M1105 - Cours n°7

Responsive Web Design (RWD)
*pour des **sites web adaptatifs***

Introduction à l'accessibilité des sites web
*pour des **sites web accessibles***

IUT 2 Grenoble Université Pierre-Mendès-France Informatique

1

Partie n°1
Responsive Web Design (RWD)
(Site Web Adaptatif)

IUT 2 Grenoble Université Pierre-Mendès-France Informatique

2

Introduction

Le problème:

- des terminaux de consultation variés: écrans, tablettes, smartphones,… de différentes tailles et résolution
- des techniques de navigation variées: souris, clavier, tactile

Comment adapter un site web aux différents dispositifs?

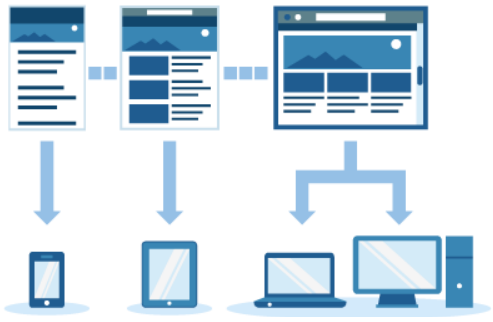
Comment concevoir une seule interface adaptable et que le parcours du site reste ergonomique pour l'utilisateur ?

Faire un ***design adaptatif (responsive)*** qui s'adapte aux différents terminaux:

=> Utilisation de la technologie "**CSS3 media queries**"

3

Adaptation à différents terminaux de consultation



http://www.lije-creative.com/bases-apprendre-responsive-web-design/

Exemple de site web adaptatif: http://getbootstrap.com/

4

Design adaptatif (responsive)

- **Principe n°1 :**
  - Avoir un design fluide
  - Avoir des dimensionnements relatifs
    - Utilisation des unités relatives (em ou %) ; pas d'unités absolues (px ou pt)
  - Les images ont également besoin de flexibilité
    => également redimensionnées en unité relative
    (en css en utilisant par exemple width:100% ou bien max-width: 100% ; height: auto)

- **Principe n°2:**
  - utilisation de **règles CSS « media queries »** différentes en fonction du terminal

5

Attention! les **2 résolutions** + le "**viewport**"

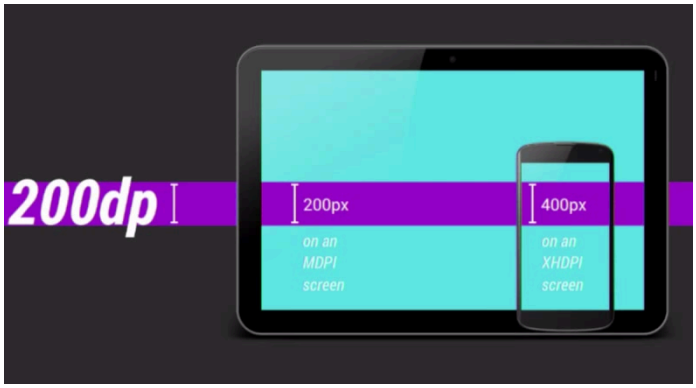
① il existe 2 « résolutions » pour un terminal :

- **La résolution native/physique** en « pixels (**px**) »:
  ⇒ c'est le nombre total de pixels tel que le constructeur décrit le terminal
  ⇒ le constructeur parle aussi de **densité** de pixels:
  nb de pixels en largeur (hauteur) / la largeur(hauteur) de l'écran en inch
- **La résolution virtuelle** : en « **density-independent** pixels (**dp**) »:
  la résolution sur laquelle un terminal se base pour l'affichage
  ⇒ définit ce qu'on appelle **la "surface virtuelle" du terminal** (device-width et device-height)

Rapports standards résolution physique/résolution virtuelle: 1x 1.5x 2x 3x :

- 240px => 240dp
- 360px => 240dp
- …

6



⇒ Les "dp" permettent d'afficher des éléments de manière uniforme sur des écrans dont la densité des pixels est différente

https://www.youtube.com/watch?v=zhszwkcay2A

7

Remarques

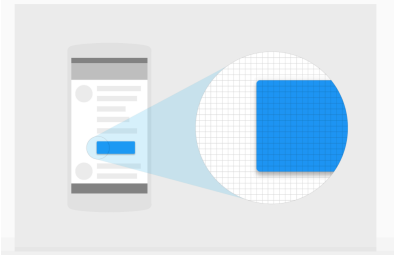
- La résolution fait référence au nb total de pixels de l'écran
- La densité de l'écran : nb de pixels en largeur (hauteur) / la largeur(hauteur) de l'écran en inch
- 1dp est égal à 1px pour un écran dont la densité est de 160.
- Pour calculer dp = (largeur en px*160) / densité

