BTS SNIR

Lycée Louis Armand
94 - Nogent sur Marne

Module Web 02

Dynamiser mon site de <geek/>

Table des matières

Les compétences visées

</>	Catégorie	Je suis capable de :	
HTML	La balise script	Ajouter une balise script pour écrire du code en JS	
		Ajouter une balise script pour inclure un fichier JS	
		Positionner la balise script dans le code HTML	
	Les balises génériques	Ajouter une balise	
		Ajouter une balise	
	Les formulaires	Ajouter un formulaire avec la balise <form>	
		Ajouter un champ dans un formulaire avec la balise <input>	
		Personnaliser le champ d'un formulaire (<input>) avec la propriété type.	
CSS	Fichier CSS et sa structure	Maîtriser la notion de class	
		Maîtriser la notion d'id	
		Rechercher par soi-même les propriétés et les valeurs associées	
		Comprendre et coder l'imbrication de sélecteurs.	
JS	Les alertes	Afficher un message d'alert	
	Les variables	Créer une variable	
		Comprendre la notion de typage des variables	
	Les fonctions	Appeler une fonction/méthode existante	
		Créer une fonction (mot clé function)	
	Gestion des éléments	Récupérer un élément HTML avec getElementById()	
		Lire/Modifier le texte d'un élément (avec innerHTML)	
		Lire/Modifier le style d'un élément (avec l'attribut style)	
	Les événements	Gérer les événements sur un élément avec addEventListener()	
	Les chaînes de caractères	Connaître le longueur d'une chaîne de caractères	
		Analyser les caractères d'une chaîne de caractères	
	Le temps et les événements répétitifs	Comprendre la notion de timestamp	
		Gérer des événements à intervalles de temps réguliers	
GIT	Le dépôt local	Créer un nouveau dépôt local (git init)	
		Ajouter des fichiers dans le dépôt local (git add)	
		Sauvegarder les modifications dans le dépôts (git commit)	
	Le dépôt distant	Créer un dépôt distant et se synchroniser avec (git remote)	
		Publier un dépôt local dans un dépôt distant (git push)	

Défi 1 - Le JS et la balise script

Le langage JS

? Donner la définition du sigle JS et expliquer l'intérêt de ce langage.

? Le JS est un langage compilé ou interprété ? Et par qui ?

La balise <script>

La balise <script> permet au développeur d'écrire du code en JS. Le JS est assez proche du C++. Il est possible de créer des variables, appeler des fonctions ou des méthodes, utiliser des structures alternatives (if) ou itératives (for, while, ...).
Commençons par un code court en JS :

```
<script>
    alert("Bienvenue sur mon site internet !");
</script>
```

</> Taper le code ci-dessus juste avant la fermeture de la balise `</head>` dans votre fichier HTML. Puis tester-le.

? Est-il possible de placer du code JS dans la balise `<head>` ou bien cela provoque une erreur ?

? D'après votre observation, expliquer ce que fait la fonction `alert()` en JS.

</> Maintenant, déplacer le code précédent avant la fermeture de la balise `</body>`, puis tester-le.

? Est-il possible de placer du code JS dans la balise `<body>` ou bien cela provoque une erreur ?

? Expliquer la différence entre placer le code dans le `<head>` ou le placer à la fin du `<body>` ?

Conclusion

Les scripts peuvent être placés dans la section <body>, ou dans la section <head> d'une page HTML, ou dans les deux. Cependant, le fait de placer les scripts au bas de l'élément <body> améliore la vitesse d'affichage, car l'interprétation des scripts ralentit l'affichage. (Source : https://www.w3schools.com/js/js_whereto.asp)

TD 1 - Les variables en JS

Créer une variable

L'instruction `var` permet de créer une variable et éventuellement de l'initialiser.

```
var texte = "Bienvenue sur mon site internet !";
```

Dans le code précédent, une variable nommée texte a été créée et elle est de type chaîne de caractères. Les doubles quotes " ou les simples quote ' peuvent être utilisées indifféremment pour délimiter une chaîne de caractères.
Le code suivant créé une variable nommée `note1` de type réel.

```
var note1 = 15.2;
```

Le type d'une variable

En JS, le type d'une variable se décide à la première affectation. Il est important de connaître le type d'une variable pour comprendre la signification de ces opérations comme le montre les applications suivantes :

Première application: les chaînes de caractères et la concaténation

Créer 2 variables `nom` et `prénom` et leur affecter votre nom et prénom.

