

Sommaire du document :

[Présentation](#)

[Cours](#)

[Exercices](#)

[Évaluation](#)

[Correction de l'évaluation](#)

[Proposition de projet](#)

Présentation

1. APPROCHE GÉNÉRALE

1.1. Positionnement dans le programme

Le programme de première en NSI se compose de sept thématiques dont, en quatrième position, *L'interaction homme-machine sur le WEB*. Cette thématique, en copie ci-dessous, se divise en quatre sous-thèmes dont celui qui nous intéresse: *Formulaires d'une page WEB*. On peut considérer que la thématique *L'interaction homme-machine sur le WEB*, qui est assez riche comparativement à d'autres, représente environ un cinquième du programme, et notre sous-thème un vingtième environ. En traitant les thématiques dans l'ordre proposé par le BO, celle-ci pourrait être abordée au second trimestre.

Interactions entre l'homme et la machine sur le Web

Lors de la navigation sur le Web, les internautes interagissent avec leur machine par le biais des pages Web.

L'Interface Homme-Machine (IHM) repose sur la gestion d'événements associés à des éléments graphiques munis de méthodes algorithmiques.

La compréhension du dialogue client-serveur déjà abordé en classe de seconde est consolidée, sur des exemples simples, en identifiant les requêtes du client, les calculs puis les réponses du serveur traitées par le client.

Il ne s'agit pas de décrire exhaustivement les différents éléments disponibles, ni de développer une expertise dans les langages qui permettent de mettre en œuvre le dialogue tels que PHP ou JavaScript.

Contenus	Capacités attendues	Commentaires
Modalités de l'interaction entre l'homme et la machine Événements	Identifier les différents composants graphiques permettant d'interagir avec une application Web. Identifier les événements que les fonctions associées aux différents composants graphiques sont capables de traiter.	Il s'agit d'examiner le code HTML d'une page comprenant des composants graphiques et de distinguer ce qui relève de la description des composants graphiques en HTML de leur comportement (réaction aux événements) programmé par exemple en JavaScript.
Interaction avec l'utilisateur dans une page Web	Analyser et modifier les méthodes exécutées lors d'un clic sur un bouton d'une page Web.	
Interaction client-serveur. Requêtes HTTP, réponses du serveur	Distinguer ce qui est exécuté sur le client ou sur le serveur et dans quel ordre. Distinguer ce qui est mémorisé dans le client et retransmis au serveur. Reconnaître quand et pourquoi la transmission est chiffrée.	Il s'agit de faire le lien avec ce qui a été vu en classe de seconde et d'expliquer comment on peut passer des paramètres à un site grâce au protocole HTTP.
Formulaire d'une page Web	Analyser le fonctionnement d'un formulaire simple. Distinguer les transmissions de paramètres par les requêtes POST ou GET.	Discuter les deux types de requêtes selon le type des valeurs à transmettre et/ou leur confidentialité.

Notre sous-thème *Formulaires d'une page WEB* se rattache plus spécifiquement aux éléments du programme officiel :

Contenus	Capacités attendues	Commentaires
Modalités de l'interaction entre l'homme et la machine Événements	Identifier les différents composants graphiques permettant d'interagir avec une application <i>Web</i> . Identifier les événements que les fonctions associées aux différents composants graphiques sont capables de traiter.	Il s'agit d'examiner le code HTML d'une page comprenant des composants graphiques et de distinguer ce qui relève de la description des composants graphiques en HTML de leur comportement (réaction aux événements) programmé par exemple en JavaScript.
Interaction avec l'utilisateur dans une page <i>Web</i>	Analyser et modifier les méthodes exécutées lors d'un clic sur un bouton d'une page <i>Web</i> .	
Formulaire d'une page <i>Web</i>	Analyser le fonctionnement d'un formulaire simple. Distinguer les transmissions de paramètres par les requêtes POST ou GET.	Discuter les deux types de requêtes selon le type des valeurs à transmettre et/ou leur confidentialité.

1.2. Notions à faire passer

Pour l'ensemble de la thématique, le programme (reproduit ci-dessus) demande d'exercer les élèves à,

1. *examiner* du code HTML,
2. à *identifier* des composants,
3. à *analyser* et *modifier* des méthodes.

Il s'agit donc davantage de donner aux élèves la faculté de lire et comprendre le code HTML que la capacité à programmer *ex nihilo*.

Concernant plus précisément le sous-thème des *formulaires HTML*, le programme demande en capacités attendues que l'élève puisse *analyser un formulaire simple*. Ce qui suppose qu'il aura eu une **expérience de lecture de formulaires** et de **reconnaissance des fonctionnalités de base** présentes dans ceux-ci.

Le programme donne une importance particulière aux requêtes POST et GET dont la différence devra être connue de l'élève : celui-ci devra donc avoir été confronté à ces deux types de requêtes et devra avoir compris la spécificité de chacune.

1.3. La progression et le nombre d'heures envisagés

L'horaire de l'enseignement de spécialité de première est de 4 heures hebdomadaires sur un total de 33 semaines environ (épreuves de français programmées le 11 juin 2026). Cela fait 132 heures au total dont un quart environ doit être consacré à l'élaboration d'un projet, soit une centaine d'heures de cours.

On peut donc compter une **vingtaine d'heures hors-projet** pour la thématique *L'interaction homme-machine sur le WEB*. Le sous-thème des *formulaires HTML* peut occuper environ **cinq heures** : les deux heures de cours que nous proposons, **deux heures d'activités / exercices** et **une heure d'évaluation** qui seront également détaillées dans ce qui suit.

1.4. Éléments d'évaluation

L'évaluation pourra comporter :

- Une partie test en classe sous forme "débranchée" avec des questions de lecture de formulaires HTML, de complétion de codes déjà créés et des exercices de création de petits formulaires simples comportant les balises de base, avec éventuellement un lexique de balises utiles.
Nous avons également réfléchi à une évaluation différenciée concernant les élèves aux connaissances plus avancées, comme on en rencontre souvent en informatique. Cette différenciation peut se faire sous forme de *questions bonus* à destination de ces élèves. Nous donnons dans ce qui suit des exemples de telles évaluations.
- Un "projet" à rendre demandant l'élaboration d'une page WEB avec un "cahier des charges" à respecter. Nous donnons dans ce qui suit un exemple de travail, assorti d'un cahier des charges, qui pourrait être proposé aux élèves dans ce cadre.

