DBWEB 2 : Architecture Web Dynamique Licence 1ère Année

Fabrice Lefèvre, CERI, 2021

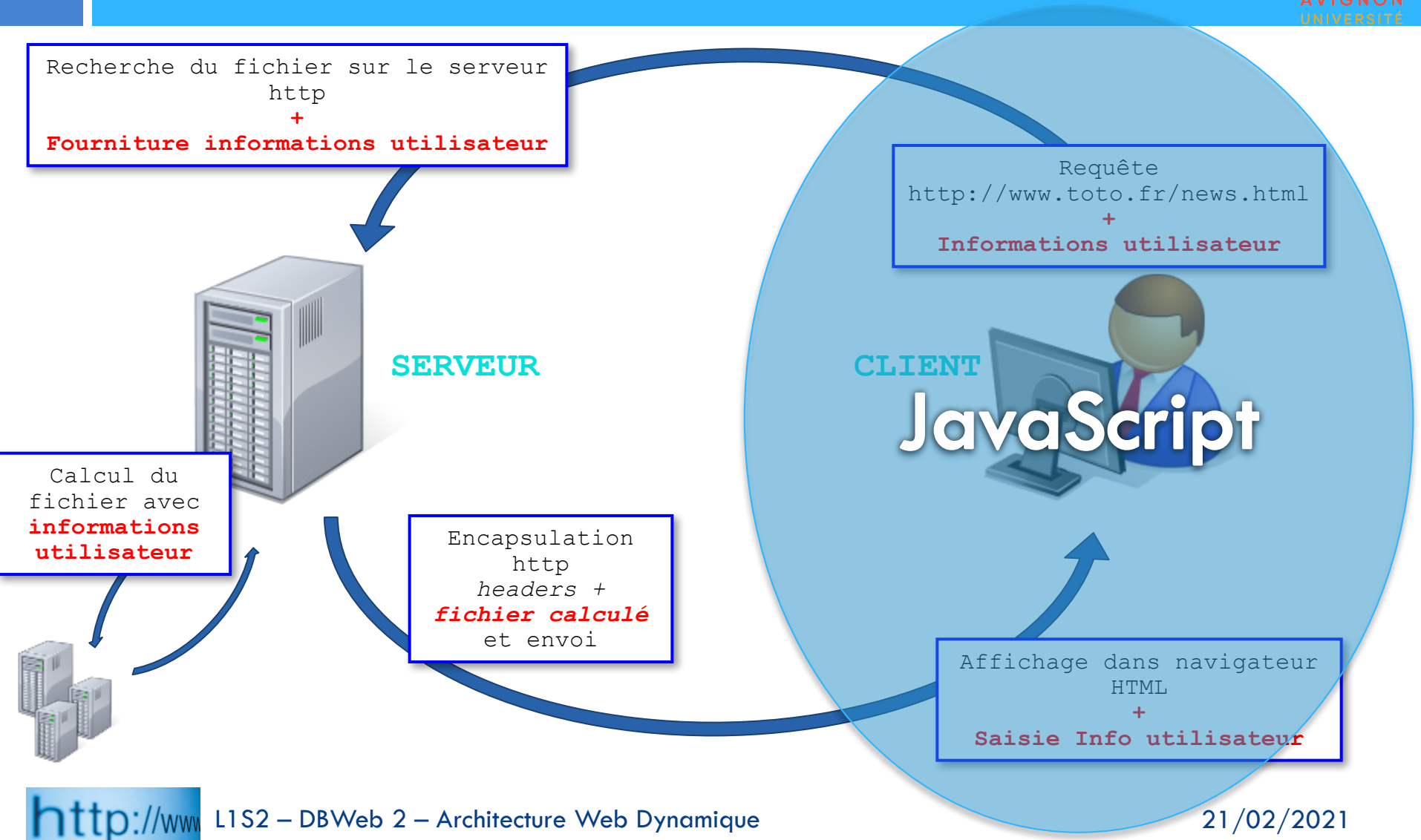
JavaScript

Fabrice Lefèvre

Fabrice.Lefevre@univ-avignon.fr

2021

Interaction dynamique et JS



Quelques exemples

Quelques exemples illustrant les capacités de Javascript :

