BTS SNIR

Lycée Louis Armand
94 - Nogent sur Marne

Module Web 02

Dynamiser mon site web en JS

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)	

PARTIE 1
INTRODUCTION AU JS

TD 1 - Présentation du langage

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 ?

Défi 1 - Mon premier code JS

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("Test d'affichage d'un message");
</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>` ?

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)

Défi 2 - Écrire du JS dans un fichier séparé

Objectif

Pour un développeur, il est essentiel de garder un code qui soit clair et réexploitable. Or mélanger dans un même fichier du HTML et du JS rend le code complexe. Il est de loin préférable de les séparer. Comment ? En créant des fichiers spécifiques pour le code écrit en JS. Ces fichiers auront pour extension .js.

Création d'un fichier test.js

Vous allez créer un nouveau fichier « test.js » qui vous servira simplement à faire des tests en JS pour découvrir ce langage et ses fonctionnalités.

</> Dans votre répertoire de travail, créer un nouveau fichier nommé test.js.

</> Mettre dans le fichier « test.js » le code JS qui se trouve entre les balises script du fichier index.html.

</> Remplacer la balise script du fichier index.html par le code suivant :

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

</> Vérifier que votre message d'alerte s'affiche toujours !

Les informations de débogage

Très souvent, le développeur veut afficher des informations, des données, des variables – juste pour en connaître l'état. Il ne souhaite pas que ces données soient par le visiteur.
En JS, il est possible d'utiliser la console de débogage.

</> Remplacer maintenant le message d'alerte du fichier test.js par :

```
console.debug("Test d'affichage d'un message");
```

? Rafraîchissez votre page. Le message apparaît-il ?

Le message apparaît dans la console de débogage !

</> Appuyer sur la touche F12 (pour afficher l'inspecteur d'objet) puis sélectionner l'onglet Console (éventuellement sélectionner d'afficher les messages de debug). Vous devez constatez que le message apparaît.

Défi 3 - Les variables en JS

Les variables en JS

</> Remplacer maintenant le code du fichier test.js par le suivant puis tester-le. Afficher la console.

```
let message = "La somme fait : ";
let val1 = 15;
let val2 = 4;
let resultat = val1 + val2;
console.log("VAL1 vaut : " + val1);
console.log("VAL2 vaut : " + val2);
console.log("RESULTAT vaut : " + resultat);
console.log(message + resultat);
console.log(message + val1 + val2);
```

? Combien de variables sont créées et quel est le type de chacune de ces variables ?

? Expliquez la différence d'affichage des 2 dernières lignes.

Le mot clé **let** permet de créer des variables. Même si en JS les variables n'ont pas de type défini lors de leur création, la notion de type reste néanmoins primordiale et détermine la signification des opérations.

PARTIE 2
ORGANISATION DES FICHIERS

Défi 4 - Organisation du répertoire de travail

Objectif

L'objectif de cette partie est d'organiser votre répertoire de travail pour s'y retrouver plus facilement.

L'organisation

Vous allez organiser votre répertoire selon le type des fichiers : tous les fichiers CSS ensembles, tous les fichiers JS ensembles et les fichiers HTML également. Nous allons aussi déplacer toutes les images utilisées dans un répertoire dédié.

Voici donc l'organisation retenue :

</> Appliquer cette organisation retenue dans vos fichiers

</> Modifier les liens vers les fichiers en ajoutant les répertoires créés pour les images, les fichiers CSS et les fichiers JS.

PARTIE 3
LA BARRE DE NAV POUR SMARTPHONE

TD 2 - Évènements et fonctions en JS

Présentation

L'objectif de cette partie est de comprendre la gestion des évènements en JS. Un évènement est déclenché généralement suite à une action de l'utilisateur : un clic sur un bouton, une saisie d'un texte, etc. Lorsqu'un évènement est déclenché, il est possible d'appeler une fonction.

Gestion des évènements

En JS, la gestion d'un évènement se fait en 2 étapes :

1. On récupère l'objet sur lequel l'utilisateur interagit
2. On indique quel évènement de cet objet nous intéresse et la fonction qui sera exécutée.

