Formation JAVASCRIPT / HTML DYNAMIQUE

Rappels HTTP + HTML

Historique

- Le World Wide Web (www) est créé par Sir Tim Berners-Lee en 1989.
- Le W3C, consortium créé en 1994, est chargé de superviser le développement des technologies du web

En 25 années

1989-1991: Invention et Implementation du Web
1992–1995: Evolution
Les premiers navigateurs (NeXT, WWWViola)
Gouvernance du Web
1996–1998 : Commercialisation sur le Web
1999–2001 : Explosion des "Dot-com" , puis la chute
2002- : Un web omniprésent
Web 2.0 : Complexité et Simplicité, Crowdsourcing, Réseaux sociaux
Web Semantic : (Berners-Lee & W3C)
Pour humain **ET** machine (metadata)
Futur : Web 3.0 / mobilité, universalité, accessibilité (WCAG2)
Depuis 2010, une multiplication des technologies

http://www.evolutionoftheweb.com/?hl=fr

Comment est structurée une page Web ?

- Le contenu HTML est constituée de balises
  - Définition: <balise>
  - Deux types
    - Les balises en paires
      <titre>Titre de ma page</titre>
    - Les balises auto-fermantes
      -
  - Les attributs
    -

http://www.w3.org/wiki/HTML_structural_elements

Un site web, comment ça marche ?

- Une architecture : Client / Serveur
- Un réseau : Internet
- Des protocoles : **HTTP**, HTTPS, ...
- Des technologies : URL, **HTML**, **CSS**, …
- Languages : Javascript, Flex, JavaFX...

Création d'un document HTML

```
<!doctype html>
<html lang="fr">
    <head>
        <meta charset="utf-8">
        <title>Titre de la page</title>
        <link rel="stylesheet" href="style.css">
        <script src="script.js"></script>
    </head>
    <body>
        ... <!-- Le reste du contenu --> ...
    </body>
</html>
```

http://w3c.github.io/html-reference/

Présentation du JavaScript

Introduction

- JavaScript est un langage interprété par le navigateur. Le JavaScript est un langage coté « **client** », c'est-à-dire exécuté chez l'utilisateur lorsque la page Web est chargée.

- Il a pour but de dynamiser les sites Internet
  - En rendant les pages interactives.
  - En permettant de modifier le contenu de la page sans la recharger.
  - On peut même créer de véritables applications riches en utilisant les derniers Framework apparus (jQuery/ExtJS)

Introduction

- Créé en 1995 par Brendan Eich pour Netscape.

Nom standard : ECMAScript

- Actuellement ECMA5 / depuis juin 2015 : ECMA6
  - a.k.a. ECMAScript 2015

http://kangax.github.io/compat-table/es5/
http://www.ecma-international.org

Les caractéristiques du JavaScript

- C'est un langage **interprété**.
- C'est un langage **orienté objet (par prototype)**.
- Il fonctionne dans un **bac à sable.**
- Exécuté par le **navigateur,** le comportement du code peut varier selon comment l'éditeur du navigateur a respecté les normes.
- En version simple, le code est interprété à la volée par le navigateur Web, **au fur et à mesure que la page est chargée**.

http://javascriptissexy.com/oop-in-javascript-what-you-need-to-know/
http://whatsmyuseragent.com/

Cas d'utilisations classiques

- Vérification des valeurs entrées dans un formulaire avant soumission .
- Mise à jour d'éléments de la page en fonction de différentes actions
  - Sélection d'un élément dans un menu déroulant comme par exemple un pays : affichage des régions
  - Affichage d'un texte lorsque la souris passe au-dessus d'un bouton.
- Requêtes au serveur après chargement de la page : **AJAX**
- Affichage de données Image, Video, GoogleMap

Premières applications

Attention

- Il faut **toujours** tester son Javascript sur tous les navigateurs que vont utiliser vos visiteurs.
- **Particulièrement** la famille **Internet Explorer**

Utiliser un **debogueur (ou débugueur ??)**

http://browsershots.org/

Syntaxe

- Le langage JavaScript est apparu en 1996 dans Netscape Navigator.

- Sa syntaxe, proche du C, est très souple et permissive :
  - « ; » finaux non obligatoires
  - Déclarations implicites possible
  - Comparaisons approximatives

Intégrer du Javascript dans une page

- Directement dans le document HTML :

```
<script language="javascript">
  console.log("It rocks");
</script>
```

- Sera ex alise en lisant le HTML.

