Spécifications FWK rédigées le 15/01

Le framework est CSS / HTML pur. Le JS est utilisé en dernier recours. Il y a possibilité de suggérer des apis tierces. L’écriture de la page se fait en html.

# Types d’affichages globaux

Avant de parler de types d’affichage globaux, on va déjà parler du header du site :

Le framework doit proposer 4 types d’affichages globaux (possiblement identifié par une classe à ajouter sur le container) différents :

* Le mode DOC, pratique pour afficher un document que l’on aura écrit en HTML (par défaut) ;
* Le mode BOOK donnant l’impression à l’utilisateur de lire un vrai livre ;
* Le mode PRES permettant la rédaction de présentation style PowerPoint en HTML ;
* … à voir si tu as d’autres idées d’affichages et si le temps le permet.

Tout type d’affichage doit proposer un système qui permet de suivre le thème/chapitre/paragraphe en cours. Cela peut se faire par la modification du texte de l’entête, mais également par la présence d’un élément supplémentaire sur la page (un fil d’Ariane) nous rappelant où on en est dans le texte.

## Mode DOC

* Le mode DOC affiche donc le document sous forme de page à la suite en hauteur. Centrée par rapport à l’écran et séparé par une marge. (10px) calculé en fonction de la hauteur des éléments ;
* background opaque gris ;
* border : gris foncé 1px ;
* border-shadow dégradé gris foncé vers couleur de background.

### Fonctionnalités supplémentaires

Le temps de lecture restant peut être affiché, il peut se calculer dynamiquement ou se basant sur le temps de lecture moyen d’un être humain.

Un mode «l’essentiel en quelques mots » aurait sa place sous deux formes. La première étant de marquer les mots essentiels au sein même du texte en les faisant ressortir par rapport au reste du texte. La seconde ne présenterait qu’eux, pour une lecture rapide des points essentiels.

## Mode BOOK

* Affichage de deux pages à la fois sauf sur mobile ;
* background opaque gris ;
* border-shadow dégradé gris foncé vers couleur de background ;
* fleche blanche à gauche et à droit pour changer de page ;
* event clavier afin de changer de page ;
* event tactile afin de changer de page facilement sur les téléphones et tablettes ;
* animation de page qui se tourne lors du changement de page.

## Mode PRES

* Affichage d’une page à la fois dans le même style que le mode DOC ;
* background opaque gris ;
* border : gris foncé 1px ;
* border-shadow dégradé gris foncé vers couleur de background ;
* border-radius sur le container ;
* event clavier afin de changer de page ;
* event tactile afin de changer de page facilement sur les téléphones et tablettes ;
* plusieurs choix d’animation pour le changement de slides ;
* au clic sur une slide dans la liste des slide, afficher en pleine écran la slide ;

### Fonctionnalités supplémentaires

* ajout d’un timer ;
* affichage plein écran ou liste des slide ;
* ajout de commentaire sur une slide ;
* transitions de texte variable à l’intérieur de chaque page.

## Mode Spritz

Il s’agit d’une application de lecture. Son approche est amusante et reprend quelques travaux sur la lecture.

On doit pouvoir changer :

* la couleur du texte ;
* la vitesse de défilement ;
* Pause/play intuitifs ;
* Régler le temps avant le démarrage.

# Différents types de page

4 types de pages différents :

* Page de garde ;
* Sommaire ;
* Page normale (par défaut) ;
* Page blanche.

A noter que seule la page de garde n’a pas de header ou footer et ne respecte pas les mêmes contraintes de marges

## La page de garde

Peux contenir un nombre important d’informations prédisposées dans la page :

* Auteur ;
* Date ;
* Titre ;
* Fond ;
* Logo ;
* Sous-titre ;
* Tuteur ;
* Lieu 🡪 université.

## Sommaire

* Contenant des titre de différentes sortes (globaux, I, A, 1, a, i)
* Gestion des clics sur tout le sommaire
* Affichage de la page sur lequel est le titre (I test ------------- page 4) variable en fonction de la taille d’écran et calculé

## Page normale

* En tete + body + footer

## Page Blanche

* Div de taille égale mais complètement vide ;
* Donc pas de header ni footer ;
* Ne compte pas dans le décompte des pages ;
* La page blanche peut contenir, paradoxalement, un message indiquant l’intention de la page vide.

# Composants d’une page

Ici on ne parlera pas forcément de composants mais plus de ce que l’utilisateur pourra faire avec le framework dans une page

Exemple :

* Positionnement du texte via une classe ;
* Gestion de la police (font-family : nomDuFont) ;
* Gestion de la taille de police (font-size : ) ;
* Gestion de la couleur du texte (color : ), de fond (background-color) ;
* Gestion du style de texte (gras (<strong></strong>), souligné (<u></u>), italique (<em></em>), lien (<a href= » » > </a>), …) ;
* Gestion du footer et header de page ;
  + Géré dans le document global 🡪 sur toutes les pages autres que la page de garde ;
  + Géré dans une page 🡪 uniquement sur la page.
* Gestion des marges et padding ;
* Gestion des images <img src= » »> </img> ;
* Gestion des diagrammes (différents types) avec Google Charts par exemple. L’utilisateur definit dans une div le type de diagramme ;
* Gestion des légendes ;
* Gestion des tableaux <table> ;
* Gestion des notes de bas de page ;
* Gestion des citations et définitions <blockquote>/ <cite> / <dl><dt><dt> ;
* … si tu as d’autres idées ou que j’ai oublié des trucs …

# Droit d’auteur

On doit laisser la possibilité de bloquer de manière «soft» la copie ou le plagiat. Plusieurs solutions sont possibles :

## Copié-collé

Remplacer le contenu du buffer par un message évoquant le droit d’auteur et la volonté de laisser le texte à sa place.

## Clique droit / ctrl-u

Afficher un message évoquant le droit de l’auteur.

# Idées de choses à ajouter si tout est fini

* Internationalisation (FRANCAIS/ANGLAIS/Russe si tu veux lol) ;
* Utilisation d’une API afin de gérer la lecture audio du mémoire (ça serait OUFFFFFFFFF) ;
* … si tu as d’autres idées je suis à ton écoute bien évidement …