Université de Bretagne Occidentale

Développement web côté client

Cours 2 : JavaScript – Animation, validation des données de formulaires

Hai Nam TRAN
Université de Bretagne Occidentale – L2 INFO

Projet

Jalon	Description
1	HTML & CSS
2	JavaScript : utiliser JavaScript pour effectuer certaines tâches samples et animations
3	Validation de données : avoir un formulaire d'inscription et valider toutes les données saisies
4	Base de données : gérer une base de données locale simple avec xml
5	Bootstrap: appliquer le framework bootstrap après avoir validé Jalon 4

Jalon 1

- **HTML & CSS**
  - **Une page d'accueil**
  - **>= 3 rubriques**
  - **Un menu de navigation**
  - **Contenu pour la page d'accueil et les rubriques**
  - **Utilisation des propriétés CSS**
  - **Appliquer la principe flexbox pour la mise en page**
    - Un gabarit basée sur une charte graphique précisée, actuelle et professionnelle
    - Page d'accueil avec un gabarit différent (facultatif)
  - **Design adaptatif (bonus)**

Objet, propriétés, et attributs

- **getElementById("objet_id")**
  - Cherche l'objet (un nœud DOM) avec l'identifiant donné et renvoie cet objet

```
<input type="text" id="fname" value="Aabecede">

<script>
    let fname = document.getElementById("fname");
    alert("Hello " + fname.value);
</script>
```

```
<input type="text" id="fname" value="ABC"/>
```

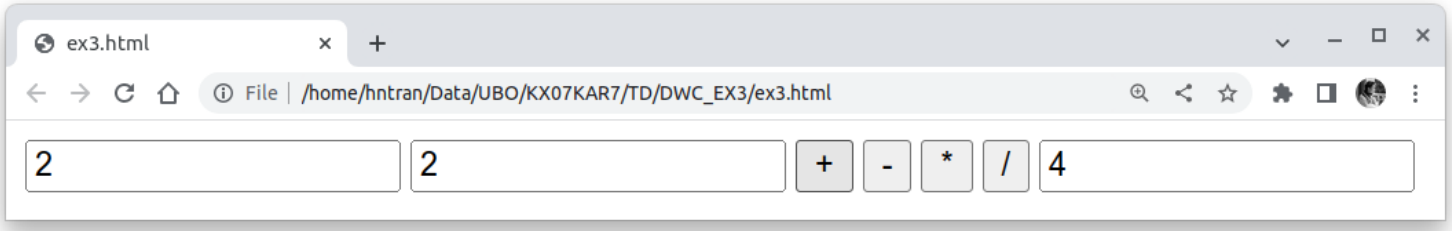
- **Attributs**
  - Les attributs qu'on peut définir pour une balise HTML
- **Propriétés**
  - Lorsqu'un navigateur analyse le code HTML, il crée un objet DOM correspondant à la balise avec des propriétés

Objet, propriétés, et attributs

- **Et maintenant ça devient compliqué…**
  - **L'objet a une propriété `attributes`**
  - **Les attributs de la balise sont les éléments de cette propriété**
  - **Plusieurs propriétés de l'objet ont les même noms que les attributs**
    - **`id`** : reflected property – lecture de la propriété = la valeur de l'attribut id
    - **`type`** : reflected property
    - **`value`** : non-reflected property - lecture de la propriété = la valeur actuelle de la balise (textbox)
      - Dans l'exemple, l'utilisateur modifie le contenu du textbox en mettant "DEF"
      - **`fname.value //return "DEF"`**
      - **`fname.getAttribute('value') //return "ABC"`**
    - **`href :`** la propriété DOM href est toujours une URL complète, même si l'attribut contient une URL relative

Objet, propriétés, et attributs

- **Les propriétés DOM sont typées**
  - **value : chaine de caractères**
  - **style : objet**
    - L'attribut style est une chaine de caractères




```
let n1 = document.getElementById("n1").value;
let n2 = document.getElementById("n2").value;
let result = n1+n2;
let txtResult = document.getElementById("result");
txtResult.value = result;
```

4 ou 22 ?

