

Intégration Web

2020 - MMI 1 – TP#5 S1





Danielo **JEAN-LOUIS**
Développeur front-end

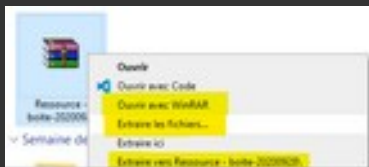
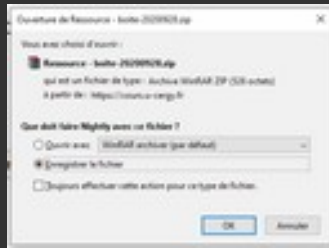
But du cours

- Voir les bases de la programmation d'un site web
 - Langage HTML
 - Langage CSS
- Sensibilisation au web : design et programmation
 - Accessibilité
 - Bonnes pratiques
- Avoir les connaissances pour développer un site web

Ressource du cours

**Travaux Pratiques #5 > "Ressource - Formulaires"
sur ENT**

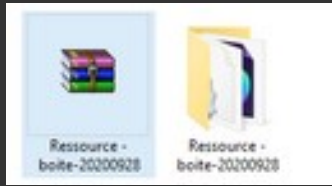
Petit point sur les ressources de cours



Menu contextuel sur l'archive (clic droit). Choisir un des choix surligné.

- Les ressources se présentent sous la forme d'une archive une fois téléchargées
- L'archive doit être impérativement extraite sinon vous ne serez pas capables de réaliser correctement les tp

Petit point sur les ressources de cours



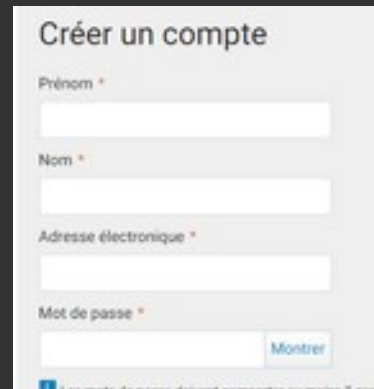
- Une fois extraite, vous devez obtenir un dossier avec le contenu de l'archive
- Libre à vous de travailler dans le dossier OU en copier le contenu pour le coller ailleurs
- **En aucun vous devez travailler dans l'archive**
- Une fois fait, glissez le dossier dans VS Code ou Sublime Text



- Web 1.0 : Web statique, l'internaute ne peut que lire les pages, il ne peut pas interagir avec
- Web 2.0 : Web social. L'internaute peut interagir avec le site notamment en postant des messages
- Web 3.0 : Web personnalisé. Le site anticipe pour vous vos besoins. E.g. les annonces basées sur vos recherches

Formulaires

- Ensemble de balises permettant à l'utilisateur d'envoyer des données à un serveur
 - Formulaire de connexion / inscription
 - Clavardage (ou chat)
 - Formulaire de contact
 - [...]



The image shows a web form titled "Créer un compte". It contains four input fields, each with a red asterisk indicating it is required: "Prénom", "Nom", "Adresse électronique", and "Mot de passe". The "Mot de passe" field has a "Montrer" button next to it. At the bottom, there is a small blue icon and a line of text that is partially cut off: "Les mots de passe doivent contenir au moins 8 caractères".

Balise <form>

- Balise principale d'un formulaire
 - Tous les éléments d'un formulaire doivent s'y trouver
- Attribut "method", trois valeurs possibles :
 - "get" : données passent dans l'url. Limitation à 255 caractères historiquement
 - "post" : données passent dans le corps de la requête. Pas de limite de caractères
 - "dialog" : ferme un formulaire si et seulement si le formulaire est dans une balise <dialog>
- Les valeurs de l'attribut "method" ne sont pas sensibles à la classe

Sources :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/Form>

A la base tous les éléments devaient être dans le formulaires pour être considérés valides, mais avec html5, d'associer un élément de formulaire sans qu'il soit dedans, il faut juste que l'élément ait l'attribut "form" et la valeur de l'attribut doit être valeur d'un id de formulaire. Le cas échéant, le champ sera rattaché au formulaire le plus proche qui le contient s'il existe.

```
<form action="traitement.php" method="get">  
  <!-- Contenu du formulaire -->  
</form>
```

