Les formulaires HTML et JavaScript

Table des matières

I. Contexte

Durée : 45 min

Environnement de travail : Repl.it

Pré-requis : Connaissance du HTML, du CSS et des bases du JavaScript

Contexte

Sans JavaScript, nos formulaires HTML sont envoyés et traités en l'état par le serveur. Avec JavaScript, nous pourrons accéder à ces éléments pour lire ou écrire une valeur, associer un gestionnaire d'événements et ainsi modifier le comportement de notre page web. Ceci avant l'envoi éventuel de nos données vers le serveur.

Dans ce cours, nous apprendrons à manipuler les différents types de champs d'un formulaire HTML, à valider les données pendant la saisie et à proposer des suggestions à l'utilisateur en fonction de sa saisie.

II. Manipuler les champs de type texte

Objectifs

- Lire/écrire la valeur d'une zone de texte
- Manipuler les écouteurs d'événements liés aux champs texte

Mise en situation

Notre formulaire HTML est en place, nous savons sélectionner ses éléments par l'intermédiaire du DOM. Il est temps d'interagir avec ces derniers. Nous allons voir comment afficher les données saisies par l'utilisateur et comment écrire dans un champ texte.

Méthode **Accès et modification de la valeur d'une zone de texte**

Afin d'accéder à la valeur d'une zone de texte (text ou textarea) on usera de la propriété value de l'élément du DOM correspondant. En modifiant `value`, nous serons donc capables de changer la valeur de la zone de texte.

Nous pourrons lier deux événements spécifiques : **focus** et **blur**. L'événement focus est appelé lors de la mise au point tandis que l'événement blur l'est lors de la perte de la mise au point.

Exemple **Accès**

```html
<form>
  <input type="text" id="short-txt" name="short-txt" value="un texte court">
  <textarea id="long-txt" name="long-txt">Un texte long</textarea>
</form>
```

```js
var shorttxt = document.getElementById('short-txt')
var longtxt = document.getElementById('long-txt')

// Accès à la valeur par la propriété value
console.log(shorttxt.value, longtxt.value)
```

Exemple **Modification**

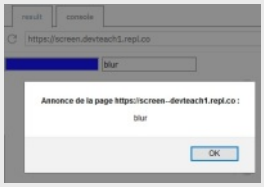
```
var longtxt = document.getElementById('long-txt')

longtxt.value = 'Un texte modifié !'

console.log(shorttxt.value, longtxt.value)
```

Les événements **focus** et **blur** peuvent être utilisés pour apporter des éléments d'information avant et/ou après la saisie d'une valeur. Pour les mettre en place, nous allons devoir créer les écouteurs d'événements associés.

Exemple **Focus et blur sur une zone de texte**

```
<form>
  <input type="text" id="txt-focus" name="txt-focus">
  <input type="text" id="txt-blur" name="txt-blur">
</form>
```

```
var txtfocus = document.getElementById('txt-focus')
var txtblur = document.getElementById('txt-blur')

txtfocus.addEventListener('focus', (event) => {
  event.target.style.background = 'blue'
})

txtblur.addEventListener('blur', (event) => {
  alert('blur')
})
```



Pour plus de précisions concernant les écouteurs d'événements, reportez-vous au cours sur la Programmation événementielle.

Remarque

Il est tout à fait possible de procéder à la modification de style lors du focus par le biais du CSS. Il s'agit d'une approche plus abordable de par sa facilité d'exécution.

Exemple

Nous retrouvons ci-dessous, un formulaire contenant une zone de saisie de type texte.

```
<form action="">
  <input type="text">
</form>
```

La pseudo-classe d'action CSS :focus va permettre d'assigner des propriétés CSS à notre élément HTML.

```
form input:focus {
 background: #f7c7cf;/* Va colorer le fond en rose */
 border-color: #af7fdd;/* Va colorer la bordure en violet */
 }
```

Attention

Gardons toujours à l'esprit que l'ajout d'écouteurs d'événements peut être coûteux pour votre navigateur. De plus, l'événement **focus** est déclenché en permanence tant que le champ en question est cliqué. Attention donc aux mauvaises surprises.