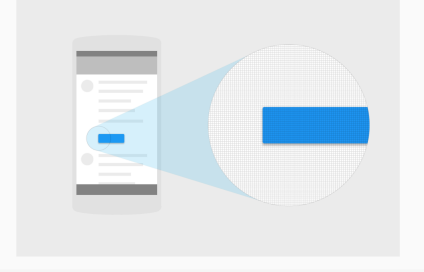
- En CSS, quand on utilise l'unité px (non recommandé d'ailleurs!), cela représente en fait des dp ! ☹
- Les dp sont explicitement utilisés quand on développe en natif Androïd

8

Low / High density screen

La densité de l'écran : nb de pixels en largeur (hauteur) / la largeur(hauteur) de l'écran en inch





9

https://material.io/guidelines/layout/units-measurements.html#units-measurements-pixel-density

A dp is equal to one physical pixel on a screen with a density of 160. To calculate dp:

dp = (width in pixels * 160) / screen density

When writing CSS, use px wherever dp or sp is stated. Dp only needs to be used in developing for Android.

Screen density	Screen width in pixels	Screen width in density-independent pixels
120	180 px	240 dp
160	240 px	
240	360 px	

If you have three screens, all 1.5 inches wide, with varying screen densities, the screen width will still be 240dp for all of them.

10

Attention! les **2 résolutions** + le "**viewport**"

② Et s'ajoute le **viewport** !!
- en « pixels CSS (**pxCSS**) », qui correspond à **la surface intérieure de la fenêtre du navigateur**.

Mais !!

- sur un terminal de bureau, la surface du viewport est égale à la surface virtuelle (définie par la résolution virtuelle)
- ☹ **sur un terminal mobile, la surface du viewport n'est pas égale à la surface virtuelle**
  ⇒ sur un terminal mobile la surface du viewport est bien plus grande !!

11

Quelques exemples

Résolution native/physique:	**Résolution virtuelle:**
iphone 3: 320px * 480px	320dp * 480dp
iPhone 4: 640px * 960px	320dp * 480dp
iPhone 5: 640px * 1136px	320dp * 568dp
iPhone 5S: 640px * 1136px	
iPad 2: 768px * 1024px	768dp * 1024dp
iPad3 : 1536px * 2048px	768dp *1024dp

12

Quelques exemples

Résolution native/physique:		**Résolution virtuelle:**
Galaxy S	480 x 800	320dp x 533dp
Galaxy S II	480 x 800	320dp x 533dp
Galaxy S3	720 x 1280	
Galaxy S4	1080 x 1920	
Galaxy Tab	1024 x 600	
Galaxy Tab 7.7/10.1	1280 x 800	
Galaxy Note 1	1280 x 800	
Galaxy Note 2	1280 x 720	
Galaxy Note 3	1920 x 1800	
Galaxy Note 10.1	1280 x 800	

13

Et on ajoute le viewport!

- Le viewport correspond à la surface intérieure de la fenêtre du navigateur (en pxCSS). Il ne dépend donc pas du terminal, mais du navigateur mobile.
- Le viewport est bien plus grand que les surfaces des terminaux afin de pouvoir y "caler" la plupart des pages web "classiques" en lui affectant un niveau de (dé)zoom.

Quelques exemples de valeurs par défaut :

- Android 1, 2 et 3 : 800 pxCSS
- Android 4 : 980 pxCSS
- Opera mini : 850 pxCSS
- Opera mobile : 980 pxCSS
- Safari mobile : 980 pxCSS
- Internet Explorer mobile : 1024 pxCSS

14

Ainsi… sur iPhone 4 par exemple:

- la résolution native est de **640px** de large
- la résolution virtuelle est de **320dp** de large
- le navigateur "safari mobile" va afficher les pages web dans une fenêtre de **980 pxCSS** de large

⇒un site statique de largeur "classique" de 980px va ainsi occuper la largeur de l'écran !!!

⇒(dé)zoom important qui rend le contenu parfois difficilement lisible

15

Que faire ?

