



**Développer une  
application  
web mobile  
en utilisant le  
Responsive web**



## Sommaire

<b>1</b>	<b><i>GENERALITES</i></b> .....	<b>5</b>
1.1	Développer une application native .....	5
1.2	Développer une application web mobile .....	6
1.3	Développer une application hybride .....	7
1.4	Les points clés du développement web mobile.....	7
<b>2</b>	<b><i>RESPONSIVE WEB DESIGN</i></b> .....	<b>9</b>
2.1	Généralités.....	9
2.2	Les principes de base .....	10

## Objectifs



Comprendre les enjeux du web mobile.  
Mettre en œuvre le « Responsive Design » avec HTML5 et CSS3.

## Pré requis

Les pré requis nécessaires afin de suivre avec profit cette formation sont la connaissance du développement web.

## Mode d'emploi

Pour chacun des thèmes abordés, ce support de formation :

- présente les points essentiels
- renvoie à des supports de cours, des livres ou à la documentation en ligne constructeur, par le symbole : 
- propose des exercices pratiques par le symbole : 

Autres symboles utilisés :



Point important qui mérite d'être souligné !



Manip à réaliser avec un utilitaire graphique (qui devrait normalement simplifier le travail !)



Approfondissement : travail de documentation ou exercice supplémentaire, à faire individuellement (non intégré dans le temps de formation moyen)

## Lectures conseillées

- Sites internet et tutoriaux sur le développement web mobile

## Outils de développement

# 1 GENERALITES

Les applications mobiles prennent de plus en plus de place dans le paysage numérique et les projets de développement mobiles explosent. Nous disposons actuellement de différents moyens pour véhiculer le contenu web sur des appareils connectés :

- développer une **application native**, qui se téléchargera et s'exécutera sur le device (Android, Iphone, etc...)
- ou développer une **WebApp**, c'est à dire une version mobile d'un site internet. Les WebApps sont très proches des applications natives de par leur ergonomie adaptée aux écrans tactiles. On peut développer différents sites dédiés, ou faire du « **responsive web design** ».
- on peut enfin développer des **applications hybrides**, qui permettront à partir d'un seul code, de compiler sur les différentes plateformes, ce qui permet d'avoir des **applications multi-plateformes**.

## 1.1 Développer une application native

Une application native est un produit développé spécifiquement dans divers "langages" (iOS, Android, WindowsPhone) et qui se télécharge et se référence au sein d'un "Store" (AppStore, Google Play, Windows Store).

Le développement natif consiste à développer avec les outils/langages propres à chaque système d'exploitation.

- Objective C avec l'IDE XCode pour iOS (sur Mac),
- Java avec ADT d'Eclipse pour Android,
- C# avec Visual Studio pour Windows Phone...

Le développement natif entraîne un certain nombre d'avantages :

- La prise en charge de fonctionnalités natives (touch, notifications, GPS, etc.), et l'accès à toutes les fonctions et capteurs de l'appareil nomade (accéléromètre, boussole, médias, téléphonie...),
- Une installation directe sur le périphérique et de bonnes performances,
- Une totale "acclimatation" au périphérique (ergonomie, performances, densité de pixels),
- On peut bénéficier des éléments propres à chaque système d'exploitation pour un développement plus rapide et plus propre. On peut créer ses propres composants en héritant des propriétés des composants natifs,
- La possibilité de 'vendre' l'application via l'AppStore ou l'Android Market.

Mais aussi certains inconvénients :

- Un développement spécifique pour chaque OS, et ce dans plusieurs langages (iOS, Android, WindowsPhone, etc.),
- En lien avec ces développements multiples : multiplication du temps de développement, et du coût de l'application, difficultés de maintenance, temps d'apprentissage important des différents langages et différentes plateformes,
- Un contenu non indexable par un moteur de recherche web classique,
- La difficulté de « mise en ligne » de l'application, et la perte de maîtrise qui en découle, liée au passage par les Stores (AppStore, Android Market...).

## 1.2 Développer une application web mobile

Il s'agit dans ce cas de développer une application web, qui sera accessible depuis différents appareils mobile. Ces applications, à la différence des applications web classiques, doivent être conçues et optimisées pour une utilisation multiple (depuis un écran d'ordinateur, une tablette, ou un smartphone....).