Créer ensuite une autre variable qui contiendra la phrase "Bonjour, je m'appelle " suivi des variables `prenom` et `nom` séparées par un espace. Vous utiliserez l'opérateur '+' pour effectuer une concaténation. Afficher ce message avec la fonction `alert()`.

Deuxième application: les chaînes de caractères et les entiers

Créer une variable nommée `numero` contenant la valeur 500.

Créer une variable nommée `texte` contenant "Tu es le visiteur n° " suivi de la valeur de `numero`. Afficher le contenu de la variable `texte` avec la fonction `alert()`.

Dans votre code précédent, donner la signification de l'opérateur '+'.

Troisième application : attention aux types !

Créer 2 variables `prix1` et `prix2` et leur affecter 5,95 et 8,90 euros.

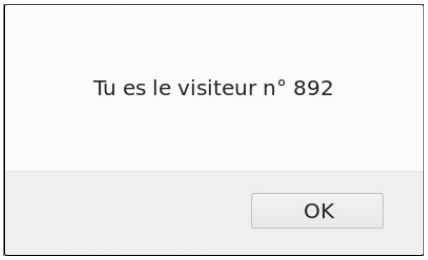
Créer un texte contenant le message "Mes 2 articles m'ont coûté" suivi de la somme de `prix1` et `prix2`. Puis afficher ce message avec le fonction `alert()`.

Conclusion : C'est le type des variables qui déterminent la signification des opérateurs. Du coup, même si en JS les variables n'ont pas de type défini lors de leur création, la notion de type reste néanmoins primordiale.

Défi 2 - Créer des variables et utiliser des méthodes

Objectif

L'objectif de ce défi est d'afficher le message suivant (voir capture ci-contre). Le numéro du visiteur doit être un nombre aléatoire compris entre 0 et 999.



Première étape – numéro de visiteur constant

</> Reprenez la deuxième application du TD précédent : créer les variables `numero` et `texte` et afficher le texte avec la fonction `alert()`. Placer ce script à la fin du body.

Deuxième étape – numéro de visiteur aléatoire

L'objet Math est un objet natif du JS : il existe sans avoir besoin de le créer. Avec cet objet, il est possible d'appeler de nombreuses méthodes dont voici la liste se trouve ici :
https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Math

Générer un nombre réel aléatoire

? En consultant la liste des méthodes disponibles avec l'objet `Math`, indiquer celle qui permet de générer un nombre aléatoire.

? D'après la documentation, le nombre aléatoire généré est compris entre quelles valeurs ?

? Modifier maintenant la valeur de la variable `numero` avec la valeur retournée par la méthode.

</> Mettre à jour votre code avec la réponse précédente.
</> Vérifier que la valeur affichée dans l'alerte soit bien entre les valeurs indiquées dans la documentation.

Un nombre réel aléatoire entre 0,0 et 999,99...

? Indiquer la valeur par laquelle il faudrait multiplier `numero` pour qu'il soit compris entre 0,0 et 999,99...

? En déduire le code permettant de générer un nombre réel aléatoire compris entre 0,0 et 999,99...

</> Modifier votre code pour que la valeur de `numero` soit comprise entre 0,0 et 999,99...
</> Vérifier que la valeur affichée dans l'alert soit bien conforme.

Un nombre entier aléatoire entre 0 et 999

Pour terminer, on souhaite que le numéro aléatoire soit un *entier*.

? Dans la liste des méthodes accessibles avec l'objet Math, trouver celle qui retourne le plus grand entier inférieur ou égal à la valeur passée en paramètre.

? En déduire le code permettant de générer un nombre entier aléatoire compris entre 0 et 999.

</> Modifier votre code pour que la valeur de `numero` soit comprise entre 0 et 999.
</> Vérifier que la valeur affichée dans l'alert soit bien conforme au cahier des charges.

TD 2 - Créer sa propre fonction

L'objectif de cette partie sera de créer une fonction appelée 'visiteur' qui contiendra le code défi précédent.
Pourquoi créer sa propre fonction ?

- Il sera ensuite possible d'appeler cette fonction n'importe quand sans retaper son code
- Il sera même possible de l'utiliser simplement dans un autre code (pour un autre site web)
- Cela permet de séparer le code écrit en JS du code HTML

Créer un fichier et déclarer une fonction

En JS, une fonction est déclarée et définie en même temps. Généralement, les fonctions sont créées dans un fichier d'extension `.js`.
Prenons un exemple. Créons un fichier `mesfonctions.js` et mettons dans ce fichier le code suivant :

```
function carre(nombre) {
  return nombre * nombre;
}
```

? Donner le nom de la fonction créée

? Donner le nombre d'argument de la fonction

? Expliquer ce que renvoie cette fonction

Appel de cette nouvelle fonction

Il est maintenant possible d'appeler cette fonction dans mon code HTML ! Comment ?
D'abord, il faut préciser à la page HTML qu'elle a besoin du fichier `mesFonctions.js` en écrivant :

```
<script src="mesFonctions.js"></script>
```

Ensuite, il est possible d'appeler cette fonction comme on a précédemment appelé la fonction alert()...

```
    <script>
        var resultat = carre(10);
        console.log("Test de la fonction : " + resultat);
    </script>
```

? Quand la fonction carre est appelée ici, quelle valeur prendra l'argument `nombre` ?