- Changement de proprietés @javascript_journuit.html
- Menu avec ancres @javascript_ancres.html
- Technique du rollover @javascript_roll-over.html
  @javascript_clignotant.html
- Combinaison formulaire HTML @js_radiocheckbox.html

Attention : comportements différents selon les réglages des navigateurs (eg pour ouverture de fenêtres)

@javascript_journuit.html

AVIGNON UNIVERSITÉ

javascript_journuit.html

```
<html>
<head>
<title>onclick</title>

<script language="javascript">
function change() {
  document.body.style.color = 'yellow'
  document.body.style.backgroundcolor = 'grey'
  document.form1.coul.value = 'gris'
}
</script>
</head>

<body>

<h1>onclick event</h1>
<br /><br />
blabla bla blabla blabla bla blabla bla blabla blabla bla blabla bla blabla blabla bla blabla
bla blabla blabla bla blablabla blabla blabla bla blabla bla blabla blabla bla blabla bla blabla
blabla bla blabla
bla blabla blabla bla blabla bla blabla blabla bla blabla bla blabla blabla bla blabla bla blabla
blabla bla blabla
<br /><br />

<form name="form1">
    <input type="text" name="coul">
    <br /><br />
    <input type="button" name="int" value="cliquez ici..." onclick="change()">
</form>
</body>
</html>
```

@javascript_clignotant.html

javascript_clignotant.html

```
<html>
<head>
<title>blink</title>
<script>
function changeimage() {
    setInterval(function(){
        if (document.theatre.src.search("images/theatre-vert.gif") != -1) {
            document.theatre.src='images/theatre-rouge.gif';
        } else {
            document.theatre.src='images/theatre-vert.gif';
        }
    }, 1000);
}
</script>
</head>

<body>
<h1>Défilement d'images automatique</h1>
<img name="theatre" src="images/theatre-rouge.gif" border="0" onclick="changeimage()">
</body>
</html>