Une première possibilité consiste à développer différents sites dédiés en fonction du dispositif visé : un site principal, un site pour smartphones, un site pour tablettes, etc... Mais cette solution impose d'avoir un contenu dupliqué, ce qui entraîne le développement et la maintenance de plusieurs versions du site. C'est donc une solution à éviter dans la mesure du possible. Néanmoins si l'on utilise une telle solution, on pourra tester le device utilisé, en vérifiant le *userAgent* du navigateur. Par exemple pour tester si on est sur iPhone :

```
if((navigator.userAgent.match(/iPhone/i)) ||  
    (navigator.userAgent.match(/iPod/i))) {  
    ...  
}
```

Il ne vous reste plus ensuite qu'à rediriger l'utilisateur vers la version adéquate de votre site internet.

L'autre solution consiste à utiliser le « **Responsive Web Design (RWD)** », qui permet de développer des pages web qui s'adaptent automatiquement au support et à ses dimensions, à la fois en terme de 'grosceur' d'affichage, mais aussi en terme de contenu, et de structuration et de positionnement du contenu dans la page.

Ce type de développement possède un certain nombre d'avantages :

- un développement unique, ce qui entrainera des coûts et des délais généralement inférieurs aux techniques citées précédemment,
- une maintenance de projet facilitée (une seule feuille de style, un seul fichier HTML, etc.),
- une mise à jour transparente et un déploiement multiplateformes

Les inconvénients ne sont cependant pas nuls :

- des performances moindres que lors de développements natifs (notamment lorsque l'on traite de gros volumes de données),
- des accès aux API natives de la plateforme (accéléromètre, etc...) qui sont limités,

- une difficulté pour tester les applications : il est nécessaire de prévoir des tests nombreux et variés tout au long du projet,
- des limites ergonomiques et de performances des navigateurs web.

Lorsque l'on développe une application web mobile, deux méthodologies peuvent être adoptées, adaptées à deux cas de figure : soit l'adaptation d'un design existant (par exemple un site web classique existant, méthode rapide mais pas forcément très optimisée), soit la démarche dite "**mobile first**" (parfaite pour débiter de zéro, mais nécessitant une réflexion initiale plus poussée).

### 1.3 Développer une application hybride

Certaines entreprises ont très bien compris le problème de compatibilité et proposent des solutions qui permettent, avec un seul code, de compiler sur les différentes plateformes.

Il s'agit de développer une application web qui va générer différentes versions d'applications natives selon la cible visée. Ces applications natives contiennent souvent un navigateur (WebView) qui va intégrer le site web, et contiennent peu (ou pas) de code natif.

Les solutions les plus connues sont PhoneGap (projet Cordova) et Titanium, solutions basées sur du Javascript.



Néanmoins, ces solutions sont assez complexes à mettre en œuvre, et les applications natives générées ne sont pas optimisées en termes de performance.

### 1.4 Les points clés du développement web mobile

Développer une application web mobile ne consiste pas à effectuer un portage pur et simple, quasi systématique et sans imagination d'applications Web classiques.

Il faut bien sûr tenir compte des contraintes matérielles d'une part : un téléphone n'offre pas les mêmes possibilités qu'un ordinateur traditionnel en terme de fonctionnalité, les performances ne sont pas les mêmes puisque même les offres mobiles les plus rapides ont des bandes passantes très faibles

Mais il faut aussi tenir compte de l'évolution des différents usages qui sont faits du mobile et intégrer le comportement de l'utilisateur, « **l'expérience utilisateur** », lors de la conception de nos applications mobiles.

- **La communication**

Le téléphone est avant tout un outil de communication. L'arrivée des forfaits à un prix abordable et la généralisation du haut débit pour les mobiles ont permis de diversifier les modes de communication à partir d'un téléphone mobile. À la voix et au SMS se sont ajoutés la messagerie instantanée, la communication via l'image, etc....

Votre application mobile devra refléter cet aspect communicant et cet aspect « social », par exemple en proposant des portails vers les principaux services de messagerie et les principaux réseaux sociaux du marché. Les fonctionnalités choisies et le mode opératoire devront respecter les pré-requis à toute application mobile : faible débit possible, taille limitée de l'écran, absence de dispositif de pointage traditionnel.