? Donner la valeur de la variable `resultat` après l'exécution de la fonction `carre()`.

? Ecrire ce qui sera affiché dans la console.

Votre fonction visiteur()

? A partir de ce qui précède, créer votre propre fonction appelée visiteur(). Cette fonction générera un nombre aléatoire et affichera dans la console et dans une alert : "Tu es le visiteur n° " suivi du nombre aléatoire.

Ce qu'il faut retenir de cette partie

En JS, le mot-clé `function` permet de créer une nouvelle fonction avec un nom et si nécessaires des arguments. Ensuite, la définition de la fonction se fait entre les accolades ouvrante et fermante.

Défi 3 - Fonction et événement en JS

Définition de la fonction visiteur() dans un fichier JS.

</> Créer un fichier nommé mesFonctions.js. Ce fichier contiendra toutes les fonctions JS que vous écrirez pour les besoins de votre site web.

</> Écrire la définition de la fonction visiteur() - voir le TD précédent.

Appel de la fonction visiteur() dans la page HTML

</> Ajouter dans votre fichier HTML que vous avez besoin du fichier mesFonctions.js.

</> Appeler la fonction visiteur() à la fin du body. Expliquer ce qui se passe.

Appel de la fonction visiteur() sur un événement

Un exemple

Il est possible d'appeler cette méthode lorsque l'utilisateur fait une action : par exemple, lorsqu'il clique sur le paragraphe du footer, ou simplement lorsque la souris le survole. Voici un exemple où la fonction visiteur() sera appelée lorsque la souris survolera le paragraphe du footer dont l'identifiant est p_footer :

```
var p_footer = document.getElementById("p_footer");
p_footer.addEventListener('mouseover', visiteur);
```

Analyse de l'exemple

? Dans le code précédent, à quoi sert la méthode getElementById() ? Que renvoie-t-elle ? (https://developer.mozilla.org/fr/docs/Web/API/Document/getElementById)

? Dans le code précédent, quel est le rôle de la méthode addEventListener() ? Que signifie les 2 arguments ? (https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener)

? En utilisant la documentation de la méthode addEventListener(), trouver notamment les événements souris qu'il est possible de gérer avec cette méthode.

Application à votre site

</> Dans votre fichier JS, en dessous de la fonction visiteurs(), ajouter le code permettant d'appeler la fonction visiteurs() lorsque l'utilisateur fait un double click sur le paragraphe du footer.

Ce qu'il faut retenir de cette partie

En JS, la méthode `getElementById()` de l'objet `document` permet de récupérer un élément de la page HTML en fournissant son identifiant (ou id).

Défi 4 - Modification en JS du code HTML d'une page

Objectif

Dans cette partie, vous allez découvrir comment modifier en JS le contenu de la page HTML ! Plus précisément, votre objectif sera de modifier le titre de l'entête du site quand la souris le survole.

Comment modifier le code HTML d'une page en JS

En JS, il est possible de modifier le code HTML se trouvant entre une balise ouvrante et une balise fermante. Dans notre cas, nous avons actuellement le code suivant dans notre header :

```
<header>
    <h1>Mes Dév! Web en SNIR</h1>
</header>
```

Nous souhaitons que lorsque la souris de l'utilisateur survole le titre de notre header, le code de la page se transforme en ceci :

```
<header>
    <h1>Par Tim Berners-Lee</h1>
</header>
```

Eh bien c'est possible : grâce au JS ! Suivez les 3 étapes suivantes :

? Mais avant, qui est Tim Berners-Lee ?

</> 1) Dans le fichier HTML, ajouter un identifiant (id) à la balise `h1` de votre `header`, par exemple '`titre_header`'

</> 2) Dans le fichier JS, créer une fonction `ChangerTitre()` qui suivra l'algorithme suivant :

```
Récupérer l'élément cible (le titre h1 du header) avec la méthode getElementById()
SI le contenu de cet élément est "Mes Dév! Web en SNIR" ALORS
    Modifier le contenu avec "Par Tim Berners-Lee"      // Remplacer par vos nom et prénom.
SINON
    Modifier le contenu avec "Mes Dév! Web en SNIR"
FIN_SI
```

Pour vous aider, utiliser l'attribut `innerHTML` de votre l'élément cible pour connaître le contenu du titre ou pour le modifier.