2. DÉTAIL DE LA SÉANCE PRÉVUE

2.1. Positionnement dans la thématique, prérequis

Pour l'ensemble de la thématique, nous pensons indispensable d'effectuer une remise à niveau sur les connaissances de base en HTML et CSS qui n'ont pas forcément été bien vues en classe de seconde. Voici un exemple de progression que l'on peut envisager pour toute la thématique :

- *Sous-thème 1: Généralités, histoire*
 1. Historique de la création de l'internet.
 2. Définition du *WEB* et sa différence avec l'*internet*.
 3. Petit panorama des différents langages (côté client et côté serveur).
 4. Signification de l'expression *développer une page WEB*.
 5. Fonction du navigateur. Proposition d'*éditeurs de code* adaptés. Visualisation / lecture de différents exemples d'édition de code (coloration, indentation, détection et correction d'erreurs...).

Ce sous-thème se rattache aux éléments du programme officiel :

Contenus	Capacités attendues	Commentaires
Modalités de l'interaction entre l'homme et la machine Événements	Identifier les différents composants graphiques permettant d'interagir avec une application <i>Web</i> . Identifier les événements que les fonctions associées aux différents composants graphiques sont capables de traiter.	Il s'agit d'examiner le code HTML d'une page comprenant des composants graphiques et de distinguer ce qui relève de la description des composants graphiques en HTML de leur comportement (réaction aux événements) programmé par exemple en JavaScript.
Interaction avec l'utilisateur dans une page <i>Web</i>	Analyser et modifier les méthodes exécutées lors d'un clic sur un bouton d'une page <i>Web</i> .	

- *Sous-thème 2: Le langage HTML*

1. Rappel sur le langage *HTML*, évocation du *W3C* et de la dernière évolution *HTML5*.
2. Rappel sur la structure d'une page, les entêtes `<!Doctype>` ; `<head>` et `<body>` Descriptif des autres balises essentielles à la composition de la page.
3. Visualisation / lecture de différents exemples de code *HTML*.

Ce sous-thème se rattache à l'élément du programme officiel :

Contenus	Capacités attendues	Commentaires
Modalités de l'interaction entre l'homme et la machine	Identifier les différents composants graphiques permettant d'interagir avec une application <i>Web</i> .	Il s'agit d'examiner le code <i>HTML</i> d'une page comprenant des composants graphiques et de distinguer ce qui relève de la description des composants graphiques en <i>HTML</i> de leur comportement (réaction aux événements) programmé par exemple en <i>JavaScript</i> .
Événements	Identifier les événements que les fonctions associées aux différents composants graphiques sont capables de traiter.	

- *Sous-thème 3 : Mise en forme avec CSS*

1. Retour sur le *W3C*.
2. Séparation du contenu de la page et de la gestion de l'apparence de la page. Visualisation / lecture de différents exemples de code *CSS*.

Ce sous-thème se rattache également à l'élément du programme officiel :

Contenus	Capacités attendues	Commentaires
Modalités de l'interaction entre l'homme et la machine	Identifier les différents composants graphiques permettant d'interagir avec une application <i>Web</i> .	Il s'agit d'examiner le code <i>HTML</i> d'une page comprenant des composants graphiques et de distinguer ce qui relève de la description des composants graphiques en <i>HTML</i> de leur comportement (réaction aux événements) programmé par exemple en <i>JavaScript</i> .
Événements	Identifier les événements que les fonctions associées aux différents composants graphiques sont capables de traiter.	

- *Sous-thème 4: Interaction client–serveur*

1. Les *requêtes client* Le protocole *HTTP* et son histoire.
2. Langages et applications côté serveur. Exemples de programmes en *PHP*.
3. Retour sur les méthodes *GET* et *POST*. Visualisation / lecture de différents exemples.

Ce sous-thème se rattache à l'élément du programme officiel :

Contenus	Capacités attendues	Commentaires
Interaction client-serveur. Requêtes <i>HTTP</i> , réponses du serveur	Distinguer ce qui est exécuté sur le client ou sur le serveur et dans quel ordre. Distinguer ce qui est mémorisé dans le client et retransmis au serveur. Reconnaître quand et pourquoi la transmission est chiffrée.	Il s'agit de faire le lien avec ce qui a été vu en classe de seconde et d'expliquer comment on peut passer des paramètres à un site grâce au protocole <i>HTTP</i> .

- *Sous-thème 5 : Formulaire d'une page WEB*

C'est l'objet de notre étude.

Ce sous-thème se rattache à l'élément du programme officiel :

Formulaire d'une page Web	Analyser le fonctionnement d'un formulaire simple. Distinguer les transmissions de paramètres par les requêtes POST ou GET.	Discuter les deux types de requêtes selon le type des valeurs à transmettre et/ou leur confidentialité.
---------------------------	--	---

- *Sous-thème 6 : Gestion d'évènements avec javascript*

Ce sous-thème se rattache également à l'élément du programme officiel :

Contenus	Capacités attendues	Commentaires
Modalités de l'interaction entre l'homme et la machine Événements	Identifier les différents composants graphiques permettant d'interagir avec une application Web. Identifier les événements que les fonctions associées aux différents composants graphiques sont capables de traiter.	Il s'agit d'examiner le code HTML d'une page comprenant des composants graphiques et de distinguer ce qui relève de la description des composants graphiques en HTML de leur comportement (réaction aux événements) programmé par exemple en JavaScript.

- *Sous-thème 7 : Sécurité et protocole SSL*

[Retour au sommaire du document](#)

2.2. Mode de travail

Compte tenu du temps très court dévolu à cette partie du programme, le mode de travail que nous proposons ici est un cours relativement standard avec de multiples activités et essais que l'élève pourra faire par lui-même.

Vient s'adjoindre à ce cours une série d'exercices :

- Des exercices d'application directe, très progressifs, à destination des élèves les plus fragiles
- Des exercices de perfectionnement et/ou de réflexion que pourront prendre les élèves les plus à l'aise avec ces notions.

2.3. Support de cours

Voici le [support de cours](#) que nous proposons.

2.4. Exercices envisagés

Voici [les exercices](#) que nous proposons.