```

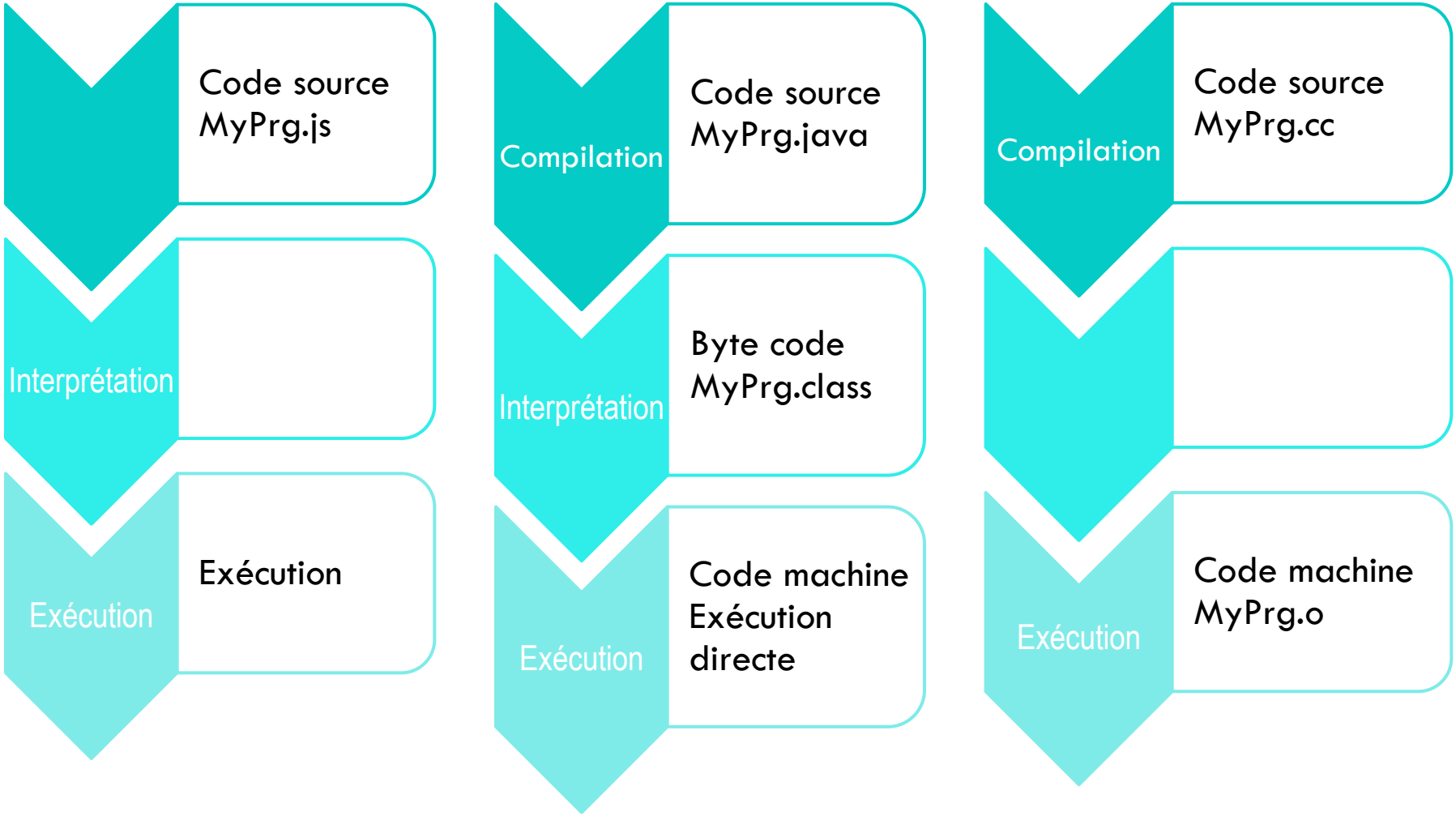
Caractéristiques de JavaScript

- Langage de script : donc interprété
- Profondément lié au langage HTML :
  - appel de JavaScript directement dans les balises HTML
  - totalité des balises HTML accessibles en JavaScript pour des manipulations diverses
    - Grâce au DOM !
- Nom trompeur : syntaxe comparable à Java mais très simplifiée
- Modélisation objet light :
  - notion de classes très allégée
  - pas d'héritage entre classes

Origine

- Développé par Netscape en association avec Sun Microsystem en 1995
- D'autres langages de scripts disponibles :
  - ActionScript (Adobe) ou JScript (Microsoft)
- JavaScript standardisé par l'ECMA (European Computer Manufactures Association)
  - nom officiel : ECMAScript
- Spécification
  - http://www.ecma-international.org/publications/standards/Ecma-262.htm

Langages interprétés et compilés



JavaScript : limites

- Code JavaScript interprété par le navigateur
- Possibilité de désactiver cette interprétation au niveau des navigateurs ➔ code JavaScript totalement ignoré par le navigateur
- JavaScript ne doit pas supporter la responsabilité de sécuriser les interactions client/serveur
  - ex de la vérification de la syntaxe des adresses e-mail : si réalisée uniquement par du code JavaScript, possibilité de désactiver ce dernier et de renvoyer une fausse adresse pour « hacker » le serveur
  - mail toto@toto.fr ; cat /etc/passwd
- Tests de validation et sécurité des données peuvent être redondants sur le client et le serveur !

JavaScript : pros and cons

- Avantages
  - bonnes possibilités d'interactivité
  - langage simple
  - facile à déployer (un simple éditeur texte et un navigateur qui le supporte)

- Inconvénients
  - se désactive par l'utilisateur
  - fonctionnement (encore) inégal entre les navigateurs
  - script ouvert. Le code que vous écrivez étant visible par tous, vous ne devez pas écrire de code sensible, par exemple la méthode de validation d'un examen

Syntaxe (1)

- Très similaire au java mais pas strictement :
  - sensible à la casse (majuscule/minuscule)
  - ";" non obligatoire en fin d'instruction (mais recommandé !)
  - commentaires : // ou /* commentaires */
- Variables
  - déclaration non obligatoire (mais recommandée !)
    - var ma_variable ;
  - Pas de type défini lors des déclarations, mais lors des affectations :
    - Nombre
    - Booléen
    - String
    - Objet (ex: array)
    - Fonction