Modification du document

- **En manipulant le DOM avec JS, on peut :**
  - **Créer de nouveaux nœuds**
    - document.createElement(nom_de_balise) : crée un élément avec la balise donnée
  - **Insertion et suppression**
    - node.append(...nodes or strings) : insère dans node, à la fin
    - node.prepend(...nodes or strings) : insère dans node, au début
    - node.before(...nodes or strings) : insère juste avant node
    - node.after(...nodes or strings) : insère juste après node
    - node.replaceWith(...nodes or strings) : remplace node
    - node.remove() : supprime le node
  - **Modifier les propriétés et attributs de nœuds**
    - Note : utilisation principale dans cette UE

Modification du document

```html
<ol id="mylist">
  <li>B</li>
  <li>C</li>
  <li>D</li>
</ol>
```

```html
<script>
  let mylist = document.getElementById("mylist");

  myList.before = "Une liste"

  let liFirst = document.createElement('li');
  liFirst.innerHTML = 'A';

  mylist.prepend(liFirst);

  let liLast = document.createElement('li');
  liLast.innerHTML = 'D';

  mylist.append(liLast);

  myList.after = "Fin de la liste"
</script>
```

Université de Bretagne Occidentale

Animation

Cours 2

Jalon 2

Hai Nam TRAN
Université de Bretagne Occidentale – L2 INFO

Animation

- **Ajouter l'aspect "dynamique" à une page web**
  - **Diaporama, animation, jeux, …**
- **Idée générale**
  - **Afficher un objet**
  - **Modifier l'objet**
    - Déplacer
    - Modifier son style
    - Modifier ses attributs
  - **après un intervalle fixe :** setInterval()
  - **après un ou plusieurs évènements :** onclick, onmouseover, keypress, keydown, keyup → à explorer

Animation

```
let timerId = setInterval(func|code, [delay], [arg1], [arg2], ...)
```

- **Exécuter une fonction de manière répétée, en commençant après l'intervalle de temps, puis en répétant continuellement à cet intervalle**
  - **func :** fonction ou chaîne de caractères représentant du code à exécuter
  - **delay :** l'intervalle entre deux appels de la fonction, en millisecondes (1000ms = 1 seconde)
  - **arg1, arg2,... :** arguments à passer à la fonction
  - **timerid :** un "identifiant de timer" que l'on peut utiliser pour annuler l'exécution de la fonction

```
clearInterval(timerId)
```

- **Annuler les appels futurs à la fonction avec l'identifiant de timer donné**

Animation – Exemple 1

```
let id = null;
let posx = 0;

function myMove() {
    id = setInterval(move, 1000/60);
}

function move() {
    let elem = document.getElementById("animate");
    if (posx == 350) {
        clearInterval(id);
    } else {
        posx++;
        elem.style.left = posx + "px";
    }
}
```

60 fps

```
<button onclick="myMove()">Animate</button>
```

Animation – Exemple 2

```
let sIndex = 0;
setInterval(autoSlide,1000);

function autoSlide() {
    sIndex = sIndex + 1;
    displaySlide();
}
```

```
<div class="slides">
    <img src="img1.jpg>
</div>

<div class="slides">
    <img src="img2.jpg">
</div>

<div class="slides">
    <img src="img3.jpg">
</div>
```

```
function displaySlide() {
    let i;
    let slides =
    document.getElementsByClassName("slides");

    if (sIndex >= slides.length) {
        sIndex = 0;
    }

    if (sIndex < 0) {
        sIndex = slides.length - 1
    }

    for (i = 0; i < slides.length; i++) {
        slides[i].style.display = "none";
    }

    slides[sIndex].style.display = "block";
}
```

Animation – Exemple 3

```
function startstop(){
      if(etat) {
        clearInterval(timerId);
        etat = false;
    }
    else {
        timerId = setInterval(tick,100);
        etat = true;
    }
}
```

```
let etat = false;
var timerId;
```

```
function tick(){
    let txtClock = document.getElementById("clock")
    let a = parseInt(txtClock.value);
    if(a<99){
        a = a +1;
    }
    else {
        a = 0;
    }
    txtClock.value = a;
}
```

requestAnimationFrame

- **Animation = redessiner la page web**
  - **Comparer**

```
setInterval(function() {
    animate1();
    animate2();
    animate3();
}, 20)
```

```
setInterval(animate1, 20);
setInterval(animate2, 20);
setInterval(animate3, 20);
```

