Responsive Web design

1- INTRODUCTION

1- Introduction
CONSTATS

- Les terminaux permettant de consulter des contenus web sont nombreux et hétérogènes
- Les terminaux ont des résolutions d'écran différentes
- On peut consulter un contenu web en mode portrait ou paysage (*Orientation)*

1- Introduction
LES TERMINAUX CONCERNES

- Ordinateurs
  - dont les très grands écrans
- Smartphone
  - Large variété de taille !
- Tablettes
  - dont les tableaux de bord auto connectés
- « Phablettes »
- TV connectées
- Imprimantes (mise en page dédiée à l'impression)
- (…)

1- Introduction
ADAPTER LES MISES EN PAGE POUR TOUS ? (1/3)

- Plusieurs solutions :
- 1/ site web dédié mobile (*web app mobile*)
  - Ergonomie pensée exclusivement pour mobile/tablette
  - Inutilisable/illisible/peu ergonomique sur ordinateur
  - À l'aide par ex d'un framework comme JqueryMobile ou de feuilles de style dédiées
  - Nécessite une url dédiée (ex http://mobile.domaine.com) ou détection du terminal client à la réception de la requête HTTP (possible en PHP ou en JS …)
  - Inconvénient : il s'agit finalement de faire plusieurs versions du site, à maintenir en parallèle

1- Introduction
ADAPTER LES MISES EN PAGE POUR TOUS ? (2/3)

- 2/ application native mobile
  - Développement Java pour Androïd ou Objective-C pour IOS …
  - Intérêts : le plus adapté aux terminaux mobiles, accès aux fonctions de la machine (photo, boussole, etc.)
  - Inconvénients : demande d'autres compétences, il s'agit d'un nouveau projet en marge du projet web, coût etc.

1- Introduction
ADAPTER LES MISES EN PAGE POUR TOUS ? (3/3)

- 3/ Site web « responsive »
  - Un seul site et un seul jeu de feuilles CSS
  - Un seul site à maintenir
  - Compétences HTML/CSS (+ JS ?) seulement

1- Introduction
DESIGN MOBILE vs DESIGN RESPONSIVE

- Un design mobile est forcément responsive mais pas l'inverse !
- Un design mobile :
  - tient compte de l'ergonomie et règles d'utilisation d'un mobile
    - Ex : plus de liens, mais des boutons !
  - Doit créer une « expérience utilisateur » (un « look and feel) proche de l'utilisation d'une application native
- Nécessaire de changer leur « rendu graphique » de certains objets HTML usuel
  - Exemple: voir ce que fait JqueryMobile

1- Introduction
LES ELEMENTS UTILISABLES

- Dimensions des blocs, images et textes
- Position et disposition des blocs

1- Introduction
BONNE PRATIQUE

- Penser « mobile » d'abord !
  - Puis version ordinateur
- En 2015 le nombre de contenus web accédés via mobile a dépassé celui des contenus accédés via ordinateur

1- Introduction
PRATIQUES INCOMPATIBLES AVEC RESPONSIVE

- Un seul design figé fait sous Photoshop puis découpé
- Développer son contenu en Silverlight ou Flash
- Un design basé sur des images fixes
- Utilisation de dimensions en pixels
- Utilisation de positionnements absolus …

1- Introduction
TESTER SES DEVELOPPEMENTS RESPONSIVES

- Plusieurs solutions
  - 1/ tester directement sur les périphériques cibles (une fois uploadé sur internet)
  - 2/ jouer sur la taille de fenêtre de son navigateur (manuellement)
  - 3/ plugins navigateur (ex de l'inspecteur d'éléments Chrome) qui permettent de « simuler » différents terminaux (marques et modele, orientation paysage et portrait)

1- Introduction
DETECTER LE TYPE DE TERMINAL

- Côté serveur (*ex: PHP, Python, NodeJS …)*
  - Grâce au *User Agent*, fourni par la requête HTTP
    - Contient infos sur navigateur, OS, terminal etc.
  - Pour gagner du temps => bibliothèque (classe) PHP : *MobileDetect.php*
- Côté client (*en JS*) :
  - Grâce au User Agent : propriété de l'objet JS *Navigator*
  - Grâce aux résolutions d'écran
    - Propriétés de l'objet *window*
    - Ou *MediaQueries CSS 3 …*

2 – RAPPELS CSS

- Sélecteurs CSS
- Positionnements (flux, *display, …)*
- Préfixes et compatibilité

2 – Rappels CSS
LES SELECTEURS CSS

- Opérateurs que l'on applique aux objets CSS afin de formaliser plus facilement les opérations à faire sur les objets du DOM
- Les plus connus
  - # : fait référence à un id
  - . : fait référence à une classe CSS
  - Par défaut si pas de sélecteur on fait références à tous les occurrences d'une balise html
- Liste complète de tous les sélecteurs :

http://www.w3schools.com/cssref/css_selectors.asp

2- Rappels CSS : Positionnement
PROBLEMATIQUE

- Mise en page CSS :
  - Le contenu est complétement indépendant de la façon dont il sera placé dans la page
  - On peut changer la mise en page sans toucher à l'HTML