Intégrer du Javascript dans une page

- Dans un fichier séparé :

```
<script language="javascript" src="animations.js">
</script>
```

- L'attribut **src** fonctionne de la même manière que celui pour les images
  - il peut être relatif
  - il peut être absolu

- Placé dans la balise **<head>** ou dans **<body>**

Les bases du langage

Variables & Types

Ces valeurs peuvent être stockées dans des variables, déclarées avec l'instruction `var` :

```
var detteMichel = 150;
```

Opérations de base :

```
var resultat = 30 - 2 * 4;

var bonjour = "Hello" + 'World';

var toto = null;
```

Variables & Types

- Chaque valeur en javascript a un type parmi les 6 types de base :
  - Les nombres (`number`)
  - Les chaînes de caracères (`string`)
  - Les booléens (`boolean`)
  - Les objets (`object`)
  - Les fonctions (`function`)
  - Les valeurs indéfinies (`undefined`)

`typeof` permet de récupérer le type d'une valeur

Conversions Implicites

Lors d'une opération ou d'une comparaison, JavaScript convertit implicitement les valeurs concernées :

```
resultat = "hello" + 4; // "hello4"
resultat = 4 + true // 5
result = 4 == '4' // true
result = '' == 0 // true
```

Pour empêcher la conversion implicite lors d'une comparaison, il faut utiliser l'opérateur **===**

Conversions Explicites

parseInt() permet de convertir une variable en nombre entier,

parseFloat() permet de convertir une variable en nombre décimal.

```
var maVariable = 5;
var maChaine = maVariable + '';
var monEntier = +maChaine;
var monBooleen = !!monEntier;
```

Les structures de contrôle

La condition if

La condition if permet de vérifier si une condition est rempli, si tel est le cas le code entre accolades sera exécuté.

```
if (condition) {
    /* code à exécuter si la condition est rempli */
} else {
    /* code à exécuter dans le cas contraire */
}
```

On peut chaîner les conditions avec && (et) ou || (ou)

while

- La boucle **while** se répète tant que la condition est validée. Cela veut donc dire qu'il faut s'arranger, à un moment, pour que la condition ne soit plus vraie, sinon la boucle se répéterait à l'infini

```
var number = 1;

while (number < 10) {
    number++;
}

alert(number);
```

Do while

- La boucle **do while** ressemble très fortement à la boucle **while**, sauf que dans ce cas la boucle est toujours exécutée au moins une fois. Dans le cas d'une boucle **while**, si la condition n'est pas valide, la boucle n'est pas exécutée.

```
var number = 1;

do {
    number++;
} while (number < 10)

 alert(number);
```

La boucle for

- La boucle **for** possède donc trois blocs qui la définissent. Le troisième est le bloc d'*incrémentation* qu'on va utiliser pour incrémenter une variable à chaque itération de la boucle. De ce fait, la boucle **for** est très pratique pour compter ainsi que pour répéter la boucle un nombre défini de fois.

```
for (var iter = 0; iter < 5; iter++) {
    console.log('num ' + iter);
}
```

Fonctions & Objets

Fonctions

- Une fonction est une suite d'instructions dans un bloc de code, pouvant ensuite être appelée autant de fois qu'on le souhaite

```
function afficherBonjour() {
    console.log('Bonjour !');
}

afficherBonjour(); // Affiche 'Bonjour !'

function afficherBonjourAvecUnPrenom(prenom) {
    console.log('Bonjour ' + prenom);
}

afficherBonjourAvecUnPrenom('Toto'); // 'Bonjour Toto'
```

Fonctions

Les fonctions sont également des valeurs et peuvent donc être stockées dans des variables

Ces deux instructions sont équivalentes :

```
function maFonction() {
    …
}

var maFonction = function () {
    …
}

maFonction();
```

Portée des variables

- Une variable déclarée au début du script, avant toutes fonctions, sera globale. Elle peut être utilisée n'importe où dans le script.

- Une variable déclarée explicitement dans une fonction aura une portée limitée à cette seule fonction, c'est-à-dire qu'elle est inutilisable ailleurs. On parle alors de « **variable locale** ».

Valeurs et Propriétés

En JavaScript, la plupart des valeurs contiennent d'autres valeurs qui leur sont associées, ce sont les propriétés.

Pour accéder à ces propriétés, il y a 2 syntaxes :

```
var texte = "Hello World";

texte.length
texte["length"]
```

Valeurs et Propriétés

Dans le cas d'une valeur de type `object`, il est possible d'ajouter, supprimer ou modifier ces propriétés :

```
var chat = {couleur: 'noir', nom: 'tosca'};

chat.age = 12;
chat.couleur = 'beige';
delete chat.age;
```

L'opérateur `in` permet de tester si un objet possède ou non une certaine propriété.