2.5. Exemple d'évaluation

Voici [un exemple d'évaluation](#) que nous proposons. Il s'agit d'une évaluation en mode "débranché" Elle se compose de quatre exercices :

1. Une lecture de code Html et de questions sur ce code.
2. Un QCM sur les notions vues en cours.
3. L'écriture d'un petit bout de code Html, en méthode GET, susceptible de générer une URL proposée.
4. L'écriture d'un bout de code Html répondant à un petit "cahier des charges".

On pourrait imaginer un barème de 8 points pour l'exercice 1 (8 questions) ; de 6 points pour le QCM (6 questions) puis de 6 et 8 points respectivement pour les exercices 3 et 4. ([accès à la correction proposée](#))

2.6. Exemple de projet

Voici [un exemple de projet](#) que nous proposons. Il s'agit d'un projet donné en début d'année, qui s'effectue par groupes de deux ou de trois élèves, pour une présentation de 15 minutes devant la classe en fin d'année.

[Retour au sommaire du document](#)

Les formulaires en HTML

Sommaire du cours

1. [Introduction aux formulaires.](#)
2. [Création d'un formulaire avec la balise <form>.](#)
3. [Les composants d'un formulaire.](#)
4. [Bonnes pratiques et accessibilité.](#)
5. [Exercice pratique.](#)

1. Introduction aux formulaires

Les formulaires HTML permettent d'instaurer un dialogue entre un utilisateur et un site web ou une application. Ils permettent à l'utilisateur d'envoyer des données au site web. La plupart du temps, ces données sont envoyées à des serveurs web mais la page peut aussi les intercepter et les utiliser elle-même.

L'analyse et le traitement du formulaire devant être effectués côté serveur, la page du formulaire sera écrite en langage PHP. En cliquant sur le bouton envoyer les contenus des zones du texte du formulaire seront transmis à une page cible PHP sous forme de paramètres directement intégrés dans la requête HTTP envoyée par le client au serveur.

Objectifs du cours :

Comprendre Le rôle des formulaires dans l'interaction utilisateur	Maîtriser Les différents types de champs et leurs attributs	Implémenter Des formulaires accessibles et fonctionnels
---	---	---

2. Création d'un formulaire avec la balise <form>

Les formulaires sont un moyen de récupérer des données à partir du client (navigateur) pour les passer à l'application serveur. Dans le fichier html, on utilise la balise `<form>` en y ajoutant l'attribut `method` qui permet d'indiquer quel est le type de requête HTTP.

2.1 Exemple de formulaire

Nom :

Email :

URL qui serait générée si la page est hébergée sur site.com :

```
site.com?nom=john&email=j%40com
```

Signification des caractères spéciaux dans votre url,

- **?** : Début des paramètres de la requête
- **=** : Séparateur entre le nom du paramètre et sa valeur
- **&** : Séparateur entre différents paramètres, ici entre nom et email
- **%40** : Encodage en hexadécimal du caractère @ dans une URL

2.2 Code du formulaire

```
<form method="get" action="traitement.php">

  <label for="nom">Nom :</label>

  <input type="text" id="nom" name="nom" placeholder="Entrez votre nom" required>

  <label for="email">Email :</label>

  <input type="email" id="email" name="email" placeholder="Entrez votre email"
required>

  <button type="submit">Envoyer</button>

  <button type="reset">Effacer</button>

</form>
```

[Retour au sommaire du cours](#)

2.3 Points clés à retenir

1. **method** : Définit comment les données sont envoyées
 - `get` : Données visibles dans l'URL, limité en taille
 - `post` : Données cachées, taille illimitée
2. **action** : Spécifie où envoyer les données qui seront traitées. Dans notre exemple, elles sont traitées par un fichier traitement.php sur le serveur. Si action n'est pas présent, les données sont envoyées vers la page elle-même.
3. **required** : Rend un champ obligatoire.

Exercice 1 - A vous de réfléchir

Après avoir vu le fonctionnement du formulaire avec la méthode `get`, dans quels cas préférez-vous utiliser POST plutôt que GET ?

Réponse :

Avec la méthode GET, les données sont visibles dans l'URL (par exemple : ?nom=Dupont&email=test@exemple.fr). En revanche, on utilise la méthode POST pour :

- Les données sensibles (mots de passe)
- Les fichiers uploadés
- Les formulaires longs
- Éviter que l'URL ne soit trop longue

Exercice 2 - Observer la différence entre la méthode get et la méthode post

Cliquer sur ce [lien n°1](#), une page apparaît, sur laquelle figure un formulaire composé de deux champs de saisie.

1. Taper `Ctrl+ U` pour visualiser le code html de cette page. Indiquer le nom de la méthode employée pour transmettre les informations. Quels sont les attributs `name` qui ont été affectés à chacun de ces deux champs ?
 2. Saisir des données puis soumettre ce formulaire, l'observation de l'URL confirme-t-elle les réponses faites à la question précédente ?
 3. Indiquer le nom du fichier vers lequel sont envoyées les informations saisies dans le formulaire. Où se trouve ce fichier ? Dans quel langage traite-t-il ces informations ?
- Ci-dessous, à titre d'illustration, un morceau de code de ce fichier. Ce bout de code récupère les valeurs saisies dans chacun des champs, repérés par leur attribut `name` et les place dans les variables

`$login_connexion` et `$mot_de_passe`. On observe que la méthode apparaît clairement dans ce code.

```
<?php
$login_connexion = $_GET['login'];
$mot_de_passe = $_GET['password']; ?>
```

Cliquez sur ce [lien n°2](#) pour obtenir une nouvelle page où figure un formulaire composé de d'un seul champ de saisie.

4. Déterminer l'attribut `name` affecté à ce champ.
 5. Saisir un mot de passe dans ce champ, valider et observer l'URL dans la barre d'adresse du navigateur. Indiquer le nom de la méthode employée pour transmettre les informations saisies ainsi que le nom du fichier vers lequel sont envoyées ces informations.
- Ci-dessous, à titre d'illustration, un morceau de code de ce fichier.

```
<?php
$mot_de_passe = $_POST['pass']; ?>
```

Cliquez à nouveau sur le [lien n°2](#) (on chargera cette page sur firefox si possible) puis faire un clic droit sur la page, choisir l'item `inspecter` puis cliquer sur l'onglet `réseau` de la zone d'inspection du code. Saisir un mot de passe dans le formulaire et valider.

