Chapitre 08

Les Media Queries *et le* Responsive Web Design

- Créer une feuille de style pour l'impression
- Lier une css print
- Les différents types de médias
- Interface fluide et médias flexibles
- Que sont les media queries ?
- Le Responsive Web Design (RWD)
- Syntaxe des Media Queries CSS3
- Chargement dynamique et conditionnel côté client
- La balise meta viewport
- L'approche mobile first
- Les tableaux responsives
- Les formulaires responsives
- Les outils du RWD - Les Scripts

1. CREER UNE FEUILLE DE STYLE POUR L'IMPRESSION

Une page web peut également prendre la forme d'un document imprimé, les CSS nous permettent également de définir les styles pour l'impression des pages web.
Les deux principes à mettre en place pour optimiser l'impression de son site est de proposer des versions «imprimables» (pages HTML épurées ou documents PDF) pour certains contenus susceptibles d'être imprimés souvent, et pour toutes les autres pages du site, prévoir une CSS Print afin de profiter de la fonctionnalité d'impression des navigateurs.

Les règles de mises en page pour l'impression

Le Web (média écran) est un média non paginé contrairement au média d'impression sur papier et de nombreux styles adaptés à l'écran (des largeurs en pixels, des textes en blanc sur une couleur de fond sombre, des tailles de texte en pixels ou en em…) ne sont pas adaptés pour l'impression.
Théoriquement, les feuilles de style pour l'impression permettent de gérer une mise en page relativement complète, avec des options proches de celles d'un logiciel de traitement de texte. Dans la pratique, il faudra se limiter à des choses beaucoup plus basiques, en raison d'un support très partiel par les navigateurs des propriétés CSS relatives à l'impression.

2. LIER UNE CSS PRINT

Une feuille de style pour l'impression est une feuille de style simple. La seule manipulation « spéciale » à faire consiste à indiquer au navigateur que la feuille de style en question doit s'appliquer au média `print`.

La méthode liée

Vous pouvez incorporer différents fichiers de feuilles de style pour les différents modes de sortie, qui peuvent contenir des définitions de formats différentes.
Si le navigateur Web par exemple affiche les pages à l'écran, il doit utiliser le fichier CSS qui fixe expressément "l'écran" comme mode de sortie, et si l'utilisateur imprime une page, le navigateur doit employer à la place le fichier CSS que vous mentionnez pour le mode "imprimante".
Exemple:

```
<html>
<head>
<title>Titre du fichier</title>
<link rel="stylesheet" media="screen" href="styles.css">
<link rel="stylesheet" media="print" href="impression.css">
</head>
<body>
```

Explication:
L'attribut `media=` est ajouté au `<link>` habituel, il détermine pour quel mode de sortie le fichier doit être utilisé, fichier que vous incorporez ensuite grâce à la mention `href=`.
Vous pouvez mentionner un ou plusieurs types de sortie pour `media=`. Plusieurs mentions sont à séparer par des virgules.
Attention à bien préciser le type de média pour toute les feuilles de styles liées car, sinon, la première feuille de style s'appliquera pour tous les médias, y compris pour l'impression.

La méthode interne

Vous pouvez ajouter à la section <style>…</style> de votre en-tête les styles spécifiques pour l'impression grâce à la règle @media :

```
@media print {
  … vos styles…
}
```

Exemple :

```
@media print {
  #menu, #footer, aside {
    display:none;
  }
```

La préparation à l'impression, quelles questions faut-il se poser ?

Que faut-il imprimer ?

De nombreux éléments utiles à l'écran ne seront pas utiles sur le papier. Il s'agit alors de les identifier, et de les «supprimer» pour l'impression.
Pour empêcher l'impression d'un élément de votre page, utilisez la propriété : **`display: none;`**
Exemples : menu de navigation, formulaire, …

Pensez toutefois à **conserver ou adapter les éléments permettant l'identification du site**, ceux-ci apparaîtront simplement plus petits, ou plus sobres.
Petit rappel : **les images de fond ne sont, la plupart du temps, pas imprimables**. Si on veut imprimer un logo du site, ce logo devra être une image présente dans le contenu HTML, via la balise <img>.

Les unités adaptées au format papier

Les unités les mieux adaptées à l'impression sont les points (`pt`), les centimètres (`cm`) et millimètres (`mm`), et les pourcentages (%).
Les points typographiques ou points sont les plus adaptés pour fixer la taille du texte. Pour les valeurs à utiliser, faire des essais avec un logiciel de traitement de texte.
Les millimètres et centimètres, ainsi que les pourcentages seront utiles pour le dimensionnement des blocs, pour déterminer les marges des éléments, etc.