Syntaxe (2)

- Instructions conditionnelles : if … else, switch … case
- Itérations (boucles) : for, for … in, while, do … while
- Gestion des fonctions :

```
function nom_fonction(par 1, par2 …) {
        par1 = par2;
        return par1;
}
```

- Gestion des objets :
  - appel au constructeur : variable = new Class()
  - accès aux propriétés par : variable.la_propriete
  - appel aux méthodes par : variable.la_methode(…)

DBWEB 2 : Architecture Web Dynamique Licence 1ère Année

Fabrice Lefèvre, CERI, 2021

JavaScript intégration

Fabrice Lefèvre

Fabrice.Lefevre@univ-avignon.fr

2021

Insertion de codes JavaScript

- Différentes manières d'inclure du JavaScript dans une page HTML :
  - Au travers des balises d'insertion <SCRIPT> </SCRIPT>
  - Appel directement de fonctions JavaScript dans les balises HTML à l'aide des attributs « capteurs d'événements » (event handler)
  - Par le pseudo protocole "javascript" avec attribut href

Insertion par <SCRIPT></SCRIPT> (1)

- Balise <SCRIPT></SCRIPT> ➔ insertion de code destiné à être exécuté par le navigateur
  - JavaScript ou autre
  - <SCRIPT language="nom_langage" src="fichier.js" > code </SCRIPT>
    - langage : JavaScript, JavaScript1.1, JavaScript1.2 (optionnel)
    - src : fichier pouvant contenir le code JavaScript à exécuter (ex : ensemble de fonctions).
    - extension : .js (configuration du serveur HTTP)
    - code : déclaration de fonctions ou de commandes JavaScript à exécuter
- Placement dans le fichier HTML :
  - on fait comme on veut
  - mais bonnes pratiques :
    - <HEAD>…</HEAD> → déclaration de fonctions stockées jusqu'à leur appel
    - <BODY>…</BODY> → exécution des commandes dès le chargement de la page

Insertion par <SCRIPT></SCRIPT> (2)

```
<HTML>
<HEAD></HEAD>
<BODY>
...
  <SCRIPT>
    /* Permet d'afficher ce message sur la page HTML */
    document.write('Voici un appel à du code JavaScript directement
exécuté dès le chargement de cette page!');
  </SCRIPT>
...
</BODY>
</HTML>
```

Insertion par <SCRIPT></SCRIPT> (2)

Insertion de code entre les balises <HEAD>…</HEAD>

```
<HTML>
<HEAD>
  <SCRIPT>
    function afficheDial(message) {
      alert(message);
    }
  </SCRIPT>
</HEAD>
<BODY>
...
  <SCRIPT>
    afficheDial('ex de fonctions !');
  </SCRIPT>
...
</BODY>
```

Insertion par <SCRIPT></SCRIPT> (3)