Dans la zone d'inspection du code, cliquer sur le lien `cible_2.php` puis sur `requête`.

6. Que remarquez-vous à propos du mot de passe que vous aviez saisi ?

Réponse :

1. La méthode est `get`. Les attributs `name` sont `login` et `password` respectivement.

2. Dans l'URL, on lit bien chacun des deux attributs `login` et `password` avec leur valeur respective. Cette apparition en clair dans l'URL est propre à la méthode `GET`.
3. Le fichier qui récupère les données de la requête est `cible_1.php` qui est localisé sur le serveur web et qui est écrit en langage `php`. Ce code n'est pas visible côté client, il est interprété par le serveur et non par le navigateur.
4. En faisant `Ctrl+ U` on lit que l'attribut `name` affecté à ce champ est `pass`
5. Cette fois-ci, c'est la méthode `post` qui est utilisée. On ne voit plus les informations saisies apparaître en clair dans l'URL. Mais on peut y lire que le fichier qui récupère les données de la requête est `cible_2.php`
6. Avec l'inspecteur de propriétés de firefox, il est toujours possible de retrouver les informations passées dans la requête, elles sont sauvegardées par le navigateur.

[Retour au sommaire du cours](#)

3. Les composants d'un formulaire

HTML5 offre de très nombreuses méthodes et attributs qui permettent de créer une grande variété de formulaires. Nous présentons ici quelques-uns de ces attributs, en particulier l'attribut `type`, utilisé principalement dans les balises `<input>` et `<button>`, qui permet de spécifier le type de données que le champ peut accepter et d'adapter le champ au recueil de ce type de données.

3.1 Tableau illustrant des types de champs et les mots clefs importants

Type	Utilisation	Attributs importants	Exemple
<code>text</code>	Saisie de texte court	<code>maxlength</code> , <code>placeholder</code> , <code>pattern</code>	<input type="text" value="Texte"/>
<code>password</code>	Saisie masquée	<code>minlength</code> , <code>required</code>	<input type="password" value="....."/>
<code>email</code>	Email avec validation	<code>multiple</code> , <code>pattern</code>	<input type="email" value="email@exemple.fr"/>
<code>number</code>	Saisie numérique	<code>min</code> , <code>max</code> , <code>step</code>	<input type="number" value="nombre"/>
<code>date</code>	Sélection de date	<code>min</code> , <code>max</code>	<input type="date" value="mm / dd / yyyy"/>
<code>checkbox</code>	Cases à cocher multiples	<code>checked</code> , <code>value</code>	<input type="checkbox"/> Entrée <input type="checkbox"/> Dessert <input type="checkbox"/> Fromage
<code>radio</code>	Choix unique	<code>checked</code> , <code>value</code>	<input type="radio"/> oui <input type="radio"/> non <input type="radio"/> je ne sais pas
<code>range</code>	Curseur de valeur	<code>min</code> , <code>max</code> , <code>step</code>	<input type="range" value="50"/>

3.2 Exemples détaillés

3.2.1 - Zone de saisie de texte

```
< input type="text">
```

Votre texte :

Il est utilisé pour créer un champ de saisie permettant à l'utilisateur d'entrer du texte sur une seule ligne.

```
<form>
  <label for="champ">Votre texte :</label>
  <input type="text" id="champ" name="champ" placeholder="Texte à saisir">
</form>
```

Une précision : L'attribut `placeholder` dans un champ input HTML sert à afficher un texte indicatif à l'intérieur du champ avant que l'utilisateur ne saisisse quoi que ce soit.

3.2.2 Case à cocher

```
<input type="checkbox">
```

- ☐ Mayonnaise
- ☐ Ketchup
- ☐ Sauce piquante

C'est un type d'élément de formulaire utilisé pour afficher **une ou plusieurs case(s)** à cocher parmi un ensemble de choix.

```
<form>
  <label> <input type="checkbox" name="sauce" value="mayonnaise">Mayonnaise </label>
<br>
  <label> <input type="checkbox" name="sauce" value="ketchup">Ketchup</label>    <br>
  <label> <input type="checkbox" name="sauce" value="piquante">Sauce piquante</label>
</form>
```

3.2.3 - Bouton radio `<input type="radio">`

Les boutons radio doivent avoir le même nom pour fonctionner en groupe : **on ne peut choisir qu'une option !** (essayez de cliquer les deux...)

- ☐ Homme
- ☐ Femme

```
<form>
  <input type="radio" id="homme" name="sexe" value="homme">
  <label for="homme">Homme</label>
  <input type="radio" id="femme" name="sexe" value="femme">
  <label for="femme">Femme</label>
</form>
```

Quelques précisions:

- Pourquoi on dit "avec étiquette" ? Parce qu'ici, si vous cliquez à côté du bouton, sur "homme" ou "femme" le bouton est aussi sélectionné. C'est l'étiquette (grâce au `<label>`). Regardez les exemples précédents, il y avait aussi des étiquettes.
- Dans les input, Un nom est associé au composant pour pouvoir récupérer les données saisies côté serveur. L'attribut `name` qui permet d'identifier chaque champ de saisies dans les paramètres qui sont envoyés au serveur,
- L'attribut `value` qui définit la valeur par défaut affichée au chargement de la page Web. De plus, la valeur de cet attribut peut être utilisée en JavaScript ou côté serveur.

3.2.4 - Bouton Poussoir

`<input type="submit" value="Valider">` ou `<button type="submit"> Valider </button>`

Créer des boutons interactifs sur une page web. (Les boutons sont souvent utilisés avec des scripts JavaScript - on verra ça plus tard !)

Cliquez-moi

```
<input type="submit" value="Cliquez-moi" id="bouton_pousse" onclick="afficherMessage()" >
```

Attention, ici il y a, à la fin, une référence à du javascript, retenez surtout le début du code. Vous pouvez faire apparaître un bouton, sans la dernière instruction... c'est juste qu'il ne se passera rien !

