Contrôle de Web Avancé / Javascript

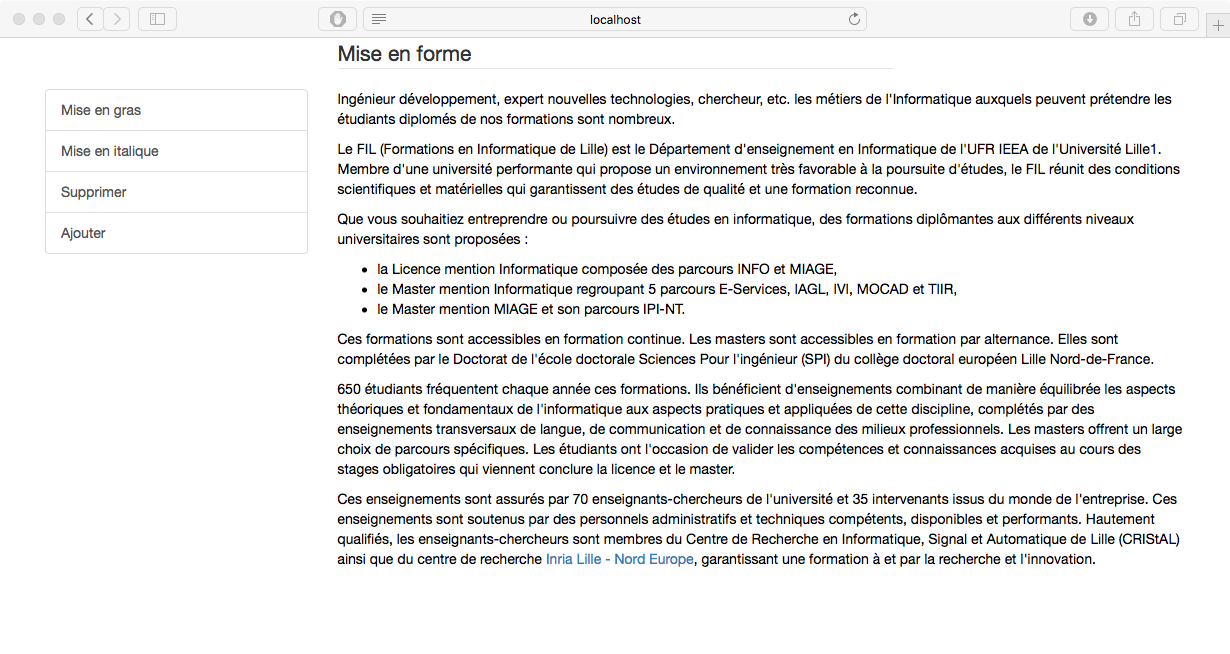
e-services FA - Jeudi 3 Novembre 2016

Nom :

Prénom :

Pour chaque question, merci de répondre sur la page de gauche correspondante.

L’application qui servira à évaluer vos connaissances en Javascript est une simple page Web où un texte (à droite) peut être légèrement formaté grâce à 4 fonctions (à gauche). La figure suivante

illustre l’affichage.

Quand le curseur de la souris passe au-dessus d’un élément HTML (paragraphe, élément de liste, lien…), celui-ci s’inverse visuellement (fond noir, texte en blanc). La barre d’outil à gauche avec la mise en gras, la mise en italique, la suppression, l’ajout fonctionne de la manière suivante : on clique sur un des outils puis sur l’élément HTML pour lequel on souhaite appliquer l’action (d’où l’intérêt de l’inversion visuelle pour une meilleure identification de la portée de l’action).

Ce contrôle n’a pas pour objectif d’évaluer vos compétences en matière d’algorithmie mais de Javascript (JS). Donc pour la plupart des questions, vous aurez une version en langage naturel d’un programme que vous devrez traduire en JS.

Les classes, dans ce contrôle, sont à créer à partir de fonctions et de prototypes (pas d’EcmaScript 6 donc, c’est-à-dire pas de mot-clé class).

L’Objet AuditeurContenu

L’objet (et non la classe) AuditeurContenu s’occupe de gérer les événements de souris pour les éléments HTML du texte à droite de la page.

