Université de Montréal

JavaScript

Jamel Eddine Jridi

Introduction

C quoi JS et c qoui ses caractersitique?

- JavaScript est un langage de script le plus souvent utilisé pour le développement web côté client.
- Qu'est ce qu'on peux faire avec ?
  - Création d'une interface utilisateur interactive dans une page Web (par exemple, menu, alerte pop-up, fenêtres, etc.)
  - Manipuler le contenu web dynamique
  - Modifiez le contenu et le style d'un élément
  - Remplacer les images sur une page sans rechargement de la page
  - Masquer/afficher le contenu
  - Générer du contenu HTML
  - La validation des formulaires
- JavaScript manipule la structure DOM du document HTML.

Document Object Model (DOM)

- Le DOM (Document Object Model) est un standard du W3C définit pour accéder aux documents.
- Permet à des programmes informatiques et à des scripts d'accéder ou de mettre à jour le contenu, la structure ou le style de documents HTML et XML
- Niveaux de la DOM Q d'examen
  - DOM 0: informelles, les premiers navigateurs
  - DOM 1: XHTML / structure XML
  - DOM 2: modèle d'événement, interface de style
  - DOM 3: modèle de contenu, validation

Puissance JavaScript

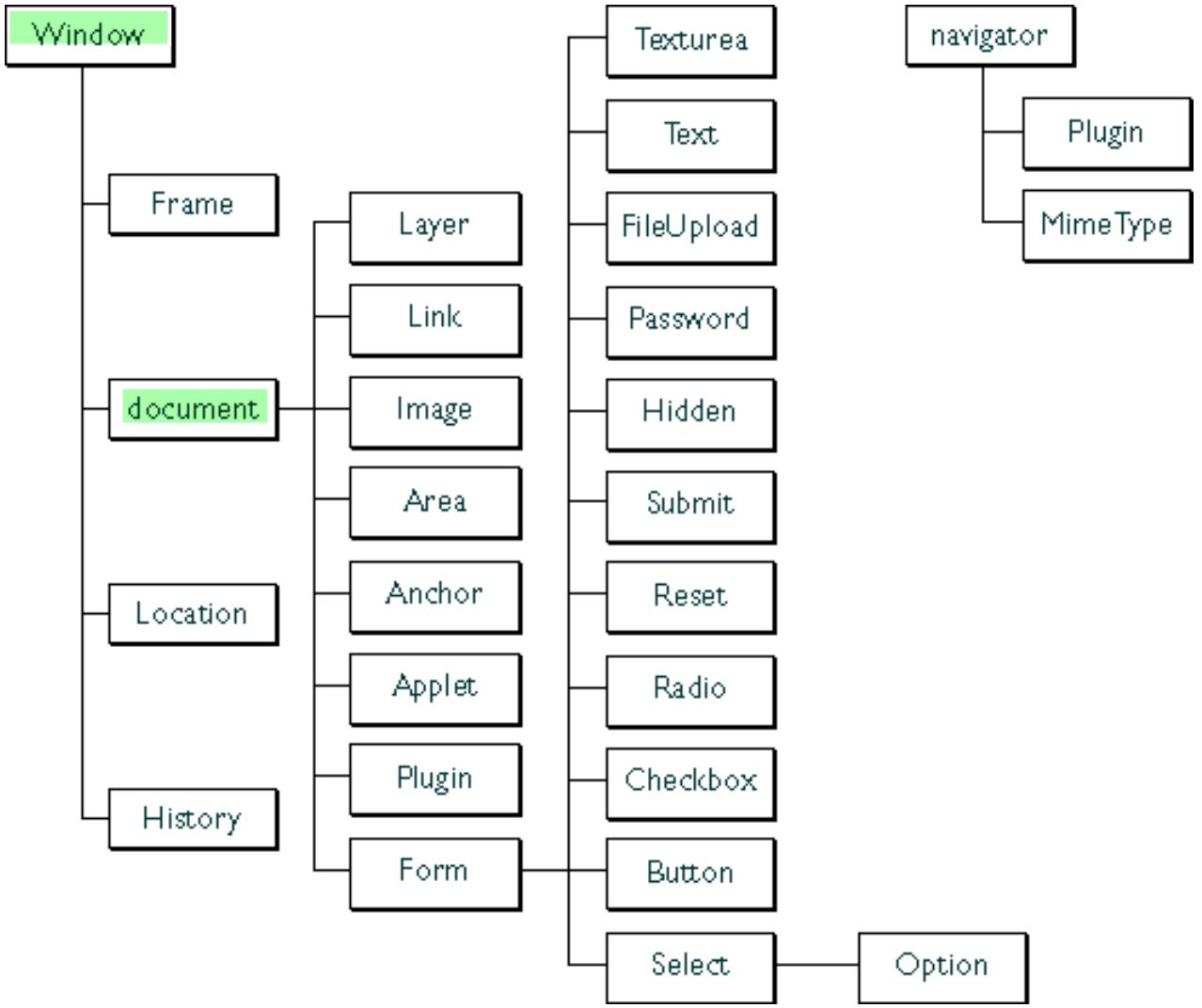
Important

- Avec la structure DOM, JavaScript obtient toute la puissance dont il a besoin pour créer HTML dynamique:
  - peut changer tous les éléments HTML dans la page
  - peut changer tous les attributs HTML dans la page
  - peut changer tous les styles CSS dans la page
  - peut supprimer des éléments et attributs HTML existants
  - peut ajouter de nouveaux éléments et attributs HTML
  - peut réagir à tous les événements HTML existants dans la page
  - peut créer de nouveaux événements HTML dans la page

Disons

Les objets du navigateur

la navigateur qu'on peut acceder par JS



Les objets du navigateur

- L'objet le plus haut dans la hiérarchie est `window` qui correspond à la fenêtre même du navigateur.
- Propriétés de l'objet de `navigator` permettent au script de déterminer les caractéristiques du navigateur dans lequel le script est en cours d'exécution