Exemple de squelette de formulaire

Balise <form> - Envoi des données

- Attribut "action"
 - Définit l'adresse du fichier qui va traiter les données
- Ensemble de paires clés / valeurs
- Clé définie par l'attribut "name"
- Valeur définie par l'attribut "value" (sauf pour la balise <textarea></textarea>)

Sources :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/Form>

A la base tous les éléments devaient être dans le formulaires pour être considérés valides, mais avec html5, d'associer un élément de formulaire sans qu'il soit dedans, il faut juste que l'élément ait l'attribut "form" et la valeur de l'attribut doit être valeur d'un id de formulaire. Le cas échéant, le champ sera rattaché au formulaire le plus proche qui le contient s'il existe.

Balise <input />

- Balise auto-fermante
- Balise à la fois "inline" et "block" → "inline-block"
- N'accepte que du texte monoligne
- Nécessite l'attribut name pour que sa valeur puisse être récupérée
 - valeur de l'attribut invisible pour l'être humain
- Attribut "value" permet de définir une valeur par défaut
 - Cette valeur sera plutôt chargée par le serveur
- Permet l'auto-remplissage. Attribut "autocomplete"

Sources :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/input/text>
- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/input>
- <https://developer.mozilla.org/fr/docs/Web/HTML/Attributes/autocomplete>

Balise <input />

- Attribut "type" pour définir le type de données qui peut être entré
 - text : texte
 - number : nombre
 - password : mot de passe
 - email : courriel (nouveau html5)
 - tel : téléphone (nouveau html5)
 - hidden : champ caché
 - [...]

Sources :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/input/text>
- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/input>

La valeur de l'attribut "type" a une grosse influence sur le type de clavier affiché sur terminaux mobiles. Avec l'affichage du bon clavier, on rend l'expérience utilisateur bien plus agréable.



Sur mobile (ici iOS) le clavier change en fonction du champ dépendamment de valeur de son attribut "type"

Sources :

- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/input/text>
- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/input>
- <https://www.smashingmagazine.com/2019/01/html5-input-types/?ref=heydesigner> - anglais

Le clavier s'est adapté au type de champ.
L'utilisateur a moins de chance de se tromper et
ça réduit les risques de frustrations et donc
d'abandon.

Balise <label></label>

- Décrit le rôle d'un champ pour un être humain
 - Ne doit en aucun cas être substitué par l'attribut "placeholder" de la balise <input />
- Attribut facultatif mais indispensable
 - "for" : associe un label à un champ
Le champ associé doit avoir l'attribut "id" avec la même valeur

```
<label for="prenom">Prénom</label>  
<input type="text" name="name" id="prenom" />
```

Sources :

- <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/label>
- https://www.w3schools.com/howto/howto_css_form_icon.asp - mauvaise pratique

En liant la balise <label> à élément de formulaire grâce à l'attribut "for", il est possible de cliquer sur le label associé pour placer le curseur dans le champ de texte.

Il est possible de se passer de l'attribut "for" si l'élément de formulaire est compris dans la balise <label>

Pratiquons ! - Ajoutons notre premier formulaire

Pré-requis :

- Avoir le contenu de la ressource "formulaire-basique" sur ENT
- Ajouter une balise `<form>` avec la valeur "#" pour l'attribut "action"
- Ajouter une balise `input` avec son label en prenant soin de rendre le tout accessible, et y entrer une valeur

Pratiquons ! - Ajoutons notre premier formulaire

Pré-requis :

- Avoir le contenu de la ressource "formulaire-basique" sur ENT
- Ajouter dans le même formulaire un autre champ input avec un type "intelligent" (exemple "email") et mettre une valeur incorrecte. Ne pas oublier le label avec l'attribut "for"

Soumission du formulaire

- Balise `<input>` avec la valeur "submit" pour l'attribut "type" et l'attribut "value" pour le libellé
- Balise `<button>` avec la valeur "submit" pour l'attribut "type"
 - Attention : en absence d'attribut "type" le navigateur tranche en fonction du contexte quant à son comportement
 - Balise fermante : Permet donc de mettre du texte + image dans le bouton par exemple
- Se place dans le formulaire à la fin de préférence

L'élément de formulaire permettant sa soumission peut se placer à l'extérieur si et seulement si, le bouton ou input possède l'attribut "form" avec comme valeur l'id du formulaire qu'il doit soumettre.

