**Exemple 1**
***Version* 1**

- Saisir le script
- Créer un dossier nommé "Javascript" à la racine du site
- Enregistrer le script dans ce dossier sous le nom :
  "**JS_page1_v1.html**"
- La tester dans un navigateur

HTML

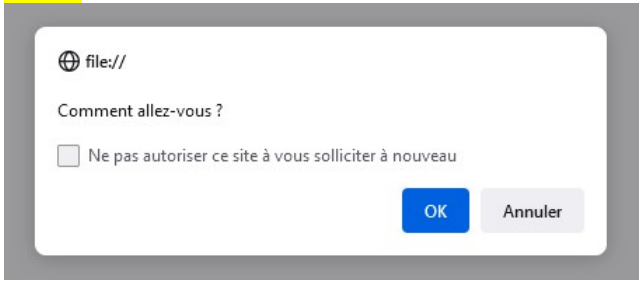
**Exemple 1**
***Version* 2**

**Exemple 1**

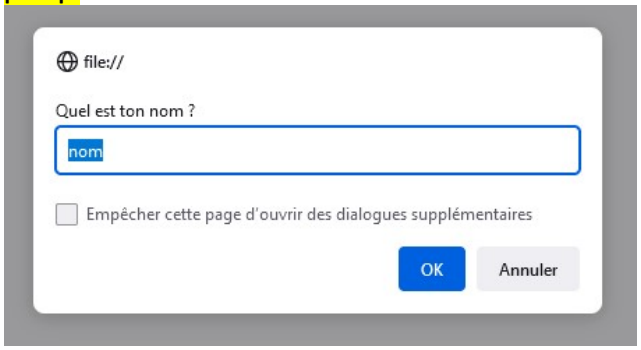
Sommaire

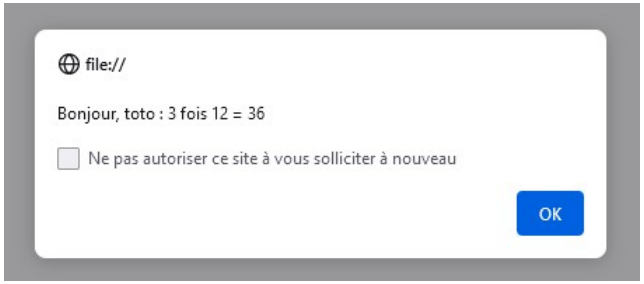
**alert**


**confirm**




**prompt**




Remarques

Le navigateur parcourt le code en commençant par l'entête.
Après avoir afficher les 4 fenêtres et tout le code Javascript, il passe au body et affiche la page…

; ➡ ne pas oublier le point-virgule à la fin d'une instruction
-> Pas obligatoire -> mais une bonne pratique
-> pour certaines instructions « compressée » : risque d'erreurs !

## 1. Des généralités

### a. Afficher des résultats en console

⇢ Pour afficher la consolesur Firefox : Menu / Développement web / Console web
On peut tester directement des actions dans la console, à partir de la zone de saisie.

⇢ L'instruction : console.log("éléments_a_afficher")

### b. Les types de données

**■ Les nombres**

⇢ Nombres et opérations : somme : + | soustration : – | produit : * | division : /

⇢ *Deux fonctions de la bibliothèque Math* :
Math.random() renvoi un réel supérieur à 0 et strictement inférieur à 1
Math.floor() renvoie la partie entière d'un réel

Exemple : pour obtenir un entier compris entre 1 et 6, il faut saisir :

➡Math.floor(Math.random()*6)+1

**■ Les chaînes de caractères**

⇢ La concaténation de 2 chaînes de caractères : "bon " + "jour" ➡ "bonjour"

*On considère la variable* : mot = "bonjour"

⇢ Longueur d'une chaîne : mot.length ; ➡ 7

⇢ Le rang d'un caractère d'une chaîne : mot[2] ; ➡"j"

⇢ Découper une chaîne : mot.slice(1,5) ; ➡ "onj"

⇢ Transformer en majuscules : mot.toUpperCase() ; ➡ "BONJOUR"

⇢ Transforme en minuscule : .toLowerCase()

**■ Les booléens**

⇢ Les opérateurs logiques : **et**⇔&& **ou** ⇔ || **non** ⇔ !

⇢ Pour comparer :
**>** ⇔ supérieur **>=** ⇔ supérieur ou égal
**<** ⇔ inférieur **<=** ⇔ inférieur ou égal
*triple égale* === ⇔ égal à (les deux objets sont identiques : type et valeur)
(*ne pas confondre avec l'affectation* : =)
*double égal* == ⇔ égal à (même si les deux objets n'ont pas le même type)
Par exemple : "5" == 5 est vraie)

⇢ Les valeurs undefined et null
Elles s'emploient toutes les deux pour dire « rien ».

```
>> var maVariable;
<- undefined
>> |
```

On définit une variable qui n'a pas de valeur initiale.

```
>> var maVariableNulle= null;
<- undefined
>> |
```

On créer une variable et on précise clairement qu'elle est vide (on n'a pas de valeurs « favorite » à saisir -> c'est le cas de la création d'un menu déroulant sans présélectionner de valeur)



**■ Les tableaux**

⇢ Liste vide ➡[ ]

⇢ Insérer des éléments
-> la concaténation…
-> les méthodes push : on ajoute un élément à la fin de la liste
et pop : on retire le dernier élément de la liste

⇢ Trouver la position d'un élément dans un tableau : on utilise la méthode .indexOf(element)
➡sa première position s'il apparaît plusieurs fois | ➡ –1 si l'élément n'est pas dans la liste

⇢ Convertir un tableau en chaîne de caractères : on utilise la méthode .join

**Exemple 2 ✱**

```
var dinoEtNombres = [
        3,
        "T-Rex",
        ["Tricératops","Stégosaure", 3627.5],
        10
        ];
```

dinoEtNombres[0] ➡3
dinoEtNombres[2][1] ➡"Stégosaure"

**Exemple 3 ✱**

```
<script>
        var animaux=["chien", "chat"];
        console.log(animaux);
        animaux.push("canard");
        console.log(animaux);
        animaux.pop();
        console.log(animaux);

        console.log(animaux.join());
        console.log(animaux.join(" - "));
</script>
```

Tester le script
console.log(animaux.join()); ➡
console.log(animaux.join(" - ")); ➡

**■ Les objets**

Ces objets ressemblent beaucoup à des tableaux : il s'agit d'un ensemble de paires clé-valeur.

**Exemple 4 ✱**

⇢ Créer une nouvelle page Web et saisir le script ci-dessous.

⇢ Tester dans un navigateur. Qu'obtient-on en console ?

➡Object { pattes: 4, nom: "Pitre", couleur: "Noir", cri: "Miaule" }

⇢ Tester les instructions suivantes :

chat.nom➡ Pitre

Rôle :Accéder aux valeurs

Object.keys(chat) ➡[ "pattes", "nom", "couleur" ]

Rôle : Accéder à l'ensemble des clés

```
<script>
        var chat = {
            pattes: 4,
            nom: "Pitre",
            couleur: "Noir"
        };
        chat.cri="Miaule";
        console.log(chat);
</script>
```

## 1. Des généralités

### a. Afficher des résultats en console

⇢ Pour afficher la console sur Firefox : Menu / Développement web / Console web
On peut tester directement des actions dans la console, à partir de la zone de saisie.

⇢ L'instruction : console.log("éléments_a_afficher")

### b. Les types de données

**■ Les nombres**

⇢ Nombres et opérations : somme : + | soustration : – | produit : * | division : /

⇢ *Deux fonctions de la bibliothèque Math* :
Math.random() renvoi un réel supérieur à 0 et strictement inférieur à 1
Math.floor() renvoie la partie entière d'un réel

Exemple : pour obtenir un entier compris entre 1 et 6, il faut saisir :

➡ `Math.floor(Math.random()*6)+1`

**■ Les chaînes de caractères**

⇢ La concaténation de 2 chaînes de caractères : "bon " + "jour" ➡ "bonjour"

*On considère la variable* : mot = "bonjour"

⇢ Longueur d'une chaîne : mot.length ; ➡ 7

⇢ Le rang d'un caractère d'une chaîne : mot[2] ; ➡"j"

⇢ Découper une chaîne : mot.slice(1,5) ; ➡ "onj"

⇢ Transformer en majuscules : mot.toUpperCase() ; ➡ "BONJOUR"

⇢ Transforme en minuscule : .toLowerCase()

**■ Les booléens**

⇢ Les opérateurs logiques : **et**⇔&& **ou** ⇔ || **non** ⇔ !