Exemple

Nous pouvons également donner le **focus** à un champ, ou l'enlever en invoquant sur notre élément les méthodes `focus()` et `blur()`.

```
let txtblur = document.querySelector('#txt-blur')
txtblur.focus() // Cible le champ txtblur
let txtfocus = document.querySelector('#txt-focus')
txtfocus.blur() // Permet de ne plus cibler le champ txtfocus
```

Syntaxe **À retenir**

- Pour lire ou écrire la valeur d'un champ nous utilisons la propriété `value` de l'élément du formulaire souhaité.
- Les événements **focus** et **blur** nous permettent de lier des actions avant ou après une saisie dans une zone de texte.

III. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :

1

Dans cet exercice, vous devrez utiliser les événements **focus** et **blur**. On dispose du code HTML suivant :

```
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width">
<title>Les formulaires HTML et JavaScript</title>
<link href="style.css" rel="stylesheet" type="text/css"/>
</head>
<body>
<div class="form-row">
<input type="text" name="text" id="text" class="inputPinkOnFocus">
<input type="textarea" name="textarea" id="textarea" class="inputPinkOnFocus">
<input type="password" name="password" id="password" class="inputPinkOnFocus">
</div>
<script src="script.js"></script>
```

1 https://repl.it/

```
</body>
</html>
```

```
.form-row{
   width: 100%;
   margin: 20px;
}

.form-row span {
  margin-left: 10px;
}
```

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail Repl.it.

Question

[solution n°1 p.17]

Dans un premier temps, vous créerez une classe commune à tous les éléments `input`.

Par la suite, au **focus**, vous devrez faire en sorte que le background de l'élément input prenne la couleur rose et que sa police prenne la couleur bleu.

Enfin, vous créerez un élément **blur** qui annulera les effets du **focus.**

IV. Manipuler les checkbox, radio et select

Objectifs

- Détecter le changement d'état d'une case à cocher, d'un bouton radio et d'une liste déroulante
- Manipuler la valeur de ces champs
- Manipuler les écouteurs d'événements liés à ces champs

Mise en situation

Nous allons voir comment afficher et modifier la valeur des autres types de champs présents dans un formulaire, à savoir les cases à cocher, les boutons radio et les listes déroulantes.

Méthode

Comme pour les champs de type texte, nous pourrons accéder à leur valeur grâce à la propriété `value` de l'élément du DOM correspondant. En modifiant `value`, nous pourrons modifier la valeur choisie.

Nous pourrons lier l'événement **change** permettant de détecter la modification du choix par l'utilisateur.

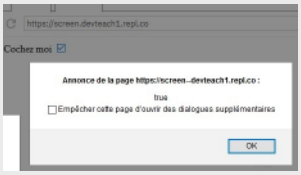
Les éléments de formulaire donnant lieu à un choix de l'utilisateur ont en commun l'événement **change**.

Exemple **Application aux Checkbox**

Ici, nous ajoutons un écouteur d'événement **change** à notre case à cocher. De plus, pour détecter si la case est cochée ou non, nous utiliserons la propriété `checked` de l'objet `Event` de JavaScript.

```
<label for="check">Cochez-moi</label>
<input type="checkbox" id="check" name="check">
```

```
var checkbox = document.getElementById('check')

checkbox.addEventListener('change', () => {
  alert(event.target.checked) // True si coché, sinon false
})
```

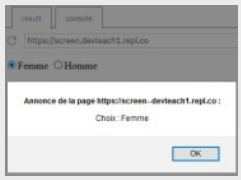


Exemple **Application aux boutons radio**

Pour les boutons radio, l'utilisateur dispose de plusieurs choix de réponses, mais ne peut en sélectionner qu'une. Nous devons donc placer un écouteur d'événements sur chacun des choix possibles.

```
<input type="radio" name="gender" value="Femme">Femme
<input type="radio" name="gender" value="Homme">Homme
```

```
var btnradio = document.getElementsByName('gender')

for (var count=0; count<btnradio.length; count++) {
    btnradio[count].addEventListener('change', (event) => {
        alert(`Choix : ${event.target.value}`);
    });
}
```



Nous pouvons aussi forcer le choix en assignant une valeur à `value`. Attention, cette option doit exister au niveau des choix utilisateurs.