Pratiquons ! - Améliorons notre premier <form>

Pré-requis :

- Avoir le contenu de la ressource "formulaire-basique" sur ENT et fait les pratiques précédentes
- Rajouter un élément de formulaire permettant sa soumission
- Jouer sur la propriété "method" et ses valeurs ("get" et "post") puis soumettre le formulaire
 - Si target="GET" regardez l'url de votre page, vous verrez les données dedans.
- Décommenter la ligne du fichier html
 - `<script src="scripts/index.js"></script>` (vers la ligne 40)
- Soumettre le formulaire
- Regarder l'onglet "console" de la console du navigateur

Bonnes pratiques – Champ de formulaire

- Un champ doit avoir :
 - Un label qui lui est associé (avec l'attribut "for")
 - Les attributs suivants
 - Un attribut "id" avec une valeur identique à celle de son label associé
 - Un attribut "name"
 - Un attribut "type", de préférence en accord avec la donnée attendue

Balise <textarea></textarea>

- Balise acceptant du texte multi-lignes
- Le retour à la ligne doit être fait par l'utilisateur
- La valeur par défaut n'est pas dans l'attribut "value" mais entre les deux balises
- Utilisation des propriétés "rows" et "cols" pour définir le nbre de lignes et colonnes visibles
- La balise, par défaut, peut se redimensionner
 - **Très mauvaise idée de désactiver cette option**

Source :
• <https://developer.mozilla.org/fr/docs/Web/HTML/Element/textarea>

L'élément de formulaire permettant sa soumission peut se placer à l'extérieur si et seulement si, le bouton ou input possède l'attribut "form" avec comme valeur l'id du formulaire qu'il doit soumettre.

Éléments à cocher

- Permettent d'effectuer des choix parmi des choix prédéfinis
- Balise input avec une valeur spécifique pour l'attribut "type" :
 - checkbox : Permet de sélectionner plusieurs éléments
 - radio : Permet de sélectionner qu'un seul élément

Source :

- https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Formulaires/Les_blocs_de_formulaires_natifs#%C3%89l%C3%A9ments_%C3%A0_cocher

L'élément de formulaire permettant sa soumission peut se placer à l'extérieur si et seulement si, le bouton ou input possède l'attribut "form" avec comme valeur l'id du formulaire qu'il doit soumettre.

Éléments à cocher

- Doivent être associés à un label respectif pour qu'ils soient compréhensibles pour l'être humain
- Attribut "checked" permet de pré-sélectionner une valeur (ou plus)
- Nécessitent l'attribut "name" pour savoir quelle est le nom de la valeur envoyée

Source :

- https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Formulaires/Les_blocs_de_formulaires_natifs#%C3%89l%C3%A9ments_%C3%A0_cocher

L'élément de formulaire permettant sa soumission peut se placer à l'extérieur si et seulement si, le bouton ou input possède l'attribut "form" avec comme valeur l'id du formulaire qu'il doit soumettre.

Element à cocher – Bouton radio

- Nécessite la valeur "radio" pour l'attribut "type" de la balise <input />
- Ne permet de sélectionner qu'un seul choix
- Ne permet pas de désélectionner un choix
- Les boutons liés doivent avoir la même valeur pour l'attribut "name"

```
<input type="radio" id="gaucher" name="habilite" value="gaucher">  
<label for="gaucher">Gaucher</label><br>  
<input type="radio" id="droitier" name="habilite" value="droitier">  
<label for="droitier">Droitier</label><br>  
<input type="radio" id="ambidextre" name="habilite" value="ambidextre">  
<label for="ambidextre">Ambidextre</label><br>
```

Source :

- https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Formulaires/Les_blocs_de_formulaires_natifs#%C3%A9ments_%C3%A0_cocher

L'élément de formulaire permettant sa soumission peut se placer à l'extérieur si et seulement si, le bouton ou input possède l'attribut "form" avec comme valeur l'id du formulaire qu'il doit soumettre.