Orientation et marges

Pour tout le contenu
Pour spécifier une orientation Portrait, utilisez la règle : `@page {size: portrait;}`
Pour spécifier une orientation Paysage, utilisez la règle : `@page {size: landscape;}`
Pour spécifier la taille des marges (ici, 2cm), utilisez la règle : `@page {margin: 2cm;}`
*Cependant, la plupart des navigateurs **n'en tiennent pas compte**, et appliquent des marges par défaut, ou bien des marges configurées par l'utilisateur.*

Couleurs

Arrière-plans
Pour spécifier un arrière-plan blanc, utilisez la propriété : `background-color: #ffffff;`
Pour spécifier un arrière-plan sans image de fond, utilisez la propriété : `background-image: none;`
La plupart des navigateurs conformes aux standards gèrent eux-mêmes correctement la suppression des arrière-plans de page à l'impression. Ces règles ne sont donc à employer que dans des cas particuliers.
Avant-plan
Pour spécifier la couleur noire d'un avant-plan, utilisez la propriété : `color: #000000;`
Pour modifier à l'impression la couleur d'une bordure utilisez la propriété : `border-color: #000000;`

Un exemple de feuille de style pour l'impression

1. Passer le fond de page en blanc, et les textes de contenu (+ titres et liens) en noir ;
2. Choisir une famille de police serif ;
3. Supprimer les éléments devenus inutiles (la bannière de l'entête, les bordures de la page, le menu général, le pied de page, les publicités, formulaires) ;
4. Adapter la largeur du contenu restant à la page ;
5. Modifier les unités (em et px vers pt).

Aller plus loin…

Sauts de page et lignes orphelines
Pour empêcher qu'un encadré soit imprimé « à cheval » sur deux pages, utilisez la propriété : `page-break-inside: avoid;`
Pour forcer un saut de page *avant* un élément, utilisez la propriété : `page-break-before: always;`
Pour forcer un saut de page *après* un élément, utilisez la propriété : `page-break-after: always;`

Impression des URL
Insérer entre parenthèses les URL des liens après chacun d'entre eux, les rend utiles à quiconque ayant sous la main une copie papier du document:

```
a:after {content: " (" attr(href) ") "; }
```

Impression des textes alternatifs des images
Pour imprimer le texte alternatif qui décrit le contenu de l'image :

```
img[alt]:after {
 content:" ("attr(alt)")";
}
```

Le contenu de l'attribut alt est affiché entre parenthèses après chaque image: `"(attribut alt)"`

Lecture Web :
http://actuel.fr.selfhtml.org/articles/css/mise_en_page_imp/index.htm
http://www.pompage.net/traduction/impression

3. LES DIFFERENTS TYPES DE MEDIAS

Il est possible, en CSS, de définir des formats pour différents médias de sortie en incorporant diverses feuilles de style séparées pour différents médias de sortie …

Mention	Signification
`media="all"`	Le fichier CSS s'applique à tous les types de médias.
`media="aural"`	Le fichier CSS s'applique à des systèmes de restitution vocale assistée par ordinateur.
`media="braille"`	Le fichier CSS s'applique à des périphériques de sortie avec ce qu'on appelle une "ligne Braille". Le texte y est porté sur la structure modifiable en surface du matériel de la ligne Braille et lue au toucher avec le doigt. Cette forme de média de sortie est conçue pour les aveugles.
`media="embossed"`	Le fichier CSS s'applique à des imprimantes en Braille. Le texte y est pressé en relief sous forme de structure de surface pouvant être reconnues au toucher sur du papier ou sur un matériau comparable. Comme braille, embossed est conçu pour les aveugles.

`media="handheld"`	Le fichier CSS s'applique à l'affichage sur des ordinateurs de poche très petits. Des agendas électroniques permettant la navigation sur la toile sont des représentants typiques de cette catégorie.
`media="print"`	Le fichier CSS s'applique à l'impression sur papier. Les navigateurs Web doivent utiliser ces définitions de format quand l'utilisateur désire imprimer la page Web.
`media="projection"`	Le fichier CSS s'applique à la projection de données avec des rétroprojecteurs ou appareils similaires.
`media="screen"`	Le fichier CSS s'applique à l'affichage à l'écran.
`media="tty"`	Le fichier CSS s'applique à des médias de sortie non graphiques avec une largeur de signes invariable comme par exemple des télex. Mais ce type de média est également intéressant pour des navigateurs orientés texte comme Lynx.
`media="tv"`	Le fichier CSS s'applique à des médias de sortie semblables à la télévision qui se distinguent par une résolution grossière et le défaut de soutien pour le défilement de l'image mais par contre par un soutien du son.

4. LE RESPONSIVE WEB DESIGN

Aujourd'hui, se limiter à catégoriser les supports ne suffit plus !

Autrefois, nous n'avions qu'un ou deux navigateurs et les tailles d'écrans variaient peu. Désormais, nous avons une multitude de navigateurs et un nombre important de résolutions (allant du 240x320 au 2515x1886, voire plus).
Les supports également : nous pouvons accéder à Internet depuis nos classiques ordinateurs de bureau, mais aussi via les smartphones, tablettes, etc. Ces derniers ont également des résolutions et tailles d'écran très variées !

L'expression **« Responsive Web Design »** inventée par Ethan Marcotte, désigne une méthode de travail à base de **grilles fluides,** d'**images flexibles** et de **media queries.**
Le « responsive » est une **méthode d'affichage en fonction de la largeur, qui fait partie du « web design adaptative »**, qui, lui, relève de l'accessibilité. On confond souvent les deux…

5. INTERFACE FLUIDE

Afin d'obtenir une **interface liquide et proportionnelle**, les tailles des zones structurelles doivent être exprimées en pourcentage et les dimensions typographiques doivent être exprimées en em ou rem.