</> 3) Dans votre fichier JS indiquer que lorsque la souris survolera le titre h1 de votre header, il faudra appeler la fonction `ChangerTitre()`.

Pour aller plus loin

</> En vous aidant d'internet, faire en sorte que le titre revienne à sa valeur de départ 1 seconde après que l'utilisateur l'ait survolé ! Ça c'est du défi !

Ce qu'il faut retenir de cette partie

En JS, l'attribut `innerHTML` d'un élément permet de lire et/ou modifier le code se trouvant entre une balise ouvrante et la balise fermante de cet élément.

Défi 5 - Modification en JS du code CSS d'une page

Objectif

En JS, il est aussi possible de modifier le style CSS ! Par exemple, si vous souhaitez que la palette de couleur de votre page web passe du bleu au orange, pas de soucis… le JS est l'outil qu'il vous faut !

Les 2 images superposées ci-contre montre le site avec les 2 thèmes. C'est l'objectif de cette partie. Bien-sûr, une fois que vous saurez faire avec du bleu et du orange, libre à vous d'ajouter d'autres palettes de couleurs qui vous conviennent mieux !

Choix du thème : HTML et CSS

Pour commencer, ajoutez 2 carrés (un orange et un bleu) dans le footer (en bas à gauche) de votre site. Chaque carré fera 20px de côté et aura une bordure blanche de 1px. Les carrés seront espacés de 10px entre eux.

Voici une capture d'écran du résultat final :

Il existe plusieurs façons d'obtenir ce résultat. On peut très bien créer 2 images puis les afficher côte à côte avec la balise `img`. Nous retiendrons une autre solution : l'utilisation de la balise `div`.
La balise `div` permet de créer des blocs. Ici, vous en créerez 2. Grâce à une classe CSS, vous les dimensionnerez comme il convient et vous les placerez à gauche du paragraphe du `footer`.

- </> Ajouter dans votre fichier HTML les 2 `div` dans le `footer`.
- </> Ajouter une classe `theme_couleur` en CSS pour a) redimensionner la div, b) lui ajouter une bordure et c) la placer à gauche et d) les espacer de 10px.
- </> Appliquer cette classe à vos 2 `div`.
- </> Créer 2 classes bleu et orange avec la propriété `background-color` à la valeur bleue (#2874a6) ou à la valeur orange (#cc5500).
- </> Appliquer ces classes à la `div` correspondante. Vérifier que l'affichage est correct !

Gestion des événements souris sur les couleurs de thème : JS

Un clic dans le carré orange provoquera l'affichage du site en orange, un clic dans le carré bleu et le site revient dans la couleur d'origine.

Gestion du clic sur les div

- </> En HTML, donner un id à chacune des 2 div précédentes : `theme_bleu` et `theme_orange`.
- </> Écrire en JS la gestion des événements 'clic' sur ces 2 div et appeler la fonction `changer_theme()` que nous allons créer par la suite.
- </> En JS, créer une fonction `changer_theme()` qui, pour le moment, affichera simplement dans le debug de la console du navigateur (utiliser la méthode `console.debug()`) le texte : « changement de couleur ».
- </> Vérifier que lors du clic dans les 2 div, le message apparaît bien dans la console du navigateur.

> En JS, `console.debug()` permet au développeur d'afficher des messages de débogage dans la console des navigateurs web.

Connaître la source de l'événement

Dans notre cas, la fonction `changer_theme` est appelée aussi bien par la div `theme_bleu` que par la `div theme_orange`. Comment savoir qui est à l'origine de l'appel de la fonction ?
En JS, il est possible de connaître la source de l'événement grâce à `this` : l'objet source. Par conséquent, si l'objet source est `theme_bleu`, `this.id` vaudra `theme_bleu` et vice versa pour `theme_orange`.

</> Dans la fonction `changer_theme()`, tester la valeur de `this.id`. S'il vaut `theme_orange`, alors afficher le message « changement de couleur en orange » dans le debug de la console du navigateur. S'il vaut `theme_bleu`, alors afficher « changement de couleur en bleu » dans le debug.

</> Vérifier que le message change en fonction de la couleur sur laquelle vous cliquez.

Dans un événement en Javascript, `this` est l'élément qui a reçu l'événement.

