Cours HTML5 / CSS3

Pré-requis

Installer sur sa machine (pour ne pas perdre de temps en cours avec le débug) :

- node.js :
  - https://nodejs.org/en/
  - avec Homebrew : `brew install node`
  - > checker avec la commande : `npm -v`
- installer git (mais ça doit déjà être fait) :
  - https://git-scm.com/downloads
  - avec Homebrew : `brew install git`
  - > pour checker : `git --version`
- installer ruby (ça doit déjà être fait aussi) :
  - depuis le site : https://www.ruby-lang.org/fr/documentation/installation/
  - ou avec Homebrew : `brew install ruby`
  - > pour checker : `ruby -v`
- installez Emmet dans votre IDE : http://emmet.io/download/
- installer Virtual Box (https://www.virtualbox.org/) et une VM (https://dev.windows.com/en-us/microsoft-edge/tools/vms/mac/)

Avoir déjà un peu vu :

- les bases de la syntaxe de HTML et de CSS : http://learn.shayhowe.com/html-css/building-your-first-web-page/
- le fonctionnement des sélecteurs CSS : http://flukeout.github.io/
- les bases de la syntaxe SASS : http://sass-lang.com/guide

Savoir se servir des DevTools, selon votre navigateur préféré :

- Chrome :
  - sur CodeSchool : http://discover-devtools.codeschool.com/
  - manuel officiel : https://developers.google.com/web/tools/chrome-devtools/?hl=en
- Firefox Developer :
  - https://www.mozilla.org/fr/firefox/developer/
  - https://developer.mozilla.org/fr/docs/Outils

Planning

1/2j ou moins : HTML
1/2j ou plus : CSS
1/2j : Responsive

1/2j : industralisation
1h d'explications / questions sur l'évaluation à la fin

Intro

On va parler :

- HTML5 et CSS : sémantique
- bonnes pratiques : accessibilité, syntaxe, validation
- responsive : contenus fluides, breakpoints
- compatibilité multi-navigateurs
- performance : outils d'audit, librairies
- industrialisation : outils d'industrialisation

La base

Qu'est ce que c'est ?

HTML - Hypertext Markup Language
Semantique / structuration de contenu
HTML5 / 2014
Nouveaux elements de contenu (header, main, article…), de media (video, audio, figure) et de from (input tel, email, range…)

CSS - Cascading Style Sheets
=> présentation de document
Style
Cascading parce que notion d'héritage
CSS 2 et ses évolutions,

HTML5

Pge HTML minimale

- doctype : grammaire de la page
- head
- body

Elements supplémentaires :

- css dans le head
- js à la fin du body
- favicon et autres icones

? Quelles sont les différentes balises ?

Balises structurantes principales :

- structure de document : header, footer, main, nav...
- niveaux de titre : h1 -> h6
- formatage de texte, de listes, de tableaux
- insertion de medias (images, videos…)
- elements de formulaire
- => éviter quand c'est pertinent des divs et les span ("divite")

-> MEMO HTML à donner (pdf et odt)
=> http://html5doctor.com/ (HTML5 Element Index en bas de la home)

Exemples de sites :

- paris web
- a list apart

Contre exemple :

- https://stgsocgen.taleo.net/careersection/sgcareers/moresearch.ftl?lang=en

Travail

- Console ouverte
- Pas de cache dans le navigateur
- Valider son code

EXO : bout de maquette à intégrer
http://perrinebocquin.fr/2015/lpdw/maquettes.zip
http://perrinebocquin.fr/2015/lpdw/MemoHTML.pdf

Bonnes pratiques

Liste de bonnes pratiques sur Opquast : https://checklists.opquast.com/fr/

Semantique permet de mieux faire comprendre le contenu au machines, pour :

- optimisation SEO, mais se joue aussi sur plein d'autres critères // demander quoi ?
  - inforgraphie : http://www.axenet.fr/img/le-seo-en-une-image-1000px.jpg (infographie extraite de l'article : http://www.axenet.fr/infographie-seo/)
  - opquast SEO : https://checklists.opquast.com/seo/
- scan de la page par machine

RDFa

Resource Description Framework dans des Attributs
Pour ajouter de la sémantique, structurer les données
Pour les adresses, les produits d'une boutique...
Syntaxe : http://schema.org/
Ex : paris web boutique