⇢ Pour comparer :
**>** ⇔ supérieur **>=** ⇔ supérieur ou égal
**<** ⇔ inférieur **<=** ⇔ inférieur ou égal
*triple égale* === ⇔ égal à (les deux objets sont identiques : type et valeur)
(*ne pas confondre avec l'affectation* : =)
*double égal* == ⇔ égal à (même si les deux objets n'ont pas le même type)
Par exemple : "5" == 5 est vraie)

⇢ Les valeurs undefined et null
Elles s'emploient toutes les deux pour dire « rien ».

```
>> var maVariable;
<- undefined
>> |
```

On définit une variable qui n'a pas de valeur initiale.

```
>> var maVariableNulle= null;
<- undefined
>> |
```

On créer une variable et on précise clairement qu'elle est vide (on n'a pas de valeurs « favorite » à saisir -> c'est le cas de la création d'un menu déroulant sans présélectionner de valeur)



**■ Les tableaux**

⇢ Liste vide ➡ [ ]

⇢ Insérer des éléments
-> la concaténation…
-> les méthodes push : on ajoute un élément à la fin de la liste
et pop : on retire le dernier élément de la liste

⇢ Trouver la position d'un élément dans un tableau : on utilise la méthode .indexOf(element)
➡sa première position s'il apparaît plusieurs fois | ➡ –1 si l'élément n'est pas dans la liste

⇢ Convertir un tableau en chaîne de caractères : on utilise la méthode .join

**Exemple 2 ✱**

```
var dinoEtNombres = [
        3,
        "T-Rex",
        ["Tricératops","Stégosaure", 3627.5],
        10
        ];
```

dinoEtNombres[0] ➡3
dinoEtNombres[2][1] ➡"Stégosaure"

**Exemple 3 ✱**

```
<script>
        var animaux=["chien", "chat"];
        console.log(animaux);
        animaux.push("canard");
        console.log(animaux);
        animaux.pop();
        console.log(animaux);

        console.log(animaux.join());
        console.log(animaux.join(" - "));
</script>
```

Tester le script
console.log(animaux.join()); ➡chien,chat
console.log(animaux.join(" - ")); ➡chien – chat

**■ Les objets**

Ces objets ressemblent beaucoup à des tableaux : il s'agit d'un ensemble de paires clé-valeur.

**Exemple 4 ✱**

⇢ Créer une nouvelle page Web et saisir le script ci-dessous.

⇢ Tester dans un navigateur. Qu'obtient-on en console ?

➡ Object { pattes: 4, nom: "Pitre", couleur: "Noir", cri: "Miaule" }

⇢ Tester les instructions suivantes :

chat.nom➡ Pitre

Rôle :Accéder aux valeurs

Object.keys(chat) ➡ [ "pattes", "nom", "couleur" ]

Rôle : Accéder à l'ensemble des clés

```
<script>
        var chat = {
            pattes: 4,
            nom: "Pitre",
            couleur: "Noir"
        };
        chat.cri="Miaule";
        console.log(chat);
</script>
```

**Remarque**

**b. Les types de données**

■ **Les nombres**

⇢ Nombres et opérations : somme : + | soustration : – | produit : * | division : /

L'objet Number est principalement utilisé dans les cas de figure suivants :
Si l'argument ne peut pas être converti en un nombre, il renverra NaN.
Dans un contexte de fonction simple (quand il n'est pas utilisé comme un constructeur avec l'opérateur new), Number peut être utilisé afin d'effectuer des conversions.

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Number

■ **Les tableaux**

Pour trier : méthode -> `.sort()`

Attention : s'assurer d'avoir des nombres (sinon, classement alphabétique…)

## 2. Les conditions et les boucles

### ■ L'instruction conditionnelle : SI… SINON

**Structure**

```
if (condition) {
        instruction;
        instruction;
        …
} else {
        Instruction;
        …
}
```

**Exemple 5**

```
<script>
    var nom= "Nicolas";
    console.log("Bonjour " +nom);
    if(nom.length>7) {
        console.log("Ton prénom est long");
    } else {
        console.log("Ton prénom n'est pas très long");
    }
</script>
```

Modifier le script ci-dessus de sorte qu'on demande à l'utilisateur son prénom… ✱

### ■ Boucle Tant que

**Structure**

```
while (condition) {
        instruction;
        instruction;
        …
}
```

**Exemple 6**

```
<script>
    varmoutonsComptes= 0;
    while(moutonsComptes<=10) {
        console.log("J'ai compté " + moutonsComptes + " moutons !");
        moutonsComptes++;
    }
    console.log("Zzzzzzzzz") ;
</script>
```

### ■ Boucle POUR

**Structure**

```
for (introduction ; condition ; incrémentation) {
        instruction;
        instruction;
        …
}
```

**Exemple 7**

```
<script>
    var fin= 3;
    for (var i= 0 ;i<= fin ; i=i+1) {
        console.log("Ligne n° "+ i);
    }
</script>
```

## 3. Les fonctions

Sommaire

**Exemple 8**

HTML

```
<script>
    var premiereFonction= function() {
        console.log("Bonjour tout le monde");
    }
    premiereFonction();
</script>
```

Console

```
Filtrer
Bonjour tout le monde
»
```

HTML : 2ème méthode

```
<script>
    function premiereFonctionBis() {
        console.log("Bonjour tout le monde");
    }
    premiereFonctionBis();
</script>
```

**Exemple 9** ->*avec un argument*

HTML

```
<script>
    var fct2= function(nom) {
        console.log("Bonjour " + nom + " !");
    }
    var quelNom= prompt("Quel est ton nom ?");
    fct2(quelNom);
</script>
```

Console

```
Filtrer
Bonjour Nicolas !
»
```

**Exemple 10**->*return*

HTML

```
<script>
    var fct3= function(nom, rang) {
        if (rang > nom.length) {
            return "Impossible";
        } else {
            return nom[rang];
        }
    }
    var quelNom= prompt("Quel est ton nom ?");
    console.log(fct3(quelNom, 5));
</script>
```

Console

```
a
»
```

**Exemple 5**

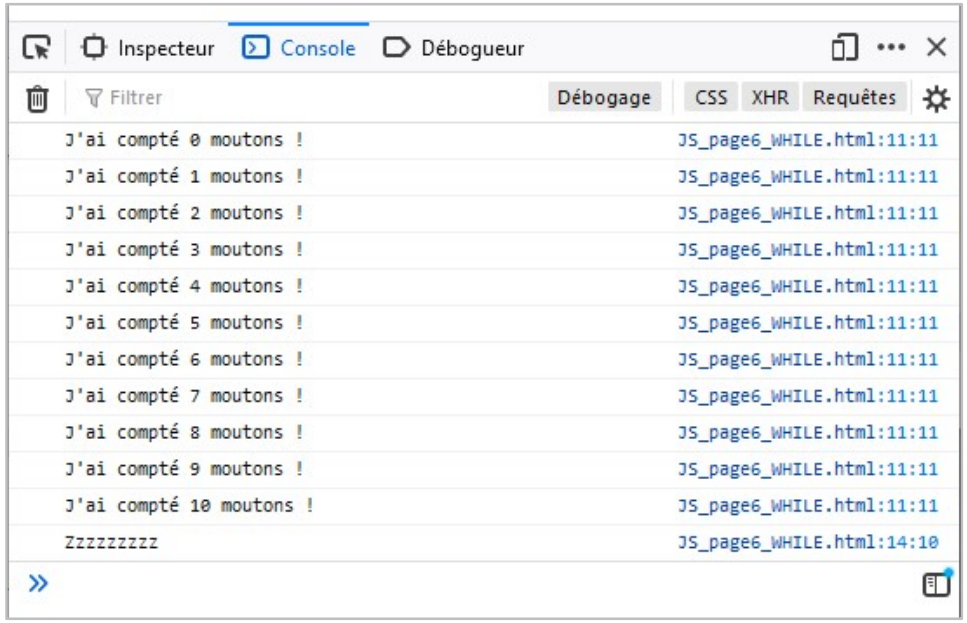
Version initiale

```
<script>
	var nom= "Nicolas";
	console.log("Bonjour " +nom);
	if(nom.length>7) {
		console.log("Ton prénom est long");
	} else {
		console.log("Ton prénom n'est pas très long");
	}
</script>
```