Modifier le style en JS

Partons d'un exemple illustrant la modification de la bordure de la barre de navigation :

```
var ma_barre_nav = document.getElementById("ma_nav");
ma_barre_nav.style.border = "2px solid #0000FF";
```

ou en 1 seule ligne :

```
document.getElementById("ma_nav").style.border = " 2px solid #0000FF";
```

Le code précédent permet de changer le style de la bordure de la barre de navigation.
En vous aidant d'internet, trouver le code permettant de modifier la couleur de fond de la barre de navigation « ma_nav ».

</> Dans votre code HTML, donner l'identifiant ma_nav à la barre de navigation.

</> Dans la fonction changer_theme(), modifier la couleur de fond de la barre de navigation en orange ou en bleu en fonction de la div cliquée.

</> Faire la même chose avec le header, le footer et la aside.

Pour aller plus loin…

</> Ajouter de nouvelles couleurs dans le choix du thème

</> Proposer un thème avec une image de fond (un motif).

Défi 6 - La barre de navigation et le formulaire de connexion

Objectif

L'objectif de cette partie est de mettre en place les formulaires de connexion et d'inscription de votre site. Ces formulaires sont accessibles grâce à la barre de navigation.
Commençons par travailler la barre de navigation.

Mise en place de la barre de navigation

La barre de navigation est composée de 4 éléments : Mon CV Mon site Inscription Connexion

</> En HTML, mettre ces 4 éléments dans une liste non-numérotée. Fournir à ces 4 éléments un id : nav_mon_cv, nav_mon_site, nav_inscription et nav_connexion.

</> En CSS, imposer un alignement horizontal de la liste, espacer les éléments de 30px, modifier le curseur de la souris pour afficher une main avec l'index levé, souligner le texte lors du passage de la souris (avec le sélecteur `hover`).

Contenu de page HTML

La section principale devra être composée de 3 div :

- une div contenant tout votre CV : son id sera `mon_cv`
- une div contenant le formulaire d'inscription : son id sera `inscription`
- une div contenant le formulaire de connexion : son id sera `connexion`

</> En HTML, ajouter les 3 div précédentes.

</> En HTML, ajouter le titre h1 « Formulaire d'inscription » dans la div inscription.

</> En HTML, ajouter le titre h1 « Formulaire de connexion » dans la div connexion.

</> Dans la feuille de style, mettre la propriété `display : none` pour ne pas afficher par défaut les div inscription et connexion.

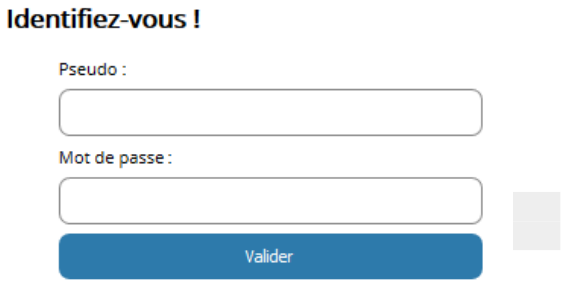
Gestion du clic sur les éléments de la barre de navigation

</> Ajouter en JS la gestion de l'événement clic sur les 4 éléments de la barre de navigation et appeler la fonction `changerSection` que nous coderons juste après.

</> Ajouter en JS une fonction `changerSection()` qui, en fonction de l'id de l'élément à l'origine de l'événement, affichera la bonne div. Par exemple, si l'id à l'origine de l'événement est `nav_inscription`, modifier la valeur de la propriété `display` de la div `inscription` à « block » et mettre la valeur de la propriété `display` des div `mon_cv` et `connexion` à « `none` ».

Le formulaire de connexion

L'objectif de cette partie est de créer un formulaire de connexion qui ressemblera au final à ceci :



? Donner le nom de la balise en HTML permettant de créer des formulaires

La balise label permet de créer un texte comme le texte Pseudo dans l'exemple :

```
<label for="pseudo">Pseudo : </label>
<input type="text" id="pseudo" name="pseudo_utilisateur">
```

? En recherchant sur internet, donner la liste des valeurs de la propriété type de la balise input qui nous intéressent dans ce formulaire.

</> En HTML, mettre en place tous les éléments du formulaire de connexion.

</> En CSS, travailler la mise en forme du formulaire pour qu'il ressemble à celui donner en exemple.

Le traitement du formulaire se fera dans un autre module.

TD 3 - Les chaînes de caractères en Javascript

Ce TD traite de quelques fonctionnalités de traitement des chaînes de caractères en Javascript.

Création d'une chaîne de caractères en JS

Il est possible d'utiliser indifféremment les guillemets doubles ou les guillemets simples pour délimiter une chaîne de caractères.