2- Rappels CSS : Positionnement
DIFFERENTS POSITIONNEMENTS

- Position dans le flux (par défaut) et sa variante en position relative
- Position absolue (CSS 2)
- Position fixe
- Position flottante

2- Rappels CSS : Positionnement
NOTION DE FLUX (1/8)

- Toutes les balises html se classent selon les deux catégories
  - En ligne (*inline*)
  - Ou en bloc (*block*)
- (Disparait en HTML 5)
- Détermine leur mode d'affichage, leur comportement, leur mise en forme par défaut
- Le positionnement sera également fonction du positionnement de la balise parente

2- Rappels CSS : Positionnement
NOTION DE FLUX (2/8)

- Un élément de type “block” peut
  - Contenir un ou plusieurs éléments de type block
  - Contenir un ou plusieurs éléments de type inline
  - Etre dimensionné (largeur et hauteur)
  - Avoir une position précise
  - Prend toute la ligne (saut de ligne avant et saut de ligne apres) : deux éléments “blocks” ne peuvent pas cohabiter sur la meme ligne

2- Rappels CSS : Positionnement
NOTION DE FLUX (3/8)

- Un élément de type “inline” peut
  - Contenir un ou plusieurs éléments de type inline seulement
  - Ne pas peut contenir d’éléments “block”
  - Ne peut pas être dimensionné (largeur et hauteur) : ses dimensions seront déterminées par son contenu
  - Ne peut pas avoir une position précise (en théorie)
  - Les éléments “inline” cohabitent sur la même ligne tant qu’il y reste de la place

2- Rappels CSS : Positionnement
NOTION DE FLUX (4/8)

- Par défaut
  - les éléments en bloc s'affichent l'un en dessous de l'autre (du haut vers le bas)
  - Les éléments en ligne s'affichent l'un à coté de l'autre (de la gauche vers la droite)
  - Dans le respect de la largeur / hauteur maximum …
- Quelques balises de type "block"
  - ,<ol>,<h1>,<h2>,<h3>, <p>, <table>, <form>,
- Quelques balises de type "inline"
  - <font>, ,<img>,
, <a>,<b>, <input>, <select>, <textarea> …

2- Rappels CSS : Positionnement
NOTION DE FLUX (5/8)

- En changeant ce comportement par défaut d'une balise on peut donc changer son mode d'affichage
  - Propriété CSS : display
  - Valeurs possibles :
    - *Block ou inline*,
    - *None* : l'élément n'est pas affiché (ne prend pas de place)
    - *inline-block* : comme un inline mais peut etre redimensionné
    - *list-element* : comme un block mais peut recevoir des propriétés propres aux listes
    - *Table, table row, table cell* : même comportement qu'un tableau notamment pouvoir centrer le contenu verticalement

2- Rappels CSS : Positionnement
NOTION DE FLUX (6/8)

2- Rappels CSS : Positionnement
NOTION DE FLUX (7/8)

- Changer le ‘display’
  - = changer le mode d’affichage
  - = permettre l’application de nouvelles propriétés
  - Exemple
    - une balise de type “block” peut d’un coup devenir de type “inline” et donc s’afficher à droite de l’élément précédent

2- Rappels CSS : Positionnement
NOTION DE FLUX (8/8)

- Exemple :
  - Changer le mode par défaut d'un élément de liste <li> de vertical (block) à horizontal (inline) pour créer un menu !
- Autre solution pour jouer sur l'affichage
  - choisir entre les balise ou (par exemple) pour des éléments qui s'affichent horizontalement ou verticalement …

2- Rappels CSS : Positionnement
POSITION RELATIVE

- Propriété :
  - `Position : relative;`
- Permet d'affiner la position de l'élt dans le flux
  - Décaler de plusieurs pixels vers le haut / bas / gauche / droite :
    - *left, top, bottom, right*
  - Appliquer des marges extérieures ou intérieures
    - *Margin-top, margin-left, margin-right, margin-top*
    - *Padding-top, padding-left, padding-right, padding-top*
  - Référence : position de base de l'élément dans le flux
  - Peut etre combinée avec un "display : …"

2- Rappels CSS : Positionnement
POSITION ABSOLUE (1/3)

- Propriété :
  - `Position : absolute;`
- Ne pas dépendre
  - Ni de la position dans le flux
  - Ni de l'ordre dans lequel apparait l'élément dans le code html
- Ne s'applique qu'à des éléments de type block

2- Rappels CSS : Positionnement
POSITION ABSOLUE (2/3)

- Permet de définir les "coordonnées" de l'élément
  - Par les propriétés : top, left, right, bottom
  - En fait on définit le décalage (en nb de pixels) par rapport à un élément de référence
    - Ex : `top : 25px;`
      - Élément sera placé 25px plus bas que son élément de réf