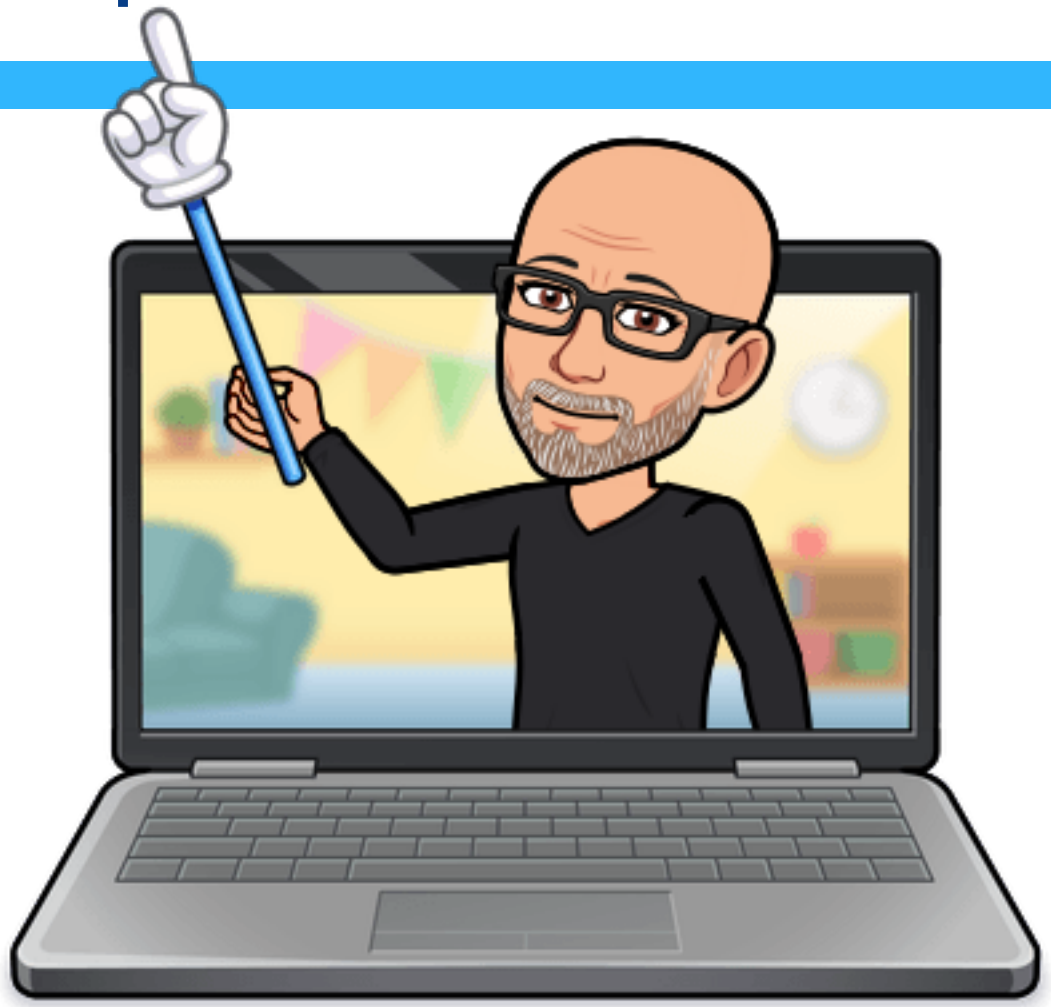
```
<HTML>
<HEAD>
  <SCRIPT SRC="my_fic.js"> </SCRIPT>
</HEAD>
<BODY>
...
  <SCRIPT>
    afficheDial('ex de fonctions !');
  </SCRIPT>
...
</BODY>
</HTML>
```

Insertion de code entre les balises <HEAD>...</HEAD> au travers d'un fichier externe

Fichier Javascript my_fic.js

```
function afficheDial(message){
   alert(message);
}
```

En pratique



Insertion par pseudo-protocole JS (1)

- Principe : remplace le protocole (http, ftp...) dans une URL par un appel de fonctions JavaScript

```
<A href="JavaScript:alert('boite de dialogue')">
    cliquer ici pour ouvrir une boîte de dialogue
</A>
```

- Conseillé pour l'appel de fonctions uniquement !
- Disponible uniquement pour la balise <A> avec IE

Insertion par pseudo-protocole JS (2)

```
<HTML>
<HEAD>
  <SCRIPT>
    function afficheDial(message){
      alert(message);
    }
  </SCRIPT>
</HEAD>
<BODY>
...
  <A href="JavaScript:afficheDial('voici un exemple d\'appel de
fonctions par le pseudo protocole JS!')">Cliquer ici pour ouvrir une
boîte de dialogue</A>
...
</BODY>
</HTML>
```

Appels directs : événements HTML

- Des événements associés directement aux balises HTML
  @javascript_onchange.htm
  - un clic de souris sur un élément
  - sélection d'un champ d'un formulaire
  - le chargement d'une page
  - ...
- Permettent d'intercepter les actions utilisateurs (mais pas que)
- Permettent d'associer les scripts à exécuter lorsque ces événements sont détectés
- Accès par un attribut portant le nom de l'évènement

Connecter JavaScript et HTML

- Gestion des événements dans partie suivante
- Mais comment “accéder” aux parties du document HTML depuis un script JavaScript sans forcément passer par un évènement ?

le DOM !