Un exemple

Supposons que nous souhaitons afficher un message popup lorsque la souris de l'utilisateur survole le footer de notre page. La première étape sera de récupérer l'objet (notre footer) avec la méthode getElementById() comme dans l'exemple suivant (p_footer est l'id du footer) :

```
let p_footer = document.getElementById("p_footer");
```

Ensuite, nous allons indiquer quel évènement sur cet objet nous intéresse et quelle fonction sera appelée :

```
p_footer.addEventListener('mouseover', mon_popup);
```

Il reste maintenant à créer la fonction mon_popup() comme suit :

```
function mon_popup() {
  alert("Gestion de l'évènement 'mouse over' sur mon footer");
}
```

A vous de jouer

Gérer l'évènement 'click' sur l'image du hamburger (son id est 'hamburger') et exécuter la fonction *afficherMasquerBarreNavigation()*. Écrire le code cette fonction qui pour le moment affichera le message « click sur hamburger » dans la console.

En JS, la méthode *getElementById()* permet de récupérer un objet du corps de la page connaissant son identifiant. La méthode *addEventListener()* permet de gérer les évènements d'un objet.

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 5 - Le hamburger du smartphone

Ajout du hamburger (HTML et CSS)

Ajouter une image, lui donner un id et le designer

</> Dans le fichier index.html, ajouter en HTML une balise *img* (juste au-dessus du titre du header) dont la source sera MyHamburger.png. Cette image aura pour id « hamburger ».

Cette image ne doit pas s'afficher pour les écrans de type bureau.

</> Dans le fichier design.css, ajouter le code CSS permettant de ne pas afficher l'image (utiliser la propriété display).

```

```

</> Dans le fichier design.css, ajouter le code CSS permettant d'afficher le hamburger pour les écrans dont la largeur est inférieure à 800px avec les propriétés suivantes : display : block (valeur par défaut des éléments de type img), *float : right ;* pour que l'image se trouve à droite, une marge de 10px tout autour de l'image, pas de marge intérieure et une hauteur de 50px.

```

```

Gestion de l'évènement 'click' sur le hamburger

</> Créer un fichier navigation.css. Ce fichier sera dédié à la gestion du hamburger et l'apparition de la barre de navigation uniquement sur smartphone.

</> En vous aidant du TD précédent, gérer l'évènement 'click' sur le hamburger et appeler la fonction *afficherMasquerBarreNavigation()*. Ajouter la fonction *afficherMasquerBarreNavigation()* qui affichera un message dans la console. Vérifier que ce message s'affiche bien à chaque fois que vous cliquez sur le hamburger.

</> Écrire la fonction *afficherMasquerBarreNavigation()* en respectant l'algorithme suivant :

```
VARIABLE maBarreNav ← document.getElementById(« barre_navigation »)
SI maBarreNav.style.display VAUT «grid» vide ALORS
    maBarreNav.style.display ← «»
SINON
    maBarreNav.style.display ← «grid»
FIN_SI
```

? Expliquer le code ci-dessus. Vérifier son fonctionnement en changeant manuellement les valeurs dans l'inspecteur d'élément (F12).

Personnalisation de la barre de navigation pour les smartphones

Vous allez maintenant personnaliser l'affichage de la barre de navigation. Au lieu d'afficher un menu horizontal, le menu devra être vertical.

</> Modifier le fichier ossature_grid.css pour que l'affichage de la barre de navigation soit sur 4 lignes pour les écrans inférieurs à 800px.

</> Dans le fichier *design.css*, ajouter les propriétés suivantes pour les balises div de la barre de navigation si la largeur de l'écran est inférieure à 800px :
- alignement du texte à gauche
- une marge intérieure de 30px en haut et en bas et de 0px sur la droite et la gauche.
- une police de 30px

</> Dans le fichier *design.css*, ajouter les propriétés suivantes pour les balises *a* de la barre de navigation si la largeur de l'écran est inférieure à 800px :
- une police de 30 px
- des petites majuscules
- pas de soulignement (propriété text-decoration)

PARTIE 4
Thème clair / Thème sombre

TD 3 - Les « Custom Properties » et les « Attributes »

Objectif de cette partie

Dans cette partie, nous allons parler de 2 concepts : les attributs et les « custom properties » qu'on pourrait simplement traduire par « variables ». La combinaison de ces 2 concepts nous permettra facilement de de réaliser 2 thèmes pour notre site (un thème clair et un thème sombre). Enfin, nous verrons comment nous pourrons passer d'un thème à un autre grâce au JS !

Les attributs

Les attributs en HTML

Vous avez déjà rencontré cette notion d'attribut dans le premier module. Reprenons cela avec un exemple :

```
<img src="MyHamburger.png" alt="Le hamburger du menu smartphone">
```

? Dans le code ci-dessus, donner le nom de la balise, les attributs et leurs valeurs.