- **La mobilité**

À l'inverse d'un ordinateur même portable, le propre d'un terminal mobile est la mobilité. Vouloir porter une application faite pour un ordinateur traditionnel sans vraie réflexion mobile est voué à l'échec. Les contraintes d'une application mobile en termes d'ergonomie, de poids, mais également de fonctionnalités ne sont pas les mêmes que pour une application desktop. Vos besoins ne sont pas les mêmes selon que vous soyez assis à votre bureau ou quelque part dans la rue.

Vous devrez donc tirer profit des fonctionnalités spécifiques des terminaux mobiles. Parmi celles-ci, la géo localisation va prendre de plus en plus de place à mesure que les appareils s'équipent en GPS, notamment pour proposer des services et des résultats géographiquement proches.

- **La simplicité**

Même si les terminaux mobiles ont fait beaucoup de progrès en termes d'utilisabilité et d'ergonomie, l'utilisation n'en est pas toujours aisée. La taille du clavier et de l'écran ne sont pas les seuls freins à une utilisation efficace des applications.

Les premières pistes de réflexion dans la création d'une application Web mobile doivent s'orienter vers la simplification :

- simplification fonctionnelle, en supprimant tout ce qui n'intéresse pas directement les utilisateurs mobiles, sans pour autant dénaturer l'application,
- simplification de l'interface, afin de rendre plus facilement accessibles les éléments importants.
- simplification de la navigation utilisateur, afin notamment de rendre accessible l'ensemble des fonctionnalités depuis le premier ou le second écran.
- 

- **La personnalisation**

Les terminaux mobiles sont des objets très personnels. Il est bon d'offrir aux utilisateurs de votre application mobile une personnalisation maximum, sans toutefois perdre en utilisabilité. Cette personnalisation peut concerner le look de l'application, via des couleurs ou des skins personnalisés, mais aussi le profil de l'utilisateur.

- **La légèreté**

Même si les points d'accès Wi-Fi et les forfaits mobiles haut débit ont tendance à se populariser, les premiers ne sont pas accessibles partout, et la couverture des seconds n'est pas universelle.

Il faut concevoir vos applications Web mobiles en prenant en compte les problèmes de performance. Pour cela, il est utile de

- redimensionner vos images,
- n'embarquer que le Javascript strictement nécessaire,
- éventuellement charger les contenus de manière progressive.



## 2 RESPONSIVE WEB DESIGN

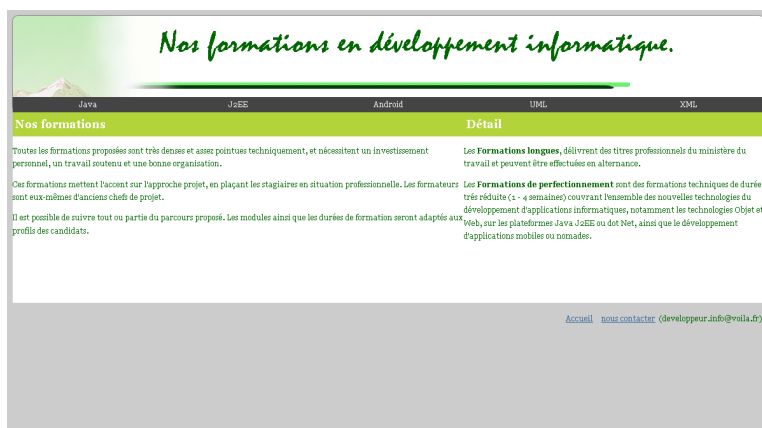
De nos jours on consulte de moins en moins le web sur un écran d'ordinateur. Les supports ont tendance à être multiples (écrans, tablettes, smartphones, et bientôt téléviseur, etc.....) ce qui oblige les designers web à prendre en considération de nombreuses contraintes, et notamment les différentes dimensions d'écran de tous ces supports.