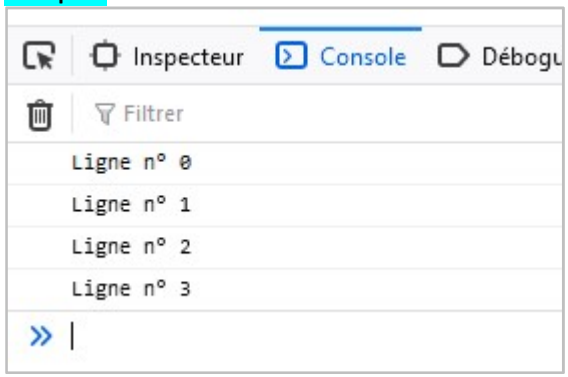
```
<!DOCTYPE html>
<html>
<head>
	<title></title>
	<metacharset="UTF-8">
</head>
<body>
<script>
	//var nom = "Nicolas";
	varnom = prompt("Notez votre prénom", "mon prénom");
	console.log("Bonjour " +nom);
	if (nom.lenght> 7) {
		console.log("Ton prénom est long")
	} else {
		console.log("Ton prénom n'est pas très long")
	}
</script>
</body>
</html>
```

**Exemple 6**

```
<script>
	varmoutonsComptes= 0;
	while(moutonsComptes<=10) {
		console.log("J'ai compté " + moutonsComptes + " moutons !");
		moutonsComptes++;
	}
	console.log("Zzzzzzzzz") ;
</script>
```




**Exemple 7**




```
<script>
	var fin= 3;
	for (var i= 0 ;i<= fin ; i=i+1) {
		console.log("Ligne n° "+ i);
	}
</script>
```

## 4. Gestion des événements

Sommaire

### a. Le principe

■ Un événement est par exemple le chargement d'une page, le passage du pointeur sur une page ou un élément de la page, un clic avec la souris sur la page où un élément de la page.

Un événement permet de déclencher l'exécution d'une fonction. Pour cela un événement est ajouté à un élément de la page, une balise par exemple, et lorsque l'utilisateur agit sur l'élément d'une certaine manière, il déclenche le code Javascript.

Ci-dessous quelques événements :

| | |
|---|---|
| Load | ➡ Chargement de la page Web |
| Click | ➡ Clic avec la souris, sélection d'un élément |
| Dblclick | ➡ Double-clic sur l'élément |
| Mouseover | ➡ Entrée du pointeur sur un élément |
| Mouseout | ➡ Sortie du pointeur de l'élément |
| Keypress | ➡ Appui sur une touche produisant un caractère |
| Submit | ➡ Envoi d'un formulaire |
| Reset | ➡ Réinitialisation d'un formulaire |
| Change | ➡ Modification de la valeur d'un élément d'un formulaire |

On utilisera les attributs onClick, onMouseover… -> voir **l'exemple 11**

■ Gestion d'un événement

*Résumons l'interaction entre l'utilisateur et la machine* :

➡ L'utilisateur déclenche un événement qui réagit en appelant un code Javascript

➡ Le code Javascript s'exécute et, par exemple, provoque une modification dans la page

➡ Même si la page provient d'une machine distante, le fichier HTML interprété dans le navigateur a été enregistré sur la machine de l'utilisateur, le client,

et le code Javascript est exécuté sur cette machine.

*On peut en déduire immédiatement les éventuels problèmes de sécurité que cela soulève* !

### b. Le DOM : Document Object Model (modèle d'objet du document)

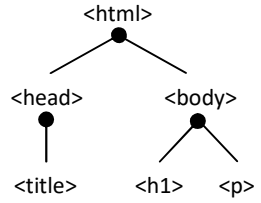
Le DOM est une interface de programmation pour les documents HTML, XML et SVG.

➡ Suivant ce modèle, le document, c'est-à-dire la page HTML, est vu comme un objet.

➡ Lorsqu'un navigateur charge une page, il crée un modèle sous la forme d'un arbre d'objets.

➡ Chaque élément de la page, appelé nœud, est un objet possédant des propriétés et des méthodes.

➡ L'objet Document contient l'objet racine <html>, qui contient deux objets <head> et <body>… etc.




➡ Javascript a un accès total sur tous les objets.

➡ Les nœuds peuvent être associés à des gestionnaires d'événements. Une fois qu'un événement est déclenché, les gestionnaires d'événements sont exécutés.

**Exemple 11**

*Le chargement d'une page est un événement qui s'appelle* :Load

*À la balise*<body>*on ajoute l'attribut*onLoad*avec comme valeur le nom d'une fonction Javascript*

HTML ->**JS_page11.html**

```
1  <!DOCTYPE html>
2  <html lang="fr">
3  <head>
4      <title>onLoad</title>
5      <meta charset="UTF-8">
6      <script>
7          function rebours(x) {
8              while (x >= 0){
9                  console.log(x);
10                 x--;
11             }
12         }
13         n = prompt("Saisir un nombre entier", "Nombre");
14     </script>
15 </head>
16 <body onLoad="rebours(n)">
17     <p>Page avec Javascript</p>
18 </body>
19 </html>
```

▪ Saisir le script et tester dans un navigateur (ne pas oublier d'afficher la console).

▪ Noter l'ordre d'exécution des instructions et des affichages (en justifiant)

**1.**le navigateur lit le code HTML en commençant par l'entête

Ligne 13 -> « prompt » : il attend la saisie de l'utilisateur

**2.** puis parcours le »body » : ligne 16

L'événement est onLoad

-> donc avant d'exécuter la fonction « rebours », le navigateur charge l'ensemble de la page

**3.**Ligne 17 : il affiche le paragraphe « Page avec Javascript »

**4.** Une fois chargée, il exécute la fonction `rebours` -> d'où l'affichage en console

**Exemple 12** Identifier des éléments par id et modifier le style d'un élément HTML✱

On considère les trois scripts ci-dessous.

HTML ->**JS_page12.html**

```
1  <!doctype html>
2  <html lang="fr">
3      <head>
4          <meta charset="utf-8">
5          <title>Modifier le style d'un élément</title>
6          <link rel="stylesheet" href="style.css">
7          <script language="JavaScript" src="JS_page12.js"></script>
8      </head>
9      <body>
10         <p id="para" onclick="change_couleur()"> cliquez ici </p>
11         <p> Paragraphe 2 </p>
12     </body>
13 </html>
```

Javascript ->**JS_page12.js**

```
1  function change_couleur(){
2      para.style.backgroundColor = "yellow";
3      para.style.color = "red";
4  }
```

CSS ->**style.css**

```
#para
{
        color: blue;
}
```

■ Questions préliminaires

▪ Quel est le rôle des lignes 6 et 7 ?

On crée le lien avec la feuille de style et le fichier Javascript

▪ Ligne 10 : le paragraphe possède l'attribut id : pourquoi ?

Il permet d'identifier de manière unique le paragraphe

Son nom est « para » -> on pourra effectuer des transformations sur ce paragraphe

▪ Quel est la couleur du premier paragraphe ? bleu

■ Saisir les trois scripts et tester.

■ Analyse

▪ Quel événement déclenche la fonction change_couleur ?

L'utilisateur déclenche la fct lorsqu'il clique sur le paragraphe

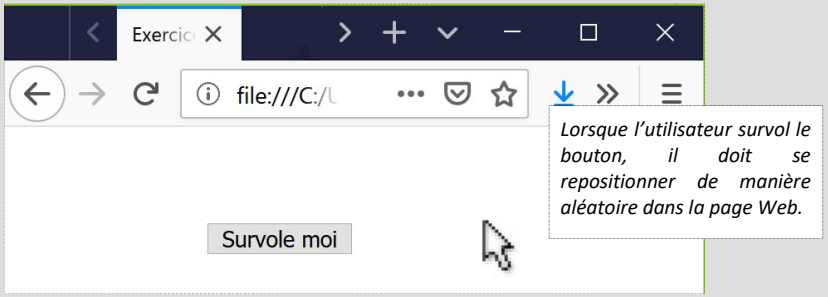
▪ Quel est son effet ?