3.2.5 - Liste déroulante

`<select>`

Pour proposer un choix parmi plusieurs options

```
<form> <select>
  <option value="question">Vous êtes...</option>
  <option value="tres_satisfait">très satisfait</option>
  <option value="satisfait">satisfait</option>
  <option value="decu">déçu</option>
  <option value="tres_decu">très déçu</option>
  <option value="autre">ne se prononce pas</option>
</select> </form>
```

[Retour au sommaire du cours](#)

4. Bonnes pratiques et accessibilité

4.1 Accessibilité

Nom complet : Saisissez votre nom tel qu'il apparaît sur vos documents officiels

```
<label for="nom-complet">Nom complet :</label>
<input type="text" id="nom-complet" name="nom_complet" aria-describedby="aide-nom"
required>
<small id="aide-nom">Saisissez votre nom tel qu'il apparaît...</small>
```

- Associez chaque champ à son label avec l'attribut `for`. Ici, en cliquant sur "Nom complet :", un curseur d'écriture apparaît dans la zone de texte. C'est grâce à la balise `<label>` qui associe la zone de texte et "Nom complet" par l'identifiant "nom-complet".
- Utilisez `aria-` (ensemble de normes destinées à rendre les pages web accessibles aux personnes handicapées. Pour en savoir plus, voir [WAI-ARIA](#)). Dans notre cas, `aria-describedby` permet d'associer une description de ce qu'il faut faire (utile par exemple pour les malvoyants qui utilisent des logiciels adaptés pour naviguer sur le net).
- Assurez un contraste suffisant pour la lisibilité (avec le fichier `style.css`).
- Organisez les champs logiquement avec `tabindex`. Si vous avez plusieurs champs, vous pouvez sauter de l'un à l'autre en utilisant la touche TAB. La fonctionnalité `tabindex` permet de choisir l'ordre des zones de texte.
- Fournissez des messages d'erreur clairs (on verra ça plus tard).

4.2 Validation côté client

Exemple:

```
<input type="number" min="18" max="120" required>  
<input type="email" required pattern="[a-z0-9._-]+@[a-z0-9.-]+\.[a-z]{2,}$">
```

Ici, on impose que l'adresse email soit écrit en trois parties,

- Une première partie ou il peut y avoir toutes les lettres, tous les chiffres et les caractères ".", "_" et "-" [a-z0-9._-]
- Une deuxième partie = le site d'hébergement (comme gmail) avec @ et toutes les lettres, tous les chiffres et les caractères "." et "-" +@[a-z0-9.-]
- Enfin, le domaine, après un point (noté \.), qui doit contenir au moins deux lettres +\.[a-z]{2,}\$. Le signe \$ indique qu'il ne doit rien y avoir ensuite.

[Retour au sommaire du cours](#)

5. Exercice pratique

Formulaire d'inscription

On veut réaliser le formulaire suivant :

Nom:

Prénom:

Date de naissance:



Email :

Téléphone :

Adresse :

Pays :



Centres d'intérêt : ☐ Sport ☐ Musique ☐ Voyage

Étape 1 : Structure de base

Créez la balise <form> avec method="post", envoyant le formulaire au fichier "inscription.php"

Étape 2 : Informations personnelles

Ajoutez les champs : nom, prénom, date de naissance

Étape 3 : Coordonnées

Ajoutez les champs : email, téléphone, adresse

Étape 4 : Options

Ajoutez une liste déroulante pour le pays et des cases à cocher pour les centres d'intérêt.

Correction :

Étape 1 :

```
<form method="post" action="/inscription.php">  
< type="submit" value="Envoyer">  
</form>
```

Étape 2 :

```
<label for="nom">Nom:</label>  
<input type="text" name="nom" id="nom" placeholder="Votre nom" required><br>  
  
<label for="prenom">Prénom:</label>  
<input type="text" name="prenom" id="prenom" placeholder="Votre prenom" required><br>  
  
<label for="date_naissance">Date de naissance:</label>  
<input type="date" name="date_naissance" id="date_naissance" required>
```

Étape 3 :

```
<label for="email">Email :</label>  
<input type="email" name="email" id="email" placeholder="Votre email" required><br>  
  
<label for="tel">Téléphone :</label>  
<input type="tel" name="tel" id="tel" placeholder="06 ** ** ** ** *"><br>  
  
<label for="adresse">Adresse :</label>  
<input type="text" name="adresse" id="adresse" placeholder="Votre adresse"><br>
```

Étape 4 :

```
label for="pays">Pays :</label>  
<select name="pays" id="pays">  
  <option value="france">France</option>  
  <option value="belgique">Belgique</option>  
  <option value="suisse">Suisse</option>  
</select><br>  
  
<label><input type="checkbox" name="interets" value="sport"> Sport</label>  
<label><input type="checkbox" name="interets" value="musique"> Musique</label>  
<label><input type="checkbox" name="interets" value="voyage"> Voyage</label>
```

Code final (toutes les étapes regroupées) :

```
<form method="post" action="/inscription.php">
```

```
<label for="nom">Nom:</label>
```

```
<input type="text" name="nom" id="nom" placeholder="Votre nom" required><br>
```

```
<label for="prenom">Prénom:</label>
```

```
<input type="text" name="prenom" id="prenom" placeholder="Votre prenom" required><br>
```

```
<label for="date_naissance">Date de naissance:</label>
```

```
<input type="date" name="date_naissance" id="date_naissance" required><br>
```

```
<label for="email">Email :</label>
```

```
<input type="email" name="email" id="email" placeholder="Votre email" required><br>
```

```
<label for="tel">Téléphone :</label>
```

```
<input type="tel" name="tel" id="tel" placeholder="06 ** ** ** ** *"><br>
```

```
<label for="adresse">Adresse :</label>
```

```
<input type="text" name="adresse" id="adresse" placeholder="Votre adresse"><br>
```

```
<label for="pays">Pays :</label>
```

```
<select name="pays" id="pays">
```

```
<option value="france">France</option>
```

```
<option value="belgique">Belgique</option>
```

```
<option value="suisse">Suisse</option>
```

```
</select><br>
```

Centres d'intérêt :

```
<label> <input type="checkbox" name="interets" value="sport"> Sport</label>
```

```
<label> <input type="checkbox" name="interets" value="musique"> Musique</label>
```

```
<label> <input type="checkbox" name="interets" value="voyage"> Voyage</label>
```

```
< type="submit" value="Envoyer">
```

```
</form>
```

[Retour au sommaire du document](#)

Exercices : Les Formulaires HTML

Niveau : 1ère NSI - Thème : Interaction Client-Serveur (Web)

Exercice 1 : Un formulaire de contact simple

Objectif : Créer un formulaire demandant le nom, l'email et un message.