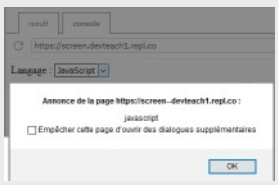
```
event.target.value = "Femme"
```

Exemple **Application aux listes déroulantes**

De la même manière que pour les boutons radio, nous allons placer un écouteur d'événement **change** et utiliser la propriété `value` pour afficher le choix de l'utilisateur.

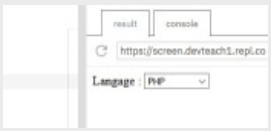
```
<label for="language">Langage :</label>
<select name="language" id="language">
    <option value="javascript" selected>JavaScript</option>
    <option value="php">PHP</option>
    <option value="python">Python</option>
</select>
```

```
var language = document.getElementById('language')

language.addEventListener('change', (event) => {
  alert(event.target.value)
})
```



Nous pouvons aussi forcer le choix en assignant une valeur à `value`. Attention, cette valeur doit exister dans la liste déroulante.

```
event.target.value = "php"
```



Syntaxe **À retenir**

- Pour lire ou modifier le choix d'un utilisateur (case à cocher, bouton radio, liste déroulante), nous utilisons la propriété `value` de l'élément du formulaire souhaité.
- Pour « écouter » les choix de l'utilisateur, nous utilisons l'événement **change**.

V. Exercice : Appliquez la notion

On dispose du formulaire HTML suivant :

```
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>Les formulaires HTML et JavaScript</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <form>
      <div class="form-row">
        <input type="checkbox" name="consentement" id="consentement">J'accepte de recevoir la newsletter.
      </div>
      <div class="form-row">
        Ainsi que des informations partenaires ?
        <input type="radio" name="btn-radio" value="oui">Oui
        <input type="radio" name="btn-radio" value="non">Non
      </div>
      <div class="form-row">
        <select name="pays" id="pays">
          <option value="France">France</option>
          <option value="Espagne">Espagne</option>
          <option value="Italie">Italie</option>
          <option value="Belgique">Belgique</option>
        </select>
        <span id="message"></span>
      </div>
    </form>
    <script src="script.js"></script>
  </body>
</html>
```

```css
.form-row{
  width: 100%;
  margin: 20px;
}

.form-row span {
  margin-left: 10px;
}
```

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail `Repl.it` :

1

Question 1

[solution n°2 p.17]

Une fois la checkbox cochée, une fenêtre pop-up indiquant *"Vous acceptez de recevoir la newsletter"* apparaît :

Dans le cas où l'utilisateur ne souhaite pas recevoir de newsletter, il pourra sélectionner le bouton radio *"non"*.

Question 2

[solution n°3 p.17]

Affichez un message dans le span `#message` indiquant le pays sélectionné lorsque le choix du pays sera modifié.

Indice :

`element.innerText` permet d'écrire une chaîne de caractères dans un élément *HTML*.

VI. Validation et soumission des formulaires HTML

Objectifs

- Maîtriser l'objet formulaire du DOM
- Soumettre un formulaire
- Valider les données saisies

Mise en situation

Une seule règle d'or, *ne jamais faire confiance à l'utilisateur.* En d'autres termes, vérifiez toujours les données saisies par l'utilisateur avant qu'elles ne soient traitées.

Lors du clic sur le bouton pour soumettre le formulaire, les données sont envoyées à la ressource indiquée dans l'attribut `action` de la balise `<form>`.

Nous allons voir comment intercepter les données et valider ou invalider le formulaire.

