

# Intégration Web

2020 - MMI 1 – TP#5 S1

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

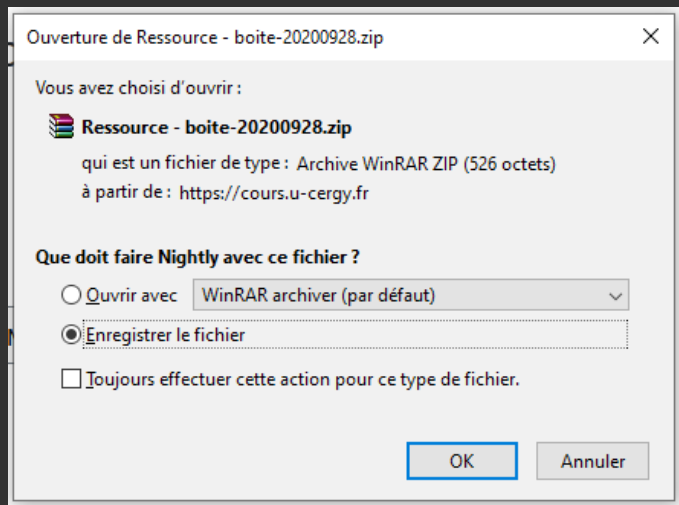
# But du cours

- 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

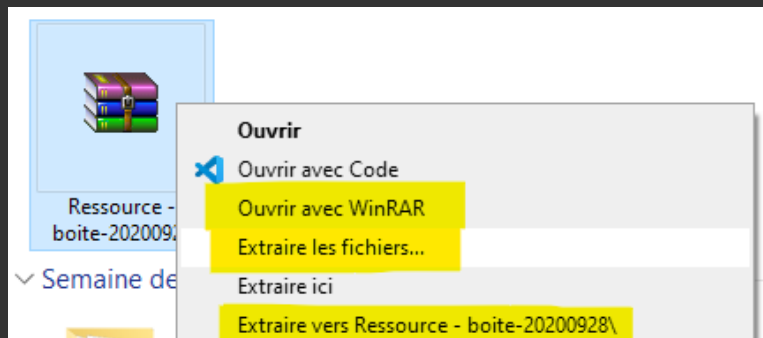
# Ressource du cours

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

# Petit point sur les ressources de cours

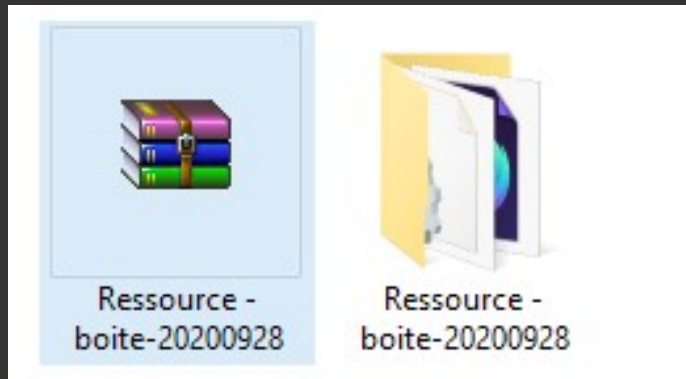


- Les ressources se présentent sous la forme d'une archive une fois téléchargées
- L'archive doit être impérativement extraite sinon vous ne serez pas capables de réaliser correctement les tp



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

# Petit point sur les ressources de cours

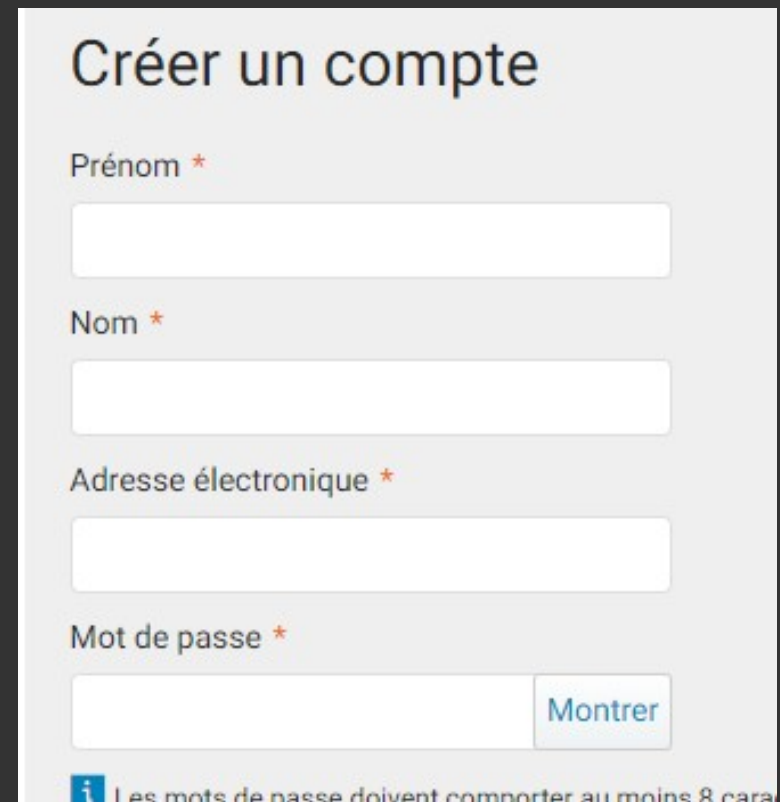


- Une fois extraite, vous devez obtenir un dossier avec le contenu de l'archive
- Libre à vous de travailler dans le dossier OU en copier le contenu pour le coller ailleurs
- **En aucun vous devez travailler dans l'archive**
- Une fois fait, glissez le dossier dans VS Code ou Sublime Text

**Entrons dans le web 2.0**

# 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 information icon followed by the text "Les mots de passe doivent comporter au moins 8 caractères".