Dans les lignes suivantes, indiquez celles qui sont correctes et celles qui ne le sont pas en justifiant.

```
var monTexte = Je découvre les chaînes de caratères en JS;
var monTexte = "Je découvre les chaînes de caratères en JS";
var monTexte = 'Je découvre les chaînes de caratères en JS';
var monTexte = "Je découvre les chaînes de caratères en JS';
```

Il existe cependant une petite différence quand même entre les doubles guillemets et les simples.

Dans l'exemple suivant, indiquez quelles instructions sont correctes en justifiant.

```
var monLivre = "Je viens de terminer de lire la saga "Star Wars" !";
var monLivre = 'Je viens de terminer de lire la saga "Star Wars" !';
var maPreference = "J'ai préféré le livre au film.";
var maPreference = 'J'ai préféré le livre au film.';
```

En utilisant le caractère d'échappement '\', transformer les instructions précédentes incorrectes en instruction correctes :

En JS, les guillemets doubles et simples permettent de délimiter des chaînes de caractères. Choisissez-en une, et tenez-vous y toutefois : du code avec des mises entre guillemets diversifiées peut amener des confusions, en particulier si vous utilisez les deux sortes dans la même chaîne !

La concaténation

En JS, il est possible de concaténer (c'est-à-dire mettre à la suite) un texte ou des nombres avec un texte. L'opérateur '+' permet la concaténation.

Compléter le code suivant pour obtenir la phrase : « Je décrocherai mon BTS à XX ans », XX étant la valeur de la variable age.

```
var age =          ;    // Compléter l'instruction l'âge que vous aurez lorsque vous serez diplômé.

var phrase =  Je décrocherai mon                                          ;
```

La longueur d'une chaîne

L'attribut length des objets de type chaînes de caractères permet de connaître la longueur d'une chaîne.

Compléter le code suivant pour afficher dans la console la longueur de l'identifiant saisi par un utilisateur dans un formulaire.

```
var identifiant = document.getElementById("form_identifiant").value;

console.debug(                                          );
```

L'extraction d'un caractère d'une chaîne

La méthode charAt(position) d'une chaîne de caractères renvoie le caractère se trouvant à la position indiquée en paramètre. Par exemple, si la variable mot contient la chaîne 'azerty', mot.charAt(1) renverra le caractère 'z'.

En utilisant la méthode charAt(), affecter à la variables initiales les initiales du nom et du prénom suivant :

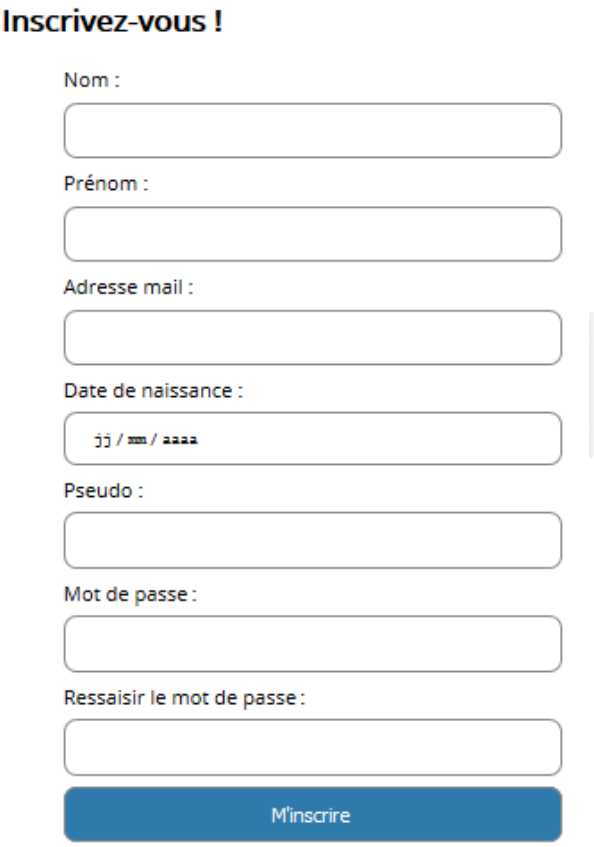
```
var nom = "Carolus";
var prenom = "Magnus";

var initiales =
```

Défi 7 - Le formulaire d'inscription

L'objectif

L'objectif de cette partie est de créer un formulaire d'inscription qui ressemblera au final à celui ci-contre.



Les champs input

? En recherchant sur internet, donner la liste des valeurs de la propriété type de la balise input qui nous intéressent dans ce formulaire.