2- Rappels CSS : Positionnement
POSITION ABSOLUE (3/3)

- Permet de définir les “coordonnées” (suite)
  - L’élément de référence est le premier élément ancètre “positionné” qu’il rencontre (en remontant dans l’arborescence)
    - Pas forcément le parent direct !
    - Positionné = position relative ou absolue ou fixe
    - Si aucun élément ancètre positionné, ce sera par rapport à la fenêtre
  - Si aucun top/left/right/bottom précisé : l’élément sera à la même place que son élément de référence
  - Ne pas oublier de positionner les éléments dans l’arborescence (position absolue ou relative)

2- Rappels CSS : Positionnement
POSITION FIXE

- Cas particulier d’une position absolue
- Syntaxe
  - `Position : fixed;`
  - Combinée à la définition de décalage (coordonnées) top/bottom/right/left
- L’élément est fixé, “accroché” à la page et ne sera pas soumis aux scrolls
  - Exemple : entête ou pied de page persistant, bouton “contact” ou “feedback” accroché sur le côté …

2- Rappels CSS : Positionnement
POSITION FLOTTANTE (1/4)

- Adaptée au positionnement en colonnes
- À l'origine : pour habiller du texte avec des images
  - Exemple : texte autour de l'image …
- largeur d'un élt flottant dictée par son contenu
- Propriété `float`
  - Permet de sortir l'élément du flux et de le positionner à l'extrême gauche ou l'extrême droite de son élément parent (le conteneur)
  - `Float : left`
  - `Float : right`

2- Rappels CSS : Positionnement
POSITION FLOTTANTE (2/4)

- Deux éléments flottants
  - Dans la même direction = s'afficheront côte à côte
    - Exemple : colonnes
  - Directions différentes = s'afficheront à l'opposé l'un de l'autre (au sein de l'élément parent)

2- Rappels CSS : Positionnement
POSITION FLOTTANTE (3/4)

- “Habillage”
  - Tout élément demeuré dans le flux et qui succède à un élément flottant sera “habillé” par l’élément flottant
  - Ex : texte flottant à gauche + image avec positionnement par défaut = l’image s’affichera à haut à gauche du texte, entourée par le texte

2- Rappels CSS : Positionnement
POSITION FLOTTANTE (4/4)

- Nettoyage des flottants
  - Propriété clear
  - permet à un élément de ne plus subir le comportement d'habillage dicté par un objet flottant qui le précède directement
    - Il va donc se caler en-dessous de ce dernier.
  - Ne marche que sur des blocks
  - `Clear : left`
  - `Clear : right`
  - `Clear : both`

2- Rappels CSS : Positionnement
BONNES PRATIQUES ? (1/2)

- Les balises DIV et SPAN permettent de définir des “boites” (très pratiques)
  - DIV notamment pour regrouper des éléments et leur affecter des propriétés de position
  - SPAN notamment pour appliquer une ancre ou des propriétés textes
- mais inutile de les multiplier !
- Exemple
  - On peut affecter les propriétés de position à toute balise de type block et pas seulement les DIV
  - Exemple : inutile d’entourer un paragraphe P avec un DIV => on peut directement positionner le paragraphe P

2- Rappels CSS : Positionnement
BONNES PRATIQUES ? (2/2)

- Souvent plusieurs méthodes pour arriver au même résultat …
- Tester sur plusieurs navigateurs et plusieurs terminaux

2- Rappels CSS : Compatibilité entre navigateurs MOTEUR DE RENDU

- Algorithme (programme) embarqué dans le navigateur
- Sert à afficher la page d'après le code source HTML
- 4 moteurs différents pour 4 rendus différents
  - Webkit : Chrome et Safari
  - Mozilla : Firefox
  - Opéra
  - Internet Explorer

2- Rappels CSS : Compatibilité entre navigateurs PREFIXES DE PROPRIETES (1/3)

- CSS3 en très récent et pas géré de la même façon par tous les navigateurs
- Certaines propriétés sont spécifiques à certains navigateurs
- Nécessité de préfixer les propriétés par le nom du moteur de rendu :
  - `-webkit-`
  - `-moz-`
  - `-o-`
  - Aucun préfixe pour IE<10
  - `-ms` (pour IE 10)

2- Rappels CSS : Compatibilité entre navigateurs PREFIXES DE PROPRIETES (2/3)

- Exemple
  - Transition
    - `-webkit-transition : 0.25s;` // chrome et safari
    - `-moz-transition : 0.25s;` // firefox
    - `transition : 0.25s;` // internet explorer et les versions récentes des navigateurs
  - Dégradé couleur
    - `-webkit-gradient ( … )`
    - `-moz-linear-gradient (…)`

2- Rappels CSS : Compatibilité entre navigateurs PREFIXES DE PROPRIETES (3/3)