Créer un compte


Prénom \*

Nom \*

Adresse électronique \*

Mot de passe \*

Montrer

 Les mots de passe doivent comporter 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>

```
<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>

# 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 :

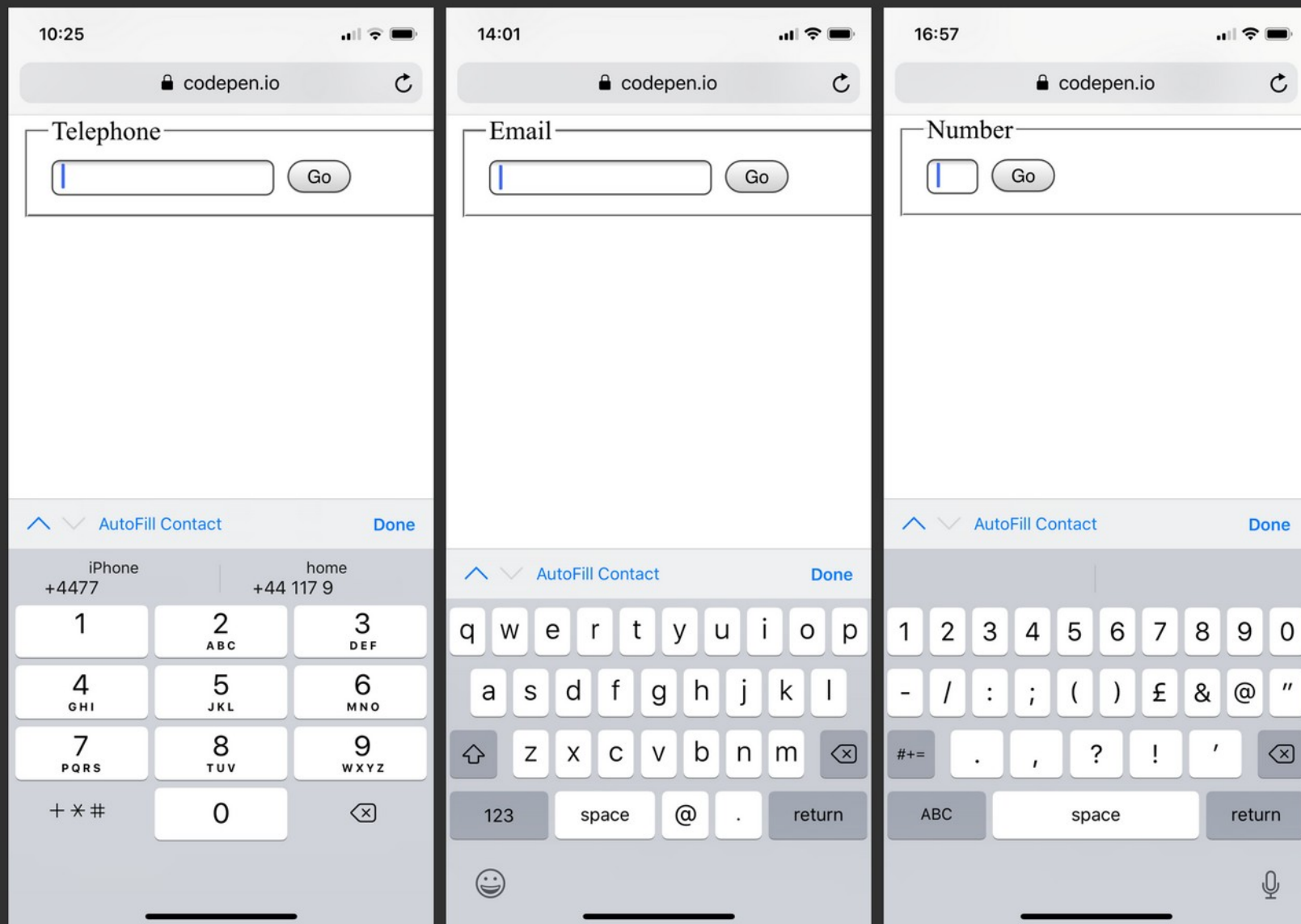
- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/Input/text>
- <https://developer.mozilla.org/fr/docs/Web/HTML/Element/Input>
- <https://developer.mozilla.org/fr/docs/Web/HTML/Attributes/autocomplete>

# 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>



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

# 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](https://www.w3schools.com/howto/howto_css_form_icon.asp) - mauvaise pratique

# 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

# 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>

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

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

# 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%89l%C3%A9ments\\_%C3%A0\\_cocher](https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Formulaires/Les_blocs_de_formulaires_natifs#%C3%89l%C3%A9ments_%C3%A0_cocher)



# 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%89l%C3%A9ments\\_%C3%A0\\_cocher](https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Formulaires/Les_blocs_de_formulaires_natifs#%C3%89l%C3%A9ments_%C3%A0_cocher)

# 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 envoyée 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

# 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>

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



# 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](https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Formulaires/Validation_donnees_formulaire)

# 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**

**On pourrait rajouter un \* à la main  
après chaque champ mais c'est  
fastidieux**

# 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>

# 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>

# 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>

# 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>



# 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 ?**