Accessibilité

ARIA : Accessible Rich Internet Applications
Pour malvoyants qui se servent d'un logiciel de synthèse vocale
Ajoute des éléments sémantiques pour décrire les rôles, les états et la fonction de contrôles d'interfaces utilisateurs répandus : menus (role="menubar", role="menuitem"...), cacher des elements inutiles à la navigation inutile (flèches d'un diaporama…)
Utiliser les bons élements html : ex :bouton et pas span pour déclancher un évènement js

Meta de description des données

Dans le header, pour le partage social :

- OpenGraph pour share sur la plupart des réseaux sociaux (fb, g+, linkedin, twitter...),
- Dublin Core : schemas de données pour décrire des ressources et établire des relations avec d'autres ressources. Utile pour interopérabilité entre systèmes concus indépendament les uns des autres.
    - ex: utilisé par la BNF, pour du catalogage et améliorer la visibilité des collections sur internet.
- Twitter Card pour share twitter

=> ex paris web, à regarder en même temps que detail

OpenGraph
http://ogp.me/
Spécifique pour fb : https://developers.facebook.com/docs/sharing/opengraph
4 meta de base :

- og:title
- og:type : website, video, image, article...
- og:image
- og:url

Mais aussi : og:description, og:locale (langue)...

Twitter Card
https://dev.twitter.com/cards/
Pour ajouter de l'info à OpenGraph :

- twitter:card : type de carte "summary", "summary_large_image", "photo", "gallery", "product", "app", or "player".
- twitter:site : @username for the website used in the card footer.
- twitter:title, twitter:description, twitter:image : di pas d'OpenGraph
- Et aussi: twitter:creator, et différents autres champs selon le type de card

Framework HTML

HTML5 boilerplate <3
Détailler et expliquer

Templating

HAML -> Pierre-Cyril ?

CSS

Cascading Style Sheets
=> héritage

Syntaxe
selecteur { propriété : valeur; }

Plusieures façons de l'ajouter à la page :

- inline
- interne
- fichier externe inclus n'importe où dans la page
- fichier externe inclus dans le header

Toutes les propriété / valeur (support pou exo) :

- CSS 2.1 : http://www.w3.org/TR/CSS21/propidx.html
- https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

=> EXO
HTML, div.box *2
reset css
div : width, height, padding, border, margin, bgimage (full syntaxe, puis shortcut)
box--important => bgc red
p dans div: font-size, color, background-color, box-shadow (h-shadow v-shadow [blur] [spread] color), width en %
div height en em plutot qu'en px ⇒ que sont les em
span dans p : color, bgi ?
span hérite couleur du p => certaines valeurs s'héritent, d'autres non
class sur un span : même si class définie avant selecteur dans CSS, class appliquée : pondération

Sélecteurs et poids :

- element : 1 =>p
- class : 10 => .paragraphe
- id : 100 => #paragraphe1
- inline : 1000 => <p style="">
- !important : 1000 => p { color: blue !important; }

https://css-tricks.com/wp-content/csstricks-uploads/cssspecificity-calc-1.png
https://css-tricks.com/wp-content/csstricks-uploads/cssspecificity-calc-2.png

Les pseudo-elements
:before, :after + content : “”
::first-letter, ::first-line, ::selection
2 syntaxes, en général bien comprises par les navigateurs (:) et (::), à la base pour distinguer pseudo-elements et pseudo-selecteurs
=> exo : dans exo préc, styler des pseudo elements
1ere lettre p plus grande (et en em)
1ere ligne p en font-variant: small-caps
div : before “coucou”
span : after

Pseudo-classes
:hover, etc

2 concepts fondamentaux : box-model et flow.
Tout le reste est de la déco.