Méthodes

Les fonctions étant des valeurs comme les autres, il est possible de les affecter comme propriété d'un objet, on parle alors de méthodes.

```
chat.miaule = function () {
    console.log('miaou');
};

chat.miaule();
```

Les valeurs possèdent aussi des méthodes prédéfinies :

```
var hey = "Hello !";
typeof hey.toUpperCase; // function
hey.toUpperCase(); // "HELLO !"
```

Les tableaux en javascript

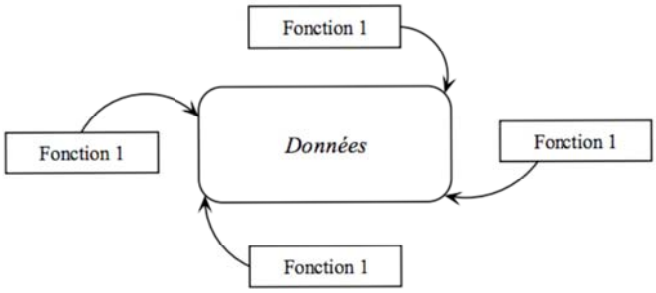
```
var tableau = [];

tableau[0] = 'toto';
tableau.push('tata');

console.log(tableau.length);
console.log(tableau.join(' '));
```

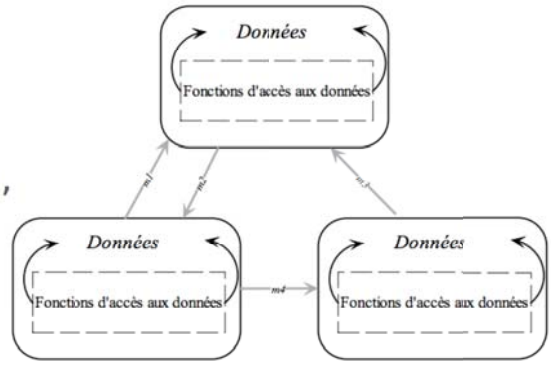
Les Objets & Javascript

Rappel : Procédural vs Objet



← Des instructions souvent réunies en fonctions qui modifient des données

Des entités avec un lien fort avec leurs données. Ces **objets** collaborent, s'envoient des informations dans un but commun →



Les objectifs ?
modularité, sécurité, lisibilité, portabilité.

Objets en Javascript

- Javascript est construit autour d'un paradigme basé sur des **objets**.
- On dispose d'objets déjà existants
  - Array, Date, Math,String, Regex
- On peut créer ses propres objets de multiples façons
- Construire une forme hiérarchie
- Utiliser puis détruire si besoin.

Construire un objet , plusieurs manières

- A partir d'une assignation.

```
monfruit = { couleur:"rouge", prix:5 };
```

- Une fonction que l'on nommera Constructeur.

```
function fruit(v) {
        var couleur = "rouge";
        this.prix = v;
}
var monfruit=new fruit(120) ;
document.write(monfruit.prix);
```

Pléthore d'autre possibilités

- Modifier un objet en cours de traitement

```
monfruit.producteur = "Naranji Spain" ;
```

- Redéfinir tous les objets d'une famille

```
fruit.prototype.origine = "Europe";
```

- Un objet peut contenir d'autre objets

```
function orange(p) { this.prix = p;  }
x = new orange(25);
monfruit.orange = x;
document.write("prix orange: " + monfruit.orange.prix);
```

Navigateur & Évènements

Les objets dans le navigateur

- `window` : objet global, il est sous-entendu.

```
window.alert("hey!")
alert("hey!")
```

- `window.navigator` : représente le navigateur directement
  - Version
  - Langue
  - Etc…
- `window.document` : représente le document HTML.

Les événements

- Les événements permettent de déclencher une fonction selon qu'une action s'est produite ou non. Par exemple, on peut faire apparaître une fenêtre alert() lorsque l'utilisateur survole une zone d'une page Web ou clique sur un lien.

- Cela peut se faire en ajoutant un attribut dont
  - Le **nom** correspond à l'événement
  - La **valeur** correspond au code ou à la fonction à exécuter

```
<a href="#" onClick="alert('hello');">Hello</a>
```

Interaction entre l'utilisateur et la page

- Exercice : Afficher une alerte lorsque la souris passe au dessus d'une image (mouseover).

Différents évènements

- load
- focus, blur and change
- mouseover and mouseout
- click, keydown

- Propre aux formulaires :
  - submit
  - reset

- Et de nombreux autres…

Les événements au travers du DOM

- Ils existent d'autres manières de gérer les événements en JavaScript, nous verrons ces méthodes au cours du prochain chapitre.