Celle-ci permet de modifier le style du paragraphe (couleur et couleur de fond).

■ Modifier les scripts afin que la couleur de fond du deuxième paragraphe devienne vert clair lorsque la souris survole le premier paragraphe ✱

**Exemple 13** Sommaire

Objectif : Déplacer un bouton en cas de survol de celui ci




➔Une partie des scripts est donnée -> les compléter ✱

➔ On pourra utiliser les instructions suivantes :

`Math.random()` qui renvoie un nombre flottant compris dans l'intervalle [0 ; 1[

`Window.innerWidht` qui donne la largeur de la fenêtre

`Window.innerHeight` qui donne la hauteur de la fenêtre

Le mot-clef `this` s'emploie à l'intérieur d'une méthode pour désigner l'objet auquel cette méthode est en train d'être appliquée.

HTML ->**JS_page13.html**

```
<!doctype html>
<html lang="fr">
<head>
    <meta charset="utf-8">
    <title>Exemple 13</title>
    <link rel="stylesheet" href="style.css">
    <script language="JavaScript" src="JS_page13.js"></script>
</head>
    <body>
        <input type="Button" value="Survole moi" onmouseover ="bouge(this);" />
    </body>
</html>
```

Javascript ->**JS_page13.js**

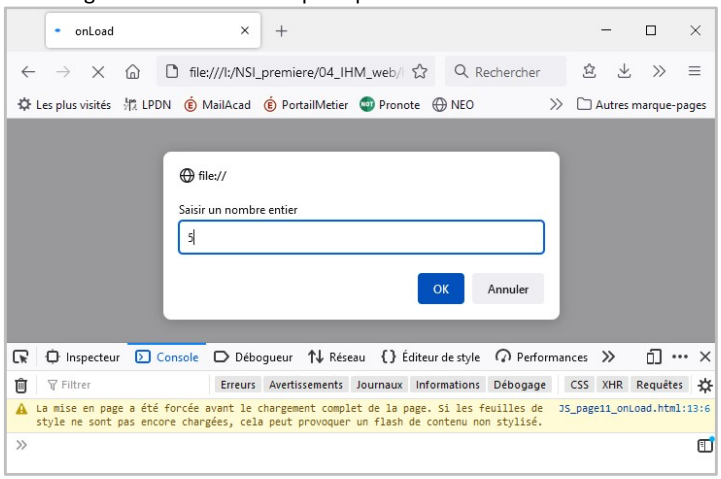
```
function bouge(element){
    x= Math.random()*window.innerWidth;
    y= Math.random()*window.innerHeight;
    element.style.position="absolute";
    element.style.left=x+"px";
    element.style.top=y+"px";
}
```

## Exemple 12

Sommaire

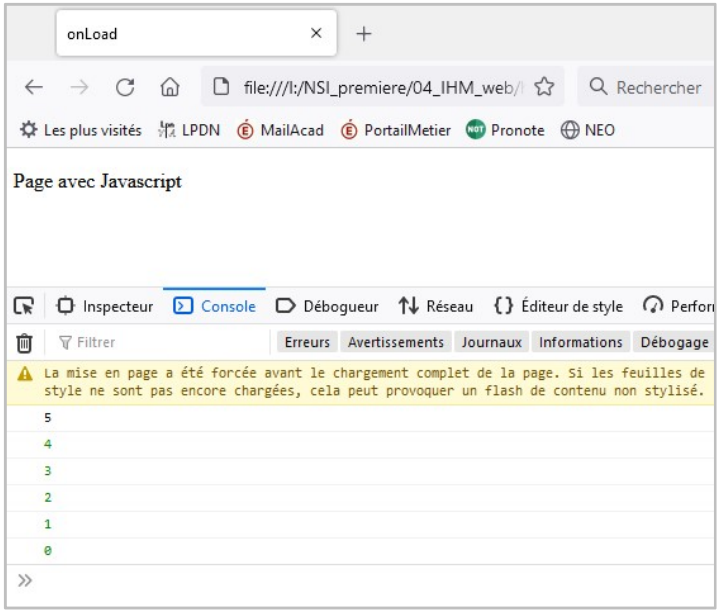
Dans l'ordre d'exécution :

**1.**le navigateur lit l'entête -> « prompt » : interaction avec l'utilisateur




```
<!DOCTYPE html>
<html lang="fr">
<head>
    <title>onLoad</title>
    <meta charset="UTF-8">
    <script>
        function rebours(x) {
            while (x >= 0){
                console.log(x);
                x--;
            }
        }
        n = prompt("Saisir un nombre entier", "Nombre");
    </script>
</head>
<body onLoad="rebours(n)">
    <p>Page avec Javascript</p>
</body>
</html>
```

**2.** puis parcours le »body »
L'événement est onLoad -> donc avec d'exécuter la fonction « rebours », le navigateur charge l'ensemble de la page : il affiche « Page avec Javascript »




**3.** Une fois chargée, il exécute la fonction -> d'où l'affichage en console

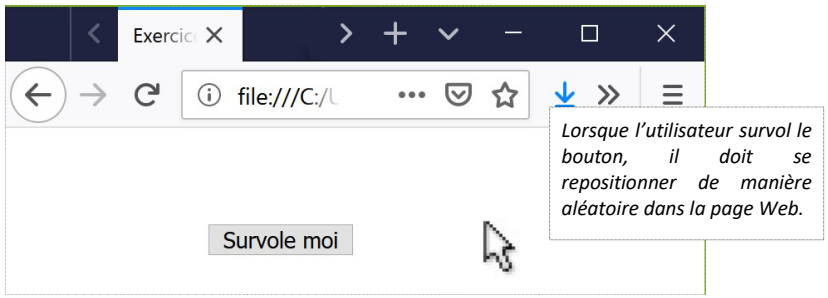
```
<!doctype html>
<htmllang="fr">
      <head>
            <metacharset="utf-8">
            <title>Modifier le style d'un élément</title>
            <link rel="stylesheet" href="style.css">
            <scriptlanguage="JavaScript"
src="JS_page12.js"></script>
      </head>
      <body>
            <p id="para" onmouseover="change_couleur2()"
            onclick="change_couleur()"> cliquez ici </p>
            <p id="para2"> Paragraphe 2 </p>
      </bodv>
```

```
functionchange_couleur(){
      para.style.backgroundColor = "yellow";
      para.style.color = "red";
}

function change_couleur2(){
      para2.style.backgroundColor = "#dcff64";
}
```

## Exemple 13

Objectif : Déplacer un bouton en cas de survol de celui ci




➔Une partie des scripts est donnée -> les compléter ✱

➔ On pourra utiliser les instructions suivantes :

`Math.random()` qui renvoie un nombre flottant compris dans l'intervalle [0 ; 1[
`Window.innerWidht` qui donne la largeur de la fenêtre
`Window.innerHeight` qui donne la hauteur de la fenêtre

Le mot-clef `this` s'emploie à l'intérieur d'une méthode pour désigner l'objet auquel cette méthode est en train d'être appliquée.