1. "Contrôler" la largeur de la fenêtre du navigateur (le viewport) et s'affranchir du comportement par défaut :
   ⇒balise meta

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

- width: adapte le "width" (largeur) de la fenêtre du navigateur (= le viewport) à la largeur du "device" c'est-à-dire la largeur du terminal exprimée en dp)
- initial-scale: fixe le niveau de zoom de la page (1 = 100%) pour s'affranchir du zoom par défaut rendant les contenus illisibles

16

Que faire?

2. Utiliser des règles « **CSS3 media queries** »: règles d'affichage basées sur les résolutions moyennes

⇒ Définition de « **points de rupture** » pour différentes largeurs de la fenêtre du navigateur ***au delà*** ou ***en deçà*** desquels les styles sont changés

17

Exemple de fichier .css

```
/* styles pour toutes les largeurs */
Règles CSS


/* styles pour des largeurs de fenêtre de navigateur  "moyennes*/
@media screen and (max-width:768px) {
Règles CSS
}

/* styles pour des largeurs de fenêtre de navigateurs "petites"  */
@media screen and (max-width : 480px) {
Règles CSS
}
```

18

Que faire ?

3. Adopter une méthodologie selon que le site existe déjà ou est à créer:

http://www.alsacreations.com/article/lire/1464-web-mobile-introduction-et-glossaire.html

1. **Méthodologie n°1** : Adapter un site existant (dégradation progressive)
2. **Méthodologie n°2** : La démarche "mobile-first" (amélioration progressive)

19

Méthodologie n°1: Adapter un site existant

Beaucoup de choses à faire et pas toujours facile…

1. Fixer le viewport du terminal (<meta name="viewport"…>)
2. Identifier les points de rupture
3. Faire un choix de navigation mobile (en haut, en bas, select, déroulant, masqué, etc.)
4. Appliquer des styles mobiles via media queries
5. Traiter un par un tous les éléments possédant des largeurs problématiques (width, min-width, height, min-height, margin, padding)

20

Méthodologie n°1: Adapter un site existant

7. Réintégrer les éléments dans le flux pour obtenir des blocs verticaux
8. Adapter les contenus et les médias pour éviter des débordements
9. Opter pour des gabarits de largeur fluide
10. Prendre en compte les écrans HD (rétina) et les performances web mobile (wifi, 3G,…) (pb du temps de chargement d'une page)

http://wdfriday.com/blog/2012/04/introduction-a-la-performance-pour-le-web-mobile/

Rmq: penser à supprimer les pseudo-éléments :hover pour les mobiles et tablettes car ils ne sont pas visibles.

21

Méthodologie n°2: La démarche "mobile-first"

Cette démarche est préconisée car elle permet de ne pas tomber dans le pièges des images inadaptées, des temps de chargements longs, des éléments superflus…

http://wdfriday.com/blog/2012/03/css-et-mobile-first-proceder-par-amelioration-progressive/

1. Faire plusieurs maquettes du site pour les différents types de terminaux: smartphones, tablettes, écrans de portables, écrans de bureau

   ⇒ pour représenter les différentes éléments du site: navigation, sidebar, header, footer, menu déroulant, slideshow, animation, icônes, boutons, images,…

22

Méthodologie n°2: La démarche "mobile-first"

2. Grâce aux « media queries », ajouter à chaque point de rupture des éléments qui ajoutent à l'expérience utilisateur: réorganisation des éléments, affichage de nouveaux contenus, de nouvelles colonnes…

Comment tester ?

Outils développeur web de Firefox

(Des alternatives?)

- Développer une application spécifique à une plateforme mobile (une application native) dans un langage spécifique à chaque plateforme :
  - Objective-C/Swift pour iOS
  - Java pour Android
  - Langages de la gamme .NET pour Windows Phone

  ⇒ Coût de développement, de maintenance

- WebApp (Application mobile Web): une application web mobile hybride (code HTML, CSS, JavaScript et accès au matériel), utilisation du navigateur web de l'appareil pour l'interpréter

  ⇒ La webApp est perçue par les utilisateurs comme une application native

  ⇒ Pas aussi performant qu'une application native

  ⇒ Voir Adobe PhoneGap framework

25

Partie n°2

Introduction à l'accessibilité des sites web

26

L'accessibilité : un droit universel

L'accessibilité du web est un droit universel, selon l'article 9 de la Convention relative aux droits des personnes handicapées adoptée en 2006 par l'Organisation des Nations Unies

En France:

La loi n° 2005-102 sur l'égalité des droits et des chances, la participation et la citoyenneté des personnes handicapées, promulguée le 11 février 2005 stipule que les informations diffusées sur les services de communication publique en ligne doivent être accessibles aux personnes handicapées

27

Accessibilité

«La nature du web est son universalité. Il doit être accessible à toutes les personnes handicapées »

Tim Berners Lee, Directeur du W3C et créateur du Web.