DOM (1)

- Modèle Objet de Document, (ou DOM, Document Object Model) est un standard du W3C permettant l'accès aux documents HTML et XML.
- Permet principalement deux choses au développeur/concepteur :
  - fournit une représentation structurée du document
  - codifie la manière dont un script peut accéder à cette structure à travers les objets qu'elle contient
    - au niveau des propriétés
    - et des méthodes
- Donc, en pratique,
  ➔Permet de lier un document web en HTML à un langage de programmation ou de script

DOM (2)

- Il s'agit d'une interface qui permet de standardiser les moyens d'accès à une page Web à travers ses éléments.
  - ≠ application particulière,
  - ≠ produit spécifique
  - ≠ présentation propriétaire d'une page Web.
- Toutes les propriétés et méthodes et tous les événements disponibles au développeur pour manipuler et créer des pages dynamiques sont organisés sous forme de hiérarchies d'objets
  - le document en entier est un nœud **document**
  - chaque élément HTML est un nœud **élément**
  - chaque attribut HTML est un nœud **attribut**
  - le texte dans les éléments HTML sont des nœuds **texte**
  - les commentaires sont des nœuds **commentaires**
  - …

Interlude…

- Peut-être utile de définir rapidement la notion d'objet en informatique ?

- Lorsqu'un objet du réel est modélisé en informatique, il se résume rarement à un type simple de variable. Comment représenter un tel objet de façon efficace ?
- Programmation Orientée Objet : Ensemble de variables descriptives regroupées, et associées aux fonctions permettant leur manipulation (par exemple création et destruction : constructeur et destructeur)
  - propriétés
  - méthodes
- Classe → type informatique structuré
- Instance d'une classe → objet
- Mode de déclaration et de désignation simple (notation pointée) :
  - MaClasse monObjet
  - monObject.maPropriete
  - monObjet.maMethode(arguments)
- Encapsulation, héritage, polymorphisme…

Interlude…

- Ecrire le code décrivant une classe COMPTE comprenant les attributs suivants :
  - Numéro de Compte : Entier
  - Nom du Titulaire : Chaine(30)
  - Solde : Réel

Interlude…

- Déclaration de la classe

compte=classe

privé :

// cette partie sera inaccessible en dehors de la classe

numéro : entier
nom : chaine(30)
solde : réel

public :

// cette partie sera accessible et constitue l'interface de la classe

procédure init(pnum:entier ; pnom:chaine(30))
procédure créditer(montant : réel)
procédure débiter(montant : réel)
fonction solde() : réel

Interlude…

- Description des méthodes

```
procédure compte.init(pnum:entier ; pnom:chaine(30) )
début
    numéro:=pnum
    nom:=pnom
    Solde:=0
fin

procédure compte.créditer(montant : réel)
début
    solde:=solde+montant
fin

procédure compte.débiter(montant : réel)
début
    solde:=solde - montant
fin

fonction compte.solde() : réel
début
    retourner(solde)
fin
```

Interlude

- Programme Exemple

```
// déclaration d'un pointeur, pas d'instanciation
var c1 : pointeur de compte

Début
    c1=nouveau(compte)  //  instanciation d'un compte
    c1.init(1,"Dupont")
    c1.créditer(1000)
    c1.débiter(500)
    afficher(c1.solde)      //interdit : solde est en partie privée !
    afficher(c1.solde())    // affiche 500
    détruire(pc)             // libération de la mémoire allouée
Fin
```

DOM (2)

- Il s'agit d'une interface qui permet de standardiser les moyens d'accès à une page Web à travers ses éléments.
  - ≠ application particulière,
  - ≠ produit spécifique
  - ≠ présentation propriétaire d'une page Web.
- Toutes les propriétés et méthodes et tous les événements disponibles au développeur pour manipuler et créer des pages dynamiques sont organisés sous forme de hiérarchies d'objets
  - le document en entier est un nœud **document**
  - chaque élément HTML est un nœud **élément**
  - chaque attribut HTML est un nœud **attribut**
  - le texte dans les éléments HTML sont des nœuds **texte**
  - les commentaires sont des nœuds **commentaires**
  - …