Consignes :

1. Créez une balise `<form>`
2. Ajoutez un champ texte pour le "Nom" (utilisez `<label>` et `<input text= ..>`).
3. Ajoutez un champ pour l'"Email" (type approprié).
4. Ajoutez un bouton d'envoi contenant le texte "Envoyer".
5. N'oubliez pas les attributs `<name>` pour chaque champ (ex: "nom", "email").

Aperçu du résultat attendu :

Nom :

Email :

Correction:

```
<form action="traitement.php" method="get">
<div>
  <label for="nom">Nom :</label>
  <input type="text" id="nom" name="nom" placeholder="Votre nom" required>
</div>
<div>
  <label for="email">Email :</label>
  <input type="email" id="email" name="email" placeholder="exemple@gmail.com" required>
</div>
<button type="submit">Envoyer</button>
</form>
```

Note : L'attribut `for` du label doit correspondre à l' `id` de l'input pour lier les deux éléments (cliquer sur le texte active le champ).

Exercice 2 : Choix multiples (Radio, Checkbox, Select)

Objectif : Manipuler différents types de sélections.

Consignes :

1. Créez un formulaire pour commander une pizza.
2. **Taille (Choix unique) :** Utilisez des boutons radio pour "Petite", "Moyenne", "Grande".
Astuce : Ils doivent avoir le même attribut `name` pour être exclusifs.
3. **Ingédients (Choix multiples) :** Utilisez des cases à cocher (checkbox) pour "Fromage", "Champignons", "Olives".
4. **Sauce (Liste déroulante) :** Utilisez un `select` pour choisir entre "Tomate", "Crème", "Barbecue".

Aperçu du résultat attendu :

Taille :

☐ Petite ☒ Moyenne ☐ Grande

Suppléments :

☐ Fromage ☐ Champignons ☐ Olives

Sauce :

Tomate ▼

Commander

Correction:

```
<form action="commande.php">
  <!-- Boutons Radio : même 'name' pour un choix unique -->
  <p>< < strong> Taille < /strong> > :
    <input type="radio" id="petite" name="taille" value="petite">
    <label for="petite">Petite</label>

    <input type="radio" id="moyenne" name="taille" value="moyenne" checked>
    <label for="moyenne">Moyenne</label>

    <input type="radio" id="grande" name="taille" value="grande">
    <label for="grande">Grande</label>
  </p>
```

```

<!-- Checkbox : choix multiples possibles -->
<p> <strong> Ingrédients </strong> >:
  <input type="checkbox" id="fromage" name="ingredients" value="fromage">
  <label for="fromage">Fromage</label>
  <input type="checkbox" id="champi" name="ingredients" value="champignons">
  <label for="champi">Champignons</label>
  <input type="checkbox" id="olives" name="ingredients" value="olives">
  <label for="olives">olives</label>
</p>

<!-- Select : liste déroulante -->
<p <strong> Sauce </strong> >.
  <label for="sauce">Sauce :</label>
  <select id="sauce" name="sauce">
    <option value="tomate">Tomate</option>
    <option value="creme">Crème</option>
    <option value="bbq">Barbecue</option>
  </select>
</p>

<button type="submit">Commander</button>
</form>

```

Exercice 3 : Structure et Méthode d'envoi

Objectif : Organiser un formulaire long et comprendre `GET` vs `POST` .

Consignes :

1. Créez un formulaire d'inscription complet.
2. Utilisez la balise `fieldset` et `legend` pour regrouper les champs "Identité" (Nom, Prénom).
3. Utilisez un autre `fieldset` pour "Message" contenant un `textarea` .
4. Configurez le formulaire pour envoyer les données de manière **invisible** dans l'URL (méthode POST).

5. L'attribut <action> doit pointer vers "traitement_inscription.php".

Aperçu du résultat attendu :

Identité

Nom :

Prénom :

Votre Message

S'inscrire

Correction :

```
<!-- method="post" est important pour ne pas afficher les données dans l'URL
      (utile pour les mots de passe ou les données volumineuses) -->
<form action="traitement_inscription.php" method="post">
  <fieldset>
    <legend>Identité</legend>
    <p>
      <label for="nom_complet">Nom :</label>
      <input type="text" id="nom_complet" name="nom">
    </p>
    <p>
      <label for="prenom">Prénom :</label>
      <input type="text" id="prenom" name="prenom">
    </p>
  </fieldset>
  <fieldset>
    <legend>Votre Message</legend>
    <!-- Textarea pour le texte sur plusieurs lignes -->
    <textarea name="message" rows="5" cols="40" placeholder="Ecrivez ici"></textarea>
  </fieldset>
  <button type="submit">S'inscrire</button>
</form>
```

Exercice 4 : Interaction Client-Serveur (PHP)

Objectif : Écrire le formulaire HTML correspondant au script code.php de traitement suivant.

Consignes : Écrire un extrait de formulaire html (balise et tout ce qui est dedans) qui puisse être interprété dans le serveur par le code PHP présenté ci-dessous. On s'efforcera de proposer un formulaire "convivial" où les champs sont bien identifiables par l'utilisateur.

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {

    $prenom = isset($_POST['prenom']) ? htmlspecialchars($_POST['prenom']) : "";

    $sexe = isset($_POST['sexe']) ? htmlspecialchars($_POST['sexe']) : "";

    $motDePasse = isset($_POST['motDePasse']) ? htmlspecialchars($_POST['motDePasse']) : "";

    if (strlen($motDePasse) < 8) {

        echo "Mot de passe trop court.";

    } else {

        echo "<h1>Données du formulaire</h1>";

        echo "<p><strong>Prénom :</strong> $prenom</p>";

        echo "<p><strong>Sexe :</strong> $sexe</p>";

        echo "<p><strong>Mot de passe :</strong> $motDePasse</p>";

    }

} else {

    echo "Aucune donnée reçue.";

}
```

Correction:

