IUT BAYONNE PAYS BASQUE

UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR | COLLÈGE STEE SCIENCES ET TECHNOLOGIES POUR L'ÉNERGIE ET L'ENVIRONNEMENT

Développement Web – Front Web et accessibilité / ergonomie

R3.01

Yann Carpentier

Objectifs du cours

- Rappel sur l'ergonomie d'un point de vue général (p.3)
- Rappel sur HTML / CSS (p.6)
- Processus d'intégration d'une maquette (p.10)
- Focus sur l'accessibilité Web (p.13)
- HTML, CSS, DOM et Javascript (p.20)

Rappel général sur l’ergonomie Web (1)

- Sur la base de spécifications
- Maquettes
- Scénarios
- Personas
- Interviews

Rappel général sur l'ergonomie Web (2)

- Principes ergonomiques à appliquer au Web
  - Généraux :
    - > Approche objet / action
    - > Fournir une aide en ligne (informer, guider, …)
    - > Réduire la profondeur des menus
    - > Organiser les menus
    - > Organiser les éléments d'interfaces en bloc logique
    - > Feedback (ex messages d'erreurs sur formulaires)
  - Visuels :
    - > Limiter le nombre de couleurs
    - > Choix des couleurs en évitant les discriminations visuelles (luminosité, contraste notamment)
    - > Sémantique des couleurs (culturelle, professionnelle)

Rappel général sur l'ergonomie Web (3)

- Principes ergonomiques à appliquer au Web
  - Textuels :
    - > Densité de texte
      - Paragraphes
      - Aérations
    - > Choix de polices
      - Adaptation de la taille à la résolution
      - Pas de serif (sauf pour les titres)
      - En général, pas de justification sur écran
    - > Eviter les effets : clignotements, variations de couleurs par animation
  - Son :
    - > Ambiance : A éviter
    - > Vidéos ou interviews : à sous titrer
  - Etc …
  - Cf : Cours de madame Voisin R1.02 Dev interface Web partie 1

Rappel sur HTML / CSS (HTML 1)

- Structure construite avec des balises choisies sur la base de leur valeur sémantique :
  - <main> <nav> <article> <aside> <header> <footer> <Section>

Rappel sur HTML / CSS (HTML 2)

- Balises HTML
  - Balises choisies en priorité pour leur valeur sémantique
  - Jamais utilisées pour leur représentation graphique (ex un H1 grand et visible), le style d'une balise sera appliqué via le CSS
  - Balises de titre
    - > H1 unique
    - > Les balises restantes toujours imbriquées (ex pas de H3 sans H2)
  - toujours dans un <ul> ou
  - Jamais de sans attribut alt
  - toujours avec un texte explicite (pas de : cliquez ici)
  - Mise en valeur des mots via des balises sémantiquement adaptées (pour emphase, ou), la mise en valeur visuelle passera par le CSS

Rappel sur HTML / CSS (CSS 1)

- De préférence dans un (ou plusieurs) fichier à part lié au document html via la balise <link>
- Style en cascade
- Appliqué aux éléments html via leur attribut « class » ou « id »
  - idéalement class pour cibler plusieurs éléments ayant la dite class déclarée en attribut
  - *Id* pour cibler un élément uniquement
  - Plusieurs *class* possibles pour un même élément
  - Cascade implique qu'un style provenant d'une classe peut être « écrasé » par une classe déclarée dans l'attribut