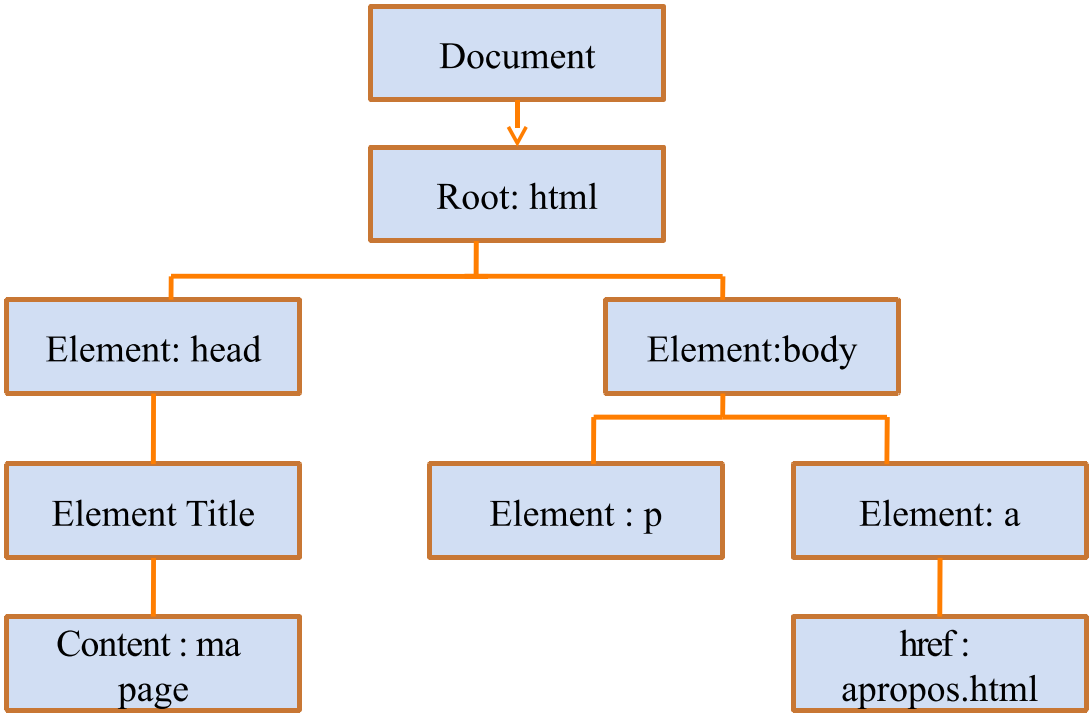
DOM

Document Object Model

DOM

- Désigne la technologie liant le HTML au JavaScript.
- Chaque balise HTML est un objet, dont les attributs sont les clés de l'objet.
- Le Document Object Model (DOM) a été normalisé par le W3C en 1998.

Architecture : en arbre



Sélectionner des objets

Deux méthodes permettent de sélectionner directement des objets sans explorer le DOM manuellement :

- document.getElementById("id");
  - Renvoie l' élément de la page dont l'id est placer en paramètre

- document.getElementsByTagName("balise");
  - Renvoie un tableau qui contient tout les éléments de la page qui correspondent au type de balises placé en paramètre.

Sélectionner des objets

Dans les navigateurs récents, 2 autres methodes sont disponibles, se basant sur les sélecteurs CSS

- document.querySelector(" img.profil");
  - Renvoie le premier élément de la page de type <img> possédant la classe « profil »

- document.querySelectorAll(".article .titre");
  - Renvoie un tableau contenant tout les éléments ayant la classe « titre » et contenus dans un élément ayant la classe « article »