Remarque :
Il est intéressant de visualiser les valeurs des 3 propriétés de l'objet `this` en console :
Inspecteur + sélection de l'élément `input`
-> position
->left en px
-> top en px

HTML ->**JS_page13.html**

```html
<!doctype html>
<html lang="fr">
<head>
    <meta charset="utf-8">
    <title>Exemple 13</title>
    <link rel="stylesheet" href="style.css">
    <script language="JavaScript" src="JS_page13.js"></script>
</head>
    <body>
        <input type="Button" value="Survole moi" onmouseover ="bouge(this);" />
    </body>
</html>
```

Javascript ->**JS_page13.js**

```javascript
function bouge(element){
    x= Math.random()*window.innerWidth;
    y= Math.random()*window.innerHeight;
    element.style.position="absolute";
    element.style.left=x+"px";
    element.style.top=y+"px";
}
```

### c. Fonction `getElementById` / méthode `innerHTML`

- Sélectionner un élément par `getElementById`
- Extraire ou modifier du texte avec la propriété `innerHTML`

Sommaire


**Exemple 14**

Objectif : Lorsque l'utilisateur clique sur un bouton, on lui demande un texte qui va remplacer le titre initial

HTML ->**JS_page14.html**

```
1  <!DOCTYPE html>
2  <html lang="fr">
3      <head>
4          <title>Modifier du texte avec le DOM</title>
5          <meta charset="utf-8" />
6          <script language="JavaScript" src="JS_page14.js"></script>
7      </head>
8      <body>
9          <h1 id="titre-principal">Bonjour le monde</h1>
10         <p>Avec le titre initial</p>
11         <input type="Button" id="bouton" value="Changer le titre ?"
12                onclick="change_titre()" />
13     </body>
14 </html>
```

Javascript ->**JS_page14.js**

```
1  function change_titre(){
2      let elementTitre = document.getElementById("titre-principal");
3      console.log(elementTitre);
4      console.log(elementTitre.innerHTML);
5      let texteNouveauTitre = prompt("Veuillez entrer un nouveau titre :");
6      elementTitre.innerHTML = texteNouveauTitre;
7  }
```

**Question 1.** Saisir et tester (ne pas oublier d'afficher la console).

Quelques explications

*Fichier HTML*

Ligne 9 : L'attribut id permet d'identifier de manière unique la balise <h1>

*Fichier JS*

Ligne 2 :

- En appelant la méthode `getElementById`, on demande au navigateur de rechercher dans la page Web l'élément HTML dont l'attribut id a pour valeur "titre-principal"
- L'appel de cette fonction renvoie cet objet du DOM.
- On stocke l'objet dans la variable `elementTitre`.

Ligne 3 : on affiche en console le contenu HTML de `elementTitre`

Ligne 4 : on **extrait** la valeur de la propriété innerHTML de l'élément qui se trouve dans elementTitre

c'est-à-dire :

Ligne 6 : on **modifie** la valeur de la propriété innerHTML de l'élément qui se trouve dans elementTitre : on lui donne pour valeur le texte stocké auparavant dans texteNouveauTitre

*Remarque*

Ligne 2 :

**Question 2** : Compléter la fonction `change_titre` avec les deux lignes suivantes :

```
let elementBonton = document.getElementById("bouton");
elementBonton.value = "Changer à nouveau ?";
```

Quel effet obtient-on ?

La variable elementBouton reçoit les propriétés HTML du bouton

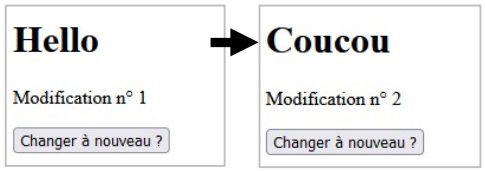
Puis on modifie sa propriété value -> c'est-à-dire le texte inscrit sur le bouton

**Question 3** : Compléter la fonction `change_titre` afin que, lorsque l'utilisateur clique sur le bouton le texte du paragraphe devienne : "Modification n° 1"(après le 1er clic)
"Modification n° 2"(après le 2ème clic)…

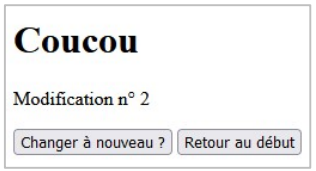
**Question 4** : Créer un nouveau bouton "Retour au début" qui permet, lorsque l'utilisateur clique dessus, de remettre la page à l'état initial.

**Question 5** : Modifier le code Javascript de sorte que si l'utilisateur ne saisit rien dans la fenêtre, la page web revienne à l'état initial.

**Question 3**




**Question 4**



inner = intérieur

## Exemple 14

**Question 3**

Convertir un entier en chaîne de caractère !!!

Voir par exemple :

https://waytolearnx.com/2019/09/convertir-int-en-string-en-javascript.html

```
var compteur = 0;

functionchange_titre(){
	letelementTitre = document.getElementById("titre-principal");
	console.log(elementTitre);
	console.log(elementTitre.innerHTML);
	// question 2
	letelementBonton = document.getElementById("bouton");
	elementBonton.value = "Changer à nouveau ?";

	// question 3
	if (texteNouveauTitre===""){
		reinitialise();
	} else {
		elementTitre.innerHTML = texteNouveauTitre;
		letelementParagraphe = document.getElementById("para");
		compteur = compteur + 1;
		elementParagraphe.innerHTML = "Modification n°
"+compteur;
	}
}

// question 4
functionreinitialise(){
	letelementTitre = document.getElementById("titre-principal");
	letelementBonton = document.getElementById("bouton");
	letelementParagraphe = document.getElementById("para");
	elementTitre.innerHTML = "Bonjour le monde";
	elementBonton.value = "Changer le titre ?";
	elementParagraphe.innerHTML = "Avec le titre initial";
	compteur = 0
}
```

```
<!DOCTYPE html>
<htmllang="fr">
	<head>
		<title>Modifier du texte avec le DOM</title>
		<metacharset="utf-8" />
		<scriptlanguage="JavaScript"
src="JS_page14_v3.js"></script>
	</head>
	<body>
		<h1 id="titre-principal">Bonjour le monde</h1>
		<p id="para">Avec le titre initial</p>
		<input type="Button" id="bouton" value="Changer le
titre ?"
		onclick="change_titre()" />
		<input type="Button" id="bouton2" value="Retour au
début"
		onclick="reinitialise()" />
```

### d. L'écouteur d'événement `addEventListener`

Les éléments sont à l'écoute d'événements avec l'attribut `onClick`à l'intérieur des balises et des fonctions sont définies pour gérer ces événements.

Avec l'objectif d'avoir un balisage plus condensé et de mieux séparer les aspects contenu et comportement d'une page, le DOM propose une méthode plus intéressante : les éléments ont un identifiant, on définit des variables et on utilise l'écouteur d'événements `addEventListener`.

**Exemple 15**

Objectif : Afficher un texte à l'aide d'une fonction suivant l'événement `onClick` au bout de 2 secondes

Quelques explications -> code Javascript

Ligne 1 : Permet d'attendre le chargement du document pour initialiser les écouteurs aux objets HTML. Une fois chargée, la fonction init est appelée.

Ligne 3 : `event` reçoit l'objet HTML ayant généré l'événement

Ligne 4 : on définit la variable `objHTML`

Ligne 5 : La méthode `addEventListener()` met en place la fonction `affiche` à chaque fois que l'événement click est utilisé.

Ligne 12 : l'argument `event` est à nouveau obligatoire

Ligne 13 : La méthode *`setTimeout`*est une méthode de l'objet `Window`

-> elle indique un délai avant exécution d'une fonction en millisecondes.

HTML ->**JS_page15.js**

```html
<!doctype html>
<html lang="fr">
<head>
    <meta charset="utf-8">
    <title>Méthode addEventListner</title>
    <link rel="stylesheet" href="style.css">
    <script src="JS_page15.js"></script>
</head>
    <body>
        <p id="para1">clique moi</p>
    </body>
</html>
```

Javascript ->**JS_page15.js**

```javascript
document.addEventListener("DOMContentLoaded", init);

function init(event) {
    let objHTML= document.getElementById("para1");
    objHTML.addEventListener("click",affiche);
}

function affiche_pause(){
    alert("coucou");
}

function affiche(event){
    setTimeout(affiche_pause,2000);
}
```

**Exercice 16**

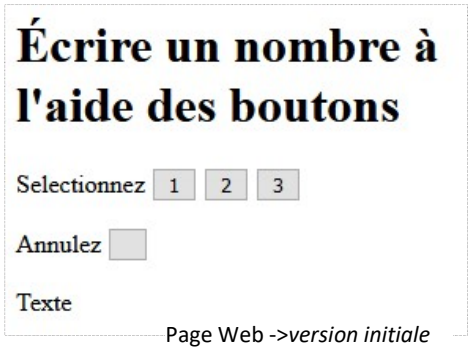
Créer un page Web contenant :

- 4 boutons : trois boutons numérotés 1, 2 et 3.
  le 4ème est initialement "vide".
- un paragraphe qui contient initialement le texte "Texte"

Lorsque l'internaute clique sur un des trois premiers boutons, le chiffre correspondant s'affiche dans le 4ème bouton et à la place de "Texte"

Si l'utilisateur clique sur le 4ème bouton (Annuler), tout doit reprendre son état initial.