  - **Parfois, le CPU est surchargé → on doit redessiner moins**
- **Une autre alternatif**

```
let requestId = requestAnimationFrame(callback)
cancelAnimationFrame(requestId);
```

  - **Programme la fonction callback pour qu'elle s'exécute au moment le plus proche où le navigateur veut faire une animation**

requestAnimationFrame

```
function animatePage() {
    // animation ici
    requestAnimationFrame(animatePage);
}

requestAnimationFrame(animatePage);
```

- **requestAnimationFrame est 1-shot**
  - **→ Utilisation : appels récursives**
  - **Utiliser cancelAnimationFrame() pour arrêter l'animation**

Jalon 2

- **JavaScript**
  - **Un "outil"**
  - **Une animation**
  - **Un diaporama**
  - **Un album photo / portfolio**

Université de Bretagne Occidentale

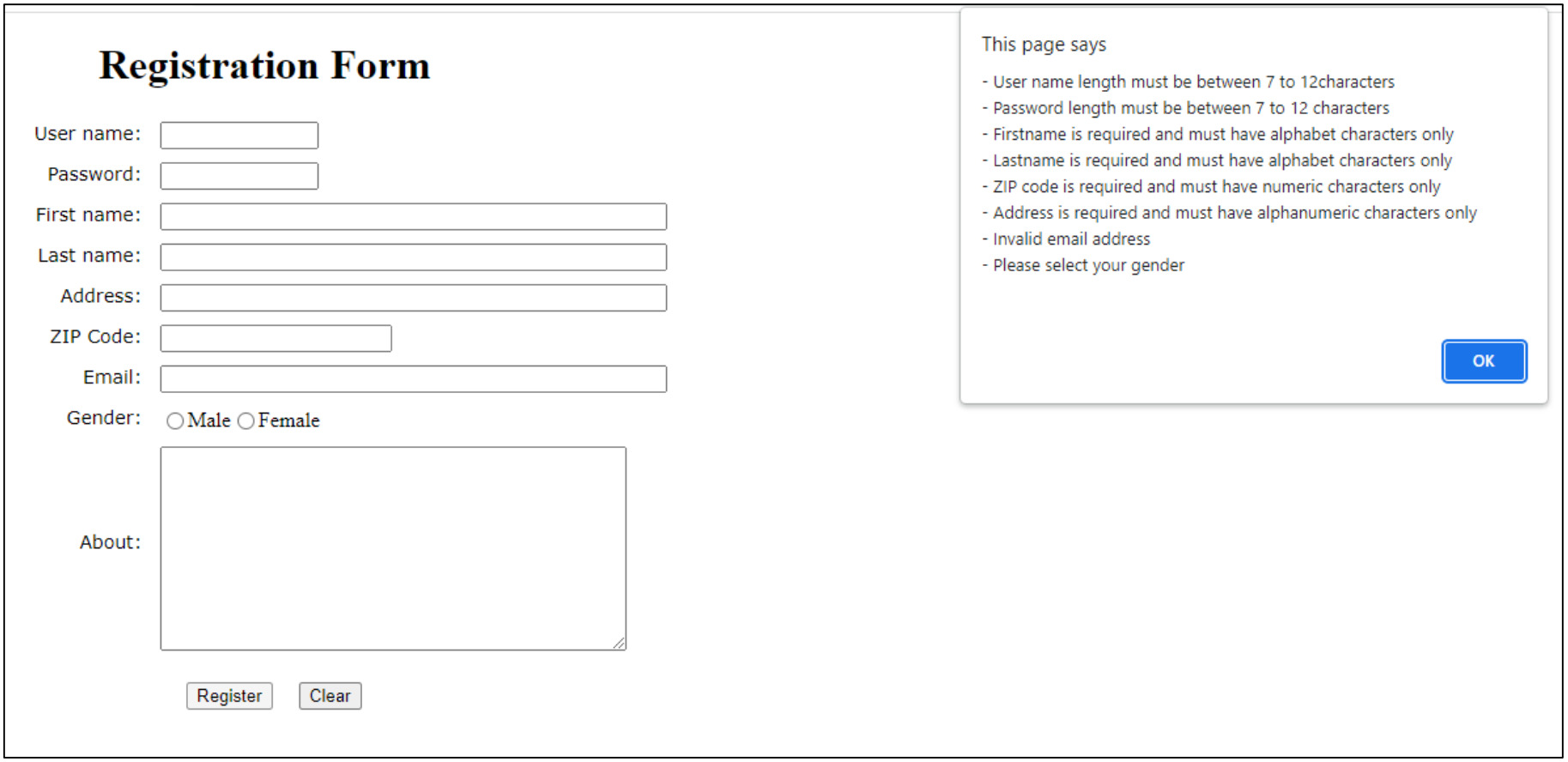
Validation des données de formulaires

Cours 2

Jalon 3

Hai Nam TRAN
Université de Bretagne Occidentale – L2 INFO

Validation des données de formulaires

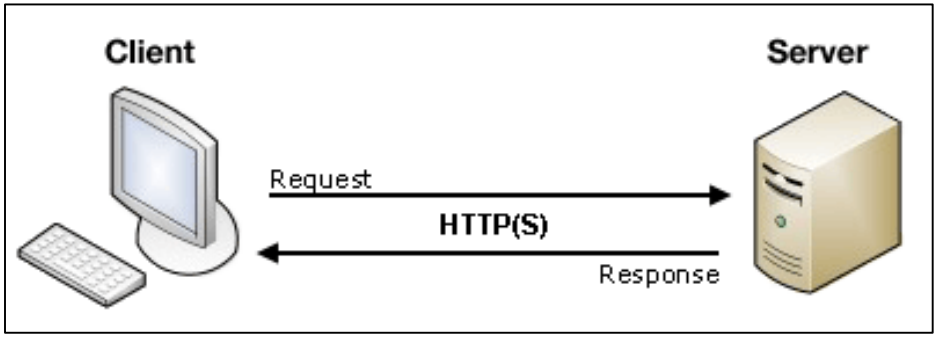


Formulaires HTML

- **Des outils pour interagir avec l'utilisateur**
  - **Un des vecteurs principaux d'interaction entre un utilisateur et un site web ou une application**
    - Permettent à l'utilisateur d'envoyer des données au site web
- **Un formulaire HTML est composé d'un ou plusieurs widgets**
  - **Zone de texte - textbox, textfield**
  - **Boîte à sélection - dropdown box**
  - **Bouton - button**
  - **Cases à cocher – checkbox**
  - **Bouton radio – radio button**

Formulaires HTML

- `<form action="..." method="...">`
  - Définit un formulaire
  - `action` : définit l'emplacement (une URL) où doivent être envoyées les données collectées par le formulaire
  - `method` : définit la méthode HTTP utilisée pour envoyer les données (get/post)



action/method : à voir dans les autres UE de développement web (côté serveur)

Formulaires HTML

- **<input> : l'élément de saisie dans un formulaire**
  - **Une liste + description complète**