- Rendre un site compatible tous navigateurs :
  - écrire plusieurs fois la même propriétés avec les préfixes différents
    - Problèmes avec IE6-7-8 (et 9?)
    - Seul IE10 ne pose plus aucun problème

2- Rappels CSS : Compatibilité entre navigateurs
CSS CONDITIONNES POUR IE (1/5)

- Rendre un site compatible tous navigateurs : mécanisme propre à IE
- Principe : mettre du code dans la page html qui ne sera interprété que par IE
  - Les autres navigateurs vont l'ignorer
- Fonctionnent avec IE5 à IE9
  - IE10 ne les interprète plus
  - Mais d'un autre côté plus besoin avec IE 10 …
- Dans la page html ou dans le fichier css

2- Rappels CSS : Compatibilité entre navigateurs
CSS CONDITIONNES POUR IE (2/5)

- Deux cas :
  - On cible IE en général

```
<!--[if IE]>
Ici : code HTML pour IE.
<![endif]-->
```

  - On cible ou on exclut une ou plusieurs versions de IE en particulier

```
<!--[if comparaison IE version]>
Ici : code HTML pour les versions d'IE
  choisies.
<![endif]-->
```

2- Rappels CSS : Compatibilité entre navigateurs CSS CONDITIONNES POUR IE (3/5)

- On cible une ou plusieurs versions de IE en particulier : (suite)
  - Syntaxe de l'opérateur de comparaison
    - = : pas de mot-clé
    - > :mot-clé `gt` pour «greater than»
    - ≥ : mot-clé `gte` pour «greater than equal»
    - < : mot-clé `lt` pour «less than»
    - ≤ : mot-clé `lte` pour «less than equal»
  - Exemple de versions d'IE
    - IE 8
    - IE 6

2- Rappels CSS : Compatibilité entre navigateurs CSS CONDITIONNES POUR IE (4/5)

- Exemples

```
<!--[if gte IE 6]>
pour IE 6.0 et version supérieures
<![endif]-->

<!--[if lte IE 6]>
pour IE 5.0, IE 5.5, IE 6.0 mais pas IE7.0 et
  supérieur
<![endif]-->

<!--[if !IE]>
si ce n'est pas IE
<![endif]-->
```

2- Rappels CSS : Compatibilité entre navigateurs
CSS CONDITIONNES POUR IE (5/5)

- Outils pour tester plusieurs versions d'IE
  - Pb car on ne peut pas installer plusieurs versions d'IE sur une même machine, sans créer de conflits
  - Outils
    - Multiple IE (*tredosoft*)
    - IETester (*debugbar*)

2- Rappels CSS : Compatibilité entre navigateurs
EVITER DE PREFIXER

- Outil qui préfixe à notre place : Prefixr.com
  - http://prefixr.com
  - On copie/colle le code CSS sans préfixe
  - Il détecte automatiquement les propriétés à préfixer et rajoute les préfixes

2- Rappels CSS : Compatibilité entre navigateurs CSS3Pie (1/2)

- “Plugin” pour donner à IE5 à 9 des propriétés qu’il n’a pas
  - Ombres, coins arrondis …
- **CSS3pie.com**
- Il suffit de télécharger et de déposer le fichier PIE.htc dans le dossier du site
- Faire le lien depuis le fichier css dans la classe
  - Propriété behavior
  - `behavior: url(/PIE.htc);`

2- Rappels CSS : Compatibilité entre navigateurs CSS3Pie (2/2)

- Propriétés supportées et offertes à IE
  - border-radius
  - box-shadow
  - border-image
  - CSS3 Backgrounds (-pie-background)
  - Gradients
  - RGBA Color Values

3- LES TYPES DE DESIGN

3- Différents types de Design
DESIGN STATIQUE

- Les unités de mesure (blocs, colonnes, textes, images …) sont exprimés en valeurs fixes
- Le positionnement des éléments est inamovible
- Tout est figé.

3- Différents types de Design
DESIGN FLUIDE (*LIQUID)*

- Les unités de mesure (dimensions des colonnes, des blocs …) sont exprimées en valeur relatives (génériques) : %, em, etc.
- Un seul design pour tous les types de terminaux
- S'adapte automatiquement à la taille de fenêtre, jusqu'à une certaine mesure (taille minimum …)
- Seules les dimensions et taille (et donc indirectement peut être la disposition) changent
  - *(impossible de changer les couleurs ou masquer / afficher des éléments en fonction de la résolution)*

3- Différents types de Design
DESIGN *ADAPTIVE*

- Basé sur la gestion des points de rupture
  - paliers de changement de résolution d'écran
- Unités de mesure exprimées en valeur fixes mais qui diffèrent selon les tailles d'écran
  - on applique différentes valeurs fixes selon le terminal
- Cela revient à avoir plusieurs designs statiques que l'on applique au choix en fonction du terminal
  - On peut modifier + que les tailles et dimensions : n'importe quelle propriété CSS !
- Les tailles d'écran (points de rupture) sont détectées grâce aux *CSS3 Media Queries*