Méthode

Avant d'être réellement soumis au serveur, un événement de type **submit** est déclenché sur l'élément du DOM correspondant au formulaire. Nous pourrons le capter en ajoutant un événement et ainsi accéder aux données du formulaire.

À partir de ce moment, la balle est dans notre camp. Nous pourrons interdire certaines saisies, suggérer des données à saisir, informer sur les données attendues ou bien plus encore selon nos besoins.

1 https://repl.it/

```
form.addEventListener('submit', (event) => {
 // ici nos traitements sur les données du formulaire
 event.preventDefault(); // Annule l'envoi des données
})
```

L'interface Event représente un événement qui se produit dans le DOM. Une fois la méthode `preventDefault()` rattachée à cet événement, elle indique que l'action par défaut ne pourra pas être exécutée.

Exemple : la redirection après le clic d'un utilisateur sur un élément `<a>` peut être annulée grâce à la méthode `preventDefault()` appliquée à l'événement.

L'objet formulaire du DOM possède une propriété `elements` qui représente l'ensemble des champs de saisie sous forme d'un tableau. Ces éléments possèdent eux-mêmes des propriétés pour identifier les champs du formulaire :

- `name` : le nom du champ,
- `type` : le type de champ (*password*, *text*, etc.),
- `value` : la valeur du champ, etc.

Exemple **Accès aux champs d'un formulaire**

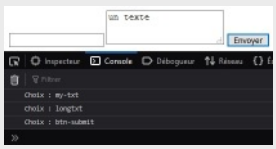
```
var form = document.querySelector('form')

// Nombre d'éléments du formulaire
console.log(`Choix : ${form.elements.length}`);

// Nom du 1er élément
console.log(`Choix : ${form.elements[0].name}`);

// Type du second élément
console.log(`Choix : ${form.elements[1].type}`);

// Valeur du second élément
console.log(`Choix : ${form.elements[1].value}`);
```

Pour optimiser notre code, nous pouvons ici, par exemple, parcourir les éléments du tableau **elements** en utilisant des structures itératives de type `for` couplées à des structures conditionnelles de type `if` ou `switch/case`.

Exemple **Soumission du formulaire**

Le formulaire envoyé donne lieu au déclenchement d'un événement de type **submit**.

```
var form = document.querySelector('form')

form.addEventListener('submit', (event => {
  for(var countElement=0; countElement<form.elements.length; countElement++) {
    console.log(`Choix : ${form.elements[countElement].name}`);
  }
  event.preventDefault()
}))
```



L'événement **submit** n'est là que pour nous permettre d'accéder aux éléments de notre formulaire. Une fois nos vérifications faites, nous pourrons stopper la soumission du formulaire en invoquant la méthode `preventDefault()` de l'objet `Event`.

Si cette dernière n'est pas appelée, nos traitements seront appliqués, puis notre formulaire envoyé au serveur.

Complément **Méthode submit()**

L'objet formulaire du DOM possède une méthode `submit()` que nous pouvons appeler pour soumettre un formulaire, couplée à un événement **click** sur un bouton, par exemple. Toutefois, avec cette méthode, aucun événement **submit** ne sera déclenché et les différentes contraintes de validation (*required*, *pattern*, *min*, etc.) que nous aurions pu mettre en place directement au niveau des champs de notre formulaire ne seront pas exécutées. Nous devrons réaliser toute la validation à la main.

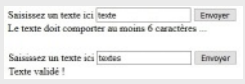
Pourquoi faire valider les données par le navigateur ?

Nous pourrons ainsi gérer des erreurs de saisie avant que le formulaire ne soit envoyé au serveur. Cela évitera la perte des données du formulaire pour l'utilisateur et des allers-retours inutiles pour le serveur.

Exemple **Validation des données pendant la saisie utilisateur**

Nous utiliserons ici l'écouteur d'événements **input**. Il sera déclenché à chaque modification de la valeur saisie.