Element à cocher – Case à cocher

- Nécessite la valeur "checkbox" pour l'attribut "type" de la balise <input />
- Permet de sélectionner tous les choix
- Permet de désélectionner d'un choix

```
<input type="checkbox" name="menu[]" value="fromage" id="fromage" />  
<label for="fromage">Fromage</label>
```

On n'oublie de mettre un <label> pour qu'on puisse cliquer dessus

Source :

- https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Formulaires/Les_blocs_de_formulaires_natifs#%C3%A9ments_%C3%A0_cocher

L'élément de formulaire permettant sa soumission peut se placer à l'extérieur si et seulement si, le bouton ou input possède l'attribut "form" avec comme valeur l'id du formulaire qu'il doit soumettre.

Case à cocher – Envoi des données

- Tableau ou valeur unique

```
<input type="checkbox" name="menu[]" value="plat" id="plat" />  
<label for="plat">Plat</label>  
  
<input type="checkbox" name="menu[]" value="fromage" id="fromage" />  
<label for="fromage">Fromage</label>
```

Les données peuvent être envoyées dans un tableau, la notation nomDuChamp[] permet ceci

```
<input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">  
<label for="vehicle1"> I have a bike</label><br>  
<input type="checkbox" id="vehicle2" name="vehicle2" value="Car">  
<label for="vehicle2"> I have a car</label><br>
```

Ou les données peuvent être envoyées dans une nom de champ distinct

L'élément de formulaire permettant sa soumission peut se placer à l'extérieur si et seulement si, le bouton ou input possède l'attribut "form" avec comme valeur l'id du formulaire qu'il doit soumettre.

Pratiquons ! - Améliorons notre premier <form>

Pré-requis :

- Avoir le contenu de la ressource "formulaire-basique" sur ENT
- Rajouter un textarea avec le label associé
- Rajouter des inputs type radio (min. 2)
- Rajouter des inputs type checkbox
- S'arranger pour que ça ressemble à quelque chose. Flexbox vous aidera

Liste déroulante – Balise `<select>`

- Permet de sélectionner un choix parmi une liste de choix définis
 - Possibilité de sélectionner plusieurs éléments (attr "multiple")
- Balise `<select>` qui contient des balises `<option></option>`
 - Possibilité de grouper des options via la balise `<optgroup></optgroup>`
 - Chaque option doit avoir un attribut "value"
 - Un `<option>` sans `<select>` n'est pas valide
- Attribut "selected" pour présélectionner un élément
- Possibilité de personnalisation très limitée

Source :
• <https://developer.mozilla.org/fr/docs/Web/HTML/Element/select>

L'élément de formulaire permettant sa soumission peut se placer à l'extérieur si et seulement si, le bouton ou input possède l'attribut "form" avec comme valeur l'id du formulaire qu'il doit soumettre.

Point accessibilité – Personnalisation

- Beau n'est pas égal à fonctionnel
- Personnaliser des éléments de formulaire pour qu'ils soient beaux est une très mauvaise idée
- Supprime / endommage le côté accessible des éléments

Sources :

- https://www.w3schools.com/howto/howto_css_form_icon.asp

Pratiquons ! - Améliorons notre premier <form>

Pré-requis :

- Avoir le contenu de la ressource "formulaire-basique" sur ENT et fait les pratiques précédentes
- Rajouter une liste déroulante simple
- Rajouter une liste déroulante avec optgroup

Source :

- <https://accessify.com/tools-and-wizards/developer-tools/insta-select/> - Pour générer une liste rapidement



Quiz

"Test - Formulaires" sur ENT

Conditions du test :

- ~8 minutes
- Une tentative

Validation des formulaires

- Do not trust user input / Ne pas faire confiance aux entrées utilisateur
- Validation automatique mais... on peut tricher
 - **Une validation côté serveur est indispensable**
- La valeur de l'attribut "type" permet (parfois) d'avoir une règle de validation explicite
- Possibilité de désactiver la validation native avec l'attribut "novalidate"

Source :

- https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Formulaires/Validation_donnees_formulaire
- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/Form#attr-novalidate>

- Il est possible de définir une valeur par défaut si la variable n'existe pas. Si la valeur par défaut n'est pas valide non plus alors la déclaration sera jugée invalide
- L'héritage existe aussi avec les variables css, si une variable est redéfinie plus tard, c'est cette valeur qui s'applique