AuditeurContenu contient d’abord une propriété **elementSurvole** qui référence l’élément HTML dont l’ID vaut contenu.

AuditeurContenu contient aussi la propriété **contenu** qui référence le même élément HTML (c’est-à-dire dont l’ID vaut contenu).

AuditeurContenu contient la fonction **init** qui a 2 instructions :

1. On veut que tout déplacement de souris (onmousemove) au dessus l’élément HTML référencé par la propriété contenu de l’objet (AuditeurContenu) invoque la fonction onmousemove de l’objet (AuditeurContenu). Mais attention, on veut que quand celle-ci est invoquée, le mot clé this à l’intérieur de celle-ci renvoie toujours vers l’objet (AuditeurContenu), quelque soit la manière dont cette méthode est invoquée. Il faut donc utiliser la fonction bind.
2. La seconde instruction est identique mais pour le clic de souris (onclick).

AuditeurContenu contient la fonction **onmousemove** qui prend un paramètre evenement et dont le comportement est le suivant.

Si l’élément HTML référencé par la propriété elementSurvole de l’objet ne vaut pas l’élément HTML référencé par la propriété target du paramètre evenement alors

Si la propriété elementSurvole de l’objet n’est pas nulle alors

Les propriétés background et color de la propriété style de l’élément elementSurvole valent respectivement ‘#FFFFFF’ et ‘#000000’.

La propriété elementSurvole de l’objet référence maintenant l’élément HTML référencé par la propriété target du paramètre evenement.

Si l’élément HTML référencé par la propriété target du paramètre evenement ne vaut pas l’élément HTML référencé par la propriété contenu de l’objet alors

Les propriétés background et color de la propriété style de l’élément elementSurvole valent respectivement ‘#000000’ et ‘#FFFFFF’.

AuditeurContenu contient la fonction **onclick** qui prend un paramètre evenement et dont le comportement est le suivant.

Si la propriété commande de l’objet n’est pas nulle alors invoquer la fonction agit de cette propriété commande avec comme argument la propriété target du paramètre evenement.

AuditeurContenu contient la fonction **fixerLaCommande** qui prend un paramètre commande et dont le comportement est le suivant.

La propriété commande de l’objet vaut le paramètre commande.

La classe Commande

La classe Commande est la classe mère des classes liées aux commandes (voir plus loin).

Le **constructeur** de cette classe prend en paramètre titre et baliseMere.

Son comportement est le suivant.

La propriété titre du nouvel objet créé vaut la valeur du paramètre titre.

Le paramètre baliseMere est lui aussi transféré comme propriété du nouvel objet.

Enfin, on invoque la fonction construireLaBaliseHTML sur le nouvel objet.

La classe définit la fonction **construireLaBaliseHTML** qui ne prend pas de paramètre et dont le comportement est le suivant.

On crée un élément HTML A (lien) qu’on met dans la propriété balise de l’objet.

On affecte la valeur "list-group-item" à la propriété className de la précédente propriété balise. Idem pour la propriété href mais avec la valeur "#".

On ajoute (au sens HTML) l’élément HTML référencé par la propriété balise à l’élément HTML référencé par la propriété baliseMere de l’objet.

On indique que l’objet devient auditeur des clicks sur l’élément HTML référencé par la propriété balise de l’objet.

Enfin on ajoute (au sens HTML) une balise textuelle *- avec comme texte le contenu de la propriété titre de l’objet -* à l’élément HTML référencé par la propriété balise.

La classe définit la fonction **(\*\*la méthode qui sera appelé en cas de click\*\*)** qui prend comme paramètre evenement et dont le comportement est le suivant.

La propriété className de l’élément HTML référencé par la propriété balise de l’objet vaut "list-group-item active".

On invoque la fonction fixerLaCommande de l’objet AuditeurContenu avec l’objet comme argument.

La classe définit la fonction **agit** qui prend comme paramètre cible et dont le comportement est le suivant.

La propriété className de l’élément HTML référencé par la propriété balise de l’objet vaut "list-group-item".

On invoque la fonction fixerLaCommande de l’objet AuditeurContenu avec null comme argument.