1. Calculer les tailles des éléments structurels de l'interface

Une interface proportionnelle s'obtient en déterminant les dimensions des blocs et espacements en % (voir chapitre sur les mises en page élastiques pour le calcul optimal de celles-ci).
Prévoir également une **largeur maximale** avec la propriété `max-width`, une trop haute résolution peut rendre un site illisible.
Prévoir également une **largeur minimale** avec la propriété `min-width` pour éviter un contenu trop comprimé et donc illisible. En dessous d'une certaine valeur c'est une autre feuille de style qui prendra le relai grâce aux média queries.

2. La taille de police

La taille de police par défaut de l'utilisateur est de 16, mais celle-ci peut être modifiée dans les paramètres par défaut ou de manière occasionnelle à l'aide du zoom.
Les unités de mesure proportionnelles, comme le « em », pris en compte par tous les navigateurs, permettent une adaptation de la taille de la police de caractère en fonction des choix de l'utilisateur (accessibilité !).
Rester proportionnelle à la taille par défaut du navigateur, grâce à un `font-size:100%` placé sur le body permet de réajuster à loisirs le em de base par rapport à l'intégralité du site.

```
body {
          font-size :100% ;
}
p    {
          font-size:1em;
     }
```

Calculer les tailles typographiques de l'interface

Un titre de 50px sur la maquette graphique va être calculé comme suit : 50px ÷ 16px (taille de base) = 3.125 em
Un sous-titre de 35px donnera 30 ÷ 16 (taille de base) = 1.875 em
Le corps de texte de 12 px donnera 12 ÷ 16 (taille de base) = 0.75 em
Ces valeurs sont à reporter dans la feuille de style (en plus d'un body : 100%).

em ou rem ?

Le **em est cumulatif** : les enfants héritent des propriétés de taille de son parent. A prendre en compte dans l'attribution des tailles pour les éléments imbriqués les uns dans les autres.
Le **rem**, lui, **se base sur la taille définie dans la racine**, il ne se cumule pas.

Des tailles de polices proportionnelles à la taille de l'écran

De nouvelles unités de taille pour les polices de caractère permettent de faire en sorte que la taille de police soit proportionnelle à la taille de l'écran : ce sont les unités **vw**, **vh**, **vmin** et **vmax** (v comme viewport, w pour la largeur, h pour la hauteur).

10vw correspond à une taille de 10% de la largeur de la fenêtre du navigateur. Si la fenêtre fait 1600px de large, c'est comme une taille de police de 160px. Si la fenêtre est ramenée à 500px de large, alors la police sera équivalente à 50px.
10vh est l'équivalent, mais pour la hauteur de la fenêtre.
10vmax correspond au maximum de 10vw et 10vh : si l'écran est plus large que haut, alors ce sera la valeur de 10vw qui sera retenu.
10vmin correspond au minimum de 10vw et 10vh.

Quand utiliser les % ou les em ?

Les zones structurelles de mes pages doivent être **proportionnelles à l'écran**, je vais donc travailler **en %**.
Les marges, espaces internes, bordures, doivent être **proportionnelles à la taille de la typo**, je vais donc travailler **en em**.

```
body {
            font-size:100%;
      }
div {
            width:90%;
            margin:1em auto;
            padding:1em;
            border:0.15em solid #4C6600;
            background-color:#F5FFD9;
      }
```

6. MEDIAS FLEXIBLES

Pour que les médias (les images, les vidéos, …) soient responsives, nous utilisons la **propriété « max-width »** qui permet de spécifier la largeur maximum de l'élément, par rapport à son parent.

```
img, object, embed, canvas, video, audio, picture {
       max-width: 100%;
      /* Ainsi, le média ne débordera jamais de son cadre */
       height: auto;
      /* Pour conserver les bonnes proportions de nos medias */
}
```

Concrètement, on redimensionne la largeur comme on le souhaite avec `width`, `min-width` ou `max-width` (le plus souvent on utilise ce dernier avec une valeur de 100%) et on indique une hauteur automatique afin que le ratio soit adapté automatiquement.

Exemple pour une image :

Les images seront contenues dans un conteneur `<figure>` afin d'isoler l'élément image. Sémantiquement, c'est plus correct qu'un div neutre car on signale par là que c'est une figure d'illustration liée au contenu (l'article) :

```
<article>
     <figure>
      <img src="image.jpg" alt="">
      </figure>
      Mauris ac vulputate velit.
     Nullam faucibus risus sed felis tincidunt nec placerat …
</article>
```

La dimension, proportionnelle, de la balise figure au sein de l'article est calculée en respectant la même méthode que celle appliquée précédemment,
L'image occupera 100% de l'espace qui lui est alloué au sein de sa balise figure. **L'image est en réalité plus grande que la taille affichée pour permettre à celle-ci de s'agrandir sur une résolution élevée sans apparaître pixélisée.**

```
figure {
     width:95.483870967%;
     margin:1.93548387% auto;
}
article figure img {
     width:100%;
     height: auto;
}
```

L'inconvénient de cette méthode est le poid des images, relativement lourd puisque nous prévoyons une dimension permettant une qualité suffisante sur haute résolution.