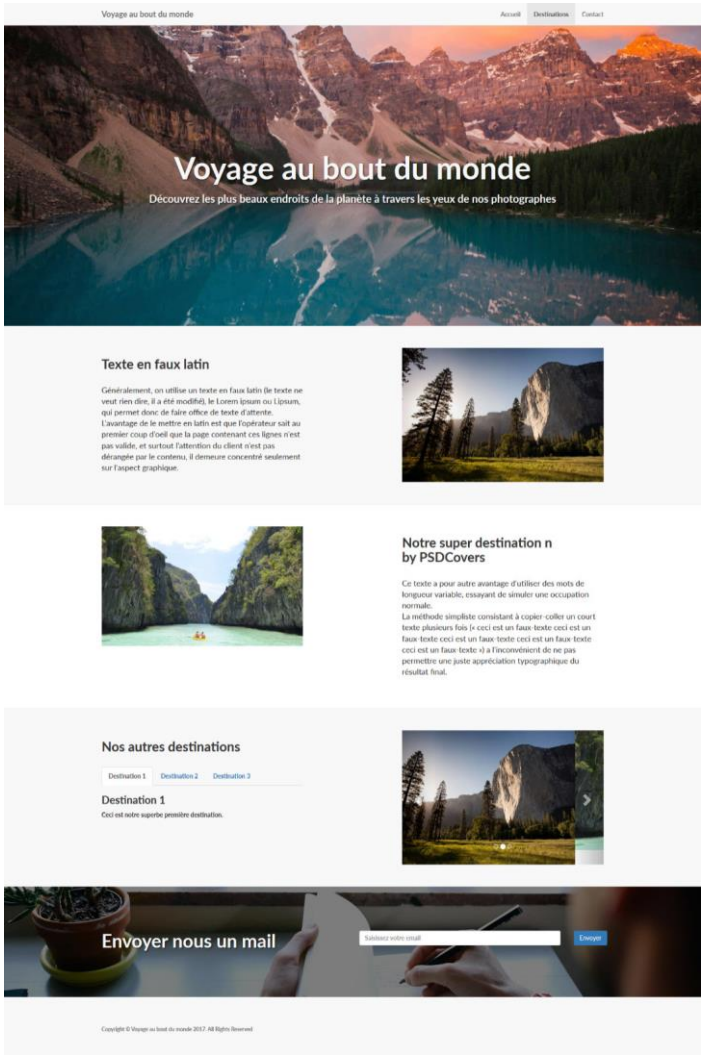
Rappel sur HTML / CSS (CSS 2)

- Structuration du flux des éléments dans la page via plusieurs mécanismes :
  - Standard historique : Inline , inline-block, block, float
  - Layout :
    > Flexbox
    > Grid
  - Cf : Cours Madame Voisin R1.02 Affichage Web et accessibilité partie 2

Intégration Web

- Point de départ : Maquette construite par un designer
  - Intégration en plusieurs grandes étapes
  - Phase 1 : découpage / identification structure
    - > Découpage de la maquette en parties sémantiquement cohérentes (-> future structure html)
    - > Découpage d'abord macro, puis micro (avec les couleurs pour faciliter l'identification des zones)
    - > En déduire le meilleur support technique pour la structure html à venir (standard , flexbox , grid)

