SCI6373 Programmation documentaire

Cours 6

Été 2025

Plan

- Retour sur OU et ET booléens
- L'objet `document` et les éléments HTML
- Entrées par formulaire
- Dynamisme par propriétés CSS
- Sorties par propriétés `innerText` et `innerHTML`

ET et OU booléens (1/2)

- En fait, ET && et OU || acceptent à gauche comme à droite n'importe quelle valeur JS
- Et peuvent aussi retourner comme résultat n'importe quelle valeur JS
- Voici comment…

ET et OU booléens (2/2)

- En fait le OU || s'évalue comme ceci :

  exp1 || exp2

  ↔ si (exp1 ≈ vrai), retourner exp1, sinon retourner exp2

- Et le ET && s'évalue comme ceci :

  exp1 && exp2

  ↔ si (exp1 ≈ faux), retourner exp1, sinon retourner exp2

`exp1` ≈ `true` veut dire que `exp1` est interprétable comme `true`

Objets de l'environnement navigateur Web

La page HTML

Objets correspondants aux éléments HTML

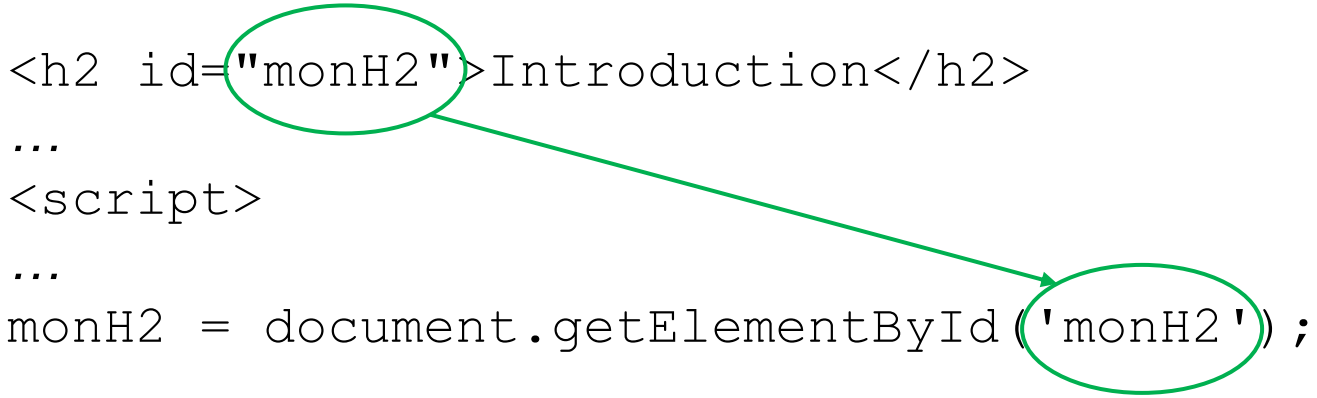
- Tous les éléments d'une page HTML ont un objet correspondant en JS
- L'accès à ces objets se fait via l'objet statutaire **`document`**, qui offre plusieurs fonctions à cette fin
- Par "accès" à un objet, on veut dire obtenir un *pointeur* vers cet objet

Accès via l'attribut HTML "id"

- La façon la **plus simple** est par la fonction **`document.getElementById()`**
- Pour l'utiliser, il faut "prévoir le coup" en mettant sur l'élément un attribut "id", dont on connaîtra donc la valeur.
- Ex.:

```
<h2 id="monH2">Introduction</h2>
```

Exemple



document.getElementById()

- Retourne comme résultat un pointeur à l'objet JS qui correspond à l'élément HTML identifié par le "id" passé en argument
- Les membres présents dans cet objet dépendent du type d'élément HTML
  - Exemple 210

Exemple : un <h2>

- Par exemple, pour un élément "h2", l'objet a les deux propriétés (de type caractère)
  - `innerText` et `innerHTML`
- Elles peuvent être manipulées par JS
  - Peuvent même être *assignées*
  - Note : assignation avec LHS qui n'est *pas un nom de variable*
  - Exemples 210 et 220

Interactions par formulaire

- On va utiliser l'accès aux éléments HTML et les *formulaires HTML* pour interagir avec l'utilisateur
- Belle alternative aux fonctions que l'on connaît déjà…
  - prompt(), alert() et confirm()