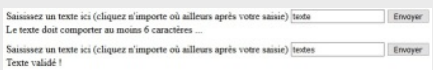
```
var mytxt = document.getElementById('my-txt')
var error = document.getElementById('error')

mytxt.addEventListener('input', (event) => {
  var mytxtValue = event.target.value
  if (mytxtValue.length < 6) {
    error.innerText = 'Le texte doit comporter au moins 6 caractères ...'
  } else {
    error.innerText = 'Texte validé !'
  }
})
```



Exemple **Validation des données après la saisie utilisateur**

La méthode sera sensiblement identique, seul l'écouteur d'événements sera différent. Nous utiliserons ici **blur.**

```
mytxt.addEventListener('blur', (event) => { ... })
```



Syntaxe **À retenir**

- La soumission d'un formulaire déclenche un événement **submit**.
- En ajoutant un écouteur d'événements, nous pourrons accéder au contenu du formulaire grâce à l'objet `Elements` de la balise `<form>` et valider les données pendant la saisie avec l'événement **input** ou après la saisie avec **blur**.

VII. Exercice : Appliquez la notion

Vous disposez du formulaire HTML suivant :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>Les formulaires HTML et JavaScript</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <form>
      <label for="email">Email</label>
      <input type="text" id="email" name="email">
      <label for="password">Mot de passe</label>
      <input type="password" id="password" name="password">
      <input type="submit" id="btn-submit" name="btn-submit">
    </form>
    <div id="error"></div>
    <script src="script.js"></script>
  </body>
</html>
```

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail `Repl.it`.

1

Question 1

[solution n°4 p.18]

Contrôlez le format de l'e-mail à l'aide de l'expression régulière suivante : `/^([0-9a-zA-Z].*?@([0-9a-zA-Z].*\.\w{2,4}))$/`.

Question 2

[solution n°5 p.18]

Contrôlez que le mot de passe comporte au moins 8 caractères.

Question 3

[solution n°6 p.19]

Une fois ces conditions respectées, le formulaire peut être validé.

VIII. Auto-évaluation

A. Exercice final

Exercice 1

[solution n°7 p.19]

Exercice

Lorsque l'on utilise JavaScript pour gérer des formulaires, ceux-ci sont alors gérés au niveau...

- ○ Du navigateur
- ○ Du serveur

Exercice

1 https://repl.it/

Quelle propriété permet d'accéder à la valeur d'un élément de formulaire tel qu'un **input** ?

Exercice

Cochez les affirmations correctes.

- ☐ L'événement **focus** est déclenché quand on clique dans une zone de texte
- ☐ L'événement **blur** est déclenché quand on soumet un formulaire
- ☐ L'événement **change** permet de détecter le choix d'un utilisateur dans une liste déroulante
- ☐ Il n'est pas possible de détecter si une case à cocher est cochée

Exercice

La soumission d'un formulaire déclenche l'événement...

- ○ blur
- ○ focus
- ○ submit
- ○ change
- ○ form

Exercice

Pour "écouter" les modifications et choix de l'utilisateur, on utilise l'événement...

- ○ update
- ○ change
- ○ modify

Exercice 7

[solution n°8 p.20]

Exercice

Grâce à laquelle de ces déclarations pourrait-on modifier la valeur d'une zone de texte ?

- ☐ document.getElementById('txt').input = 'mon texte'