Intégration Web



Titre et menu

En-tête de page

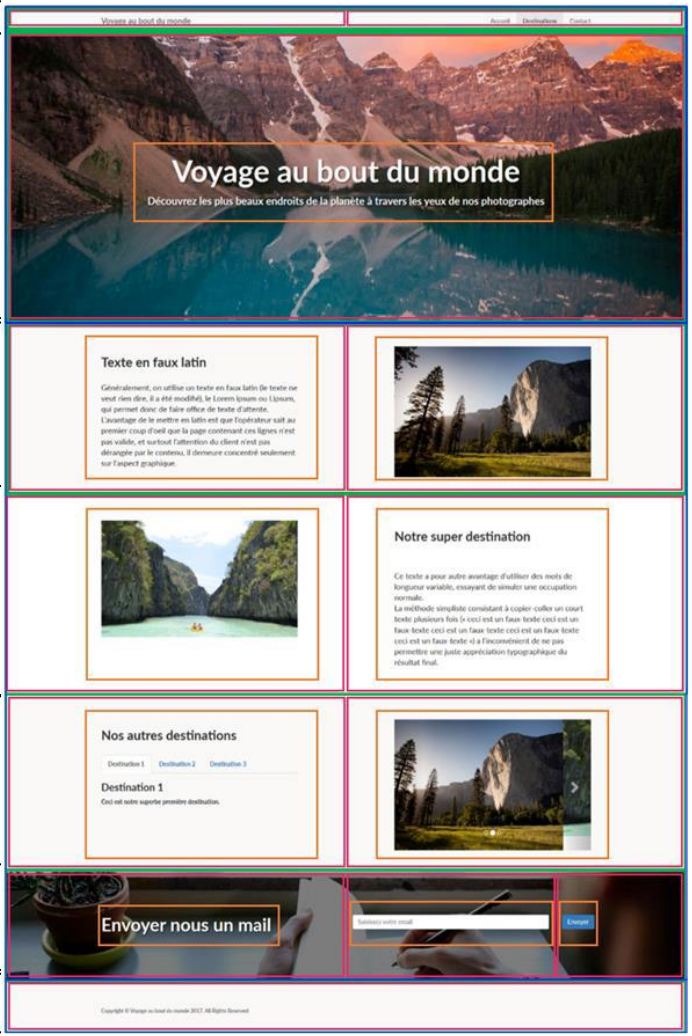
Contenu

Contenu

Contenu

Contenu

Pied



Intégration Web

- Phase 2 : Structure HTML
  - > Choix des balises en fonction des zones identifiées (ex *footer* pour un pied de page, *nav* pour un menu (cf cours R1.02 et mémentos fournis)
  - > Ajout du contenu sans préoccupation du style
    - Paragraphes
    - Images
    - Liens
    - Etc…
- Phase 3 : ajout des styles CSS
  - > En premier : ceux liés à la structure et au comportement des éléments dans le flux : layout grid / flex / standard
  - > Ensuite : les styles concernant l'affichage :
    - Couleurs
    - Polices : famille, poids et taille
    - Marges internes et externes (margin / padding)
    - Alignement des textes
    - Etc …

Accessibilité Web

- **Accessibilité ->** on pense souvent aux personnes ayant des **troubles visuels**. Mais le principe même de **l'accessibilité est de rendre un site ou une application web utilisable par le plus grand nombre**.

- Adapter les éléments d'interface (en taille notamment) pour faciliter leurs manipulations

- Rendre la lecture possible via un lecteur d'écran par inclusion d'information non affichée -> landmark ARIA

Accessibilité Web – Règles générales

- Doctype valide : <!DOCTYPE html>
- Balises html avec un attribut lang : <html lang="fr">
- Le code source doit être valide :
  - balises conformes au type de document déclaré
  - Imbrication correcte des balises(ex: lien imbriqué dans un autre lien interdit)
  - Chaque balise ouverte doit être fermée : <html> </html> (Attention aux quelques balises auto-fermantes : ex <img />)
  - Pas d'id dupliqué
  - Pas de balises obsolètes
- Vérification possible sur https://validator.w3.org/

Accessibilité Web – Landmark Aria

- Selon le W3C, les attributs ARIA peuvent définir :
- **la nature** de l'élément (un menu de navigation, par exemple) ;
- **l'action** de l'élément (il ouvre une modale) ;
- **l'état** de l'élément (un menu dropdown ouvert ou fermé, par exemple)

- Identification des zones principales (navigation, en-tête , contenu principal, pied de page, moteur de recherche) de la page avec "role" comme attribut et comme valeur :
  - Pour l'entête : banner
  - Pour le pied de page : contentinfo
  - Pour le contenu principal : main
  - Pour le moteur de recherche : search
  - ATTENTION : banner, contentinfo et search sont uniquee dans la page même si il y a plusieurs balises <nav> <header> ou <footer> (dans les <articles> par exemple)

Accessibilité Web – Landmark Aria - Navigation

- L'attribut aria-label permet de nommer de façon 'invisible' une région dans une page.
  Cela permet de différencier plusieurs menu de navigation en leur donnant un nom, ou de nommer des parties de page significative, pour lesquelles aucune balise HTML5 sémantiquement adaptée n'existe.
  Ex : dans un <nav> aria-label="Navigation Principale" et aria-label="Navigation Secondaire" dans l'autre.

- L'attribut aria-haspopup (avec true ou false en valeur), permet d'indiquer si un item de menu a un sous menu
- L'attribut aria-expanded (avec true ou false en valeur), permet d'indiquer l'etat déplié ou non de ce dernier
- L'attribut aria-current avec la valeur « page » permet de définir le lien correspondant à la page courante (**y compris dans le fil d'ariane**)

- Via le CSS il faudra également faire un feedback visuel aux personnes n'ayant pas besoin d'être assistées.