</> En HTML, mettre en place tous les éléments du formulaire de connexion.

</> En CSS, travailler la mise en forme du formulaire pour qu'il ressemble à celui donner en exemple.

Traitement du formulaire avant l'envoi

Le traitement de ce formulaire se fera dans un autre module. Cependant, avant de l'envoyer, il peut être utile de vérifier que les informations saisies sont complètes et correctes.

Imposer des champs requis

? En vous aidant d'Internet, trouver quel attribut de la balise input permet d'imposer que le champ soit rempli avant l'envoi du formulaire. Donner un exemple.

</> Ajouter ce champ dans tous les input du formulaire d'inscription.

Vérifier que les mots de passe sont identiques

Il est possible également en Javascript de vérifier les données saisies avant d'envoyer le formulaire.

</> En HTML, donner l'id mdp1 et mdp2 aux 2 input permettant de saisir le mot de passe.

</> En JS, ajouter la gestion de l'événement 'submit' du formulaire. Appeler la fonction VerifierFormulaireInscription() que vous écrirez ensuite.

</> En JS, ajouter la fonction VerifierFormulaireInscription() selon l'algorithme suivant :

```
SI le mot de passe saisi dans mdp1 est différent de celui saisi dans mdp2 ALORS
    Afficher un messsage d'alert : Les mots de passe sont différents
    Empêcher l'envoi du formulaire  // voir l'aide ci-dessous
FIN_SI
```

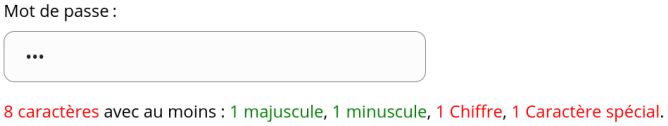
En JS, il est possible d'empêcher un événement d'accomplir sa mission grâce à la méthode `event.preventDefault()`. Cette méthode empêche la propagation de l'évènement vers sa destination normale. Dans notre exemple, lorsque les mots de passe sont différents, on bloque l'envoi du formulaire.

</> Vérifier que le la vérification des mots de passe fonctionnent correctement.

Mot de passe valide

Pour améliorer la sécurité des membres de votre site, vous leur imposerez des contraintes sur leurs mots de passe. Les mots de passe devront contenir au moins :

- 8 caractères,
- 1 lettre minuscule,
- 1 lettre majuscule,



- 1 chiffre,
- et 1 caractère spécial.

Ces contraintes seront visibles à l'utilisateur comme sur le screenshot ci-contre. Dès qu'une contrainte est remplie, elle passe en vert, sinon elle reste en rouge.

Le code HTML, la balise span et les classes vert et rouge

</> En HTML, ajouter un paragraphe contenant le texte : « 8 caractères avec au moins : 1 majuscule, 1 minuscule, 1 chiffre et 1 caractère spécial. »

</> En CSS, ajouter 2 classes : vert et rouge qui modifie la couleur du texte l'une en vert et l'autre en rouge.

</> En HTML, ajouter la balise span pour délimiter les 5 parties du texte qui seront colorés en rouge ou en vert :

1. « 8 caractères » avec l'identifiant mdp_longueur
2. « 1 majuscule » avec l'identifiant mdp_majuscule
3. « 1 minuscule » avec l'identifiant mdp_miniscule
4. « 1chiffre » avec l'identifiant mdp_chiffre
5. « 1 caractère spécial » avec l'identifiant mdp_special.

La balise `<span>` est un boîte HTML en ligne (`inline`). Il peut être utilisé pour grouper du texte afin de le mettre en forme (grâce aux attributs `class` ou `id` et aux règles CSS) ou parce qu'ils partagent certaines valeurs d'attribut comme `lang`. `<span>` est très proche de l'élément `<div>`, mais l'élément est un élément de bloc, alors que `<span>` est un élément en ligne.

</> Tester les class vert et rouge sur les balises span et vérifier leur bon fonctionnement. Puis appliquer la classe rouge sur les 5 `span` (par défaut, aucune condition n'est respectée lorsque le mot de passe est encore vide).

La gestion des événements

La vérification du mot de passe se fera à chaque fois que l'utilisateur ajoute un nouveau caractère dans son mot de passe.

En vous aidant d'Internet, trouver le nom de l'événement à utiliser dans `addEventListener()` indiquant qu'un nouveau caractère a été saisi dans l'`input` du mot de passe.

```
```

</> En JS, ajouter la gestion cet événement. Appeler la fonction `VerifierMotDePasse()` que vous créerez juste après.