- ☐ document.getElementById('txt').valeur = 'mon texte'
- ☐ document.getElementById('txt').value = 'mon texte'
- ☐ event.target.value = 'mon texte'

Exercice

Grâce à laquelle de ces déclarations pourrait-on ajouter un écouteur d'événement **submit** à un formulaire en bloquant l'envoi des données au serveur ?

- ○ form.addEventListener('click', (event) => {
  event.preventDefault();
  })

- form.addEventListener('submit', (event) => {
  event.preventDefault();
  })
- form.addEventListener('submit', () => {
  event.preventDefault();
  })

Exercice

Comment pourrait-on accéder au type du champ situé en seconde position dans le formulaire ?

- form.elements[2].type
- form.elements[1].name
- form.elements[1].type

Exercice

Grâce à quel événement est-il possible de valider les données pendant la saisie utilisateur ?

Exercice

`event.target.value.length` permet...

- [ ] D'afficher la longueur d'un texte saisi
- [ ] De mettre en place un contrôle sur une longueur minimum
- [ ] D'afficher le nombre de champs d'un formulaire

B. Exercice : Défi

Maintenant, vous allez devoir mettre en place la validation du formulaire de création de compte du site de e-commerce dont vous êtes le développeur, spécialiste du JS.

Vous disposez du formulaire HTML suivant comme base écrite par votre prédécesseur :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>Les formulaires HTML et JavaScript</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <form>
      <div class="field">
        <label for="identifiant">Identifiant</label>
        <input type="text" id="identifiant" name="identifiant">
      </div>
      <div class="field">
        <label for="password">Mot de passe</label>
        <input type="password" id="password" name="password">
      </div>
      <div class="field">
        <label for="email">Email</label>
        <input type="text" id="email" name="email">
```

```
      </div>
      <div class="field">
        <label for="email-confirm">Confirmation de l'email</label>
        <input type="text" id="email-confirm" name="email-confirm">
      </div>
      <div class="field">
        <label for="nom">Nom</label>
        <input type="text" id="nom" name="nom">
      </div>
      <div class="field">
        <label for="prenom">Prénom</label>
        <input type="text" id="prenom" name="prenom">
      </div>
      <div class="field">
        <label for="annee">Année de naissance</label>
        <select id="annee" name="annee">
          <option value="1980">1980</option>
          <option value="1981">1981</option>
          <option value="1982">1982</option>
          <option value="1983">1983</option>
          <option value="1984">1984</option>
          <option value="1985">1985</option>
          <option value="1986">1986</option>
          <option value="1987">1987</option>
          <option value="1988">1988</option>
          <option value="1989">1989</option>
          <option value="1990">1990</option>
        </select>
        <div id="age"></div>
      </div>
      <input type="submit" id="btn-submit" name="btn-submit">
    </form>
    <script src="script.js"></script>
  </body>
</html>
```