3- Différents types de Design
DESIGN *RESPONSIVE*

- Basé sur la gestion des points de rupture ET sur l'approche fluide (*Liquid)*
- Cela revient à avoir plusieurs designs fluides l'on applique au choix en fonction du terminal grâce aux MediaQueries
- On peut modifier + que les tailles et dimensions : n'importe quelle propriété CSS

3- Différents types de Design
SYNTHESE

- Illustration grâce au site
- http://www.liquidapsive.com/

4 – DESIGN FLUIDE

4- Design Fluide
UNITEES CSS (1/9)

- Unités absolues
  - px : pixels
  - in : pouces (inch)
  - pt : point
  - cm : centimètre (et mm : millimètre)
  - pc : pica
- Unités relatives : les seules permettant un design fluide !
  - %
  - em (et rm)

4- Design Fluide
UNITEES CSS (2/9)

- Unités absolues : quel usage ?
  - in / pt / cm / pc sont des unités qui ne serviront que dans des styles CSS dédiés à l'impression (*media print*)
    - formats d'impression imposent des dimensions fixes
  - px : peut être utilisé pour des tailles de police, des largeur de bordure, des marges, des largeurs et hauteurs
    - => dans le cas de dimensions, on tombe alors dans un design « statique »

4- Design Fluide
UNITEES CSS (3/9)

- Unités relatives: quel usage ?
  - % : idéal pour définir les dimensions (largeur/hauteur) d'un élément de type bloc
    - Le % s'effectue par rapport aux dimensions du *body* ou de l'élément parent, le cas échéant
  - em, rm : concernent les tailles des polices de caracteres (voir plus loin)
    - Note (déconseillé) : Peut être aussi utilisé pour la définition de marges, de taille de bordure … mais déconseillé car le résultat est peu intuitif et compliqué à débugguer

4- Design Fluide
UNITEES CSS (4/9)

- Utilisation de *em* (et *rm*)
  - Permet de définir une taille de police en fonction de la taille de police par défaut du navigateur (ou du document html en cours, ou du bloc parent)
  - Par défaut taille de police à l'écran = 12 px
    - Équivaut à 1 e.m
  - Se définit avec la propriété *font-size* comme pour les polices en px
  - S'applique également sur les google fonts ou autres

4- Design Fluide
UNITEES CSS (5/9)

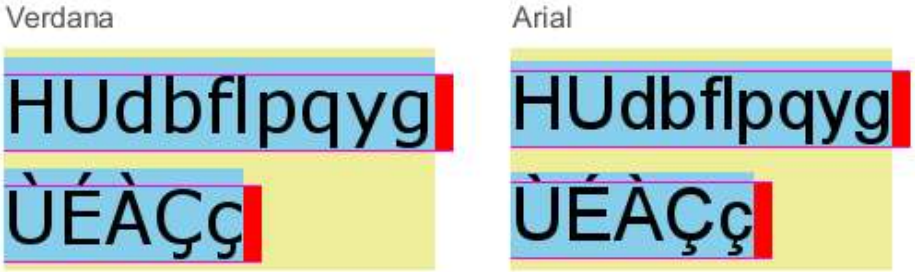
- Utilisation de *em* (et *rm*) (suite)
  - Avantage par rapport au px :
    - 1/ Tous les navigateurs permettent de définir une taille personnalisé par défaut pour le texte (pbs de vues par ex)
    - 2/ Exprimer nos tailles de police en *em* permet de se calibrer sur la taille définie dans le navigateur de l'internaute le cas échéant
    - 3/ Permet également de définir au sein du document des règles de taille de texte

4- Design Fluide
UNITEES CSS (6/9)

- Utilisation de *em* (et *rm*) (suite)
  - La taille effective d'une ligne de texte (hauteur de ligne à l'écran) va varier en fonction de
    - 1/ de la taille du texte par défaut
      - paramètre du navigateur
    - 2/ puis des tailles définies dans le CSS sur les blocs parents (*font-size*)
      - css défini sur balises *html* ou *body* ou sur *div* parents
    - 3/ de la taille de l'espace réservé (sorte de padding) au dessus du texte lui-même, espace dont l'importance varie d'une police à l'autre
  - Note : la taille du texte ne varie pas d'une police à l'autre

4- Design Fluide
UNITEES CSS (7/9)

- Utilisation de *em* (et *rm*) (suite)
  - Illustration



4- Design Fluide
UNITEES CSS (8/9)

- Utilisation de *em* (et *rm*) (suite)
  - Cas de rm :
    - comme em
    - mais ne prend que la taille du texte navigateur
    - N'est pas impacté par la taille du texte des conteneurs parents

4- Design Fluide
UNITEES CSS (9/9)