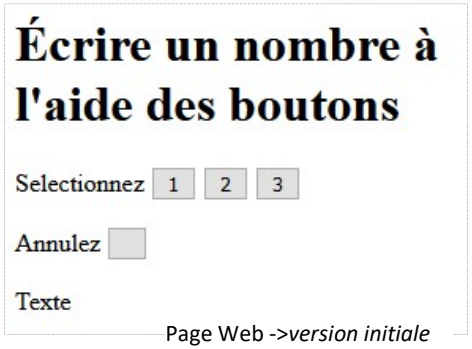
Enregistrer les fichiers sous les noms JS_page15.html et JS_page15.js




Page Web ->*version initiale*

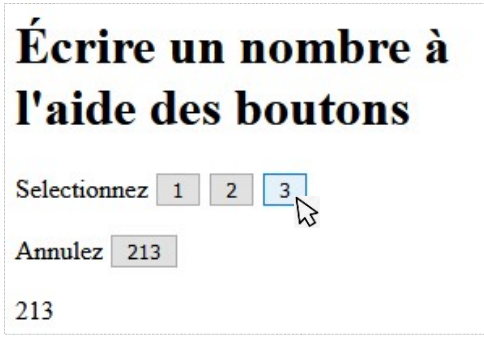
## Correction exercice 16

```
<html>
	<head>
		<metacharset="utf-8"/>
		<title>Titre</title>
		<scriptlanguage="JavaScript" src="exo16.js"></script>
	</head>
	<bodybgcolor="#ffffff" text= "#0080ff">
		<h1>Écrire un nombre à l'aide des boutons</h1>
		<p>Selectionnez<input type="button" value="1" onClick="nombre(1)">
			<input type="button" value="2" onClick="nombre(2)">
			<input type="button" value="3" onClick="nombre(3)">
		</p>
		<p>Annulez <input type="button" id="B4" value=" " onClick="reset()"></p>
		<p id="lab1">Texte</p>
	</body>
</html>
```




Page Web ->*version initiale*

```
var texte= "";                  //initialisation
function nombre(x){
	texte= texte+x;             //concaténation
	obj1 = document.getElementById("B4")
	obj1.value=texte;                  //on modifie la valeur d'un objet de type button
	obj2 = document.getElementById("lab1")
	obj2.innerHTML=texte;       //on modifie le contenu d'un balise HTML
}
function reset(){
	texte= "Texte";
	document.getElementById("B4").value= " ";//on re-passe la valeur du bouton B4 à " "
	document.getElementById("lab1").innerHTML= texte;//on re-initialise le texte du dernier
paragraphe
	texte= ""//Ré-initialisation
}
```



## 5. Formulaires dans une page Web

Sommaire

➔ Un formulaire est un des principaux moyens d'interaction entre un utilisateur et un site Web.

➔ Avec un formulaire HTML, les données sont envoyées aux serveurs qui doit ensuite les traiter.

➔ Un formulaire HTML est composé d'un ou plusieurs widgets.

Ceux-ci peuvent être des *zones de texte* (sur une seule ligne ou plusieurs lignes), des *boîtes à sélection*, des *boutons*, des *cases à cocher* ou des *boutons radio*. Etc.

### a. Créer un formulaire

■ Créer un formuaire : `<form> … </form>`

■ L'élément HTML `<input>`

Il est utilisé pour créer un contrôle interactif dans un formulaire web qui permet à l'utilisateur de saisir des données. La façon dont un élément <input> fonctionne dépend grandement de la valeur de son attribut type.

Des types de champs disponibles sont :

| | |
|---|---|
| text | Un champ texte sur une seule ligne.<br>Les sauts de ligne sont automatiquement retirés par défaut |
| checkbox | Une case à cocher qui permet de sélectionner/désélectionner une valeur |
| radio | Un bouton radio qui permet de sélectionner une seule valeur parmi un groupe de différentes valeurs |
| date | Un contrôle qui permet de saisir une date<br>composé d'un jour, d'un mois et d'une année -> exemple : "2018-07-22" |
| email | Un champ qui permet de saisir une adresse électronique |
| password | Un champ texte sur une seule ligne dont la valeur est masquée<br>Les attributs maxlength et minlength définissent la taille maximale et minimale de la valeur à saisir dans le champ |
| submit | Un bouton qui envoie le formulaire |
| button | Un bouton |
| reset | Un bouton qui réinitialise le contenu du formulaire avec les valeurs par défaut |

Autres attributs possibles

▪ autofocus : un attribut booléen qui passe le focus sur le champ lorsque le formulaire est affiché
▪ required : un attribut booléen qui indique que le champ doit être renseigné avant de pouvoir envoyer le formulaire
▪ name : le nom du champ qui sera rattaché à la donnée envoyée via le formulaire
▪ value : la valeur du champ (valeur initiale qui est modifiée par l'utilisateur)

**Exemple 16**

```
<p>Texte avant le formulaire</p>
<form>
    <p>Texte à l'intérieur du formulaire</p>
    <inputtype="text"/>
</form>
<p>Texte après le formulaire</p>
```

Texte avant le formulaire

Texte à l'intérieur du formulaire

Texte après le formulaire

On ajoute à la balise `<input>` d'autres arguments :

➔ `name` -> donner un nom à la zone de texte (indispensable pour la suite pour reconnaître d'où viennent les informations, ou pour créer un groupe…)

➔ `id` -> permet d'identifier de manière unique la balise (CSS, javascript, PHP…)

```
<inputtype="text"name="pseudo"id="psdo"/>
```

Mais aussi :

➔ `size` -> la taille du champ

➔ `maxlength` -> le nombre maximum de caractères

➔ `value` -> préremplir le champ avec une valeur par défaut

➔ `placeholder` -> donner une indication sur le contenu du champ

✱ Compléter afin que la zone de texte contienne une indication permettant de comprendre ce qu'il faut saisir ; que le champ fasse 30 caractères de long mais l'on ne puisse écrire que 10 caractères maximum à l'intérieur

■ L'élément HTML `<label>`

Cette zone de texte est bien jolie mais si votre visiteur tombe dessus, il ne sait pas ce qu'il doit écrire. C'est justement le rôle de la balise `<label>`

**Exemple 17**

```
<form>
    <label>Votre pseudo :</label>
    <inputtype="text"name="pseudo"id="psdo"/>
</form>
```

Votre pseudo :

➔ Pour lier le label au champ, il faut lui donner un attribut `for` qui a la même valeur que l'`id` du champ

```
<labelfor="psdo">Votre pseudo :</label>
```

✱ Compléter le formulaire afin de demander au visiteur son mot de passe et sa date de naissance.

## 5. Formulaires dans une page WebREMARQUES

Sommaire

### a. Créer un formulaire

■ Créer un formuaire : `<form> … </form>`

■ L'élément HTML `<input>`

https://developer.mozilla.org/fr/docs/Web/HTML/Element/input

Il est utilisé pour créer un contrôle interactif dans un formulaire web qui permet à l'utilisateur de saisir des données. La façon dont un élément <input> fonctionne dépend grandement de la valeur de son attribut type.

Des types de champs disponibles sont :

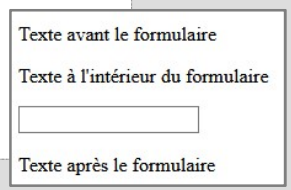
| | |
|---|---|
| text | Un champ texte sur une seule ligne.<br>Les sauts de ligne sont automatiquement retirés par défaut<br>Par défaut, lorsque l'attribut type n'est pas présent, il aura la valeur implicite text. |
| checkbox | Une case à cocher qui permet de sélectionner/désélectionner une valeur<br>https://developer.mozilla.org/fr/docs/Web/HTML/Element/Input/checkbox |
| radio | Un bouton radio qui permet de sélectionner une seule valeur parmi un groupe de différentes valeurs<br>https://developer.mozilla.org/fr/docs/Web/HTML/Element/Input/radio |
| date | Un contrôle qui permet de saisir une date<br>composé d'un jour, d'un mois et d'une année -> exemple : "2018-07-22"<br>https://developer.mozilla.org/fr/docs/Web/HTML/Element/input/date |
| email | Un champ qui permet de saisir une adresse électronique |
| password | Un champ texte sur une seule ligne dont la valeur est masquée<br>Les attributs maxlength et minlength définissent la taille maximale et minimale de la valeur à saisir dans le champ |
| submit | Un bouton qui envoie le formulaire |
| button | Un bouton |
| reset | Un bouton qui réinitialise le contenu du formulaire avec les valeurs par défaut |