Le box-model

Exo

Ajoute une reset / eric meyer
Lien vers css : <link rel="stylesheet" href="assets/css/main.css">
Bonnes pratiques structure assets
http://learn.shayhowe.com/assets/images/courses/html-css/opening-the-box-model/box-model.png => image box model
Faire des divs de même taille fixe avec bordure :

- ajouter padding
- ajouter margin
- ajouter bg // bg hexa, bg rgba
- ajouter box-sizing (content-box vs border-box)
- display inline
- display table // avec 2 div dedans display table-cell // vs comportement display block

2 div à bordure noire, 1 avec un p, l’autre avec un span, avec du 20 mots de faux texte dedans
  float: left; width: 40%; margin: 20px 4%; border: 1px solid black;
=> ajouter 1 border ou un bg aux elements à l’interieur pour bien visualiser
=> test les propriétés + devtools permettent de visualiser la boite
=> ajouter d’autres élements span & p
=> changer le display (block, inline, inline-block, table + table-cell…)

Parler du box-sizing (box-sizing: content-box | padding-box | border-box)
=> tester le box-sizing : div avec 5% de margin (=> 100%), contenu
=> pour se faciliter la vie : *, *:before, *:after { box-sizing: border-box; }

Le Flow

position : relative | absolute | fixed | static
absolute, width => par rapport au parent le plus proche positionné
height naturelle en fct du contenu
width naturelle d'un element en absolute ou en float
Elements imbriqués avec le float : comment réagit le parent, comment corriger son flow (overflow: auto; || clearfix)
Clear d'un element float

EXO
header, main, section, aside, footer
main avec bg
section & aside en float left et right

La Typographie

Déclaration :
font- family: "Font", fallback (dans les websafe font, une ou plusieures), generic (par ordre de prépérence : sans-serif, serif, monospace, cursive, fantasy)
font-family: "Nom de la font", Arial, sans-serif
font-weight (normal, bold, 100 -> 900)
font-style (ital)
Activation des ligatures avec : font-variant-ligatures: common-ligatures

Websafe fonts
http://www.cssfontstack.com/

Webfonts pour avoir des typos non-websafe
google fonts : optimisé web, package css tout fait, optim possible (dwld possible en haut, mais récup toute la famille, et pas de package CSS => uniquement leur import js pour faire ça)
font squirrel : grosse bibli pour tous les usages, optim fine mais dans un 2nd temps
Exo live avec la ALEGREYA SANS (http://www.fontsquirrel.com/fonts/alegreya-sans)
Light, Light ial, Medium, Medium ital
=> décli pour les différentes variantes, bold en 1er, ital en 2eme
flash de texte avant chargement de la bonne typo => http://bramstein.com/

Propritétés supplémentaires pour un meilleur rendu

- hyphens: auto; /* césure propre */
- word-wrap: break-word; /* passage à la ligne forcé */
- font-variant-ligatures: common-ligatures
- -webkit-font-smoothing: antialiased
- -moz-osx-font-smoothing: grayscale

icofonts : fontawesome (mais donnent les exemples sur des <i> au lieu des span, et n'utilisent pas aria-hidden)

Les élements de formulaire

Souvent obligé de compléter avec du js
Select : select2
radio / checkbox

Les transitions

Easing : http://easings.net/fr
=> Exo : icone menu open-close (au hover)

Les animations => exo anim en demo

Les maquettes

Les designers ne pensent pas toujours à tout
-> exemples à analyser :

- home : menu trop serré, surtout en DE // quel est le comportement
- home : 3 boites de même taille : taille naturelle, taille minimum qui s'alonge si nécessaire, resize en js ?
- acces : map callée sur la hauteur du texte
- boutique : prix ne seront pas forcément sur 2 chiffres

Bonnes pratiques

vendor-prefix : sensés être seulement utilisés pour de l'experimentation de dev, pas pour de l'implémentation en prod : ex : transform dans caniuse

- -webkit- (Chrome, newer versions of Opera.)
- -moz- (Firefox)
- -o- (Old versions of Opera)
- -ms- (Internet Explorer)

code DRY vs WET : Don't Repeat Yourself VS We Enjoy Typing

- utiliser de préférence les syntaxes courtes (margin au lieu de ml, mr…)
- utiliser des modifieurs plutôt que tout rettapper : btn, btn--warning

class sémantiques

Syntaxe BEM => expliquer
Etroitement connectée au Design Atomique-> qui permet de créer : atomes, molecules, modules…