Exemple pour une vidéo :

L'élément video issu de HTML5 peut, comme une image, être redimensionné sans altérer son ratio.

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

7. QUE SONT LES MEDIA QUERIES ?

La spécification CSS3 Media Queries du W3C définit les techniques pour l'application de feuilles de styles en fonction des périphériques de consultation utilisés pour des pages HTML. On nomme également cette pratique **Responsive Web Design**, pour dénoter qu'il s'agit **d'adapter dynamiquement le design à l'aide de CSS**.

Exemples de sites adoptant cette technique : http://mediaqueri.es

Ces bonnes pratiques permettent d'exploiter encore plus les avantages de la séparation du contenu et de la présentation: l'intérêt est de pouvoir satisfaire des contraintes de dimensions, de résolutions et d'autres critères variés pour améliorer l'apparence graphique et la lisibilité (voire l'utilisabilité) d'un site web.

Lecture Web :
http://www.w3.org/TR/css3-mediaqueries/
https://developer.mozilla.org/en-US/docs/CSS/Media_queries

8. SYNTAXE DES MEDIA QUERIES CSS3

La philosophie des media queries (ou **requêtes de media**) en CSS3 est d'offrir un panel de critères plus vaste et plus précis, à l'aide de propriétés et de valeurs numériques, ainsi que de combinaisons multiples de ces mêmes critères. **Le but est de cibler plus finement les périphériques de destination en fonction de leurs capacités.**

L'écriture de ces requêtes est relativement explicite (en anglais) : **une media query est une expression dont la valeur est toujours vraie ou fausse**. Il suffit d'associer les différentes déclarations possibles avec un opérateur logique pour définir l'ensemble des conditions à réunir pour l'application des styles compris dans le bloc adjacent.
Les opérateurs logiques peuvent être "*and* "(et) "*only*" (uniquement) et "*not*" (non). Pour obtenir l'équivalent du "ou", il suffit d'énumérer différentes media queries à la suite, séparées par des virgules : si l'une d'entre elles est valable, alors l'ensemble de la règle sera appliquée.
En général, on combine ensemble un type de média (*screen*, *all*...) et une expression grâce à *and*, bien qu'une expression seule puisse être utilisée. L'expression est toujours écrite entre parenthèses.

Exemple 1 : cibler les écrans de largeur inférieure à 640 pixels grâce à la règle max-width associée à la valeur 640px.

```
<link rel="stylesheet" media="screen and (max-width: 640px)"
href="smallscreen.css" type="text/css" />
```

ou

```
@media screen and (max-width: 640px) {
  .bloc {
    display:block;
    clear:both;
  }
}
```

Exemple 2 : cibler les écrans dont la résolution en largeur est comprise entre 200 et 640 pixels :

```
<link rel="stylesheet" media="screen and (min-width: 200px) and (max-width:
640px)" href="smallscreen.css" type="text/css" />
```

ou

```
@media screen and (min-width: 200px) and (max-width: 640px) {
  .bloc {
    display:block;
    clear:both;
  }
}
```

Fonctionnalités

La plupart des critères (ou fonctionnalités) peuvent être préfixés par *min-* et *max-* lorsqu'elles acceptent des valeurs numériques pour définir des valeurs minimales ou maximales à respecter.

Lecture Web :
http://stuffandnonsense.co.uk/blog/about/hardboiled_css3_media_queries propose des requêtes « prêtes à l'emploi » pour différents types d'écran

9. CHARGEMENT DYNAMIQUE ET CONDITIONNEL COTE CLIENT

Être responsive, c'est **faire en sorte que le site soit le plus rapide possible au chargement et dans son temps d'exécution**. Pour cela nous n'allons charger les feuilles de styles et le JavaScript que lorsque que ce sera nécessaire.

Ce type de code, placé en tête de notre page HTML, permet de ne le charger que quand c'est nécessaire :

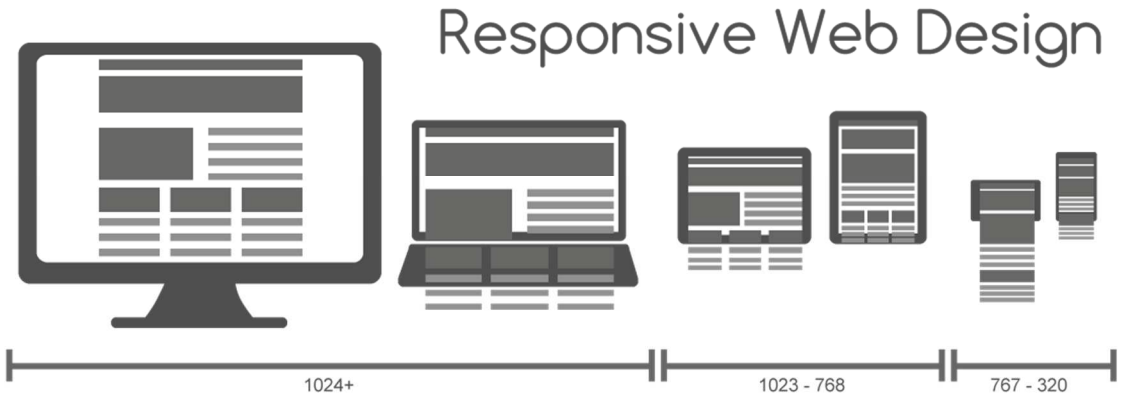
```
<link rel="stylesheet" media="screen" href="layout.css" />
<link rel="stylesheet" media="screen and (max-width: 640px)" href="small-screen-
layout.css" /><!-- Media Queries -->
<link rel="stylesheet" media="screen and (min-width: 1240px)" href="large-screen-
layout.css" /><!-- Media Queries -->
<link rel="stylesheet" media="print" href="print.css" />
```