Autres attributs possibles

- autofocus : un attribut booléen qui passe le focus sur le champ lorsque le formulaire est affiché
- required : un attribut booléen qui indique que le champ doit être renseigné avant de pouvoir envoyer le formulaire
- name : le nom du champ qui sera rattaché à la donnée envoyée via le formulaire
- value : la valeur du champ (valeur initiale qui est modifiée par l'utilisateur)

**Exemple 16**

```
<p>Texte avant le formulaire</p>
<form>
    <p>Texte à l'intérieur du formulaire</p>
    <inputtype="text"/>
</form>
<p>Texte après le formulaire</p>
```




On ajoute à la balise `<input>` d'autres arguments :

➔ `name` -> donner un nom à la zone de texte (indispensable pour la suite pour reconnaître d'où viennent les informations, ou pour créer un groupe…)

➔ `id` -> permet d'identifier de manière unique la balise (CSS, javascript, PHP…)

```
<inputtype="text"name="pseudo"id="psdo"/>
```

Mais aussi :

- ➔ `size` -> la taille du champ
- ➔ `maxlength` -> le nombre maximum de caractères
- ➔ `value` -> préremplir le champ avec une valeur par défaut
- ➔ `placeholder` -> donner une indication sur le contenu du champ

✱ Compléter afin que la zone de texte contienne une indication permettant de comprendre ce qu'il faut saisir ; que le champ fasse 30 caractères de long mais l'on ne puisse écrire que 10 caractères maximum à l'intérieur

■ L'élément HTML `<label>`

https://developer.mozilla.org/fr/docs/Web/HTML/Element/input

Cette zone de texte est bien jolie mais si votre visiteur tombe dessus, il ne sait pas ce qu'il doit écrire. C'est justement le rôle de la balise `<label>`

**Exemple 17**

```
<form>
    <label>Votre pseudo :</label>
    <inputtype="text"name="pseudo"id="psdo"/>
</form>
```


➔ Pour lier le label au champ, il faut lui donner un attribut `for` qui a la même valeur que l'`id` du champ-> garantir l'unicité

```
<labelfor="psdo">Votre pseudo :</label>
```

✱ Compléter le formulaire afin de demander au visiteur son mot de passe et sa date de naissance.

## Exemple 16

✱ Compléter afin que la zone de texte contienne :
- une indication permettant de comprendre ce qu'il faut saisir ;
- que le champ fasse 30 caractères de long
- mais l'on ne puisse écrire que 10 caractères maximum à l'intérieur

```
<!doctype html>
<htmllang="fr">
<head>
    <metacharset="utf-8">
    <title>Formulaires</title>
</head>
    <body>
        <p>Texte avant le formulaire</p>
        <form>
            <p>Texte à l'intérieur du formulaire</p>
            <input type="text" name="pseudo" id="psdo"
            placeholder="ex : toto" size="30" maxlength="10" />
        </form>
        <p>Texte après le formulaire</p>
    </body>
</html>
```

## Exemple 17

Sommaire

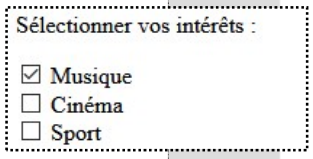
Rattacher un libellé à un élément de saisie (<input>) offre différents avantages :
Le texte du libellé n'est pas seulement associé visuellement au champ, il est techniquement associé avec le champ. Ainsi, lorsque l'utilisateur a le focus sur le champ, un lecteur d'écran pourra énoncer le contenu du libellé et permettre à l'utilisateur de disposer d'un meilleur contexte.
Vous pouvez cliquer sur le libellé pour passer le focus voire activer le champ. De cette façon, on dispose d'une meilleure ergonomie car la surface d'utilisation du champ est agrandie, ce qui s'avère utile sur les petits appareils comme les téléphones portables.
https://developer.mozilla.org/fr/docs/Web/HTML/Element/label

✱ Compléter le formulaire afin de demander au visiteur son mot de passe et sa date de naissance

```
<!doctype html>
<htmllang="fr">
<head>
    <metacharset="utf-8">
    <title>Formulaires</title>
</head>
    <body>
        <form>
        <div>
            <label for="psdo">Votre pseudo :</label>
            <input type="text" name="pseudo" id="psdo" />
        </div>
        <div>
            <label for="mdp">Votre mot de passe :</label>
            <input type="password" name="mdp" id="mdp" />
        </div>
        <div>
            <label for="dateNaissance">Dater de naissance :</a>
            <input type="date" id="dateNaissance" />
        </div>
        <div>
            <input type="color" />
        </div>
        </form>
    </body>
```

**Exemple 18** ->checkbox

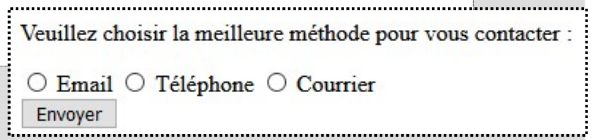
```
<form>
    <p>Sélectionner vos intérêts :</p>
    <div>
        <inputtype="checkbox" id="musique" name="musique" checked>
        <labelfor="musique">Musique</label>
    </div>
    <div>
        <inputtype="checkbox" id="cinema" name="cinema">
        <labelfor="cinema">Cinéma</label>
    </div>
    <div>
        <inputtype="checkbox" id="sport" name="sport">
        <labelfor="sport">Sport</label>
    </div>
</form>
```




✱ Saisir et tester

**Exemple 19** ->bouton radio

```
<form>
    <p>Veuillez choisir la meilleure méthode pour vous contacter :</p>
    <div>
        <inputtype="radio" id="Choix1" name="contact" value="email">
        <labelfor="Choix1">Email</label>
        <inputtype="radio" id="Choix2" name="contact" value="telephone">
        <labelfor="Choix2">Téléphone</label>
        <inputtype="radio" id="Choix3" name="contact" value="courrier">
        <labelfor="Choix3">Courrier</label>
    </div>
    <div>
        <input type="button"value="Envoyer"/>
    </div>
</form>
```




✱ Saisir et tester

**Remarques**

➔ Pour définir un **groupe** de boutons radio, on leur donne le même nom via l'attribut `name`. Une fois qu'on a formé un groupe de boutons radio, on ne pourra sélectionner qu'une seule des options de ce groupes.

➔ L'attribut `value` est une chaîne de caractères qui contient la valeur du bouton radio. Cette valeur n'est pas montrée à l'utilisateur par le navigateur, elle permet d'identifier l'option sélectionnée.

HTML5 apporte de nombreuses fonctionnalités nouvelles relatives aux formulaires. De ***nouveaux types*** de champs sont en effet apparus avec cette version. Il suffit de donner à l'attribut `type` de la balise `<input />` l'une des nouvelles valeurs disponibles :

`email`, `url`, `tel`, `number`, `range` (curseur), `color`, `date`…

■ L'élément HTML `<textarea>`

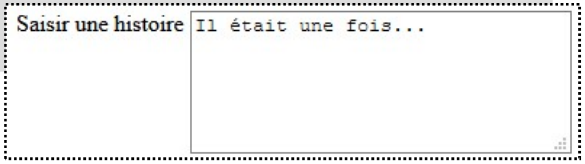
Sommaire

Il représente un contrôle qui permet d'éditer du texte sur plusieurs lignes

**Exemple 20** ->zone de texte sur plusieurs lignes

```
<form>
    <labelfor="histoire">Saisir une histoire</label>
    <textareaid="histoire" name="histoire" rows="5" cols="30"
        style="vertical-align: top;">Il était une fois...</textarea>
</form>
```

✱ Saisir et tester




■ L'élément HTML `<button>`

Il s'agit d'un élément HTML équivalent à celui utiliser dans l'exemple 19 permettant à l'utilisateur d'envoyer les données renseignées dans le formulaire.