API des élements

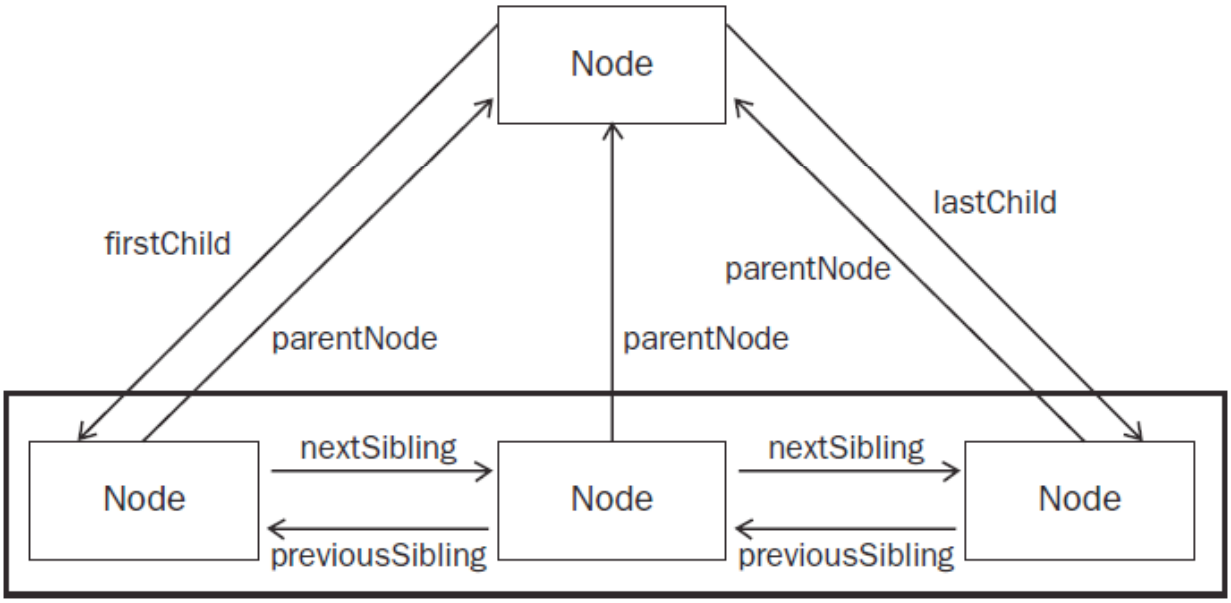
- innerHTML pour accéder au contenu d'un élément
- id, className, … pour les propriétés classiques
- Explorations des éléments
  - firstChild, lastChild, children…
  - nextSibling, previousSibling
  - parentNode
- Création d'éléments
  - createElement()
  - createTextNode(), appendChild()
  - removeChild()

Exemple

```
<p id="para">Ceci est un paragraphe</p>
<p>Ceci est un <em>autre</em> paragraphe</p>

<script>
    var p = document.getElementById("para");
    console.log(p.value);
    var pTab = document.getElementsByTagName("p");
    console.log(pTab[1].children[0].innerHTML);
</script>
```

Navigation DOM



DOM ELEMENT : BALISE XML (ÉLÉMENT)

- Le type Elément est plus utilisé dans le développement.
- Ce type représente un élément XML ou HTML, qui fourni les renseignements sur le nom de balise, ses enfants ou ses attributs.

DOM ELEMENT : BALISE XML (ÉLÉMENT)

- Propriété
  - parentNode
  - innerHTML
- Méthodes
  - getAttribute(string name)
  - hasAttribute(string name)
  - removeAttribute(string name)
  - setAttribute(string name,string value)

DOM NODELIST : SÉQUENCE DE NOEUDS

- Nodelist représente le type d'un groupe de nœud lors d'une sélection:
- Exemple de fonction retournant nodelist:
  - attributes;
  - getElementsByTagName();

- Méthodes
  - item(int i) : retourne le noeud à l'index i
- Attribut
  - length : nombre de noeuds

Les événements au travers du DOM

- Après avoir vu comment déclarer des événements directement dans les balises html, nous allons voir comment déclarer ces événements dans le code JavaScript en se servant tout d'abord de l'interface Dom-0 implémenter par Netscape, puis du standard actuel Dom-2.

- Le but de cette méthode va être de cloisonner le code, en effet on ne trouvera aucune trace de JavaScript dans la page html.

Exemple avec DOM-0

- On récupère tout d'abord l'élément HTML dont l'ID est clickme ;
- On accède ensuite à la propriété onclick à laquelle on assigne une fonction anonyme ;
- Dans cette fonction anonyme on fait un appel à la fonction alert() avec un String en paramètre.

```
<span id="clickme">Clic!!!!</span>
<script>
  var element = document.getElementById('clickme');

  element.onclick = function() {
      alert("clic clic");
  };
</script>
```

Suppression d’un événement avec Dom-0

- Pour supprimer un événement avec Dom-0, il suffit d’assigner une fonction vide a cet événement.
  - element.onclick = **function**() {};

Dom-2

- Première différence par rapport au dom-0, nous n'utiliserons plus une propriété pour déclarer notre événement mais la fonction addEventListener().

- Cette fonction prend trois paramètres :
  - L'événement
  - La fonction appeler au déclenchement de l'événement
  - Un booléen

Dom-2 exemple

```
var element = document.getElementById('clickme');
var myFunction = function() {
    alert("Vous m'avez cliqué !");
};
element.addEventListener('click', myFunction, false);
```

Dom-2 événements multiples

- Avec le dom-2, nous allons avoir la possibilité de déclarer plusieurs événements pour le même élément.

```
var element = document.getElementById('clickme');
// Premier événement click
element.addEventListener('click', function() {
    alert("Premier evenement");
}, false);
// Deuxième événement click
element.addEventListener('click', function() {
    alert("Deuxieme evenement");
}, false);
```

Supprimer un événement avec dom-2

- Pour supprimer un événement avec dom-2, on va utiliser la méthode removeEventListener();

```
element.removeEventListener('click', myFunction, false);
```

- Les evenements declares avec une fonction anonyme sont impossibles à supprimer.