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

  - **<input type="text">**
    - Champs de saisie pour du texte sur une seule ligne
    - Exemple : <input type="text" id="txtfname" value="Aabecede">
  - **<input type ="button">**
    - Boutons qui peuvent être programmés afin de contrôler des fonctionnalités de la page via un gestionnaire d'évènement (la plupart du temps pour l'évènement **onclick**)
    - Exemple : <input type="button" value="Hello" onclick="helloJS()">
  - **<input type="password">**
    - Champs de saisie pour un mot de passe sans que celui-ci ne soit lisible à l'écran
    - Exemple : <input type="password" minlength="8" required>
  - **…**

Devoir 2 : Créer un formulaire HTML et tester les différents <input>

Formulaires HTML

```
<table>
    <tr>
        <td><label for="username">User name:</label></td>
        <td><input type="text" id="txtUsername" size="12" /></td>
    </tr>
    <tr>
        <td><label for="passid">Password:</label></td>
        <td><input type="password" id="txtPassword" size="12" /></td>
    </tr>
    ...
</table>
```

- **Un formulaire est souvent crée en utilisant une liste ou un tableau (invisible) pour placers ses éléments**

Validation des données de formulaires

- **Définition**
  - **Vérifier la correction de données saisies par l'utilisateur avant**
    - (1) envoyer les données au serveur
    - (2) modifier/sauvegarder les données dans une base de données
  - **Quelques exemples**
    - Ce champ est obligatoire
    - Veuillez entrer votre numéro de téléphone au format xx-xx-xx-xx-xx
    - Veuillez entrer une adresse e-mail valide
    - Votre mot de passe doit comporter entre 8 et 12 caractères et contenir une majuscule, un symbole et un chiffre
    - Nom d'utilisateur est déjà pris
    - Cette adresse e-mail est déjà utilisée
- **Objectif**
  - **Obtenir de bonnes données dans un bon format**
  - **Protéger nos utilisateurs**
  - **Protéger nos serveurs/base de données/application web**