- Tableau de correspondance des unités
  - *Sur la base de taille par défaut du texte à l'écran =12px*
  - *Voir ici :*
  - *http://reeddesign.co.uk/test/points-pixels.html*

4- Design Fluide
DISPOSITION DES BLOCS (1/3)

- Jouer sur
  - *Les positionnements : flottants , relatifs*
  - *Ou Le mode d'affichage :*
    - `display : inline-block`
    - `display : none`
- Définir des largeurs en % ou en em
- Définir des marges fixes ou auto

4- Design Fluide
DISPOSITION DES BLOCS (2/3)

- Attention // Cas des marges extérieures (*margin)*
  - Les marges ne sont pas incluses quand on définit une largeur en % ou en px !
  - Il faut donc inclure dans le calcul la largeur des marges
- Cas des marges intérieures (*padding*)
  - Par contre le padding est compté (inclus) dans la définition de la largeur du bloc

4- Design Fluide
DISPOSITION DES BLOCS (3/3)

- Solution 1 :
  - « wrapper » chacun des blocs au sein d'un nouveau bloc parent
  - Remplacer les marges extérieures sur le bloc initial par un padding sur le nouveau bloc parent
- Solution 2 :
  - Utiliser la propriété `box-sizing` (voir plus loin)

4- Design Fluide
IMAGE RESPONSIVE (1/5)

- Utiliser les largeurs en %
  - L'image se dimensionne par rapport aux dimensions du conteneur parent
  - Bonne pratique : mettre max-width:100px pour que l'image s'adapte à la taille du bloc parent
- On garde le ration en fixant une des deux dimensions et en mettant l'autre à auto

```
img {
      max-width: 100%;
      height: auto;
}
```

4- Design Fluide
IMAGE RESPONSIVE (2/5)

- Nécessaire dans tous les cas de passer ses images à travers un service comme JPEG Mini pour optimiser leur poids
- www.jpegmini.com/
- Attention : redimensionner une image à l'écran avec des largeurs en % ne diminue pas son poids
  - *Pb de bande passante surtout sur réseau 3G*

4- Design Fluide
IMAGE RESPONSIVE (3/5)

- Solution : plusieurs versions d'images
  - Rq : on ne peut pas changer le *src* d'une image en CSS.
  - 1/ changement des *src* d'images en JS en fonction de la détection du type de terminal
  - 2/ les afficher en *background-image* de *div* et changer le *background-image* dans une règle *media query*
  - 3/ mettre les images dans des *div* dont un seul est affiché, les autres étant en *display : none* (signifié ignorés et images non téléchargées). Les règles *media queries* servent alors à masquer/afficher les bons *div*

4- Design Fluide
IMAGE RESPONSIVE (4/5)

- Les dernières évolutions HTML/CSS
  - https://responsiveimages.org/
  - L'attribut `srcset` :
    - Permet définir d'autres images possibles en fonction de la largeur de l'écran
    - Le navigateur ne téléchargera que l'image nécessaire
    - Si le navigateur ne supporte pas cet attribut il charge l'image du src

```
<img src="small.jpg"
     srcset="large.jpg 1024w, medium.jpg 640w,
small.jpg 320w"
     alt="A rad wolf">
```


4- Design Fluide
IMAGE RESPONSIVE (5/5)

- Les dernières évolutions HTML/CSS (2)
  - Les balises <picture> et <source> :
    - Remplacent la balise <img> et l'attribut <src>
    - On définit autant de <source> que l'on veut pour un <picture> donné

```
<picture>
<source media="(min-width: 480px)"
        srcset="big.jpg 1x, big-hd.jpg 2x">
<source srcset="small.jpg 1x, small-hd.jpg
        2x">
<img src="fallback.jpg" alt="">
</picture>
```

4- Design Fluide
BACKGROUND RESPONSIVE

- Utiliser la valeur « cover » sur la propriété background-size
- Prévoir une image de fond tres large (ex: 2000px * 1300px)
- Code :

```
background: url('./bg.jpg') no-repeat center fixed;
-webkit-background-size: cover;
-moz-background-size: cover;
-o-background-size: cover;
background-size: cover;
```


4- Design Fluide
BOX-SIZING

- Constat : cas d'un élément enfant qui déborde du parent car width+bordure+padding > largeur du parent
- `box-sizing: border-box;`
- Permet d'inclure les *paddings* et bordures dans le calcul de la valeur *width* (longueur), tout en excluant la marge
- À appliquer sans modération sur tout élément CSS dont on fixe les dimensions largeur et/ou hauteur
- Evite les débordements

4- Design Fluide
GERER LES DEBORDEMENTS DE TEXTE (1/3)

- On n'est pas maître de sa mise en page texte :
  - possibilité par ex à l'utilisateur d'augmenter la taille des polices dans les parametres navigateur ou réduction taille fenetre
- Solutions au texte qui déborde
- 1/ Césure basique (retour à la ligne forcé =couper les mots qui débordent en deux)
  - Utiliser `word-wrap: break-word;`