Le langage HTML impose des noms aux attributs : c'est le cas des attributs src et alt.

? Comment connaître la liste de tous les attributs possible d'une balise ?

Mais chose intéressante, le langage HTML nous autorise à créer nos propres attributs (à condition qu'ils commencent par « data- ») avec nos propres valeurs. Par exemple, il est possible d'écrire :

```
<body data-theme="light">
```

ou selon le besoin :

```
<body data-theme="dark">
```

? Donner le nom de l'attribut qui est utilisé ici. En vous aidant de sites internet de référence, quel nom donne-t-on à ce genre d'attribut ?

Personnalisation CSS en fonction des attributs

En CSS, il est possible de personnaliser le style d'une balise en fonction de la valeur de ses attributs. Prenons l'exemple suivant :

```
body[data-theme="light"]{
    backgroung-color: white;
}

body[data-theme="dark"] {
    backgroung-color: black;
}
```

? D'après le code ci-dessus, quel changement sur la page y-aura-t'il suivant la valeur de l'attribut data-theme ?

Les « custom properties » ou les variables en CSS

L'état des lieux sans les custom properties

En CSS, le code peut-être parfois répétitif et difficile à maintenir. C'est le cas notamment lorsqu'on définit la palette de couleur d'un site internet. Par exemple, quand l'entête, la barre de navigation et le pied de page par exemple ont la même couleur de fond, ça donne quelque chose comme :

```
header {
    background-color : #54679A;
}
nav {
    background-color : #54679A;
}
footer {
    background-color : #54679A;
}
```

? Combien de lignes doivent être modifiées si l'on souhaite changer la couleur de fond de notre page ?

Et avec les custom properties ?

Voici le même code avec les custom properties :

```
body {
    --bg-color: #54679A;
}
header {
    background-color : var(--bg-color);
}
nav {
    background-color :  var(--bg-color);
}
footer {
    background-color :  var(--bg-color);
}
```

? D'après le code ci-dessous, comment définit-on une variable en CSS ?

? D'après le code ci-dessus, comment fait-on référence à une variable en CSS.

? Combien de lignes doivent être modifiées si l'on souhaite changer la couleur de fond de notre page ?

? Quelle est la portée d'une variable CSS déclarée dans le body comme dans l'exemple ci-dessus ?

Défi 6 - Le thème clair et le thème sombre

Objectifs

L'objectif de cette partie est d'ajouter un thème clair/sombre au design de votre site. L'utilisateur pourra passer de l'un à l'autre en cliquant sur un bouton dans le footer.

Dans le fichier index.html

</> Modifier la balise body pour ajouter 2 attributs : un id valant « body » et un data-theme valant «light»

</> Ajouter dans le footer un bouton dont l'id vaut « dark_light » et dont le texte est « Mode sombre »

Dans le fichier design.css

Il y a ici pas mal de travail. Faites-le consciencieusement. Nous allons définir dans notre fichier 7 variables contenant des couleurs :

- Pour le header et le footer :
  - *--hf-bg-color* : *h*eader *f*ooter *b*ackground *color*
  - *--hf-text-color* : *h*eader *f*ooter *text color*;
- Pour la barre de navigation :
  - *--n-bg-color* : *n*avbar *b*ackground *color*
  - *--n-text-color* : white;
- Pour la section et le aside :
  - *--sa-title-color* : section *a*side *title color*
  - *--sa-text-color* : section *a*side *text color*
  - *--sa-bg-color* : section *a*side *b*ackground *color*

</> Ajouter dans le fichier design.css les styles conditionnels body[data-theme="light"] et body[data-theme="dark"].

</> Dans la partie body[data-theme="light"], définir les variables avec les valeurs adaptées pour le thème clair.

</> Dans la partie body[data-theme="dark"], définir les variables avec les valeurs adaptées pour le thème sombre.

</> Dans le code CSS, changer toutes les couleurs par la variable CSS qui convient.

</> Tester votre code en passant du thème clair au thème sombre manuellement dans le fichier HTML.

Gestion de l'évènement sur le bouton

</> Ajouter un nouveau fichier nommé « dark_ligth.js » qui servira uniquement à la gestion du thème du site.

</> Dans ce fichier JS, ajouter la gestion de l'évènement « click » sur le bouton dont l'id est dark_ligth, puis appeler la fonction *dark_ligth()*.