### 2.1 Généralités

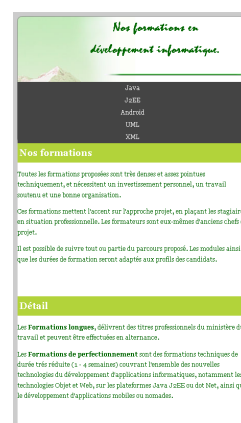
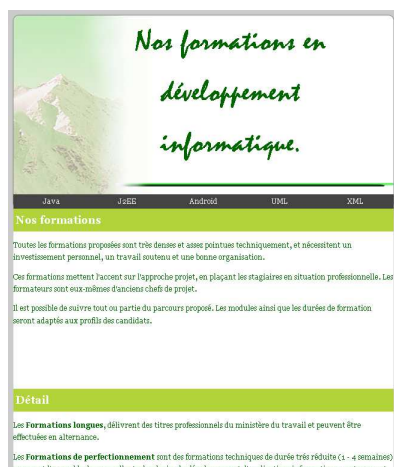
Le **Responsive Web Design** est une technique de conception de sites web qui regroupe différents principes et technologies permettant à un site d'offrir au visiteur un accès optimal au site en terme de lecture et de navigation, et ce quel que soit le support physique de lecture (ordinateur, smartphones, tablettes, téléviseur, etc.). Ceci doit se faire en évitant le défilement horizontal ou le zoom (sur les appareils tactiles notamment).

Il faut donc développer des pages web qui s'adaptent automatiquement au support et à ses dimensions, à la fois en terme de 'grosseur' d'affichage, mais aussi en terme de contenu et de structuration et de positionnement du contenu dans la page.

Par exemple, la même page pourra s'afficher comme suit en fonction des supports :



mode tablette



mode smartphone

mode smartphone large ou notebook



**Taille des écrans** : en général, on peut estimer, en mode portrait, que :

- une largeur d'écran de moins de 480px correspond à un smartphone ;
- une largeur d'écran entre 480px et 1024px correspond à une tablette ;
- une largeur d'écran supérieure à 1024px correspond à un ordinateur de bureau ou un portable.

## 2.2 Les principes de base

Pour développer une application Web qui soit **Responsive Web Design**, nous pouvons nous appuyer sur les trois points ci-dessous :

- Un ensemble de règles CSS3 basé sur **Media Queries**, qui vont servir à afficher/masquer certains éléments de nos pages, voire changer le rendu de notre application Web, en fonction du type de support utilisé et/ou des dimensions de l'écran.
- Le concept de « **grille fluide** » ou grille d'affichage flexible, qui permet un redimensionnement relatif des différents blocs de la page. On utilisera pour cela des unités relatives comme les pourcentages ou les **ems** en lieu et place des unités fixes comme les pixels ou les points.
- Des **médias flexibles** (et notamment des images), afin de prévenir un éventuel débordement du contenu en dehors du cadre de notre grille d'affichage.

### Les Media Queries CSS3

Les **Media Queries CSS3** sont destinées à simplifier la création de pages web pour les rendre consultables sur des supports variés (tablette, portable...). On va pouvoir définir des styles différents selon différents types de Media Queries, en ciblant notamment :

- la résolution,
- le type de media,
- la taille de la fenêtre,
- la taille de l'écran,
- le nombre de couleurs,
- le ratio de la fenêtre (par exemple le 16/9).

Une **media query** est une expression dont la valeur est soit vraie, soit fausse. Elle peut contenir les opérateurs logiques **and**, **only**, **not**. Pour obtenir l'équivalent du **or**, 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.

Le but des media queries est de cibler les périphériques de destination en fonction de leurs capacités. Pour cela, on peut indiquer quel est le type de média visé par l'intermédiaire des attributs suivants :

<b>screen</b> :	Écrans
<b>handheld</b> :	Périphériques mobiles ou de petite taille
<b>print</b> :	Impression
<b>aural / speech</b> :	Synthèses vocales
<b>braille</b> :	Plages braille

<b>embossed :</b>	Imprimantes braille
<b>projection :</b>	Projecteurs (ou présentations avec slides)
<b>tty :</b>	Terminal/police à pas fixe
<b>tv :</b>	Téléviseur

On peut aussi (ou en plus) indiquer la règle (l'expression) pour laquelle on va appliquer la feuille de style. Cette expression est entre parenthèses, et peut être basée sur les principaux critères suivants :