Validation des formulaires – Champs requis

- Attribut "required"
 - Empêche la validation d'un formulaire si la valeur du champ est absente
- Pseudo-classe ":required" pour styliser dans le CSS
 - ":valid" : quand le champ est valide
 - ":invalid" : quand le champ est invalide
 - ":optional" : quand le champ est facultatif
 - ":checked" : quand le champ a été coché

Sources :

- <https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/required> – anglais
- https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Formulaires/Validation_donnees_formulaire

- Il est possible de définir une valeur par défaut si la variable n'existe pas. Si la valeur par défaut n'est pas valide non plus alors la déclaration sera jugée invalide
- L'héritage existe aussi avec les variables css, si une variable est redéfinie plus tard, c'est cette valeur qui s'applique

Pratiquons ! - Améliorons notre premier <form>

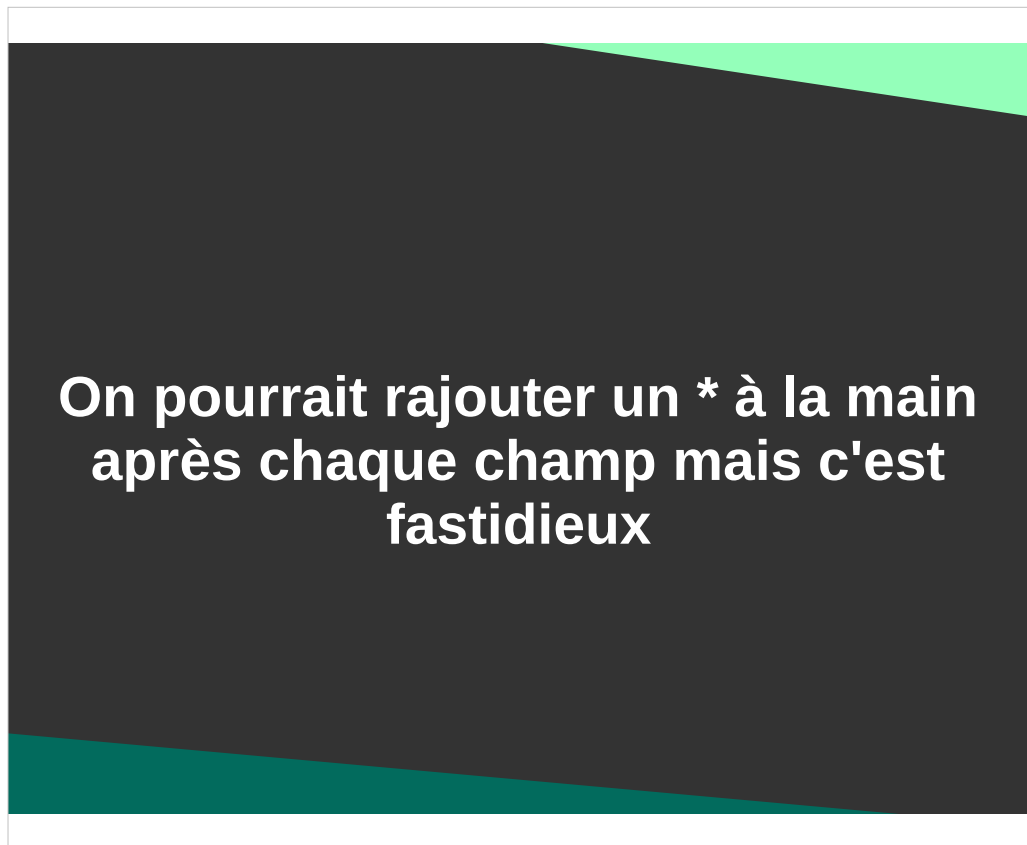
Pré-requis :

- Avoir le contenu de la ressource "formulaire-basique" sur ENT et fait les pratiques précédentes
- Rajouter l'attribut "required" sur un ou plusieurs champs
- Appliquer une pseudo-classe ":required" dans le CSS et appliquer des déclarations CSS
- Valider le formulaire en ne remplissant pas les champs requis