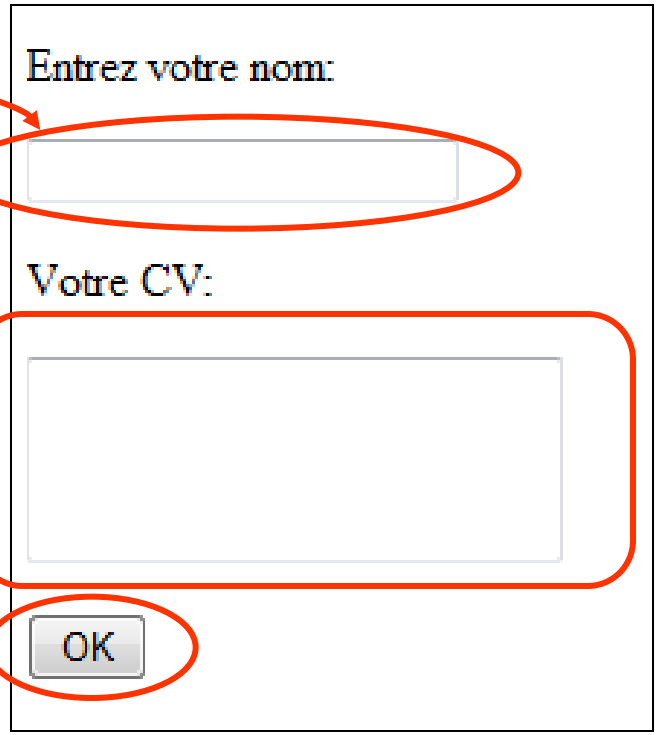
Entrées par formulaire

- Utilise un formulaire (`form`) HTML
- Un formulaire HTML comporte des *contrôles* de divers types, notamment:
  - zone de saisie de type `input type="text"`
    - on peut aussi utiliser `type="number"`
  - zone de saisie de type `textarea`
  - bouton de type `input type="submit"`

Rappel: formulaires HTML

En navigateur:

```
<form>
   <p>Entrez votre nom:</p>
   <p><input type="text"></p>
   <p>Votre CV:</p>
   <p><textarea></textarea></p>
   <p><input type="submit" value="OK"></p>
</form>
```



Formulaires et scriptage (1/3)

- Pour pouvoir accéder au contenu saisi dans un contrôle de type `input type="text"` ou `textarea`, on va l'identifier en lui ajoutant un attribut `id`

```
<form>
   <p>Entrez votre nom:</p>
   <p><input type="text" id="entrée"></p>
   <p><input type="submit" value="OK"></p>
</form>
```

Formulaires et scriptage (2/3)

- L'attribut `action` de `form` détermine ce qui se passe quand on clique sur le bouton de type `submit`

```
<form action="javascript:alert('Vous avez cliqué OK');">
  <p>Entrez votre nom:</p>
  <p><input type="text" id="entrée"></p>
  <p><input type="submit" value="OK"></p>
</form>
```

Exemple 369

Formulaires et scriptage (3/3)

- Dans cette action, on peut accéder à la chaîne saisie dans un contrôle du formulaire (propriété `value`):

```
document.getElementById("id").value
```

```
<form action='javascript:alert("Bonjour " +
        document.getElementById("entrée").value);'>
  <p>Entrez votre nom:</p>
  <p><input type="text" id="entrée"></p>
  <p><input type="submit" value="OK"></p>
</form>
```

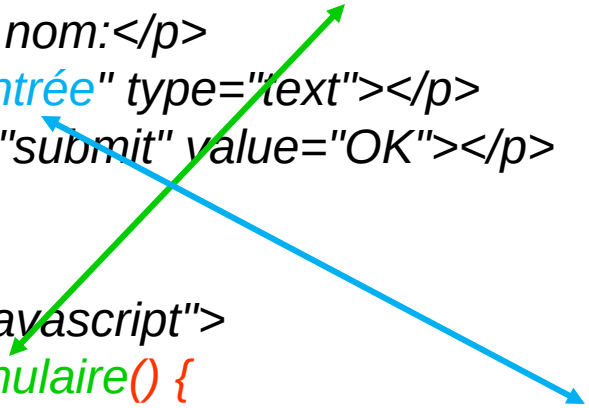
Exemple 375

Limites de l'attribut "action"

- L'attribut "action=…" du formulaire est peu pratique si on veut faire des traitements complexes, qui demandent *plusieurs* énoncés JS
  - La question des guillemets, aussi, est une nuisance
- Solution: mettre comme "action=…" un *appel de FDP*, laquelle peut être aussi complexe que l'on veut