```
<form action="code.php" method="post">

  <label for="votrePrenom">Prénom :</label>

  <!-- Attention : le 'name' doit être 'prenom' pour correspondre au PHP -->

  <input type="text" id="votrePrenom" name="prenom" required>


  <label for="sexe">Genre :</label>

  <select name="sexe" id="sexe">

    <option value="Femme">Femme</option>

    <option value="Homme">Homme</option>

  </select>


  <label for="motDePasse">Mot de passe :</label>

  <input type="password" id="motDePasse" name="motDePasse" minlength="8" required>


  <input type="submit" value="Envoyer">

</form>
```

[Retour au sommaire du document](#)

EXEMPLE D'ÉVALUATION

LIRE DU CODE HTML

Exercice 1

Quelques questions sur l'extrait de code html reproduit ci-dessous.

1. Que va produire cet extrait lors de son exécution sur un navigateur ?
2. Expliquer le sens de la balise `<form>` et de l'attribut *action* présents en première ligne de cet extrait
3. Expliquer ce que signifie l'attribut *method* à l'intérieur de la balise `<form>`, quelle autre valeur aurait-il pu prendre ?
4. Quelle est l'utilité de l'attribut *name* à l'intérieur de la balise `<input>` ?
5. Quelle est l'utilité de l'attribut *type* à l'intérieur de la balise `<input>` ?
6. Quelle est l'utilité de la balise `<label>` présente dans cet extrait.
7. Quelle est l'utilité de l'attribut *required* présent dans la balise `<input>` ?
8. Quelle est l'utilité de l'attribut *placeholder* ?

Extrait :

```
<form action="https://www.exemplesite.com/" method="get">

  <label for="username">Nom :</label>

  <input type="text" id="username" name="nom utilisateur" required>

  <label for="password">Mot de passe :</label>

  <input type="password" id="password" name="motdepasse" placeholder="Mot de passe" required>

  <input type="submit" value="Envoyer">

</form>
```

CONNAISSANCES DU COURS

Exercice 2

Cet exercice est un QCM. Cocher la (ou les) bonne(s) réponse(s)

1. Parmi les réponses suivantes, que permet d'effectuer la méthode POST du protocole http ?
 - Définir le style d'une page web.
 - Pirater des données bancaires.
 - Envoyer une page web vers le client.
 - Envoyer les données saisies dans un formulaire HTML vers un serveur.

2. Un internaute clique sur un lien qui envoie la requête http suivante à un serveur :

`http://jaimelaneige.com/ma_planche/traitement.php?nom=Snow&prenom=Jon`

Que demande cette requête au serveur ?

- De renvoyer le fichier traitement.php en identifiant nom et prénom à Snow et Jon.
 - D'exécuter le fichier traitement.php en identifiant nom et prénom à Snow et Jon.
 - D'indiquer si Jon Snow a bien pris son traitement.
 - De renvoyer le fichier traitement.php en affichant prénom et nom : Jon
3. Que peut-on affirmer au sujet des formulaires de pages Web ?
- Les variables d'une requête POST sont affichées dans l'URL.
 - Une requête GET peut être utilisée pour transmettre des données sensibles.
 - Les variables d'une requête GET sont affichées dans l'URL.
 - Une requête POST peut être utilisée pour transmettre des données sensibles.
4. On considère le formulaire ci-dessous :

Quel est votre langage préféré ?

Python ☐ Java ☐ Php ☐

Quelle balise parmi les quatre suivantes a été utilisée pour les cases à cocher ?

- `<input type="radio">`
 - `<input type="circle">`
 - `<input type="checkbox">`
 - `<input type="square">`
5. Un élément form (un formulaire) d'une page HTML contient un élément button de type submit. Un clic sur ce bouton :
- envoie les données du formulaire vers la page définie par l'attribut action de l'élément form
 - efface les données entrées par l'utilisateur dans le formulaire
 - envoie les données du formulaire vers la page définie par l'attribut method de l'élément form
 - ne fait rien du tout si un script javascript n'est pas associé au bouton
6. Dans un formulaire sur un page web, pour transmettre des données sécurisées comme un mot de passe ou un numéro de carte bancaire, il vaut mieux utiliser la méthode :
- HEAD
 - GET
 - HTTPS
 - POST

PASSER DU FORMULAIRE AU CODE HTML

Exercice 3

Écrire un extrait de formulaire html (balise <form> et tout ce qui est dedans) afin qu'après avoir ouvert puis rempli ce formulaire sur le navigateur, ce dernier affiche l'URL ci-dessous. On s'efforcera de proposer un formulaire "convivial" où les champs sont bien identifiables par l'utilisateur.



Exercice 4

Écrire un extrait de formulaire (balise <form> et tout ce qui est dedans)) afin que ce formulaire demande :

- Un nom composé de trois à douze lettres majuscules, à renseigner obligatoirement
- Un mot de passe à renseigner obligatoirement
- Un pays à sélectionner dans une liste déroulante de trois pays

On fera en sorte que le mot de passe soit transmis au serveur de manière la plus sécurisée possible.

[Retour au sommaire du document](#)

Proposition de Correction : EXEMPLE D'ÉVALUATION

LIRE DU CODE HTML

Exercice 1

Quelques questions sur l'extrait de code html reproduit ci-dessous.

1. Que va produire cet extrait lors de son exécution sur un navigateur ?
Il va produire un formulaire avec un champ de saisie pour le nom de l'utilisateur, un champ de saisie pour le mot de passe et un bouton de soumission.
2. Expliquer le sens de la balise `<form>` et de l'attribut *action* présents en première ligne de cet extrait
La balise `<form>` est une balise de formulaire qui permet de créer un formulaire. L'attribut *action* est un attribut qui permet de spécifier l'URL du fichier qui va traiter les données du formulaire.
3. Expliquer ce que signifie l'attribut *method* à l'intérieur de la balise `<form>`, quelle autre valeur aurait-il pu prendre ?
L'attribut *method* à l'intérieur de la balise `<form>` est un attribut qui permet de spécifier la méthode de transmission des données du formulaire. Il aurait pu prendre la valeur "post".
4. Quelle est l'utilité de l'attribut *name* à l'intérieur de la balise `<input>` ?
L'attribut *name* à l'intérieur de la balise `<input>` est un attribut qui permet de spécifier le nom de la variable qui va contenir les données du formulaire.
5. Quelle est l'utilité de l'attribut *type* à l'intérieur de la balise `<input>` ?
L'attribut *type* à l'intérieur de la balise `<input>` est un attribut qui permet de spécifier le type de données (composant) que le champ de saisie va contenir.
6. Quelle est l'utilité de la balise `<label>` présente dans cet extrait, et des attributs *for* et *id* présent dans la balise `<label>`?
La balise `<label>` présente dans cet extrait est une balise qui permet de créer une étiquette associée à un champ de saisie, cet attribut fournit un lien clair entre un texte de description et un élément de formulaire comme `<input>`.
7. Quelle est l'utilité de l'attribut *required* présent dans la balise `<input>`?
L'attribut *required* présent dans la balise `<input>` est un attribut qui permet de spécifier que le champ de saisie est obligatoire.
8. Quelle est l'utilité de l'attribut *placeholder*?
L'attribut *placeholder* est un attribut qui permet de spécifier un texte qui apparaît dans le champ de saisie lorsque celui-ci est vide.

Extrait :