- http://bradfrost.com/wp-content/uploads/2013/06/atomic-design.png
- Atomes : http://bradfrost.com/wp-content/uploads/2013/06/atoms.jpg
- Molécule : http://bradfrost.com/wp-content/uploads/2013/06/molecule.jpg
- Organisme : http://bradfrost.com/wp-content/uploads/2013/06/organism2.jpg
- Template : http://bradfrost.com/wp-content/uploads/2013/06/template1.jpg
- Page (instance de template) :

http://bradfrost.com/wp-content/uploads/2013/06/page1.jpg
=> article sur le sujet http://bradfrost.com/blog/post/atomic-web-design/

-> qui permet d'avoir un code organisé pour avoir : des style guide. Ex : styleguide de github
http://primercss.io/
Ou aussi : Bootstrap (pour twitter)

Les emails : on oublie toutes les bonnes pratiques
https://www.campaignmonitor.com/css/

=> EXO : inté bout de maquette
Prépare les blocks pour tout le site : header, footer, main, section, aside
=> mais ne travaille que sur une partie

Inspiration

- Tableau HTML devient chart : http://codepen.io/team/css-tricks/pen/289ddf6daa8575b3c44914921f4a741f
- Interface en cercle & perspective : http://codepen.io/jcoulterdesign/pen/pjQdGb
- Dessin : http://codepen.io/rachel_web/pen/VveQPW
- Login : https://dash.readme.io/login (ralentir l'amin à 3s pour la décortiquer)

Performance

- taille de simages
- 1 seul css
- 1 seul js, en footer
- fichiers minifiés
- réduction des appels au serveurs : moins de fichiers (utilisation sprite, font icones…)

- https://checklists.opquast.com/webperf/

Plugin de test : PageSpeed Insights
Test les performances de plusieurs sites en live :

- paris web
- http://www.lp-dw.com/
- votre site ?

La Compatibilité

Différence entre front-end et backend : front-end, on ne maitrise pas l'environnement dans lequel le code est executé.
=> différents navigateurs, différents affichages.
Quelle est la config de celui qui râle ? http://supportdetails.com/

Les navigateurs

Reset vs Normalize
Modernizr
outil pour savoir si propriété / fonctionnalité gérée par navigateur : http://caniuse.com

Commentaires conditionnels

Backward compatibility
Pour viser une version de IE // ne marche plus à partir de IE10 :

- pour viser un navigateur avec CSS : modif des headers
- pour charger des fichiers qui aident à la compatibilité :
    - IE9 : Placeholders.js
    - IE8 : HTML5 et Respond.js (ou cacher le contenu et dire à l'utilisateur de mettre à jour son navigateur)

=> montrer code

Test

IE anciens à prendre en compte (de moins en moins, heureusement)
Comment débuger IE quand on est sous mac ?

- Machine Virtuelle : VirtualBox
- ModernIE : https://dev.windows.com/en-us/microsoft-edge/tools/vms/mac/

Config des host : C:\WINDOWS\system32\drivers\etc\hosts en admin : faire pointer localhost vers IP machine mac (marche avec un serveur local et des URL relatives)
F12 pour les DevTools IE

Le Responsive

On doit aussi gérer la taille de la fenêtre et la taille du texte variable selon les configs, les choix des utilisateurs.

Concerne surtout les plus petits appareils à écran tactile : smartphone, tablettes
Toutes les tailles et résolutions d'écran possibles : http://screensiz.es/

Teste avec DevTools // reload pour avoir la bonne apparence

Pourquoi ? le site web n'est pas adapté à son mode de consultation :
- scroll horizontal
- obligé de zoomer sur les textes pour lire
- élements en fixed qui cachent de larges parts de la page
- menu inadapté, navigation complexe
- liens trop petits, incliquables…

Objectifs :
- Garder un site toujours agréable à consulter
- Assurer une lecture confortable
- Avoir des interactions adaptées aux surfaces tactiles (elements cliquables de 44x44, evènement touch & co…)
- Faire attention aux perfomances

Bonnes pratiques pour le mobile :
- W3C : http://www.w3.org/TR/2008/REC-mobile-bp-20080729/
- Opquast : https://checklists.opquast.com/web-mobile/

=> Penser dès le début qu'on ignore sur quelle taille d'écran le site sera consulté

Ajustement de la taille des contenus en fonction de l'espace disponible :
- taille des textes relative : em, rem et pourcentages, mais pas de px
- blocks adaptatifs, fluides : pas de largeur en px mais en %, utiliser max-width (cas images à traiter, bgsz cover)...
- réorganisation des contenus
- points de rupture : mediaqueries : breakpoints pas connus à l'évance, dépendent du contenu. Faut resizer et vérifier quand ça pète. Mais soin particulier pour certains devices populaires, comme ipad et iphone. 45-90 caractères par ligne pour une bonne lisibilité.