Oui, nos champs requis sont personnalisés, mais ce n'est pas forcément clair

- Web 1.0 : Web statique, l'internaute ne peut que lire les pages, il ne peut pas interagir avec
- Web 2.0 : Web social. L'internaute peut interagir avec le site notamment en postant des messages
- Web 3.0 : Web personnalisé. Le site anticipe pour vous vos besoins. E.g. les annonces basées sur vos recherches



- Web 1.0 : Web statique, l'internaute ne peut que lire les pages, il ne peut pas interagir avec
- Web 2.0 : Web social. L'internaute peut interagir avec le site notamment en postant des messages
- Web 3.0 : Web personnalisé. Le site anticipe pour vous vos besoins. E.g. les annonces basées sur vos recherches

Pseudo-élément

- Permet d'ajouter du contenu sans pour autant ajouter de nouvelles balises
- Syntaxe :
 - un sélecteur html + "::" + le nom de la pseudo classe
 - exemple : `mon-sélecteur::pseudo-element { color: blue; }`

Sources :

- <https://www.creativejuiz.fr/blog/css-css3/difference-entre-pseudo-element-et-pseudo-classe>
- <https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-%C3%A9l%C3%A9ments>
- <https://accessibleweb.com/question-answer/how-is-css-pseudo-content-treated-by-screen-readers/> - anglais
- <https://developer.mozilla.org/fr/docs/Web/CSS/::first-line>

- Pour le premier point l'avantage est que ceci est calculé par le navigateur. Par exemple il existe le pseudo-élément "::first-line" qui permet d'appliquer des déclarations uniquement sur la première ligne d'un texte. Si on fait ça avec du code, donc une balise , on ne peut pas s'assurer de la première ligne.
- Il faut faire attention aux deux points. Cette syntaxe est arrivée avec CSS3, de fait, les moteurs de rendu des navigateurs tolèrent qu'on utilise ":" au lieu "::"

Pseudo-éléments

- Vous ne pouvez pas associer plusieurs fois le même pseudo-élément à une balise
 - Le navigateur tranchera
- Ne s'affiche pas dans le code source
- `::before` / `::after` sont certainement ce que vous utiliserez le plus
- Si vous vous trompez sur la syntaxe, le navigateur est suffisamment malin pour faire la différence entre le `::` et le `:` grâce au nom

Sources :

- <https://www.creativejuiz.fr/blog/css-css3/difference-entre-pseudo-element-et-pseudo-classe>
- <https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-%C3%A9l%C3%A9ments>
- <https://accessibleweb.com/question-answer/how-is-css-pseudo-content-treated-by-screen-readers/> - anglais
- <https://developer.mozilla.org/fr/docs/Web/CSS/::first-line>

- Pour le premier point l'avantage est que ceci est calculé par le navigateur. Par exemple il existe le pseudo-élément `::first-line` qui permet d'appliquer des déclarations uniquement sur la première ligne d'un texte. Si on fait ça avec du code, donc une balise ``, on ne peut pas s'assurer de la première ligne.
- Il faut faire attention aux deux points, mais de lles moteurs de rendu des navigateurs ne font pas la différence entre `:` et `::`, ils se réfèrent au nom pour savoir si c'est une pseudo-classe ou un pseudo-élément

Pseudo-éléments - ::before /::after

- Permettent d'ajouter des éléments avant ou après un contenu, On peut mettre les deux en même temps
- N'existent pas pour les lecteurs d'écran → Ne pas placer de données critiques dedans

Sources :

- <https://developer.mozilla.org/fr/docs/Web/CSS/::after>
- <https://developer.mozilla.org/fr/docs/Web/CSS/::before>

- Si vous mettez du code html comme valeur de la propriété "content" ce dernier ne sera pas interprété. Exemple "content: <h1>Hello</h1>" affichera "<h1>Hello</h1>" sur la page
- La valeur de la propriété "content" sera nulle si vous voulez, par exemple, juste afficher un bloc. Car oui, presque toutes les déclarations vues en cours fonctionnent.