```
<form action="https://www.exempledesite.com/" method="get">

<label for="username">Nom :</label>

<input type="text" id="username" name="nom utilisateur" required>

<label for="password">Mot de passe :</label>

<input type="password" id="password" name="motdepasse" placeholder="Mot de passe" required>

<input type="submit" value="Envoyer">

</form>
```

CONNAISSANCE DU COURS

Exercice 2

Cet exercice est un QCM. Cocher la (ou les) bonne(s) réponse(s)

1. Parmi les réponses suivantes, que permet d'effectuer la méthode POST du protocole http ?
 - Définir le style d'une page web.
 - Pirater des données bancaires.
 - Envoyer une page web vers le client.
 - Envoyer les données saisies dans un formulaire HTML vers un serveur.
2. Un internaute clique sur un lien qui envoie la requête http suivante à un serveur:

`http://jaimelaneige.com/ma_planche/traitement.php?nom=Snow&prenom=Jon`

Que demande cette requête au serveur ?

- De renvoyer le fichier traitement.php en identifiant nom et prénom à Snow et Jon.
 - D'exécuter le fichier traitement.php en identifiant nom et prénom à Snow et Jon.
 - D'indiquer si Jon Snow a bien pris son traitement.
 - De renvoyer le fichier traitement.php en affichant prénom et nom : Jon
3. Que peut-on affirmer au sujet des formulaires de pages Web ?
 - Les variables d'une requête POST sont affichées dans l'URL.
 - Une requête GET peut être utilisée pour transmettre des données sensibles.
 - Les variables d'une requête GET sont affichées dans l'URL.
 - Une requête POST peut être utilisée pour transmettre des données sensibles.
 4. On considère le formulaire ci-dessous :

Quel est votre langage préféré ?

Python ☐ Java ☐ Php ☐


Quelle balise parmi les quatre suivantes a été utilisée pour les cases à cocher ?

- `<input type="radio">`
 - `<input type="circle">`
 - `<input type="checkbox">`
 - `<input type="square">`
5. Un élément form (un formulaire) d'une page HTML contient un élément button de type submit. Un clic sur ce bouton :
 - envoie les données du formulaire vers l défini par l'attribut action de l'élément form
 - efface les données entrées par l'utilisateur dans le formulaire
 - envoie les données du formulaire vers la page définie par l'attribut method de l'élément form
 - ne fait rien du tout si un script javascript n'est pas associé au bouton
 6. Dans un formulaire sur un page web, pour transmettre des données sécurisées comme un mot de passe ou un numéro de carte bancaire, il vaut mieux utiliser la méthode :
 - HEAD
 - GET
 - HTTPS
 - POST

PASSER DU FORMULAIRE AU CODE HTML

Exercice 3

Écrire un extrait de formulaire html (balise <form> et tout ce qui est dedans) afin qu'après avoir ouvert puis rempli ce formulaire sur le navigateur, ce dernier affiche l'URL ci-dessous. On s'efforcera de proposer un formulaire "convivial" où les champs sont bien identifiables par l'utilisateur.

 <https://www.notresite.com/formulaire?nomUtilisateur=Jules&motDePasse=chut>



```
<form action="cible.php" method="get">

<label for="username">Nom :</label>

<input type="text" id="username" name="nomUtilisateur" placeholder="identifiant" required>

<label for="password">Mot de passe :</label>

<input type="password" id="password" name="motDePasse" placeholder="Mot de passe" required>

<input type="submit" value="Envoyer">

</form>
```

Exercice 4

Écrire un extrait de formulaire (balise <form> et tout ce qui est dedans)) afin que ce formulaire demande :

- Un nom composé de trois à douze lettres majuscules, à renseigner obligatoirement
- Un mot de passe à renseigner obligatoirement
- Un pays à sélectionner dans une liste déroulante de trois pays

On fera en sorte que le mot de passe soit transmis au serveur de manière la plus sécurisée possible.

```
<form action="https://exemplesite.com" method="post">

<label for="nom">Nom :</label>

<input type="text" id="nom" name="nom" minlength="3" maxlength="12" pattern="[A-Z]+" required>

<label for="password">Mot de passe :</label>

<input type="password" id="motDePasse" name="motDePasse" required>

<label for="pays">Pays :</label>

<select name="pays" id="pays">

  <option value="Liban">Liban</option>

  <option value="Serbie">Serbie</option>

  <option value="Allemagne">Allemagne</option>

</select>

<input type="submit" value="Soumettre">

</form>
```

Projet final

Par groupe de deux ou de trois, créer une page web sur un personnage qui a joué un rôle dans l'histoire de l'informatique (Alan Turing, John Von Neumann, Grace Hooper, Ada Lovelace, Guido Van Rossum, Tim Berners-Lee...) avec le cahier des charges suivants :

L'évaluation portera sur les points suivants :

- La richesse et la qualité des informations présentées dans le site.
- Le site doit être lisible, clair et bien structuré (pensez à utiliser les balises <div> et <section>).
- La mise en forme : Utilisation du langage css (couleurs, police....).
- Ajouter des photos, des liens hypertextes, au moins deux composants graphiques.
- Ajouter au moins un gestionnaire d'événement qui fait appel à une fonction javascript pour rendre la page interactive. Ex : Cliquer sur un bouton change la couleur du fond.
- Le site doit posséder un formulaire qui permet de récupérer des valeurs dans une fonction JS. Les valeurs de chaque champ seront traitées et le résultat de ce traitement apparaîtra dans votre site.
- La présentation orale, l'explication du code et l'interaction.

[Retour au sommaire du document](#)