Le cas internet-explorer

- Avec les versions d'internet-explorer antérieures à la version 9, les méthodes addEventListener() et removeEventListener() ne fonctionnent pas.
- Nous allons donc devoir utiliser également les méthodes attachEvent () et detachEvent().
- Ces méthodes fonctionnent de la même manières que les deux que nous avons vu précédemment mais elles ne prennent que deux paramètres.

Exemple dom-2 pour ie<9

```
var element = document.getElementById('clickme');
//creation de l'evenement
element.attachEvent('onclick', function() {
    alert('Tadaaaam !');
});
//suppression de l'evenement
element.detachEvent('onclick', function() {
    alert('Tadaaaam !');
});
```

Exemple

- Afin de garantir une compatibilité avec l'ensemble des navigateurs du marché noua allons devoir adapter notre code :

```
var element = document.getElementById('clickme');
function addEvent(element, event, func) {
    if (element.addEventListener) {
        element.addEventListener(event, func, false);
    } else {
        element.attachEvent('on' + event, func);
    }
}
addEvent(element, 'click', function() {
    alert("Cette alert apparaitra dans TOUS les navigateurs");
});
```

L’objet event

- L’objet event est l’objet qu’on pourra trouver en paramètre d’une fonction de callback, cet objet est très pratique, puisqu’il contient de nombreuses informations, comme notamment l’élément déclencheur de l’évènement, l’évènement lui-même, les coordonnées de la souris ou alors la touche du clavier qui a été pressée.

L’objet event

- Récupération du type d’événement déclenché

```
var element = document.getElementById('clickme');
element.addEventListener('click', function(e) { //e = objet event
    alert(e.type); // affiche le type de l'evenement
}, false);
```

L'objet event

- Récupération de l'élément qui a déclenché l'évènement.
- Malheureusement les anciennes versions d'ie ne supportent pas cette écriture.

```
var element = document.getElementById('clickme');
element.addEventListener('click', function(e) {
    e.target.innerHTML = 'Vous avez cliqué !';
}, false);
```

L'objet event

- Pour récupérer l'attribut target de l'objet event sous ie, nous allons utiliser l'attribut srcElement

```
var element = document.getElementById('clickme');
addEvent(element, 'click', function(e) {
    var target = e.target || e.srcElement;
    target.innerHTML = 'Vous avez cliqué !';
});
```

L’objet event

- De la même manière si l’on souhaite récupérer la position de la souris nous devrons aller récupérer les attributs :
  - clientX
  - clientY

- Afin de récupérer la touche que l’utilisateur a pressé c’est l’attribut keycode qu’il faudra utiliser.

Gestion des formulaires

Les propriétés

- La propriété value, cette propriété permet de définir une valeur pour différents éléments d'un formulaire comme les **<input>**, les **<button>**, et d'autres.

```
<input id="text" type="text" size="60" value="Vous n'avez pas le focus !" />
<script>
  var text = document.getElementById('text');
  text.addEventListener('focus', function(e) {
      e.target.value = "Vous avez le focus !";
  }, true);
</script>
```

Les propriétés

- Les propriétés booléenes :
  - Disabled
  - Checked
  - readonly

```
<input id="text" type="text" />
<script>
    var text = document.getElementById('text');
    text.disabled = true;
</script>
```

Exemple checked avec boutton radio

```
<label><input type="radio" name="check" value="1" /> Case n°1</label><br />
<label><input type="radio" name="check" value="2" /> Case n°2</label><br />
<label><input type="radio" name="check" value="3" /> Case n°3</label><br />
<label><input type="radio" name="check" value="4" /> Case n°4</label>
<br /><br />
<input type="button" value="Afficher la case cochée" onclick="check();" />
<script>
  function check() {
      var inputs = document.getElementsByTagName('input'),
          inputsLength = inputs.length;
      for (var i = 0 ; i < inputsLength ; i++) {
          if (inputs[i].type == 'radio' && inputs[i].checked) {
              alert('La case cochée est la n°'+ inputs[i].value);
          }
      }
  }
</script>
```

Listes déroulantes

- 2 propriétés :
  - Option qui donne une liste des éléments dans un tableau.
  - selectedIndex qui donne l'index de l'élément sélectionné.

Exemple d'utilisation

```
<select id="list">
  <option>Sélectionnez votre sexe</option>
  <option>Homme</option>
  <option>Femme</option>
</select>
<script>
  var list = document.getElementById('list');
  list.addEventListener('change', function() {
      alert(list.options[list.selectedIndex].innerHTML);
  }, true);
</script>
```

Les méthodes de l'élément form

- L'élément form possède deux méthodes qui lui sont propre :
  - Submit
  - Reset

- Ces deux méthodes ont le même rôle que les élément input de type submit ou reset.
  - element.submit();
  - element.reset();

Les événements de l'élément form

- L'élément form possède deux événements :
  - Submit
  - Reset

- Attention, envoyer un formulaire avec la méthode submit de JavaScript ne déclenchera pas l'événement submit.