10. UTILISER LA BALISE META VIEWPORT POUR CONTROLER LA MISE EN PAGE SUR LES NAVIGATEURS MOBILES

Les différentes surfaces d'un mobile

La surface réelle d'un mobile est le nombre physique de pixels qui composent la matrice de l'écran, telle que le constructeur le décrit dans les caractéristiques (sa définition).

La surface en "pixels CSS" appelée **device-width** ou **screen.width** est le nombre de pixels virtuels que le terminal pense avoir et sur lequel il fonde son affichage.
Cette surface ne correspond pas toujours à la surface réelle, ainsi, **un "pixel CSS" n'est donc pas égal à un pixel physique.**



Lecture Web : panel de périphériques
http://screensiz.es/phone
http://mydevice.io/devices/

Le Viewport

Par défaut **la taille du viewport d'un terminal mobile ne correspond ni à la taille de son écran réelle ni celle en "pixels CSS"**. Les pages web s'affichent par défaut de manière à ce que toute la surface entre dans celle de l'écran en appliquant **un niveau de zoom automatique**.
Ce niveau de zoom initial correspond à une simple division mathématique de `device-width` (largeur de l'écran en pixels) / `viewport` (surface de la fenêtre du navigateur).

La balise meta viewport

Il est possible de modifier et d'imposer la taille de la surface du viewport d'un périphérique mobile à l'aide d'un élément <meta>. Les différentes valeurs de cet élément `meta` et de son attribut `content`, offrent la possibilité de fixer la largeur de `viewport` à la valeur souhaitée, voire de l'adapter automatiquement à la valeur de `device-width` du terminal.

```
<meta name="viewport" content="width=device-width">
```

Le **`Viewport`** désigne schématiquement la surface de la fenêtre du navigateur.
La propriété **`width`** contrôle la taille du **`viewport`**, elle peut être réglée à une valeur précise de pixels (width=600).
La propriété **`device-width`** permet un réglage adaptatif car il correspond à la largeur de l'écran en pixels CSS à l'échelle 100%.

```
<meta name="viewport" content="initial-scale=1.0">
```

La propriété **`initial-scale`** contrôle le niveau de zoom lorsque la page est chargée pour la première fois.

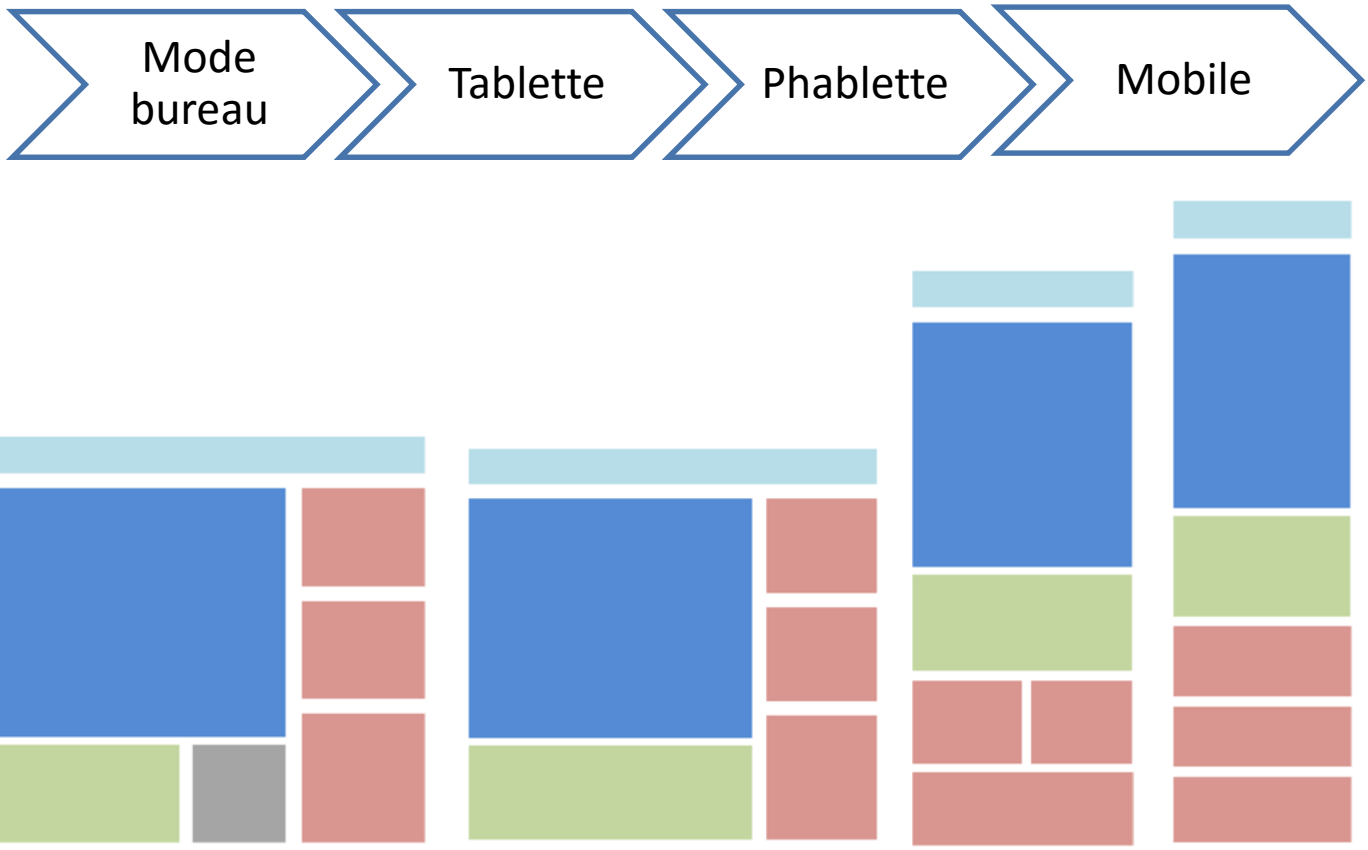
Conclusion
On peut préciser à Firefox, Chrome, Safari, Safari Mobile, Android, Opera Mini, Opera Mobile, Firefox mobile (Fennec) de s'adapter aux résolutions via le méta viewport et de contrôler le zoom via:

```
<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0">
```