Accessibilité Web – Landmark Aria - Tableau

- Il est possible dans un tableau de données de préciser le sens de lecture à l'aide de *scope* avec la valeur
  - Avec la valeur col pour une lecture en colonne, par ex : <th scope="col"> pour les colonnes d'un tableau
  - Avec la valeur row pour les lignes du tableau dans la première cellule de la ligne, ex :

```
<tr>
  <th scope="row">Carmen</th>
  <td>33 ans</td>
  <td>Espagne</td>
</tr>
```

Accessibilité Web – Landmark Aria - formulaire

- En dehors d'aria :
  Toujours faire le lien entre un input et son label (<label>) via l'attribut for ayant pour valeur l'id du input suivant, ex :
  <label for="nom">Quel est votre nom ?</label>
  <input type="text" name="nom" id="nom" />
- Avec aria :

  Possible regroupement de champs sur des thématiques communes grâce au rôle *group et l'attribut aria-labelledby* pour la balise englobante, ex :

```
<div role="group" aria-labelledby="coordonnees">
  <p id="coordonnes">Vos coordonnées</p>
  <label for="nom">Quel est votre nom ?</label>
  <input type="text" name="nom" id="nom" />

  <label for="prenom">Quel est votre prénom ?</label>
  <input type="text" name="prenom" id="prenom" />
</div>
```

- Aria-invalid (true ou false) permet de faire un feedback sur un champs contenant une valeur incorrecte lors de la soumission du formulaire

Accessibilité Web – Landmark Aria – en général

- L'usage de pictogramme doit être systématiquement accompagné d'un aria-hidden à true. (sinon il sera lu par le lecteur d'écran).
  Ex :

- Indiquez les changements de langue dans le texte avec l'attribut lang, ex : * un gamer souhaite bonne chance en écrivant gl hf, qui signifie : Good luck, have fun *

- On peut ajouter le sens de lecture avec l'attribut dir suivi de ltr ou rtl pour lecture de gauche à droite (ltr) ou de droite à gauche (rtl).

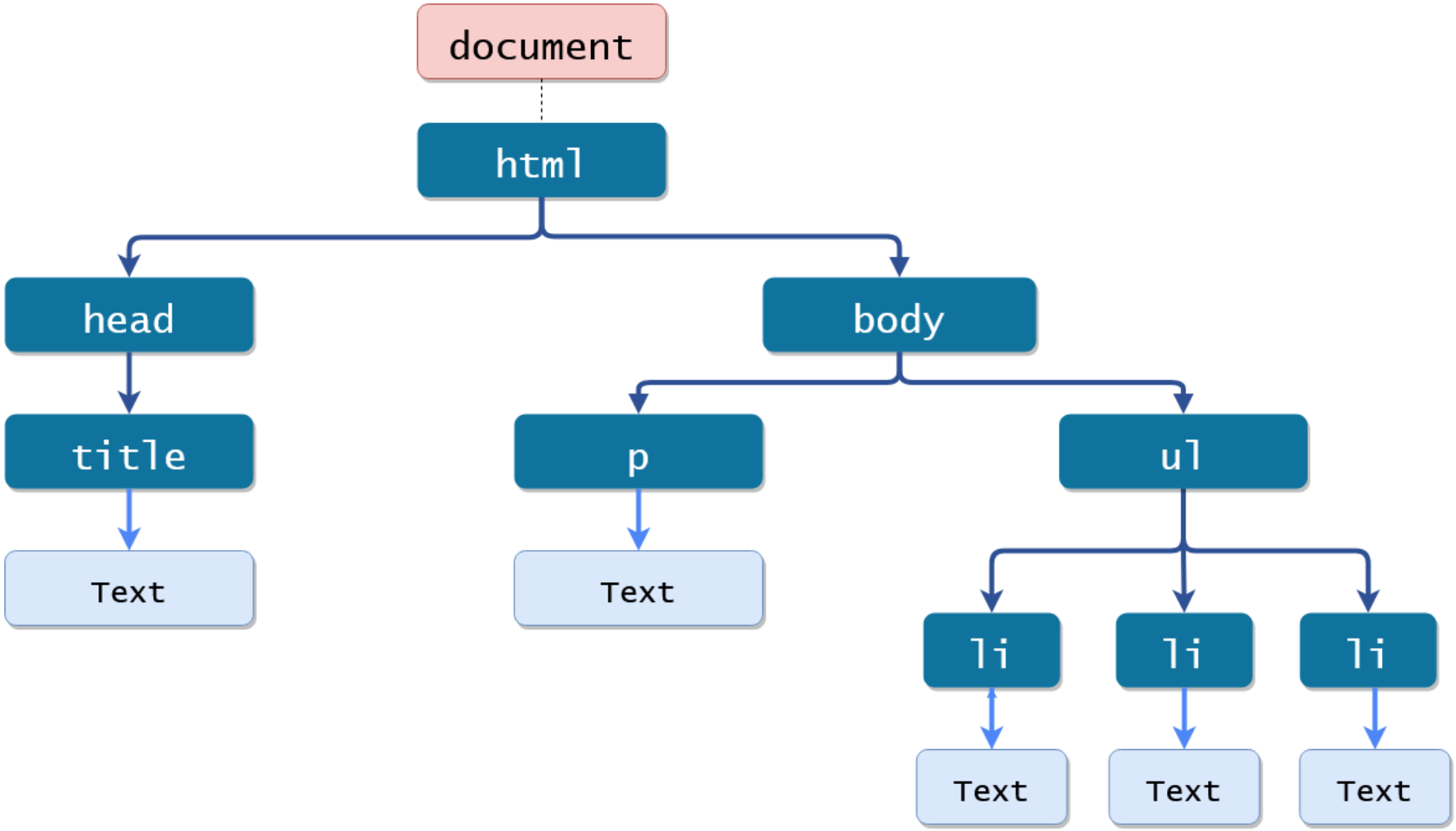
- Pour plus de détails et des compléments :
  - Elearn : R3.01 - Développement web (Web Front - Ergo)
    - > Différents mémentos, supports pdf
  - https://github.com/DISIC
  - https://disic.github.io/guide-integrateur/