4- Design Fluide
GERER LES DEBORDEMENTS DE TEXTE (2/3)

- 2/ masquer le « débordement »
  - Mais perte d'une partie du texte
  - Utiliser `overflow:hidden`
  - utiliser en sus la propriété *text-overflow* pour faire apparaître des (…) à l'endroit où l'overflow coupe le texte

```
overflow:hidden
-o-text-overflow: ellipsis;
text-overflow: ellipsis;
```

4- Design Fluide
GERER LES DEBORDEMENTS DE TEXTE (3/3)

- 3/ la césure intelligente
  - Propriété `hyphens`
  - Comme *word-wrap* mais ajuste la césure au meilleur endroit (meilleure lettre) en fonction de la langue
  - + rajoute un « - » au moment du retour à la ligne
  - Peu supporté encore par les navigateurs
  - Bonne pratique : combiner *word-wrap* et *hyphens*

```
-webkit-hyphens: auto;
-moz-hyphens: auto;
-ms-hyphens: auto;
-o-hyphens: auto;
hyphens: auto;
```


4- Design Fluide
GERER LES DEBORDEMENTS CELLULES

- Par défaut, une cellule prends les dimensions de son contenu
  - => risque que la cellule bouscule le design
- Utiliser la propriété `table-layout` pour empêcher une cellule de se redimensionner à cause de son contenu

4- Design Fluide
GERER LES DEBORDEMENTS DE TOUT CONTENU

- Penser à faire des clear après des blocs flottants
- Utiliser `box-sizing`
- Utiliser les propriétés liés au texte (voir ci avant)
- Utiliser `overflow:hidden` pour que la partie qui déborde soit masquée
- Fixer les coordonnées maximum des éléments par rapport au conteneur parent
  - `max-width:100%`
  - `max-height:100%`

4- Design Fluide
REGLAGE COMPLEMENTAIRE : LE VIEWPORT

- Dans le cas des mobiles / tablettes
- Le terminal va dézoomer par défaut, pour faire tenir tout le contenu dans la page
  - Contournement : configurer le *viewport (largeur de fenêtre)*

```
<meta name="viewport" content=
"width=device-width,initialscale=1" />
```

- Plus d'infos plus loin (chapitre *media queries*) sur le *viewport*

4- Design Fluide
BONNES PRATIQUES

- Simplifier son code HTML et éviter div superflus
- Utiliser les *media queries* (cf ci apres)
- Utiliser une grille et une mise en page fluide avec des unités relatives
- Optimiser ses images
- Sur versions mobiles : éliminer les blocs et sections inutiles et penser navigation linéaire
- Fixer *min-width* et *max-width* pour éviter débordements
- Définir le viewport
- Utiliser *box-sizing*
- Éviter les positionnements absolus

4- Design Fluide
A PROPOS DESIGN MOBILES/TABLETTES (1/2)

- Ne pas utiliser la sous-classe css *Hover*
  - ca n'existe pas sur écran tactile
  - le doigt n'est pas vu comme une souris donc pas de survol
- Les gens ont des gros doigts :
  - utiliser des boutons plutot que des liens
  - ou des liens stylisés en boutons

4- Design Fluide
A PROPOS DESIGN MOBILES/TABLETTES (2/2)

- Les gens ont des gros doigts (2) :
  - penser à espacer vos éléments cliquables
  - éviter conflits du type "le mauvais lien est enclenché"
- Les gens sont en majorité droitiers
  - Les menus à gauche ne sont donc plus une bonne idée puisque cliqués avec les doigts

5 – MEDIA QUERIES

5- Media Queries
LE PRINCIPE (1/2)

- Intégrer des blocs conditionnels au sein du CSS
- Appliquer (ou pas) de façon conditionnelle des ensemble de propriétés CSS
- En fonction de paramètres liés au terminal utilisé pour accéder à la ressource
  - Orientation (paysage, portrait …)
  - Largeur ou hauteur de l'écran
  - Type de terminal : écran, imprimante …
  - (…)
- Apporté par CSS 3

5- Media Queries
LE PRINCIPE (2/2)

- Chaque cas possible est alors une *media query* :
  - Dans le cas d'un design *adaptive* : la media query applique un style statique à la page (mais propre au terminal en question)
  - Dans le cas d'un design *responsive* : la media query applique un style fluide à la page (mais propre au terminal en question)

5- Media Queries
MEDIA

- Les types de *media* possibles :
  - ***All*** : tous
  - ***Screen*** : ordinateur, smartphone, tablette, tv ...
  - ***Speech*** : liseuses de textes (restitution audio)
  - ***Print*** : imprimantes
  - Tous les autres sont désormais *deprecated* :
    - *aural, braille, embossed, handheld, projection, tty, tv*
    - Cas des mobiles : bien que tenants dans la main (*handeld),* tous les smartphones et tablettes se définissent eux-mêmes désormais comme des écrans standards *(screen)*