</> Toujours dans ce fichier JS, créer la fonction dark_ligth(). Cette fonction respectera l'algorithme suivant :

```
Si l'attribut data-theme du body vaut « light » ALORS
    Changer l'attribut data-theme par « dark »
    Changer le texte du bouton par « Mode clair »
SINON
    Changer l'attribut data-theme par « ligth »
    Changer le texte du bouton par « Mode sombre »
FIN_SI
```

Pour vous aider : pour traduire ce code en JS, vous devrez trouver les fonctions en JS qui permettent :

- de récupérer la valeur d'un attribut d'une balise
- de modifier la valeur d'un attribut d'une balise
- de modifier le texte d'un bouton (le texte qui se trouve entre les balises <button> et </button>)

Par aller plus loin

Au chargement de la page, récupérer l'heure actuelle. Si nous sommes entre 7h et 19h, appliquer le thème clair, sinon appliquer le thème sombre.

PARTIE 5
VÉRIFICATION DES FORMULAIRES AVANT ENVOI

TD 4 - 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.

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

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

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

let 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.

```
let 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 :

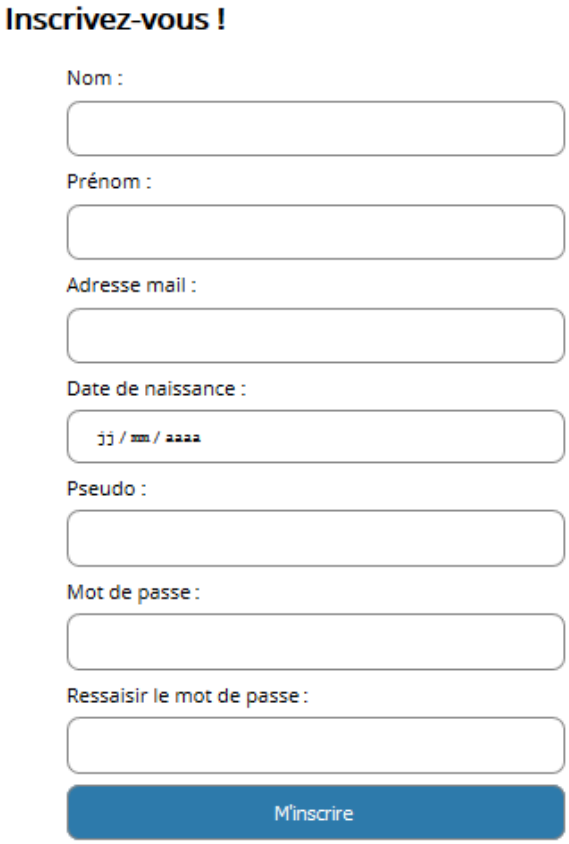
```
let nom = "Carolus";
let prenom = "Magnus";

let initiales =
```

Défi 7 - Le formulaire d'inscription

L'objectif

L'objectif de cette partie est vérifier le formulaire d'inscription avant de l'envoyer.



Vérification 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 soient 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
```

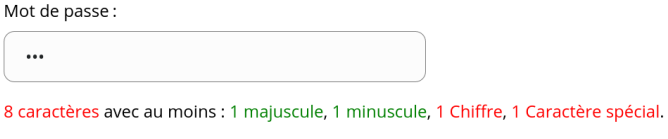
En JS, il est possible d'empêcher un événement d'accomplir sa mission grâce à la méthode `event.preventDefault()`. Cette méthode empêche la propagation de l'évènement vers sa destination normale. Dans notre exemple, lorsque les mots de passe sont différents, on bloque l'envoi du formulaire.

</> 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.

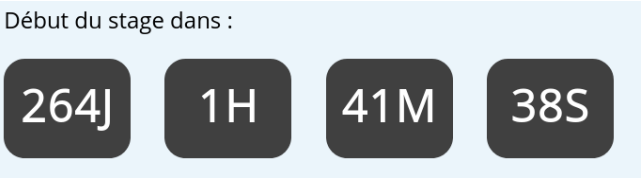
</> Dans la fonction `VerifierFormulaireInscription()`, appeler la fonction `VerifierMotDePasse()`. Si la fonction renvoie `false`, alors afficher une `alert` : « La sécurité sur le mot de passe n'est pas respectée. » puis empêcher l'envoi du formulaire.

PARTIE 5
UN COMPTEUR

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 l'hébergement de votre site internet sur les mini-pc lors du module web 3. 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.