  - `appName` donne le nom du navigateur
  - `appVersion` donne la version du navigateur
- L'objet `document` fait référence au contenu de la fenêtre. Base de DOM
- `document` regroupe au sein de propriétés l'ensemble des éléments HTML présents sur la page.
  - **soit des méthodes propres à l'objet** `document`, comme la méthode `getElementById( )`, qui permet de trouver l'élément en fonction de son identifiant (`id`);
  - **soit des collections d'objets** qui regroupent sous forme de tableaux Javascript tous les éléments de type déterminé.

getElementByTagName( ) /
getElementByClassName( )

Insertion dans une page HTML

- Le code peut être inséré où vous le désirez dans votre page Web
- Insertion pour exécution directe :
  - Le code s'exécute automatiquement lors du chargement de la page HTML dans le navigateur en même temps que le contenu de la page HTML s'affiche à l'écran.

```
<html>
<head><title></title></head>
<script type="text/javascript">
    ……
</script>
<body></body></html>
```

```
<html>
<head><title></title></head>
<body>
<script type="text/javascript">
    ……
</script>
</body></html>
```

- Insertion par appel de module externe

```
<html>
<head><title></title></head>
<script type="text/javascript"
        src="file.js" />
<body></body></html>
```

```
<html>
<head><title></title></head>
<body>
<script type="text/javascript"
        src="file.js" />
</body></html>
```

Exemple JavaScript

- Les scripts à l'intérieur d'un document HTML est interprété dans l'ordre où ils apparaissent dans le document.
- Scripts dans une fonction est interprétée lorsque la fonction est appelée.

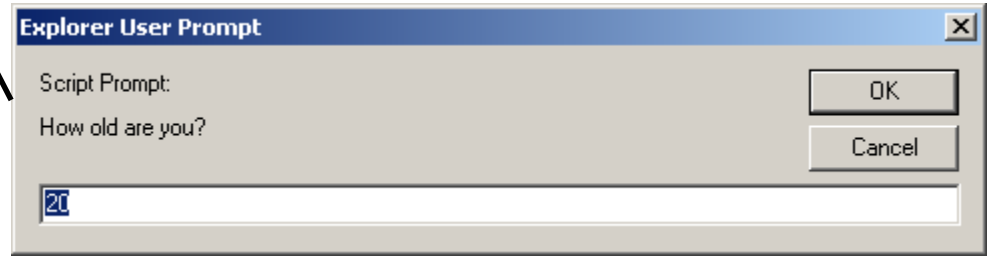
```
<html>
<head><title></title>
<script type="text/javascript">
    function bonjour()
    {
        alert('bonjour');
    }
</script>
</head>
<body>
  <button onclick='bonjour()'>click</button>
</body>
</html>
```

alert(), confirm(), prompt()

```
<script type="text/javascript">

    alert("This is an Alert method");
    confirm("Are you OK?");
    prompt("What is your name?");
    prompt("How old are you?","20");

</script>
```