La classe CommandeFormatage

La classe CommandeFormatage existe pour des outils comme *Mise en gras* ou *Mise en Italique*.

Elle hérite de la classe Commande.

*Pour cette héritage, que va-t-il se passer pour le constructeur de Commande et/en l’absence de paramètre (question délicate et non évoquée en cours :-)). Que faudrait-il ajouter et où ?*

Le **constructeur** de cette classe prend en paramètre titre, baliseMere et format.

Son comportement est le suivant.

La propriété format du nouvel objet créé vaut la valeur du paramètre format.

On invoque le constructeur de la classe mère avec les arguments titre et baliseMere.

La classe définit la fonction **agit** qui prend comme paramètre cible et dont le comportement est le suivant.

On définit une variable locale nomPropriete qui contient la valeur de la propriété propriete de la propriété format de l’objet.

On définit une autre variable locale laValeur qui contient la valeur de la propriété valeur de la propriété format de l’objet.

On affecte laValeur à la propriété X de la propriété style de l’objet cible, où X vaut ici la valeur de nomPropriete.

Enfin on invoque la fonction agit définie par la classe mère sur l’objet avec l’argument cible.

La classe CommandeAction

La classe CommandeAction existe pour des outils comme *Ajouter* ou *Supprimer*.

Elle hérite de la classe Commande.

Le **constructeur** de cette classe prend en paramètre titre, baliseMere et fonction.

Son comportement est le suivant.

La propriété fonction du nouvel objet créé vaut la valeur du paramètre fonction.

On invoque le constructeur de la classe mère avec les arguments titre et baliseMere.

La classe définit la fonction **agit** qui prend comme paramètre cible et dont le comportement est le suivant.

On invoque la fonction fonction de l’objet avec l’argument cible.

Enfin on invoque la fonction agit définie par la classe mère sur l’objet avec l’argument cible.

Le démarrage

Le code qui est décrit ici doit être exécuté quand la page HTML est totalement chargée.

* On invoque la fonction init de l’objet AuditeurContenu.
* On définit une variable barreAOutil qui référence l’élément HTML dont l’ID vaut ‘barreAOutil’.
* On crée une instance de CommandeFormatage avec les arguments suivants : le texte "Mise en gras", l’objet barreAOutil, et un objet Javascript avec 2 propriétés : 1) propriete dont la valeur est "font-weight", 2) valeur qui est égale à "bold".
* On crée une autre instance de CommandeFormatage avec les arguments suivants : le texte "Mise en italique", l’objet barreAOutil, et un objet Javascript avec 2 propriétés : 1) propriete dont la valeur est "font-style", 2) valeur qui est égale à "italic".
* On crée une instance de CommandeAction avec 3 arguments. Les 2 premiers sont "Supprimer" et l’objet barreAOutil. Le troisième est une fonction qui prend un paramètre cible et a une seule instruction : retirer (au sens HTML) l’élément HTML cible de sa balise mère.
* On crée une instance de CommandeAction avec 3 arguments. Les 2 premiers sont "Ajouter" et l’objet barreAOutil. Le troisième est une fonction qui prend toujours un paramètre cible mais qui est un peu plus compliquée que la fonction pour la suppression :
  + On définit une variable nouveauTexte qui a comme valeur le résultat de l’invocation de la fonction prompt (voir plus bas), qui demande à l’utilisateur le texte du nouveau paragraphe.
  + Si nouveauTexte est nulle alors on sort de la fonction.
  + On définit la variable baliseMere qui vaut la balise mère de l’argument cible.
  + On crée un élément HTML P (paragraphe) qu’on met dans une nouvelle variable locale nouveauParagraphe.
  + On crée une balise textuelle dont le texte est nouveauTexte. Cette balise est ajoutée (au sens HTML) à nouveauParagraphe.
  + On ajoute nouveauParagraphe (au sens HTML) à baliseMere juste avant cible. Pour cela on utilise la fonction insertBefore (nouvelElement,élémentSuivant).

Note : la fonction prompt de JS prend comme 1er paramètre le titre du prompt/fenêtre de dialogue et comme 2ème paramètre la valeur de base qui sera dans la zone de saisie.