11. LES GRANDES LIGNES D'UN THEME MOBILE

L'approche mobile-first consiste à optimiser le site pour la **navigation et la consultation en mobilité avant tout**.

1. Placez les éléments les uns sous les autres




```
#sidebar {
   float: right;
   width: 200px;
}
#main {
   margin-right: 200px;
}
```

```
@media (max-width: 1024px) {
   /* CSS appliqué aux petits
écrans */
   #sidebar {
      float: none;
      width: auto;
   }
   #main {
      margin-right: 0px;
   }
}
```

L'affichage multi-colonnes est à proscrire, les flottants sont à afficher empilés et non plus côte à côte (l'ordre naturel de navigation devra être correct en empilant les blocs auparavant positionnés comme flottants).
Penser à un design adaptatif dès le départ est la clé d'un responsive design facilité !

2. Réduisez les marges

Réduisez les marges latérales (margin- et padding-) à leur minimum afin de gagner de l'espace sur l'écran. Favorisez l'aération verticale : l'aspect aéré sera conservé.
Utilisez des hauteurs de ligne assez importantes : un téléphone étant tenu à la main, lire est plus difficile que sur un écran d'ordinateur qui lui est fixe : **plus la lisibilité est importante, moins le lecteur se perd dans la page**.
Affichez des caractères larges et lisibles... même si l'écran est petit, les caractères ne doivent pas l'être trop.

3. Images et vidéos adaptatives

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

De cette façon, les images ne dépasseront jamais de la largeur de l'écran : si l'écran est plus petit que l'image, alors c'est l'image qui se redimensionne en conservant ses proportions.

4. Éviter le débordement des images de fond

Utilisez la propriété CSS3 `background-size` : celle-ci va alors décider de la façon dont l'image de fond occupe l'espace de son conteneur[1].

5. Changer l'organisation des tableaux

Sur un écran de mobile, quand l'affichage normal ne permet plus de placer toutes les cellules du tableau côte à côte, on linéarise le tableau :

```
table, tbody { display: block; }
tr { display: table; }
td { display: table-row; }
```

Pour les très grands tableaux, une solution (un peu moins propre) peut-être de masquer certaines colonnes de moindre importance au fur et à mesure que la page est rétrécie :

```
@media (max-width: 600px) {
    table tr th:nth-of-type(2),  /* Cellules de moindre importance */
    table tr td:nth-of-type(2) {
        display: none;
    }
}

@media (max-width: 500px) {
    table tr th:nth-of-type(3), /* Cellules de moindre importance */
    table tr td:nth-of-type(3) {
        display: none;
    }
}

@media (max-width: 400px) {
    table tr th:nth-of-type(4), /* Cellules de moindre importance */
    table tr td:nth-of-type(4) {
        display: none;
    }
}
```

6. Briser les lignes trop longues

Une ligne de texte sans espace restera sur une seule ligne même si elle dépasse de son conteneur. Si l'écran est trop petit, cette ligne peut casser tout le design de votre site, forcez le retour à la ligne :

```
body { word-wrap: break-word; }
```

7. Préférez min-height à height

Si vous avez un bloc de texte dont vous réduisez la largeur, c'est sa hauteur qui s'agrandira pour contenir tout le texte. Il ne faut donc jamais donner une hauteur fixe à un bloc de texte en responsive-design, sous peine de voir le contenu sortir de son cadre. Utilisez plutôt une hauteur minimale : la hauteur sera alors au moins de cette valeur, mais augmentera pour tout contenir quand le texte aura subitement besoin de plusieurs lignes pour s'afficher.