Accès au éléments en JavaScript

- Les éléments dans le document XHTML correspondent à des objets en JavaScript
- Les objets peuvent être traitées de différentes façons:
  - Les tableaux `forms` et `elements` réseau défini en DOM 0
  - Utiliser l'attribut `name` du formulaire et ces éléments
- Les noms sont nécessaires pour les éléments de formulaire fournissant des données au serveur
- Utilisation de `getElementById` avec un attribut `id`
  - la valeur de l'attribut `id` doit être unique pour un élément

Accès au éléments en JavaScript (**forms**)

- Supposons ce simple formulaire:

```
<form action = "">
   <input type = "button"  name = "pushMe">
</form>
```

- L'élément peut être référencé comme suit (`forms`) :

```
document.forms[0].element[0]
```

Accès au éléments en JavaScript (attribut **`name`**)

- Exemple

```
<form name = "myForm"  action = "">
    <input type = "button"  name = "pushMe">
</form>
```

- Référencer l'élément `<input>`

```
document.myForm.pushMe
```

Accès au éléments en JavaScript (attribut **`id`**)

- Définissez l'attribut `id` pour l'élément `<input>`

```
<form action = "">
   <input type="button"  id="turnItOn">
</form>
```

- Utilisation de `getElementById`

```
document.getElementById("turnItOn")
```

Gestion des événements en JavaScript

- La programmation événementielle est un style de programmation dans lequel des morceaux de code, des gestionnaires d'événements, sont écrits pour être activé lorsque certains événements se produisent
- Les événements représentent l'activité dans l'environnement, y compris, en particulier, les actions de l'utilisateur telles que le déplacement de la souris ou en tapant sur le clavier
- Un gestionnaire d'événements est un segment de programme conçu pour exécuter lorsqu'un certain événement se produit
- Les événements sont représentés par des objets JavaScript
- Attribuer un événement pour un noeud DOM

Gestion des événements en JavaScript

- La capture d'un événement consiste à exécuter une action lorsque l'événement surveillé se produit dans le document.
- Les événements capturables du DOM sont :
  - Événements page et fenêtre
    - `onload` — après la fin du chargement de la page
  - Événements souris
    - `onclick` — sur un simple clic
    - `ondblclick` — sur un double clic
    - `onmouseover` — lorsque la souris est sur l'élément
  - Événements clavier
    - `onkeydown` — lorsqu'une touche est enfoncée
    - `onkeypress` — lorsqu'une touche est pressée et relâchée
  - Événements formulaire
    - `onselect` — quand du texte est sélectionné
    - `onsubmit` — quand le formulaire est validé

Gestion des événements en JavaScript

- Exemple :

```
<input type="button" name="myButton" id="idButton"
onclick="alert('You clicked the button!')"/>
```

- Un appel de fonction peut être utilisée si le script est plus longue qu'une seule instruction

```
<input type="button" name="myButton" id="idButton"
   onclick="myHandler()"/>
```

- Utilisation de `getElementById` (propriété de la DOM)

```
document.getElementById("idButton").onclick =
   myHandler
```

  - Notez que le nom de la fonction est une référence à la fonction

Traversée et Modification de la structure DOM

- Chaque élément dans un document XHTML a un objet `element` correspondant dans la représentation DOM
- L'objet `element` a des méthodes pour soutenir (voir ce lien)
  - La traversée de l'arborescence du document
    - `parentNode` : le noeud parent de `element`
    - `previousSibling` et `nextSibling`
    - `firstChild` et `lastChild`
  - Modification du document (ajout et suppression des noeuds)
    - La méthode `insertBefore` insère un nouvel enfant du noeud cible
    - `replaceChild` remplacera un nœud enfant avec un nouveau noeud
    - `removeChild` supprime un nœud enfant
    - `appendChild` ajoute un nœud en tant que nœud enfant à la fin des enfants