Validation des données de formulaires

- **Côté client** **UE Développement Web côté client**
  - **La validation est effectuée dans le navigateur**
  - **Avant que les données n'aient été soumises au serveur**
    - Moins de circulations de données
    - Moins de traitements à faire côté serveur
    - … mais, pas possible de vérifier tout
- **Côté serveur**
  - **La validation est effectuée sur le serveur (après que les données ont été soumises)**
  - **Nécessite un aller-retour vers le serveur**
    - Le serveur est plus chargé
    - Plus de "outils" de validation

Dans la pratique, on utilise une combinaison de validations côté client et côté serveur (en maximisant la validation côté client)

Validation des données de formulaires

- **Côté client – plusieurs implémentations possibles**
  - **Utiliser des attributs de validation sur les éléments de formulaire – HTML5**
    - Built-in – pas de besoin d'implémentations
    - Moins de flexibilité
    - Meilleure performance
    - Exemples
      - **required** : une valeur doit-elle être remplie
      - **minlength**, **maxlength** : longueur minimale et/ou maximale des données
      - **min**, **max** : valeur minimale et/ou maximale des données
      - **type** : type des données
      - **pattern** : une **expression régulière**
  - **Validation de formulaires avec JavaScript**
    - Implémentations nécessaires
    - (beaucoup) Plus de flexibilité

Approche à utiliser pour le projet

Validation des données de formulaires

```
function username_validation() {
    let uname = document.getElementById("txtUsername");
    var uname_len = uname.value.length;
    if (uname_len == 0 || uname_len >= 6 || uname_len < 12) {
        alert ("- Username must be between 6 to 12 characters \n");
    }
}
```

- La plupart des validations **simples** peuvent être effectuées en utilisant la valeur d'un objet et des opérateurs de comparaisons
- Pour les validations plus avancées
  - → expressions régulières

Expressions régulières

- **Des motifs (pattern) décrit, selon une syntaxe précise, un ensemble de chaînes de caractères possibles**

```
regexp = new RegExp("pattern", "flags");
```

```
regexp = /pattern/flags;
```

  - **Les slash /.../ indique à JavaScript que l'on crée une expression régulière.**
  - **Les expressions régulières peuvent avoir des flags qui affectent la recherche**

```
function name_validation() {
    let name = "JeanBon007";
    let regexp = /^[A-Za-z]+$/;
    if (!regexp.test(name) {
        alert("- Name must have alphabet characters only");
    }
}
```

Expressions régulières

- **Les flags**

Flag	Description
i	la recherche est insensible à la casse
g	la recherche liste toutes les correspondances, sans lui – seulement la première
m	mode multiligne
s	Active le mode “dotall”, qui permet à un pattern : . de correspondre au caractère de nouvelle ligne \n
u	Active le support complet Unicode
y	mode “Sticky” : chercher à la position exacte dans le texte

Expressions régulières

- **regexp.test**
  - **Recherche au moins une correspondance; si elle est trouvée, retourne true, sinon false**

```
let str = "Toto tata";
let regexp = /TOTO/i;
alert(regexp.test(str)); // true
```

Expressions régulières

- **Exemples de regexp**
  - **/^[A-Za-z]+$/** : la chaine ne contient que des lettres
    - ^ : début d'une ligne
    - $: fin d'une ligne
    - + : un groupe qui existe une ou plusieurs fois
    - [liste] : Un des caractères entre crochets
  - **/^[0-9]+$/** : la chaine ne contient que des numéros
  - **/^\w+[\+\.\w-]*@([\w-]+\.)*\w+[\w-]*\.([a-z]{2,4}|\d+)$/i** : adresse email valide

- **Apprendre regexp**
  - **Regular Expressions Cookbook, 2nd Edition by Jan Goyvaerts, Steven Levithan (612 pages)**

Jalon 3

- **Un formulaire d'inscription**
  - **Validation de données**
  - **Réaffichage/récapitulatif des données saisies**