<b>device-height :</b>	dimension en hauteur du périphérique
<b>device-width :</b>	dimension en largeur du périphérique
<b>grid :</b>	périphérique bitmap ou grille (ex : lcd)
<b>height :</b>	dimension en hauteur de la zone d'affichage
<b>width :</b>	dimension en largeur de la zone d'affichage
<b>monochrome :</b>	périphérique monochrome ou niveaux de gris (bits/pixel)
<b>orientation :</b>	orientation du périphérique (portrait ou landscape)
<b>resolution :</b>	résolution du périphérique
<b>scan :</b>	type de balayage des téléviseurs (progressive ou interlace)

...

La plupart de ces critères peuvent être préfixés par **min-** ou **max-** (s'ils sont numériques), pour définir des valeurs minimales ou maximales à respecter.

Voici un exemple dans du code **CSS** :

```
@media screen and (max-width: 767px)
{
    ...
}
```

Ici l'expression est (**max-width: 767px**), elle se comprend facilement, elle est "vraie" si le device sur lequel s'affiche la page a une largeur d'affichage inférieure ou égale à 767px.

Autre exemple : si on veut appliquer des règles lorsque l'orientation est "portrait" voici la syntaxe à utiliser :

```
@media (orientation: portrait) { ... }
```

L'objectif étant de faire des sites web parfaitement adaptés à des périphériques comme des smartphones par exemple, il est de bonne pratique de désactiver les fonctions de zoom (que l'on utilise systématiquement lorsque l'on accède à un site web 'classique' depuis un smartphone...). Ceci se fera en rajoutant la balise meta suivante :

```
<meta name="viewport" content="user-scalable=no,initial-scale=1,
    minimum-scale=1, maximum-scale=1, width=device-width">
```

On bloque le zoom : **user-scalable=no**

On fixe le zoom minimum : **minimum-scale = 1**

On fixe le zoom maximum : **maximum-scale = 1**

On ouvre la fenêtre à la largeur de l'écran : **width=device-width**

## Le concept de « grille fluide »

La grille d'affichage de notre site, c'est-à-dire la partie de l'écran dans laquelle les données vont s'afficher, doit être flexible. Pour cela, nous n'utilisons en aucun cas de largeur ou hauteur fixe, mais nous allons utiliser des pourcentages.

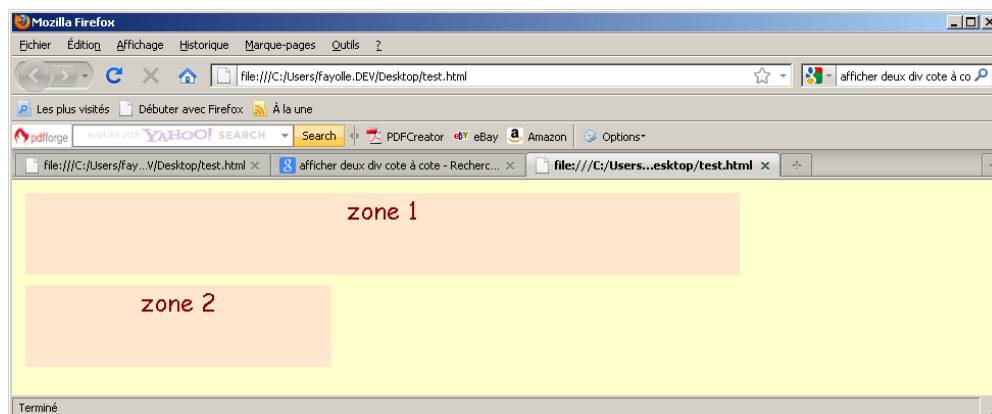
Notre 'grille' est la suivante :

```
<body>
  <div class="div">
    <div class="zone1">zone 1</div>
    <div class="zone2">zone 2</div>
  </div>
</body>
```

Dans un premier temps, les largeurs sont fixes : si la largeur de la fenêtre est plus grande que celle d'une ligne, on constate juste l'apparition d'une marge à droite