[1] Voir le chapitre sur la gestion des images de fond en CSS3

8. Optimisez pour un smartphone

Préparer un site pour les Smartphones sera d'autant plus facile si vous avez déjà bien travaillé sur votre site et les démarches qualité propres aux sites internet : utiliser le **jeu de caractères UTF-8**, **réduction du poids des pages**, **CSS** et **images optimisées** (bonne utilisation des sélecteurs, décomposition logique des différentes parties, propriétés factorisées, etc.), **diminution des requêtes HTTP**, **séparation structure/présentation**, **ordre logique de votre structure**, etc.).
Surfer sur un Smartphone n'est pas seulement une question de résolution plus petite : **la connexion est moins performante et souvent plus onéreuse** que sur un PC connecté à un réseau.
Le flash est à oublier pour de nombreux Smartphones, notamment l'iPhone qui ne l'affiche tout simplement pas. Idem pour le **Java**.

9. Tirez parti des capacités tactiles (avec JavaScript)

Certains événements Javascript n'ont pas de sens sur un Smartphone : `Mouseover` et `Mousemove` ne réagissent pas car le mouvement du doigt sur l'écran fait défiler la page, `Doubleclick` ne se déclenche pas car tapoter deux fois déclenche en général un zoom.
En revanche, il est possible de faire plein de choses nouvelles sur un écran tactile, comme le « `swipe` » pour passer d'une page à une autre ou utiliser diverses actions utilisant le `multi-touch`. Pensez-y quand vous faites un site mobile !

Lecture Web : adapter un site à un smartphone
http://openweb.eu.org/articles/adapter_site_smartphones
http://www.w3.org/2007/02/mwbp_flip_cards.html.fr

12. LES TABLEAUX RESPONSIVES

Les tableaux HTML sont plutôt flexibles, mais attention, pour les petits supports - comme les smartphones et les tablettes, les tableaux deviennent souvent illisibles.
Il existe cependant quelques astuces pour pouvoir les consulter de manière productive même sur les plus petits formats.

1. Utiliser des pourcentages pour la largeur des colonnes

Les tableaux HTML sont plutôt flexibles, il est possible d'utiliser des pourcentages pour la largeur des colonnes, mais attention, le contenu en devient très vite illisible !

2. Masquer ce qui est moins important

Une première approche consisterait à cacher ce qui pourrait être considéré comme des colonnes "moins importantes" à l'aide de la propriété `display :none ;`
Vous pouvez également proposez un système de checkboxes à l'utilisateur pour qu'il puisse choisir lesquelles il souhaite afficher (nécessite l'ajout de javascript).

3. ré-agencer les tableaux

Réagencez si besoin les colonnes en lignes afin de favoriser une orientation verticale et présenter le plus de données possibles. Vous pouvez également découper les colonnes en ce qui ressemble à des listes avec des titres (technique utilisée pour le "reflow mode" de jQuery mobile).

4. Des tableaux qui défilent

Une autre approche est basée sur l'idée de tableaux qui défilent. Elle consiste à fixer une seule colonne à gauche, et laisser une barre de défilement sur une petite partie du tableau à droite, à l'aide de la propriété CSS overflow:auto, pour que l'utilisateur puisse afficher le reste des données.

Le gros problème avec la propriété CSS overflow:auto est que la barre de défilement créée n'est pas visible sur certains appareils mobiles et tablettes. L'utilisateur peut toujours faire défiler la partie de droite, mais il n'a aucune indication visuelle qu'il peut s'y cacher plus de contenu que ce qu'il voit. Il faut donc, si on utilise ces techniques, trouver un moyen d'indiquer visuellement à l'utilisateur qu'il peut faire défiler le contenu à droite.

Aller plus loin :

Plusieurs solutions détaillées :
http://www.web-eau.net/blog/tableaux-responsive-quelles-approches-pour-quelles-solutions
Sélection de data à cacher via une liste déroulante :
http://www.filamentgroup.com/lab/responsive-design-approach-for-complex-multicolumn-data-tables.html

13. Les formulaires

Plus long sera le formulaire, plus il sera compliqué à adapter pour des petits appareils… quelques

1. Adapter le viewport

L'utilisation de la Viewport Meta dans le <head> est primordiale pour éviter les formulaires beaucoup trop petits.

2. ré-agencer les formulaires

Placez les éléments les uns sous les autres en une seule colonne et élargir les champs pour qu'ils prennent toute la taille de l'écran.

3. Utiliser les attributs HTML5 "*type*"

Utilisez les attributs du HTML5 type="email", type="tel" ou type="url" : cela adaptera le type de clavier que l'écran tactile affichera : si c'est une adresse email, les boutons « @ » ou « .com » seront mis sur le clavier principal.

14. Les outils du RWD - Les Scripts

Internet Explorer, jusqu'à sa version 9, ne comprend absolument pas les media queries.