Mediaqueries

Permettent la réorganisation du contenu & ajustement des styles

@media [ONLY | NOT] *type_de_media* [AND (*critère* : *valeur*)]
=> spec mediaquery : http://www.w3.org/TR/css3-mediaqueries/

NOT, ONLY : facultatif, Si on cette media query sert à exclure ou limiter certains types de medias. Rarement utilisé en responsive.

type : all, screen, print, etc. : facultatif : vaut all quand il n'est pas précisé. Pour cibler un type de media en particulier.
Critères :
width, min-width, max-width, height, etc. : taille en px, em, rem. Pour cibler en fonction de la taille de la fenêtre.
orientation : portrait, landscape. Pour cibler par l'orientation
resolution, min-resolution, max-resolution & -webkit-device-pixel-ratio : Nombre, en dpi. La résolution de l'écran, pour cibler les écran rétina et leur servir des images plus grandes.
Ex retina : (min-resolution : 120dpi)
(-webkit-device-pixel-ratio : 1.5)
=> votre écran est-il rétina ? https://bjango.com/articles/min-device-pixel-ratio/
Séparateur : And, virgule.
@media screen and (min-width : 450px), screen and (orientation : portrait) and (max-width : 1024px) => si mon média est un écran et que la fenêtre fait au minimum 450px de large, ou si mon media est un écran en orientation portrait dont la fenêtre fait au maximum 1024px de large

Exo

Exo de réorgansation imposible si on utilise pas table head, body, footer
-> laisser gamberger 5min, donner des indices, puis donner la solution

Exo de menu caché
Toggled avec class

Viewport

Ex iphone 4 :

- largeur de l'écran réelle en px : 640px
- device-width : 320px (parce que écran retina)
- viewport appliqué par défaut par Safari Mobile: 980px

Balise viewport explique au navigateur mobile comment il doit gérer sa fenêtre, et comment il doit afficher le contenu.

- autorisation du zoom ou pas, et dans quelle mesure
- simulation d'un écran plus grand ou plus petit

Valeurs :

- width=XXXpx : Affiche le site dans la résolution demandée
- width=device-width : Affiche le site dans la résolution virtuelle de l'écran
- initial-scale=X.X, minimum-scale=X.X, maximum-scale=X.X : Contrôle du zoom, Zoome initial sur la page, et zoom/dézoom max autorisés
- user-scalable=yes ou no: Autorisation du zoom

<meta name="viewport" content="width=device-width, initial-scale=1.0">
=> avec un travail responsive bien fait !

Exo de viewport : tester le comportement en le changeant
<meta name="viewport" content="width=800px, user-scalable=yes">
<meta name="viewport" content="width=800px, user-scalable=no">
<meta name="viewport" content="width=200px">
<meta name="viewport" content="width=1500px, user-scalable=no">
<meta name="viewport" content="width=device-width">

Et aussi

Gestion du touch : http://hammerjs.github.io/ &
https://github.com/hammerjs/jquery.hammer.js

Performances :

- Mobile first
- Images responsive : Picturefill http://scottjehl.github.io/picturefill/ pour mediaquery sur les images

Webapp
Mobile Boilerplate : https://github.com/h5bp/mobile-boilerplate
DÉPRÉCIÉE, mais utile quand même
Métadonnes
Titre de la page : <meta name="apple-mobile-web-app-title" content="Titre de la page">
Différentes tailles d'icone en fonction du device
Startup image en fonction du device
Guidelines apple :
https://developer.apple.com/library/ios/documentation/AppleApplications/Reference/SafariWebContent/ConfiguringWebApplications/ConfiguringWebApplications.html

Les framework

Frameworks CSS

Bootstrap / http://foundation.zurb.com/
Qu'est ce que c'est, pourquoi, comment ?
Avantages :

- rapide à mettre en place et à prendre en main
- super doc
- permet d'avoir un design avancé rapidement
- formulaires bien gérés
- modularité