Pseudo-éléments - ::before /::after

- **Nécessite la propriété "content"** pour s'afficher. Contenu pouvant être :
 - une chaîne de caractères, une image (qui ne peut pas être redimensionnée, nécessite la fonction "url()"), rien (une chaîne vide), la valeur d'un attribut...
 - Le texte mis n'est pas sélectionnable
- Ne fonctionne pas sur les éléments émulés comme les images, les vidéos ou champ de texte

Sources :

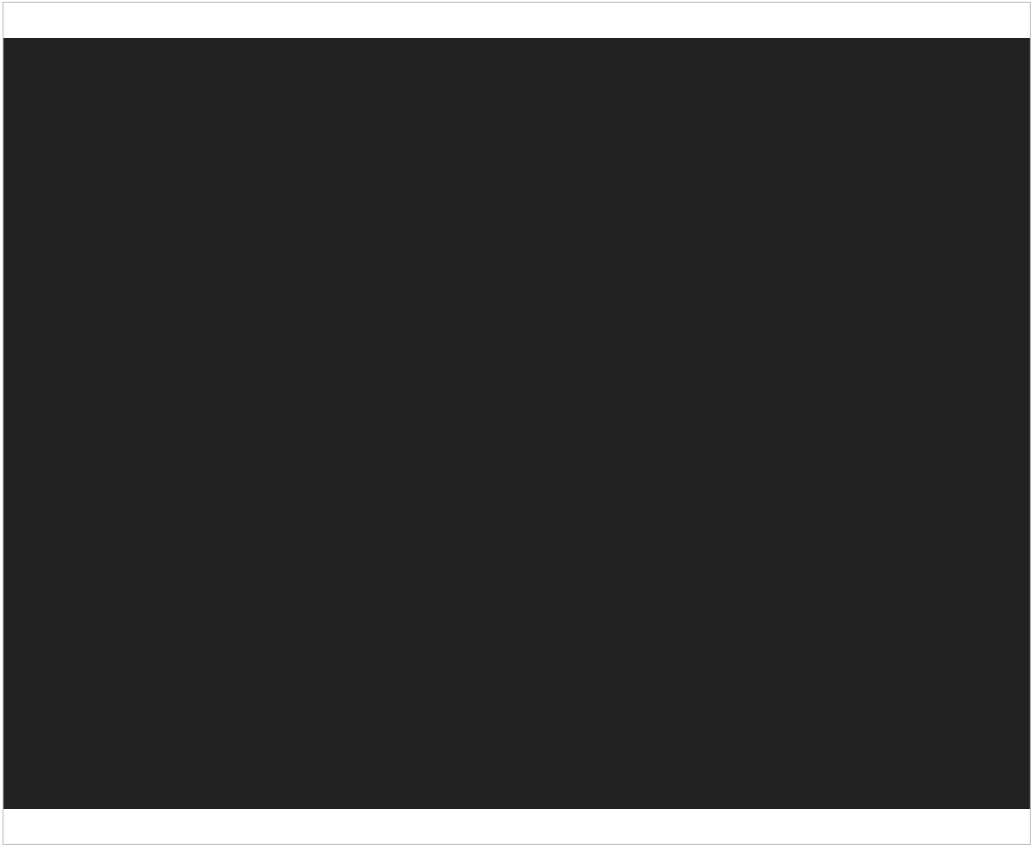
- <https://developer.mozilla.org/fr/docs/Web/CSS/::after>
- <https://developer.mozilla.org/fr/docs/Web/CSS/::before>

- Si vous mettez du code html comme valeur de la propriété "content" ce dernier ne sera pas interprété. Exemple "content: <h1>Hello</h1>" affichera "<h1>Hello</h1>" sur la page
- La valeur de la propriété "content" sera nulle si vous voulez, par exemple, juste afficher un bloc. Car oui, presque toutes les déclarations vues en cours fonctionnent.

Pratiquons ! - Améliorons notre premier <form>

Pré-requis :

- Avoir le contenu de la ressource "formulaire-basique" sur ENT et fait les pratiques précédentes
- Rajouter un pseudo-élément "::before" ou "::after" pour ajouter un "*" après tous les labels
- Rajouter un pseudo-élément "::before" ou "::after" pour ajouter un "*" après tous les labels **dont le champ est requis**
 - Jouer sur la propriété "order" des enfants flexbox
 - Jouer sur les combinateur de voisin



Questions ?