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail `Repl.it`.

1

Question 1

[solution n°9 p.21]

Vous commencerez par mettre en place le contrôle sur l'identifiant. Afin que le formulaire puisse être valide, l'identifiant doit être obligatoire.

Petite subtilité : n'utilisez pas l'attribut `required`, car un utilisateur qui s'y connaît pourrait modifier la source HTML ! Il est toujours préférable d'effectuer les vérifications côté JS, puis côté back-end à la récupération des données.

Attention, l'envoi du formulaire doit être bloqué si les conditions ne sont pas respectées.

Question 2

[solution n°10 p.21]

Maintenant, ajoutez une vérification sur le mot de passe. Il est obligatoire et doit comporter au moins 6 caractères pour des raisons de sécurité.

---

1 https://repl.it/

Question 3 [solution n°11 p.21]

Afin de s'assurer que l'utilisateur ne fasse pas d'erreurs, et que son e-mail puisse être utilisé pour lui envoyer une newsletter, il va falloir mettre en place une vérification sur les adresses e-mails saisies : elles doivent être identiques et remplis par l'utilisateur.

Question 4 [solution n°12 p.22]

Rédigez un code regroupant tous les éléments de réponses des questions précédentes ainsi qu'un événement de type submit afin de valider le formulaire.

Solutions des exercices

p. 6 Solution n°1

```
let allinputs = document.getElementsByClassName('inputPinkOnFocus');

Array.from(allinputs).forEach(function(input){
input.addEventListener('focus', (event) => {
event.target.style.background = 'pink';
event.target.style.color = 'blue';
});

input.addEventListener('blur', () => {
event.target.style.background = '';
event.target.style.color = '';
});
});
```

p. 9 Solution n°2

```
let checkbox = document.getElementById('consentement')
let radio = document.getElementsByName('btn-radio')

checkbox.addEventListener('change', (event) => {
  if (event.target.checked) {
    alert('Vous acceptez de recevoir la newsletter')
    for(item of radio) {
      if (item.value == 'oui') {
        item.checked = true
      }
    }
  } else {
    for(item of radio) {
      item.checked = (item.value === 'non');
    }
  }
})
```

p. 9 Solution n°3

```
let checkbox = document.getElementById('consentement')
let radio = document.getElementsByName('btn-radio')
let select = document.getElementById('pays')
let message = document.querySelector('#message')

checkbox.addEventListener('change', (event) => {
  if (event.target.checked) {
    alert('Vous acceptez de recevoir la newsletter')
    for(item of radio) {
      if (item.value == 'oui') {
        item.checked = true
      }
    }
  } else {
    for(item of radio) {
      item.checked = false
```

```
    }
  }
})

select.addEventListener('change', (event) => {
  message.innerText = `Vous avez indiqué être en ${event.target.value}`
})
```

p. 12 Solution n°4

```
let email = document.getElementById('email')
let form = document.querySelector('form')
let error = document.getElementById('error')

var regexEmail = /^([0-9a-zA-Z].*?@([0-9a-zA-Z].*\.\w{2,4}))$/

email.addEventListener('input', (event) => {
  if (!regexEmail.test(event.target.value)) {
    error.innerText = 'Le format de l\'email est incorrect'
  } else {
    error.innerText = ''
  }
})
```

Cette expression régulière est assez simple, et certaines adresses e-mail fonctionnelles peuvent être considérées comme non valides.

Si besoin, on peut se tourner vers une expression régulière plus complète, comme celle disponible sur ce site : https://emailregex.com/[1]

p. 12 Solution n°5

```
let email = document.getElementById('email')
let password = document.getElementById('password')
let form = document.querySelector('form')
let error = document.getElementById('error')

var regexEmail = /^([0-9a-zA-Z].*?@([0-9a-zA-Z].*\.\w{2,4}))$/

email.addEventListener('input', (event) => {
  if (!regexEmail.test(event.target.value)) {
    error.innerText = 'Le format de l\'email est incorrect'
  } else {
    error.innerText = ''
  }
})

password.addEventListener('input', (event) => {
  if (event.target.value.length < 8) {
    error.innerText = 'le mot de passe doit contenir au moins 8 caractères'
  } else {
    error.innerText = ''
  }
})
```

1 https://emailregex.com

```
23
```

p. 12 **Solution n°6**

```
let email = document.getElementById('email')
let password = document.getElementById('password')
let form = document.querySelector('form')
let error = document.getElementById('error')

var regexEmail = /^([0-9a-zA-Z].*?@([0-9a-zA-Z].*\.\w{2,4}))$/

email.addEventListener('input', (event) => {
  if (!regexEmail.test(event.target.value)) {
    error.innerText = 'Le format de l\'email est incorrect'
  } else {
    error.innerText = ''
  }
})

password.addEventListener('input', (event) => {
  if (event.target.value.length < 8) {
    error.innerText = 'le mot de passe doit contenir au moins 8 caractères'
  } else {
    error.innerText = ''
  }
})

form.addEventListener('submit', (event) => {
  if (error.innerText !== '') {
    event.preventDefault()
    alert("Le formulaire contient des erreurs et n'a pas été envoyé")
  }
})
```

Exercice p. 12 **Solution n°7**

Exercice

Lorsque l'on utilise JavaScript pour gérer des formulaires, ceux-ci sont alors gérés au niveau...

- (●) Du navigateur
- ( ) Du serveur

Exercice

Quelle propriété permet d'accéder à la valeur d'un élément de formulaire tel qu'un **input** ?

value

Exercice

Cochez les affirmations correctes.

