Quelques ingrédients "responsive"

Contrairement à une feuille A4 aux dimensions définies et limitées, une page Web a des dimensions variables et élastiques. La variété des écrans, passant du 30 pouces pour un desktop au 5 pouces pour un smart-phone, complique le travail du développeur qui doit proposer un site lisible sur n'importe quel appareil.

Adapter un site nécessite de modifier le code CSS pour l'essentiel, avec quelques modifications sur le code HTML. Un peu de JavaScript est parfois nécessaire pour gérer les interactions.

Bases du site adaptatif

width

Avant de se lancer dans des notions compliquées, la première intention élastique consiste à mettre des pourcentages dans les valeurs des largeurs de boîtes plutôt que des valeurs fixes.

Par exemple, on peut écrire :

```
body { width: 80%; }
```

plutôt que

```
body { width: 1000px; }
```

Cette notion est applicable pour les images :

```
img { width: 100%; }
```

Remarque : si l'on utilise les *medias queries* (détaillées plus bas), il est possible d'utiliser des pixels comme unité de distance, tout en proposant des variations de présentations selon les écrans.

min-width et max-width

Une zone de texte, trop étirée ou rétrécie, atteint vite ses limites en confort de lecture. Il est possible de limiter l'élasticité avec les propriétés *min-width* et *max-width*.

```
body {  
  width: 70%;  
  min-width: 500px;  
  max-width: 1100px;  
}
```

Notez le mélange de pourcentages et de pixels qui permet de régler finement les dimensions de la page. L'astuce consiste à donner des dimensions globales en pourcentage puis à les limiter avec des pixels.

La propriété *max-width* peut être utilisée avantageusement pour les images pour éviter de les étirer au delà d'une résolution :

```
img {  
  width: 100%;  
  max-width: 800px;  
}
```

Images au format SVG

Il n'est pas simple de gérer la résolution d'une image sur une surface variable, le risque étant d'avoir des images trop étirées et floues. Je conseille, quand c'est possible, d'utiliser des images au format SVG. Ces images au format vectoriel ont l'avantage d'être réduite ou étirée à l'infini sans détérioration. Il suffit d'utiliser la propriété CSS *width*.

Ci dessous, un logo SVG de FireFox d'une largeur de 50% du conteneur, suivi d'une autre version à 25%. La même source d'image étant utilisée.

Tailles de polices en *em* plutôt qu'en *px*

Pour le confort d'utilisation, préférez toujours les *em* que les *px* pour définir les tailles des polices. Par défaut, un *em* mesure 16 pixels. Cependant, si l'appareil visualisant votre site a décidé qu'un *em* était de taille 18 pour améliorer le confort de lecture, votre site s'adaptera. Autre avantage, vous pourrez facilement changer les tailles des polices dans les *media queries* (voir ci-dessous) en appliquant un *font-size* pour une balise parente.

Les medias queries

Les medias queries permettent de fabriquer des mises en formes spécifiques en fonction des tailles d'écrans. Le développeur pourra définir des seuils pour son site au delà desquels la mise en forme s'adaptera.

Le code ci-dessous permet de faire une mise en forme spécifique pour les écrans dont la largeur est inférieure ou égale à 768 pixels. L'endroit idéal pour positionner ces medias queries est à la fin de votre CSS externe. Ainsi les médias queries appliqueront les réglages en amont, complétés ou contredits par les réglages des medias queries. Attention aux doubles accolades.

```
@media screen and (max-width: 768px) {  
  body {  
    width: 90%;  
  }  
}
```

On peut aussi détecter plusieurs seuils et appliquer les réglages en conséquence, le *font-size: 20px* permet d'augmenter la taille des *em* :

```
@media screen and (max-width: 1100px) {
```

```

body {
    margin: 10px;
    width: calc(100% - 20px);
}
}

@media screen and (max-width: 768px) {
    body {
        font-size: 20px;
    }
}

```

Il est également possible de détecter le mode portrait ou paysage d'un smartphone ou d'une tablette et d'y appliquer des mises en forme :

```
@media (max-width: 768px) and (orientation: portrait) { ... }
```

Le menu burger (de la mort)

Le menu burger est devenu un incontournable du responsive. Le principe est de faire apparaître le menu quand on en a besoin, puis de le faire disparaître lors de la consultation. Voici un petit morceau de code qui pourra vous être utile.

index.html

```

<!doctype html>
<html>

<head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta charset="utf-8">
    <title>Menu Burger</title>
    <link rel="stylesheet" href="style.css">
    <script src="burger.js"></script>
</head>

<body>
    <nav id="menu">
        <a href="index.html">Accueil</a>
        <a href="index.html">Menu 2</a>
        <a href="index.html">Menu 3</a>
        <a href="index.html">Menu 4</a>
    </nav>
    <div id="icone_burger">☰</div>
    <main>
        <h1>Le site vitrine</h1>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed non risus. Suspendisse lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat. Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent blandit odio eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit.</p>
    </main>
</body>

</html>

```

Ce code HTML très simple affiche un menu, un icône burger, un titre et un texte.

style.css

```
body {
  font-family: 'Arial';
  transition-property: translateX();
  transition-duration: 0.3s;
}

#icone_burger {
  display: none;
  cursor: pointer;
  position: absolute;
  top: 0px;
  right: 10px;
  font-size: 3em;
}

@media screen and (max-width: 768px) {
  nav {
    display: none;
    position: absolute;
    top: -20px;
    left: 100vw;
    width: 250px;
    height: 100vh;
    background-color: blueviolet;
  }

  nav a {
    display: block;
    color: white;
    padding: 1em;
  }

  #icone_burger {
    display: block;
  }
}
```

Le code CSS redessine le menu verticalement, avec un fond violet et sur la droite de la page. Il est masqué au départ grâce à la propriété *display: none*. Il apparaîtra suite au clic sur l'icône burger sur les écrans dont la largeur est inférieure ou égale à 768 pixels.

burger.js

```
window.onload = function () {
  document.getElementById("icone_burger").addEventListener('click', function () {
    document.getElementById("icone_burger").style.display = 'none';
    document.getElementById("menu").style.display = 'block';
    document.getElementsByTagName("body")[0].style.transform = 'translateX(-250px)';
  });
}
```

Ce code JavaScript guète le clic sur l'icône burger. En cas de clic, il fait disparaître l'icône, puis fait apparaître le menu en défilant vers la droite de 250 pixels. Le décalage se fait grâce à la méthode *style.transform* qui va appliquer la propriété CSS *translateX*. L'effet d'animation est géré par la propriété CSS *transition* écrite précédemment dans le code CSS.

Les mobiles dézooment

Pour parvenir à afficher des sites aux dimensions larges sur des petits écrans, les smartphones dézooment, ce qui réduit fortement la taille des textes. Cette option devient moins utile quand on fabrique un site responsive, car justement, nous venons de coder une version adaptée aux petites résolutions, alors autant profiter de textes lisibles sur smartphones.

Il faudra donc ajouter la balise suivante dans la zone d'entête pour désactiver l'action de dézoomer :

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Notez que cette balise est déjà présente dans le code HTML du burger ci-dessus.

Voici l'équivalent CSS pour désactiver l'action de dézoomer :

```
@viewport {  
  width: device-width;  
  zoom: 1;  
}
```

Si vous avez une feuille de style externe reliée à toutes vos pages HTML, il serait plus judicieux et plus moderne d'opter pour ce code CSS plutôt que d'ajouter une balise `<meta name= »viewport »>` sur chaque page.

Petit entraînement

Pour vous exercer, essayer de transformer **le site d'Hitchcock** en site responsive.