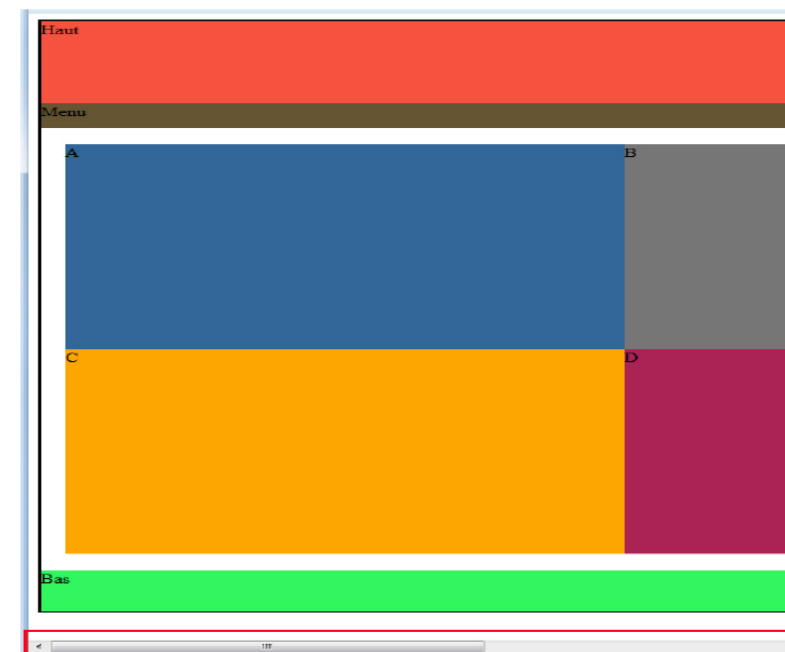
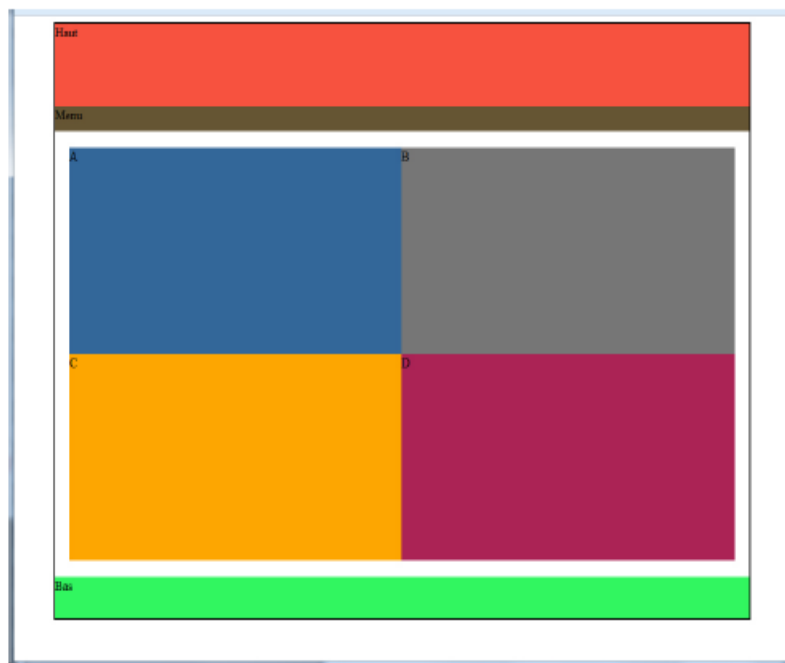
```
@media (orientation: portrait) { ... } /* règle pour les affichages en mode portrait */
```

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 ?

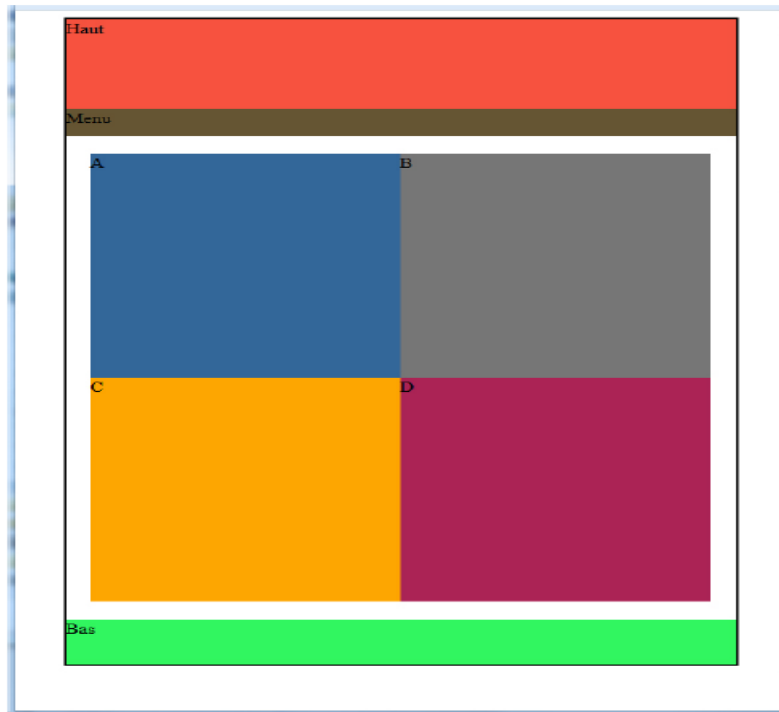
Pour cela il suffit de redimensionner la fenêtre du navigateur et de la réduire afin de voir si le contenu s'adapte.  
Si le site n'est pas responsive design, vous verrez une barre de défilement horizontale s'afficher.  
Exemple en plein écran :



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







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

### Une structure Fixe (non responsive)

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>
        <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.



# 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; }
```

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

Haut

Menu

A

C

Bas



## 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 :

```

```