Respond.js

La librairie Respond.js est une librairie JavaScript téléchargeable qui permet de remplacer la gestion des media queries pour les navigateurs qui ne le supportent pas encore.
Télécharger le script : https://github.com/scottjehl/Respond
Placez le script dans un dossier JS dans lequel vous stockerez tout le JavaScript de votre site. Pour appeler le script dans votre code, entrez simplement cette ligne de code :

```
<!--[if lt IE 9]>
<script src="js/respond.js"></script>
<![endif]-->
```

Attention, uniquement `max-width` et `min-width` pourront être utilisées avec vos media queries (`orientation`, `device-width`, `resolution`, etc ...sont donc ignorées),

CSS3-MediaQueries-js

CSS3-MediaQueries-js permet une compatibilité des anciens navigateurs avec toutes les media queries. Il est recommandé pour les designs fluides complexes, devant être compatibles avec le plus grand nombre possible de configurations.

Cette flexibilité a toutefois un coût : son poids plus important (15 Ko) et sa relative lenteur à parser le CSS comparativement à Respond.js.
Son fonctionnement est identique à Respond.js.

ResponsiveSlides.js

Ce script va vous permettre d'inclure un slider 100% responsive dans votre site.
Très léger, il dispose de nombreuses options, comme les réglages de transition, le temps d'affichage de chaque image, le défilement manuel ou automatique, la possibilité d'inclure les images (ou photos) dans des liens ...

Téléchargez le script "ResponsiveSlides.js" : https://github.com/viljamis/ResponsiveSlides.js et placez le fichier "responsiveslides.min.js" dans votre dossier JS.

Dans le <head> de votre page, appelez ce fichier et la bibliothèque JQuery à l'aide de ces deux lignes de code :

```
<script
scr="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
<script scr="js/responsiveslider.min.js"></script>
```

Utilisez une liste non ordonnée, à laquelle vous ajouterez la class "rslides", pour intégrer les photos dans votre code HTML. Nous considérerons que les images sont stockées dans un dossier "images" :

```
<ul class="rslides">
<li><img scr="images/photo1.jpg" alt="" /><li>
<li><img scr="images/photo2.jpg" alt="" /><li>
<li><img scr="images/photo3.jpg" alt="" /><li>
<ul>
```

Dans votre feuille de style CSS, ajoutez ces quelques lignes de code :

```
.rslides { position:relative; list-style:none; overflow:hidden; width:100%;
padding:0; margin:0; }
.rslides li { position:absolute; display:none; width:100%; left:0; top:0; }
.rslides li:first-child { position:relative; display:block; float:left; }
.rslides img { display:block; height:auto; float:left; width:100%; border:0; }
```

Enfin, activez le slider en ajoutant cette ligne de code JavaScript avant votre balise </body> :

```
<script>
$(function(){
$(".rslides").responsiveSlides();
});
</script>
```

Vous pouvez personnaliser les différentes caractéristiques de fonctionnement du slider en ajoutant vos réglages[1]. Exemple :

```
<script> $(function(){
$(".rslides").responsiveSlides({
auto: true,
speed: 1000,
timeout: 4000
});
}); </script>
```

FitVids

Permet de rendre tous vos lecteurs vidéo fluides (YouTube et Vimeo entre autres).

[1] Vous pouvez retrouver tous les réglages disponibles sur la page officielle de ce script

Déposez le fichier "jquery.fidvids.js" (http://fitvidsjs.com/) dans votre dossier JS.
Pour fonctionner, FidVids.js à, lui aussi, besoin de la librairie JQuery. Appelez celle -ci ainsi que ce script en ajoutant ces lignes de code dans le <head> de votre page HTML :

```
<script
scr="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
<script scr="js/jquery.fitvids.js"></script>
```

La vidéo (dans cet exemple nous utiliserons une <iframe> récupérée sur YouTube) devra être placée dans un conteneur. Pour ce faire, nous utiliserons une avec l'id "video" :

```
<div id="video">
<iframe width="640" height="350" scr="url-de-la-video" frameborder="0"
allowfullscreen></iframe>
</div>
```

Pour terminer, ciblez le conteneur "video" pour lui appliquer FitVids et le rendre totalement fluide :

```
<script>
$(document).ready(function(){
$("#video").fitVids();
});
</script>
```

Lecture Web :
Mettre en place des medias queries dans son site :
http://www.alsacreations.com/xmedia/tuto/exemples/mediaqueries/layout.html
Comment organiser les pages de son site en fonction du média : http://mediaqueri.es/

15. LES FLEXBOX POUR UN DESIGN RESPONSIVE

Le module Flexbox Layout fournit une façon plus efficace de *disposer*, *aligner* et *distribuer l'espace* entre les items d'un container, même lorsque leurs dimensions sont inconnues et/ou dynamiques - d'où le terme "flex".

L'idée principale est de donner à un élément contenant (container) la possibilité de changer les largeurs et hauteur des éléments contenus (items), afin de remplir au mieux l'espace disponible, et *s'adapter à tous les devices et toutes les tailles d'écrans*. Un container flexible permet aux items de s'étendre pour occuper la place disponible ou au contraire les réduit pour leur éviter de déborder.

Un exemple concret ici :
http://jackintheflexbox.tumblr.com/post/105291132189/un-gabarit-simple-fluide-et-responsive-avec-un