Ça devient

```
...
  <form action='javascript:traiterFormulaire();'>
    <p>Entrez votre nom:</p>
    <p><input id="entrée" type="text"></p>
    <p><input type="submit" value="OK"></p>
  </form>
...
<script type="text/javascript">
function traiterFormulaire() {
     ctrlEntree = document.getElementById("entrée");
     alert("Bonjour " + ctrlEntree.value);
};
</script>
```



Exemple 385

Petit raffinement pour la convivialité

- Le curseur n'est pas dans le contrôle de saisie au chargement de la page
- Peut être fait avec la fonction `.select()` du contrôle `<input>`
- Ajouts mineurs pour intégrer cette commodité: exemple 386

Notes sur les variables locales (1/2)

- Dans une page scriptée avec interaction par formulaire, la FDP appelée lors de la soumission du formulaire **ne sert qu'à regrouper des énoncés**, et non à encapsuler un traitement
- Il est donc acceptable de déroger **dans ce cas** à la bonne pratique consistant à déclarer *locales* toutes les variables de travail d'une FDP

Notes sur les variables locales (2/2)

- En effet, laisser les variables globales permet de les visualiser à la console JS après la fin de l'interaction, une fois que la FDP a terminé son exécution
- Ceci facilite la mise au point de la page scriptée !
  - N.B.: Il est possible de visualiser les variables locales d'une fonction à la console, mais c'est plus compliqué que les variables globales

Exercice 1

- On reprend nos bonnes habitudes :
  1. À partir de l'Exemple 386, faire en sorte que si l'utilisateur tape des blancs inutiles avant et/ou après son nom, ceux-ci sont ignorés
  2. Vérifier si un nom vide a été donné
     - Si c'est le cas, simplement ne rien afficher du tout (aucune alert() ne s'exécute)

Exercice 2

- Prendre votre traducteur de saison (p.ex. 100) et le modifier pour que l'entrée soit par formulaire

Exercice 3

- Un script avec deux entrées distinctes (donc, deux <input>) dans un formulaire, qui accepte deux nombres et en affiche la somme et le produit
  - Recommandé : input type="number"
- Explorer les différents attributs qui permettent d'encadrer la saisie (step, min, max, lang, etc.)

Dynamisme par propriétés CSS (1/2)

- Goodman BC232-BC257
- document.getElementById('monElem').style donne accès à *toutes* les propriétés CSS d'un élément HTML. Ex.:

```
document.getElementById('monElem').style.font =
  "bold sans-serif 16px";
document. getElementById('monElem').style.textAlign =
  "center";
```

Dynamisme par propriétés CSS (2/2)

- Les noms des propriétés doivent être "JSisés" (car "-" est un opérateur en JS !) :
  - text-align → textAlign
  - background-color → backgroundColor

Exemples 480 et 490

Dynamisme par classes

- Soit :

```
v = document.getElementById('monElem');
```

- Alors `v.classList` donne accès à un objet permettant d'ajouter, retirer ou tester la présence des classes CSS à l'élément, ex.:

```
v.classList.add('maClasse');
v.classList.remove('maClasse');
v.classList.contains('maClasse'); <--> true/false
```

Exercice 4

- À partir de l'infrastructure vide avec formulaire (exemple 006), faire une page scriptée qui va accepter un nom de couleur (en anglais) et régler la couleur de fond *du contrôle d'entrée* à cette couleur
  - Pas besoin de valider le nom de couleur !
  - Les espaces inutiles nuisent-elles ? Expérimentez pour le découvrir !

Exercice 5

- Modifier le traducteur de saison de l'Exercice 2 (avec entrée par formulaire) pour que, si un entrée invalide est fournie, en plus d'un message d'erreur, *la couleur de fond du contrôle d'entrée est modifiée pour attirer l'attention*
  - Ne pas oublier de ramener à la couleur normale une fois l'erreur corrigée

Exercice 6

- Pouvez-vous deviner comment on peut utiliser un contrôle de formulaire

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

  pour **afficher** un message ?
  - Ce serait une forme de *sortie* par formulaire

`.innerText` et `.innerHTML`

- Propriété des objets (correspondant à des éléments) HTML
- Fonctionnent en lecture ET écriture
- Permet de modifier dynamiquement ce qui s'affiche dans la fenêtre de navigation
- Donc, permet d'afficher un message
- Remplace avantageusement **`alert()`**

Exemples 295 et 297