- [x] L'événement **focus** est déclenché quand on clique dans une zone de texte
- [ ] L'événement **blur** est déclenché quand on soumet un formulaire
- [x] L'événement **change** permet de détecter le choix d'un utilisateur dans une liste déroulante
- [ ] Il n'est pas possible de détecter si une case à cocher est cochée

Exercice

La soumission d'un formulaire déclenche l'événement...

- ○ blur
- ○ focus
- ◉ submit
- ○ change
- ○ form

Exercice

Pour "écouter" les modifications et choix de l'utilisateur, on utilise l'événement...

- ○ update
- ◉ change
- ○ modify

Exercice p. 13 Solution n°8

Exercice

Grâce à laquelle de ces déclarations pourrait-on modifier la valeur d'une zone de texte ?

- [ ] document.getElementById('txt').input = 'mon texte'
- [ ] document.getElementById('txt').valeur = 'mon texte'
- [x] document.getElementById('txt').value = 'mon texte'
- [x] event.target.value = 'mon texte'

Exercice

Grâce à laquelle de ces déclarations pourrait-on ajouter un écouteur d'événement **submit** à un formulaire en bloquant l'envoi des données au serveur ?

- ○
```
form.addEventListener('click', (event) => {
event.preventDefault();
})
```
- ◉
```
form.addEventListener('submit', (event) => {
event.preventDefault();
})
```

- form.addEventListener('submit', () => {
  event.preventDefault();
  })

Exercice

Comment pourrait-on accéder au type du champ situé en seconde position dans le formulaire ?

- form.elements[2].type
  Attention, l'indice commence à 0 dans un tableau (élément 1 => 0, élément 2 => 1).
- form.elements[1].name
- form.elements[1].type

Exercice

Grâce à quel événement est-il possible de valider les données pendant la saisie utilisateur ?

input

Exercice

`event.target.value.length` permet...

- [x] D'afficher la longueur d'un texte saisi
- [x] De mettre en place un contrôle sur une longueur minimum
- [ ] D'afficher le nombre de champs d'un formulaire

p. 15 **Solution n°9**

```
let identifiant = document.getElementById('identifiant')

 function validIdentifiant(value) {
   if (value === '') {
     return 'l\'identifiant est obligatoire\n'
   }

   return '';
 }
```

p. 15 **Solution n°10**

```
let password = document.getElementById('password')
function validPassword(value) {
 if (value.length < 6) {
   return 'le mot de passe doit contenir au moins 6 caractères\n'
 }

 return '';
}
```

p. 16 **Solution n°11**

```
let email = document.getElementById('email')
let emailConfirmation = document.getElementById('email-confirm')

function validEmail(email, emailConfirmation) {
  if (email !== emailConfirmation) {
    return 'Les 2 adresses emails doivent être identiques\n'
  }

   return '';
}
```

p. 16 **Solution n°12**

```
let identifiant = document.getElementById('identifiant')
let password = document.getElementById('password')
let email = document.getElementById('email')
let emailConfirmation = document.getElementById('email-confirm')
let annee = document.getElementById('annee')
let age = document.getElementById('age')
let form = document.querySelector('form')

let error = '';

  function validIdentifiant(value) {
    if (value === '') {
      return 'l\'identifiant est obligatoire\n'
   }

  return '';
}

function validPassword(value) {
  if (value.length < 6) {
        return 'le mot de passe doit contenir au moins 6 caractères\n'
      }

    return '';
  }

function validEmail(email, emailConfirmation) {
  if (email !== emailConfirmation) {
     return 'Les 2 adresses emails doivent être identiques\n'
  }

  return '';
}

    form.addEventListener('submit', (event) => {
      error = ''
      error += validEmail(email.value, emailConfirmation.value)
      for(let count=0; count<form.elements.length; count++) {
       if (form.elements[count].name === 'identifiant') {
          error+= validIdentifiant(form.elements[count].value)
        } else if (form.elements[count].name === 'password'){
          error += validPassword(form.elements[count].value)
          }
 }
```

```
45
46    if (error !== '') {
47          alert(error)
48          event.preventDefault()
49      }
50  })
```