Exemple d'utilisation

```
<form id="myForm">
  <input type="text" value="Entrez un texte" />
  <input type="submit" value="Submit !" />
  <input type="reset" value="Reset !" />
</form>
<script>
  var myForm = document.getElementById('myForm');
  myForm.addEventListener('submit', function(e) {
      alert('Vous avez envoyé le formulaire !);
  }, true);
  myForm.addEventListener('reset', function(e) {
      alert('Vous avez réinitialisé le formulaire !');
  }, true);
</script>
```

La gestion du focus et de la selection

- Il existe trois méthodes qui permettent de gérer le focus d'un élément du formulaire :
  - Focus() Cette méthode va donner le focus à l'élément
  - Blur() Cette méthode va retirer le focus de l'élément
  - Select() Cette méthode va non seulement donner le focus à l'élément mais en plus elle va sélectionner le texte si cela est possible

Interaction avec les feuilles de style css

Editer le style d'un élément

- Pour éditer une propriété css d'un élément nous allons manipuler la propriété style de cet élément.
  - element.style;

Editer le style d’un élément

- Une fois cette propriété récupérée il va falloir modifier une propriété css de notre élément.
  - Par exemple modifier propriété width
    - element.style.width = '150px';
    - element.style.backgroundColor = 'red';

Editer le style d'un élément

- Cependant, il est considéré comme une meilleure pratique de passer par les class et autres id pour modifier le style des éléments:

  - element.id = 'nouvelId';
  - element.className += ' blueBox';
  - element.classList.add('redBox');
  - element.classList.remove('blueBox');

AJAX

Asynchronous Javascript And Xml

AJAX

- La technologie AJAX permet de communiquer avec un serveur de manière asynchrone.
- Le développeur pourra donc envoyer ou récuperer des données depuis le serveur sans forcer un rafraichissement total de la page.

Principe

- Navigation Web classique
  1. Manipulation utilisateur
  2. Envoie requête au serveur
  3. Réception du serveur Web qui effectue des calculs
  4. Retour du résultat sous forme de page Web
  5. Affichage dans le navigateur
- Chaque action utilisateur entraîne la transmission et l'affichage d'une nouvelle page.
- L'utilisateur attend l'arrivée de la réponse pour effectuer une nouvelle opération.

Principe

- Navigation Web classique
  1. Manipulation utilisateur
  2. Envoie requête au serveur
  3. Réception du serveur Web qui effectue des calculs
  4. Retour du résultat sous forme de page Web
  5. Affichage dans le navigateur
- Chaque action utilisateur entraîne la transmission et l'affichage d'une nouvelle page.
- L'utilisateur attend l'arrivée de la réponse pour effectuer une nouvelle opération.

Principe

- Navigation Web AJAX
  1. Demande envoyée au serveur par un programme en JavaScript incorporée dans une page Web
  2. Réception du serveur Web qui effectue des calculs
  3. Le contenu de la page Web est modifié par le programme JavaScript fonction du retour du serveur Web.
- Les requêtes sont asynchrones, l'utilisateur n'attend pas l'arrivée de l'information et peut continuer à effectuer d'autres manipulations.

Historique

- XMLHttpRequest est initialement un composant ActiveX créé par Microsoft en 1998 pour Microsoft Outlook Express. Il est intégré à la norme ECMAScript relative au JavaScript par la suite et sera utilisé par la plupart des navigateurs du marché entre 2002 et 2005.

- Un informaticien américain (Jesse James Garett) introduit le 18 février 2005 le terme AJAX dans un article du site Web Adaptive Path.

Technologies Web utilisées

- La méthode Ajax consiste à utiliser de manière conjointe diverses technologies normalisées ouvertes et disponibles sur la plupart des navigateurs du marché:
  - Le langage de programmation JavaScript,
  - Le XMLHttpRequest, un objet JavaScript permettant d'effectuer la communication entre client et serveur Web,

Technologies Web utilisées

- Le DOM (Document Object Model), une collection d'objets ou chaque objet représente un élément structurel d'un page Web ou d'un document XML,

- Le XML (eXtensible Markup Language), un langage de balisage utilisé pour structurer les informations envoyées par le serveur Web. (De + en + remplacé par JSON)

But d'Ajax

- Diminution des temps de latence
  - Permet de récupérer simplement l'information utile et de gérer au mieux la répartition des ressources entre client et serveur et donc ce qui transite sur le réseau.
- Augmentation de la réactivité de l'application Web
  - Ajax permet de modifier partiellement la page affichée par le navigateur. Il n'est plus nécessaire de recharger une page Web à chaque manipulation de l'utilisateur.
- Apport de nouvelles fonctionnalités

Avantage

- Ajax utilise des technologies présentes « de série » dans les navigateurs Web actuels. Son utilisation ne nécessite donc pas de plugins. C'est un avantage des technologies concurrentes tels qu'Adobe Flex ou bien Microsoft Silverlight.