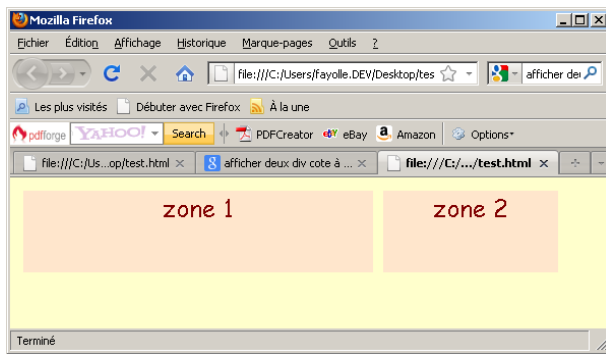


Lorsque l'on réduit la largeur de la fenêtre, les colonnes ayant des largeurs fixes, le deuxième élément passe sous le premier puisqu'il n'a plus suffisamment de place



Lorsque l'on utilise une grille fluide, on a une occupation totale du contenant, c'est-à-dire en utilisant des pourcentages :

```
div :      width: 100%;
.zone1 :   width: 60%;
.zone2 :   width: 30%;
```



Cette fois les éléments occupent toute la largeur de la fenêtre dans tous les cas étant donné qu'ils sont affectés en pourcentage.



Attention : il faut aussi modifier la taille des polices pour permettre le redimensionnement du texte. Nous allons faire en sorte que les polices aussi soient exprimées en pourcentage (en utilisant ce que nous appelons les « em », où 1em = 100 %). La valeur demandée, en *em*, est un coefficient multiplicateur. Concrètement, un paragraphe dont la taille du texte serait 2em aura un texte deux fois plus haut que le texte de l'élément qui contient ce paragraphe.

## Les médias flexibles

Il faut enfin s'assurer que les différents médias (les images, les vidéos, ...) utilisés dans notre site Web soit Responsive Design, pour éviter par exemple d'avoir des images qui débordent de leur cadre.

Afin de résoudre ce problème, nous allons utiliser la propriété CSS « **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%;
    height: auto;
}
```

Il existe aussi des techniques pour avoir des images avec une très petite taille de fichier, mais qui puissent s'afficher correctement, sans effet de flou ou de créneaux de pixels.

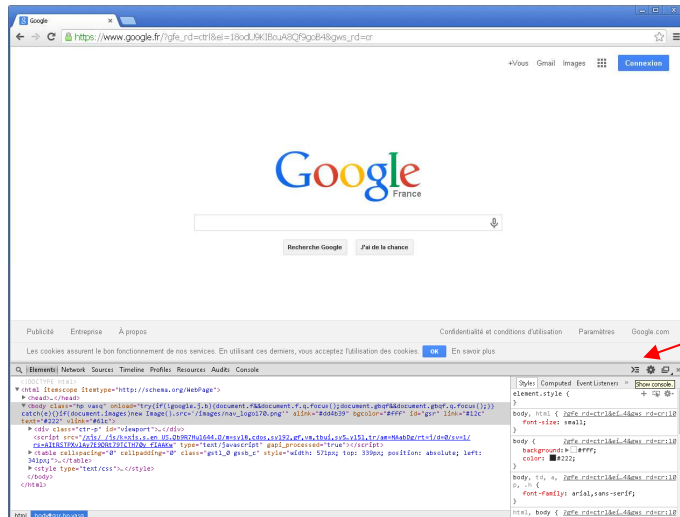
Le principe est très simple : si nous avons une image avec une très grande résolution et un très fort taux de compression, sa taille sera plus petite qu'une même image deux fois plus petite avec une compression normale.

## Tester les pages « Responsive »

Pour mettre en œuvre les tests d'un site responsive, nous pouvons simplement tester notre responsive design directement dans notre navigateur (Chrome ou Firefox), sans installer d'extension ou d'avoir à passer par un autre site, et qui fonctionne même si l'on travaille en local.

### Sous Chrome :

Ouvrir la console développeur (F12 sous Windows) et cliquer sur l'icône *show console* comme suit :



Puis cliquer sur l'onglet **Emulation** et choisir le device voulu dans la liste

### Sous Firefox :

Il faut appuyer sur Alt, pour afficher le menu, puis aller dans **Outils > Développeur web** et sélectionner **Vue Adaptative**. On peut ensuite sélectionner dans la liste déroulante la résolution voulue :