Web Accessibillity Initiative (WAI) du W3C (1997)

http://www.w3.org/WAI/

Les recommandations (Directives pour l'accessibilité aux contenus Web) => Web Content Accessibility Guidelines (**WCAG**)

En 2008 : **WCAG 2.0**
http://www.w3.org/TR/WCAG20/

Traduction : http://www.braillenet.org

28

En France : *RGAA*

- RGAA = Référentiel Général d'Accessibilité pour les Administrations (http://www.references.modernisation.gouv.fr/rgaa-accessibilite)
  Les règles du référentiel RGAA reposent sur le WCAG 2.0

- Calendrier RGAA, décret et arrêté :
  - 14 mai 2009 (publié au JO du 16 mai 2009) : Publication du décret sur l'accessibilité des sites publics, créant le référentiel d'accessibilité des services de communication publique en ligne donne obligation aux collectivités locales de se mettre en conformité avec ce référentiel d'ici trois ans, **soit avant le 16 mai 2012 (un délai réduit à deux ans pour l'Etat).**

  - 29 octobre 2009 : Publication de l'arrêté relatif au RGAA au Journal Officiel (JORF n° 0251 du 29 octobre 2009 page 18329 texte n° 31)

29

Pour donner une idée des difficultés…

Une séquence vidéo mettant « en situation » une utilisatrice handicapée (Université de Nice).

La parole à Isabelle Krempp:

30

II - WACG 2.0 : 12 directives d'accessibilité

12 directives structurantes selon 4 principes fondamentaux :
Perceptible, Utilisable, Compréhensible, Robuste

Principe 1 : Perceptible

L'information et les composants de l'interface utilisateur doivent être présentés à l'utilisateur de façon à ce qu'il puisse les percevoir

Principe 2 : Utilisable

Les composants de l'interface utilisateur et de navigation doivent être utilisables

Principe 3 : Compréhensible

Les informations et l'utilisation de l'interface utilisateur doivent être compréhensibles

Principe 4 : Robuste

Le contenu doit être suffisamment robuste pour être interprété de manière fiable par une large variété d'agents utilisateurs, y compris les technologies d'assistance

31

3 niveaux de conformité d'un document

Chaque principe regroupe une série de règles. Chaque règle est détaillée sous la forme de "critères de succès" reliés à des tests unitaires. Chaque critère est d'un "niveau" de conformité WCAG (A, AA, AAA)

3 niveaux de conformité :

- Niveau de conformité " A " (niveau minimum) : critères de succès essentiels pouvant raisonnablement s'appliquer à toutes les ressources Web.
- Niveau de conformité " AA " : critères de succès supplémentaires pouvant raisonnablement s'appliquer à toutes les ressources Web.
- Niveau de conformité " AAA " : critères de succès ne s'appliquant pas à toutes les ressources Web.

32

IV - Validation

Valider l'accessibilité

- avec des outils automatiques
- la vérification humaine.

Remarques :

- Les méthodes de validation doivent être utilisées dès le début du développement. Plus les problèmes d'accessibilités potentiels sont identifiés tôt, plus il sera facile de les corriger ou de les éviter.
- La vérification réalisée par un humain complète la validation automatique (pour assurer la clarté du langage et la facilité de la navigation par exemple)

33

Quelques outils

- Logiciel de lecture d'écran (par exemple VoiceOver d'Apple)
- Ajout d'une extension pour l'accessibilité dans les outils de développement des navigateurs (**wave** par exemple)
- Outils de validation en ligne:
  https://wave.webaim.org/
  http://www.508checker.com/
  …
- Cercle chromatique Accessible :
  http://gmazzocato.altervista.org/fr/colorwheel/wheel.php

⇒ documentation:
https://developer.mozilla.org/fr/Apprendre/a11y
https://developer.mozilla.org/fr/docs/Accessibilité

34

M1105… c'est fini

Pour continuer :

- Référencement
- Gestionnaires de contenu (Content Management System - CMS)
  - WordPress, Drupal, Joomla, CMS Made Simple, Plume CMS…
- Framework CSS/javascript et pour webApp :
  - Bootstrap, Materialize,… PhoneGap, Ionic

Rendez-vous aux semestres 3 et 4 :

- programmation côté serveur (PHP 5)
- programmation côté client (javascript / jquery)

35