Inconvénients

- Javascript peut être désactivé par l'utilisateur ou par des logiciels de protection. Sans JavaScript, plus d'Ajax.

- Les moteurs d'indexation utilisés par les moteurs de recherche (Google, Bing...) n'exécutent pas le code JavaScript. Les données dynamiquement récupérées en Ajax ne sont donc par défaut pas indexées.

Comment cela fonctionne?

La communication avec le serveur Web est réalisé *via* l'objet XMLHttpRequest.

- Pour recueillir des informations sur le serveur cet objet dispose de deux méthodes:
  - open: établit une connexion.
  - send: envoie une requête au serveur.

  - Les données fournies par le serveur seront récupérées dans les champs **responseXml** ou **responseText** de l'objet XMLHttpRequest. S'il s'agit d'un fichier XML, il sera lisible dans responseXml par les méthodes de Dom.

  - Noter qu'il faut créer un nouvel objet XMLHttpRequest, pour chaque fichier que vous voulez charger.

Comment cela fonctionne?

La communication étant asynchrone, il est nécessaire d'attendre la disponibilité des données. L'état d'attente est donné par l'attribut **readyState** de l'objet XMLHttpRequest.

Les états de **readyState** sont les suivants (seul le dernier est vraiment utile):
0: non initialisé.
1: connexion établie.
2: requête reçue.
3: réponse en cours.
- 4: terminé.

L'objet XMLHttpRequest

- Il permet d'interagir avec le serveur grâce à ses méthodes et ses attributs.
- Attributs:
  - **readyState** le code d'état passe successivement de 0 à 4 qui signifie "prêt".
  - **status** 200 est ok 404 si la page n'est pas trouvée.
  - **responseText** contient les données chargées dans une chaîne de caractères.
  - **responseXml** contient les données chargées sous forme xml, les méthodes de DOM servent à les extraire.
  - **onreadystatechange** propriété activée par un évènement de changement d'état. On lui assigne une fonction.

L'objet XMLHttpRequest

- Méthodes:
- **open**(mode, url, boolean) mode: type de requête, GET ou POST
- url: l'endroit ou trouver les données, un fichier avec son chemin sur le disque.
- boolean: true (asynchrone) / false (synchrone).

  - **send**("chaine") null pour une commande GET.

Communication Ajax pas à pas

- **Première étape: créer une instance**

```
if (window.XMLHttpRequest)      // Objet de la fenêtre courant {
    xhr = new XMLHttpRequest();      // Firefox, Safari, ...
}
else if (window.ActiveXObject)     // Version Active {
 xhr = new
     ActiveXObject("Microsoft.XMLHTTP"); // Internet Explorer
}
```

- Différent Objet à instancier en fonction du navigateur

Communication Ajax pas à pas

- **Seconde étape: attendre la réponse**
  - Le traitement de la réponse et les traitements qui suivent sont inclus dans une fonction, et la valeur de retour de cette fonction sera assignée à l'attribut **onreadystatechange** de l'objet précédemment créé.

```
xhr.onreadystatechange = function() {
    // instructions de traitement de la réponse
   if (xhr.readyState == 4 && xhr.status == 200) {
        // Recu, OK
   }
   else {
        // Attendre...
   }
};
```

Communication Ajax pas à pas

- **Troisième étape: envoyer la requête au serveur**
- Deux méthodes de XMLHttpRequest sont utilisées:
- **- open**: commande GET ou POST, URL du document, true pour asynchrone.
- **- send**: avec POST seulement, données à envoyer au serveur.

```
xhr.open('GET', 'http://www.xul.fr/fichier.php', true);
xhr.send(null);
```

Communication Ajax pas à pas

- **Quatrième étape: récupérer le données et les insérer dans la page Web**

```
xhr.onreadystatechange = function(){
    // instructions de traitement de la réponse
    if (xhr.readyState==4 && xhr.status==200) {
        document.getElementById("myDiv").innerHTML = xhr.responseText;
    }
};
```

  - Note: Si la réponse correspond à un document XML, la propriété responseXml va contenir cette réponse déjà analysée en tant que tel.

Exercices / Questions / Approfondissements….