HTML, CSS, DOM et Javascript (JS)

- En préambule
  - Il est possible de faire des animations simples à complexes en pure CSS
  - Le javascript ne sert pas qu'à faire des animations en modifiant des attributs
  - Cette partie du cours a pour objectif de vous faire prendre conscience de la synergie puissante existant entre HTML, CSS et JS

- Quelques définitions
  - DOM : Document Object Model
    - > Représentation Objet de la page HTML au niveau du navigateur une fois la page chargée
  - Javascript (Ecmascript) : couramment appelé JS, c'est langage web (étudié plus tard dans l'année), permettant entre autre (mais pas uniquement), la manipulation d'un document HTML :
    - > Changement de tout ou partie du contenu dynamiquement sans rechargement de la page
    - > Dynamisation du contenu
    - > Contrôle de saisie
    - > Interaction déclenchable automatiquement ou sur action de l'utilisateur

HTML, CSS, DOM et Javascript (JS)



HTML, CSS, DOM et Javascript (JS)

- Procédure (très simplifiée)
  - Identification d'un élément html à manipuler
    - > Se fait via un id existant ou dédié à cette finalité
    - > Ou une classe CSS (et donc implique un ensemble d'éléments)
  - Déclaration d'une variable en JS (non typées) :
    - > var mavariable = 5 ou var mavariable2 = ''Tzeentch''
  - Récupération dans un objet javascript nommé *elt* d'un élément html disposant d'un id en vue de le manipuler (avec querySelector):
    - > var elt = document.querySelector(''#id'') (le # est primordial ici comme en CSS)

HTML, CSS, DOM et Javascript (JS)

- Procédure (suite)
  - On peut donc par exemple :
    - > Définir l'attribut *style* d'une balise (via sa récupération en JS comme vu précédemment et à la méthode setAttribute, changer la valeur en CSS d'une couleur : elt.setAttribute('style','color : #0C0C0C');
    - > Définir la valeur d'un attribut *aria-required* ou *aria-expanded*
    - > Changer la visibilité d'un élément de la page en modifiant la valeur de son style *display* à *none*
    - > Remplacer le contenu d'une balise <p> ou

HTML, CSS, DOM et Javascript (JS)

- Procédure (fin)
  - Ou écrire le JS :
    - > Dans des fonctions déclarées entre balises <script> </script>, balises placées de préférence à la fin de la page html, juste avant la balise de fermeture, fonctions appelées ensuite via des événements sur les balises.
      Cf : cours de Javascript à venir dans l'année

Pour simplifier exceptionnellement, dans notre cas nous placerons ce code, dans un attribut *onclick,* sur un bouton par exemple (sur une seule ligne):

<button onclick = ''var elt = document.querySelector('#le-span');
elt.setAttribute('color','#FF0000');''> Changer la couleur en rouge </button>