Il faut saisir :

```
<div>
    <buttontype="submit">Envoyer</button>
</div>
```

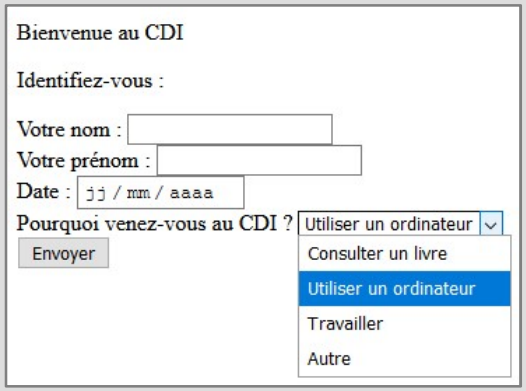
■ Les listes déroulantes `<select>`

On utilise la balise `<select></select>` qui indique le début et la fin de la liste déroulante.
On ajoute l'attribut `name` à la balise pour donner un nom à la liste.

Puis, à l'intérieur du `<select></select>`, on place plusieurs balises `<option></option>` (une par choix possible). On ajoute à chacune d'elles un attribut `value` pour pouvoir identifier ce que le visiteur a choisi.

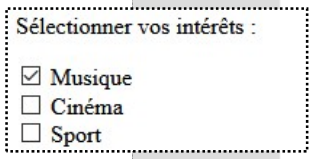
**Exemple 21**
-> listes déroulantes

✱ Créer le formulaire HTML ci-contre.



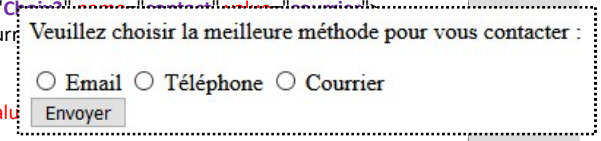
**Exemple 18** ->checkbox**REMARQUES**

```
<form>
    <p>Sélectionner vos intérêts :</p>
    <div>
        <inputtype="checkbox" id="musique" name="musique" checked>
        <labelfor="musique">Musique</label>
    </div>
    <div>
        <inputtype="checkbox" id="cinema" name="cinema">
        <labelfor="cinema">Cinéma</label>
    </div>
    <div>
        <inputtype="checkbox" id="sport" name="sport">
        <labelfor="sport">Sport</label>
    </div>
</form>
```




**Exemple 19** ->bouton radio

```
<form>
    <p>Veuillez choisir la meilleure méthode pour vous contacter :</p>
    <div>
        <inputtype="radio" id="Choix1" name="contact" value="email">
        <labelfor="Choix1">Email</label>
        <inputtype="radio" id="Choix2" name="contact" value="telephone">
        <labelfor="Choix2">Téléphone</label>
        <inputtype="radio" id="Choix3" name="contact" value="courrier">
        <labelfor="Choix3">Courrier</label>
    </div>
    <div>
        <input type="button"value="Envoyer">
    </div>
</form>
```




**Remarques**

➔ Pour définir un **groupe** de boutons radio, on leur donne le même nom via l'attribut `name`. Une fois qu'on a formé un groupe de boutons radio, on ne pourra sélectionner qu'une seule des options de ce groupes.

➔ L'attribut `value` est une chaîne de caractères qui contient la valeur du bouton radio. Cette valeur n'est pas montrée à l'utilisateur par le navigateur, elle permet d'identifier l'option sélectionnée.

Un autre name crée un autre groupe.

Par exemple, en ajoutant :

```
<inputtype="radio" id="Choix3" name="contactB" value="pigeon">
<labelfor="Choix3">Pigeon voyageur</label>
```

-> on ne peut choisir qu'un seul bt pour les 3 premières

-> et sélectionner le 4-ième (pas le même groupe)



Sommaire

HTML5 apporte de nombreuses fonctionnalités nouvelles relatives aux formulaires. De ***nouveaux types*** de champs sont en effet apparus avec cette version. Il suffit de donner à l'attribut `type` de la balise `<input />` l'une des nouvelles valeurs disponibles :

`email`, `url`, `tel`, `number`, `range` (curseur), `color`, `date`…

https://openclassrooms.com/fr/courses/1603881-apprenez-a-creer-votre-site-web-avec-html5-et-css3/1607171-creez-des-formulaires

■ L'élément HTML `<textarea>`

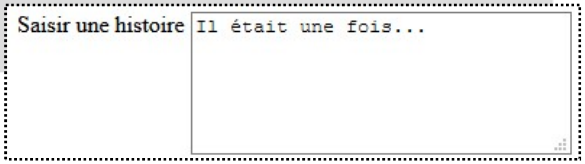
https://developer.mozilla.org/fr/docs/Web/HTML/Element/textarea

Il représente un contrôle qui permet d'éditer du texte sur plusieurs lignes

**Exemple 20** ->zone de texte sur plusieurs lignes

```
<form>
    <labelfor="histoire">Saisir une histoire</label>
    <textareaid="histoire" name="histoire" rows="5" cols="30"
        style="vertical-align: top;">Il était une fois...</textarea>
</form>
```

✱ Saisir et tester




■ L'élément HTML `<button>`

Il s'agit d'un élément HTML équivalent à celui utiliser dans l'exemple 19 permettant à l'utilisateur d'envoyer les données renseignées dans le formulaire.

Il faut saisir :

```
<div>
    <buttontype="submit">Envoyer</button>
</div>
```

■ Les listes déroulantes `<select>`

On utilise la balise `<select></select>` qui indique le début et la fin de la liste déroulante. On ajoute l'attribut `name` à la balise pour donner un nom à la liste.

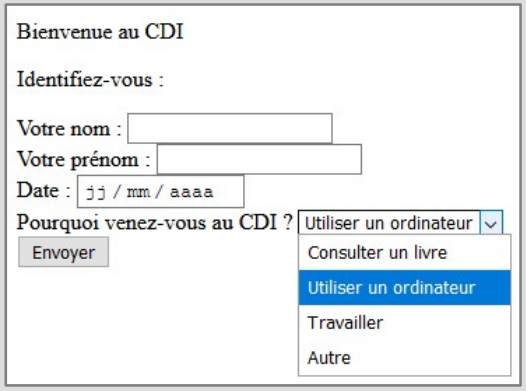
Puis, à l'intérieur du `<select></select>`, on place plusieurs balises `<option></option>` (une par choix possible). On ajoute à chacune d'elles un attribut `value` pour pouvoir identifier ce que le visiteur a choisi.

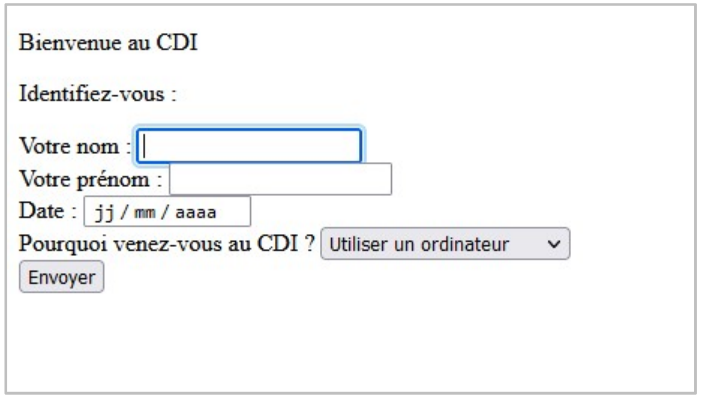
## Correction

**Exemple 21**

-> listes déroulantes

✱ Créer le formulaire HTML ci-contre.




```
<!doctype html>
<htmllang="fr">
<head>
    <metacharset="utf-8">
    <title>Entrée CDI</title>
</head>
    <body>
        <p>Bienvenue au CDI</p>
        <p>Identifiez-vous : </p>
        <form>
        <div>
            <label for="nom">Votre nom :</label>
            <input type="text" name="nom" id="nom" autofocus/>
        </div>
        <div>
            <label for="prenom">Votre prénom :</label>
            <input type="text" name="prenom" id="prenom" />
        </div>
        <div>
            <label for="date">Date :</a>
            <input type="date" id="date" />
        </div>
        <div>
            <label for="motif">Pourquoi venez-vous au CDI ?</label>
            <selectname="motif" id="motif">
              <option value="livre">Consulter un livre</option>
              <option       value="ordi"       selected>Utiliser       un
ordinateur</option>
              <option value="travail">Travailler</option>
              <option value="autre">Autre</option>
            </select>
        </div>
        <div>
            <input type="button" value="Envoyer" />
        </div>
        </form>
```