Inconvénients:

- lourd, souvent plein de trucs inutiles
- tous les sites se ressemblent

- globalement inutile si trop de customisation par dessus (on n'a pas besoin de boostrap pour faire des colonnes, des boutons et une lightbox)
- noms de class reliés à l'apparence

⇒ bien pour
- un backoffice
- démarer rapidement un projet et le prototyper
- quand on ne sait pas faire de l'inté

⇒ nul pour :
- avoir un code léger
- passer pour un vrai dev front

Framework HTML

HTML5 boilerplate <3
Détailler et expliquer

Outils d'industrialisation : les préprocesseurs

SASS : pré-processeur pour CSS : variables, fonctions (mixins), petits fichiers
Installer package SASS dans l'editeur
Revenir sur les bases de SASS : import de fichiers, variables, calculs, mixins (=fonctions)
Belle sturcture SASS ici :
https://github.com/thierrymichel/workflow/tree/master/2-guidelines#folder-structure-srcstyles

Nécessite un compilateur

Version logiciel : CodeKit...

Version ligne de commande : Grunt (le plus utilisé), Gulp, Webpack (meilleur candidat à venir) et les autres.
Sert à compiler du sass, mais pas seulement : bon pour toutes les tâches automatiques :
- copies de fichier
- linter
- test unitaires (sécurisent la maintenance et les évolutions)

Exo préprocesseur

`npm init` pour créer le package.json
Grunt => installer dans le projet : `npm install grunt --save-dev`
Créer à la mano le Gruntfile.js => `grunt server` marche
Crée dossier src content les sass, js, img, fonts
Plugin NPM pour Grunt : http://gruntjs.com/plugins

- SASS pour compiler le SASS en CSS (+ sourcemaps) :
  https://www.npmjs.com/package/grunt-contrib-sass
  Structure SASS : http://perrinebocquin.fr/2015/lpdw/sass.zip
  Montrer réaction de la console si erreur
  Récup le index.html et l'ajoute
  récup les css et les remets au bon endroit
- Watch pour écouter les changements dans les fichiers :
  https://www.npmjs.com/package/grunt-contrib-watch
- BrowserSync pour recharger les pages du projets automatiquement, et synchroniser tous les navigateurs (y compris sur mon smartphone) :
  http://www.browsersync.io/docs/grunt/
  placer avant le watch
  Montrer les option UI, les URL à ouvrir (console)
- PostCSS pour améliorer réduire la tailled es CSS, ajouter les préfixes :
  https://www.npmjs.com/package/grunt-postcss + npm install grunt-postcss pixrem autoprefixer cssnano
- Copy pour les images, les fonts, les js :
  https://www.npmjs.com/package/grunt-contrib-copy
  config assez custom
  Structure JS : http://perrinebocquin.fr/2015/lpdw/js.zip
  tache : pas de copie des images en build
  ajout des taches au watch
- ImageMin pour réduire le poids des images pour le build :
  https://www.npmjs.com/package/grunt-contrib-imagemin
- Concat pour les vendors js (du coup plus de copy) :
  https://www.npmjs.com/package/grunt-contrib-concat
  Ajout de la tache au watch
- Uglify pour minifier les js : https://www.npmjs.com/package/grunt-contrib-uglify
  En build
- et puis tant qu'on y est, ajout de vendors en npm :
  - Normalize : npm install --save normalize.css
  - jQuery : npm install --save jQuery
  - Placeholders pour IE9 : npm install --save placeholder
  - etc.

2 tâches :

- server : pour les travail en cours (avec watch, reload, mais sans les tâches lourdes comme la minification)
- build : pour livrer avec des fichiers performants (minification…)

On .gitignore les fichiers générés et compilés (sass-cache, node_modules, assets)

=> création d'un workflow faite
=> possibilité aussi de récupérer un workflow : récup du gruntfile et du package.json + npm install

Evaluation

À la maison
Choisir une des maquettes du cours et l'intégrer.
A faire en responsive // menu mobile est donné, le reste est à improviser => questions ?
Si inspiré, faire des animations au hover sur les elements qui s'y prettent

Font : à choisir sur google-font ou fontsquirrel, typo proche, selon l'implémentation préférée.
4 styles.
gris : #575757
bleu : #1FBED6