5- Media Queries
PROPRIETES

- Voir document joint pour la liste exhaustive (officielle W3C)
- Principales propriétés :
  - Hauteur et largeur de la zone d'affichage
  - Hauteur et largeur du terminal
  - orientation (portrait ou paysage)
  - Resolution d'écran
  - Nombre de couleurs
- Chaque (presque) propriété existe également en version *min-propriété* et *max-propriété*

5- Media Queries
SYNTAXE (1/3)

- Les *media queries* peuvent être appliquées
  - À une balise <LINK …>
  - Au sein d'un bloc CSS (dans une balise *style* dans le *head* ou dans un fichier css externalisé)
  - Le choix du media et des propriétés sont des conditions que l'on peut combiner à l'aide d'opérateurs logiques
    - AND, NOT, ONLY
  - On peut créer autant que *queries* que l'on souhaite
  - Le processeur CSS les évalue une par une en séquence
    - Possibilité de rentrer dans plusieurs *queries*

5- Media Queries
SYNTAXE (2/3)

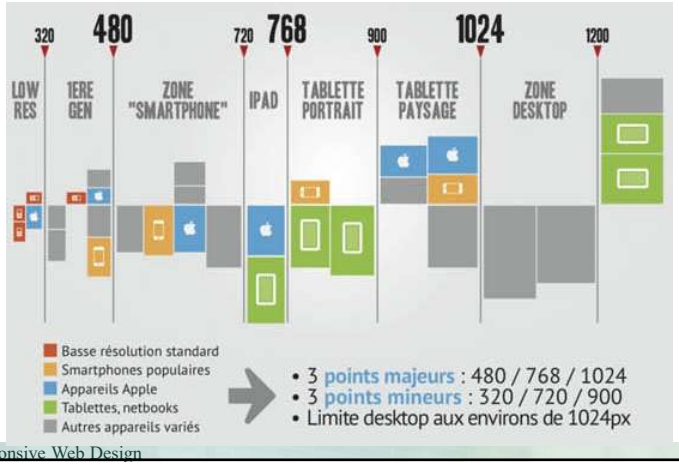
```
body { color: black; }

@media only screen
and (min-width : 321px) {
/* Styles */
      p { font-size:1.5em; }
      // (…)
}
@media only screen
and (max-width : 1024px) {
/* Styles */
      p { color: red; }
      // (…)
}
```

5- Media Queries
SYNTAXE (3/3)

```
<link rel="stylesheet"
      media="only screen and ((min-width :
                  321px))"
      href="example.css" />
```


5- Media Queries
POINTS DE RUPTURE (1/3)

- Paliers de résolutions (largeur en px) pour changer de catégorie de terminal



5- Media Queries
POINTS DE RUPTURE (2/3)

- Points de rupture « courants » (basé sur *min-width* ou *max-width*) :

1. **< 320px** : appareils très basse résolution
2. **< 480px** : smartphones 1ère génération
3. **< 768px** : smartphones récents et tablettes en mode portrait
4. **entre 768px et 1024px** : tablettes en mode paysage ou grandes tablettes en mode portrait
5. **> 1024px** : design très large pour le *full desktop*

5- Media Queries
POINTS DE RUPTURE (3/3)

- Au minimum, il gérer au moins
  - < 480px
  - < 768px
  - Et > 768px pour tout le reste

5- Media Queries
TAILLE ECRAN MOBILES ET TABLETTES (1/2)

- La réalité : un écran mobile ne dispose non pas d'un seul type de largeur (et de hauteur), mais de trois (en px) :
  - largeur "constructeur"
  - largeur "device-width" (ou screen.width en JS)
    - (plus basse)
  - largeur de fenêtre (viewport)

5- Media Queries
TAILLE ECRAN MOBILES ET TABLETTES (2/2)

- Ressources : découvrir résolutions et *device-width* des principaux terminaux mobiles et tablettes
- http://mydevice.io/devices/
- http://screensiz.es/phone
- http://www.alsacreations.com/article/lire/1490-comprendre-le-viewport-dans-le-web-mobile.html

5- Media Queries
DEFINIR UN TERMINAL EN PARTICULIER

- Chaque terminal est défini par une combinaison particulière de propriétés
  - Résolution + présence d'un écran HD ou pas + etc.
- Voir trame *media queries* type fournie

5- Media Queries
CAS DES SMART TV

- Va devenir un terminal important
- A prendre en compte dès maintenant
- Exemple de Query

```
@media tv (min-width:1919px) and (max-
resolution:72dpi) {
    // ...
}
```

- Encore en évolution …

5- Media Queries
RESSOURCES

- https://developer.mozilla.org/fr/docs/Web/CSS/Media_queries
- http://www.alsacreations.com/article/lire/1490-comprendre-le-viewport-dans-le-web-mobile.html