Vérification du mot de passe : sa longueur

</> En JS, créer la fonction `VerifierMotDePasse()`. Vérifier la longueur du mot de passe (voir le TD précédent). Si la longueur est supérieure ou égale à 8, afficher dans le `debug` « nombre de caractères corrects », sinon afficher « nombre de caractères incorrects.

Ajoutons maintenant la gestion des couleurs. L'exemple ci-dessous montre comment changer la classe d'un élément en CSS. On peut le traduire ainsi : Si `mon_header` contient la classe bleu, alors je supprime cette classe de sa liste et je lui ajoute la classe orange.

```
if (document.getElementById("mon_header").classList.contains('bleu') )
{
  document.getElementById("mon_header").classList.remove('bleu');
  document.getElementById("mon_header").classList.add('orange');
}
```

</> Sur le même principe, gérer le changement de couleur si le nombre de caractères atteint les 8.

Vérification du mot de passe : le nombre de majuscules, minuscules, chiffres et caractères spéciaux.

Il faut maintenant compter dans le mot de passe le nombre de majuscules, minuscules, chiffres et caractères spéciaux pour savoir si les règles sont respectées.

</> Dans la fonction `VerifierMotDePasse()`, faire une boucle pour parcourir tous les caractères du mot de passe puis les afficher dans le `debug` avec la méthode `CharAt()` - voir le TD précédent. Tester et vérifier.

</> Dans la boucle, vérifier si le caractère est une majuscule. Si oui, incrémenter de 1 la variable nbMajuscules (cette variable sera préalablement déclarée et initialisée à 0). Une fois la boucle terminée, afficher dans le debug le nombre de majuscule. Tester votre code et vérifier.

</> Procéder de la même façon pour les minuscules, les chiffres et les caractères spéciaux.

</> Ajouter ensuite la gestion des classes vert et rouge sur les `span`.

Vérification du mot de passe avant l'envoi

Dans l'état actuel de votre code, l'utilisateur peut envoyer le formulaire d'inscription même si toutes les règles concernant le mot de passe ne sont pas respectées (Faites le test pour le constater…)
Voici la solution que nous retiendrons :
Lorsque l'utilisateur envoie le formulaire, la fonction `VerifierFormulaireInscription()` doit en plus vérifier que le mot de passe respecte les règles. Elle appellera alors la fonction `VerifierMotDePasse()` qui lui renverra `true` si tout est bon ou `false` sinon.

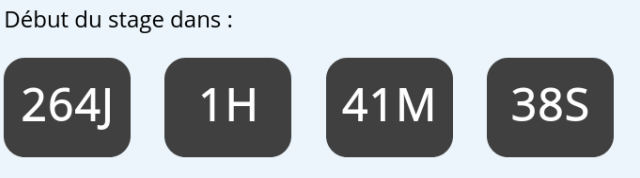
</> A la fin de la fonction `VerifierMotDePasse()`, renvoyer `true` si le mot de passe est correct ou `false` sinon.

</> Dans la fonction `VerifierFormulaireInscription()`, appeler la fonction `VerifierMotDePasse()`. Si la fonction renvoie `false`, alors afficher une `alert` : « La sécurité sur le mot de passe n'est pas respectée. » puis empêcher l'envoi du formulaire.

Défi 8 - Un compteur de <geek/>

Objectifs

L'objectif de ce défi est de rajouter un compteur dans votre aside. Ce compteur affichera le nombre de jours restant avant le début de votre stage. Il devra s'actualiser toutes les secondes. Il pourra ressembler au compteur ci-contre.



HTML et CSS

</> En HTML, ajouter 4 div, donner leur des id qui ont un sens.
</> En CSS, personnaliser ces div avec une classe.
</> Tester l'affichage en mettant un texte fixe dans vos 4 div.

Rendre dynamique le compteur avec du JS

Pour commencer, vous allez mettre à jour l'affichage en cliquant sur la première div.

</> Gérer l'événement clic sur la première div en JS et appeler la fonction MettreAJourLeCompteur()
? En vous aidant d'Internet, trouver ce qu'est un timestamp en JS.

</> Dans la fonction MettreAJourLeCompteur(), créer 2 variables contenant le timestamp du moment actuel et le timestamp du début du stage (vous pouvez vous servir de Date.now() et Date.UTC() - consulter la doc !). Afficher ces 2 valeurs dans le debug.
</> Calculer le nombre de jours qui séparent ces 2 timestamp et mettre à jour la div correspondante.
</> Faire de même avec les div suivantes.
</> Enfin, en JS, appeler la fonction MettreAJourLeCompteur() toutes les secondes.