Interprétation DOM (1)

```
<TABLE>
<TBODY>
<TR>
<TD>ShadyGrove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the river, Charlie</TD>
<TD>Dorian</TD>
</TR>
</TBODY>
</TABLE>
```

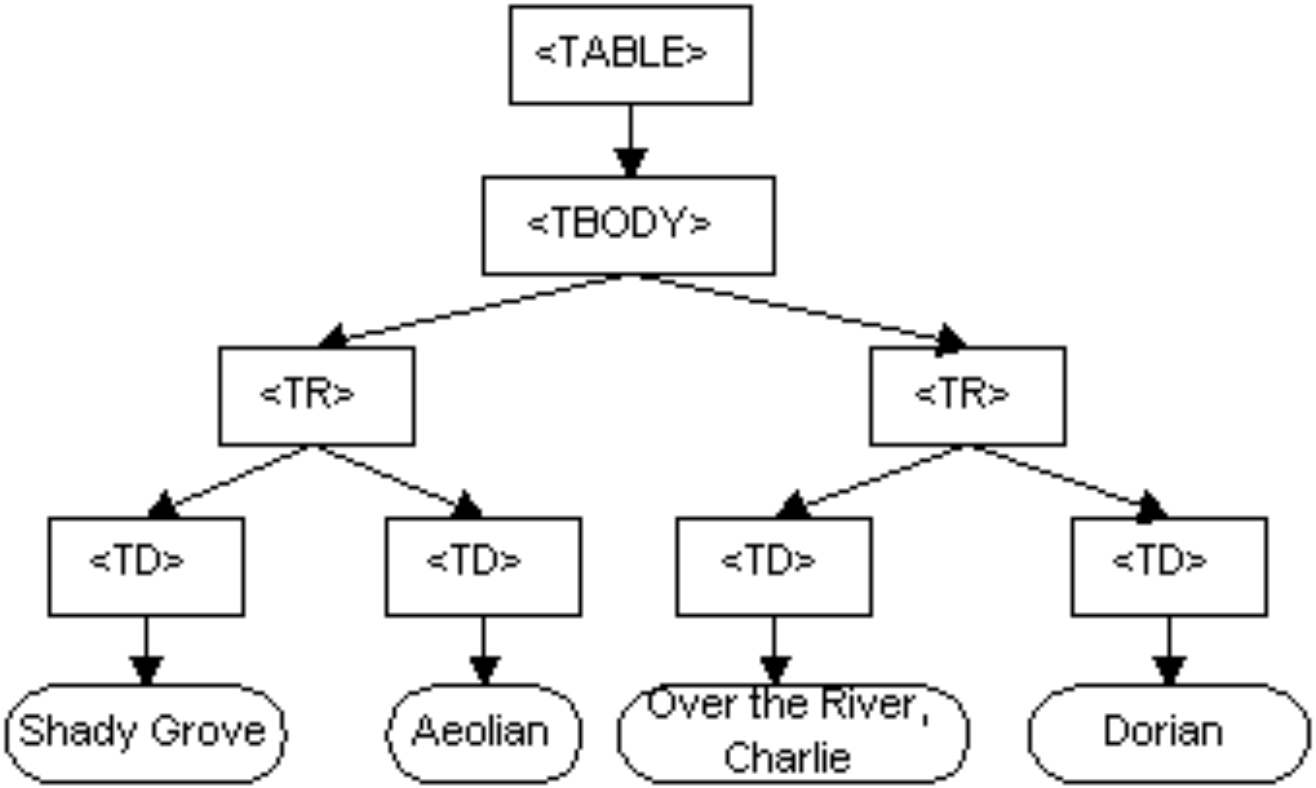
Interprétation DOM (1bis)

```
<TABLE>
  <TBODY>
    <TR>
      <TD>ShadyGrove</TD>
      <TD>Aeolian</TD>
    </TR>
    <TR>
      <TD>Over the river, Charlie</TD>
      <TD>Dorian</TD>
    </TR>
  </TBODY>
</TABLE>
```

Indentation rend lisible le code :

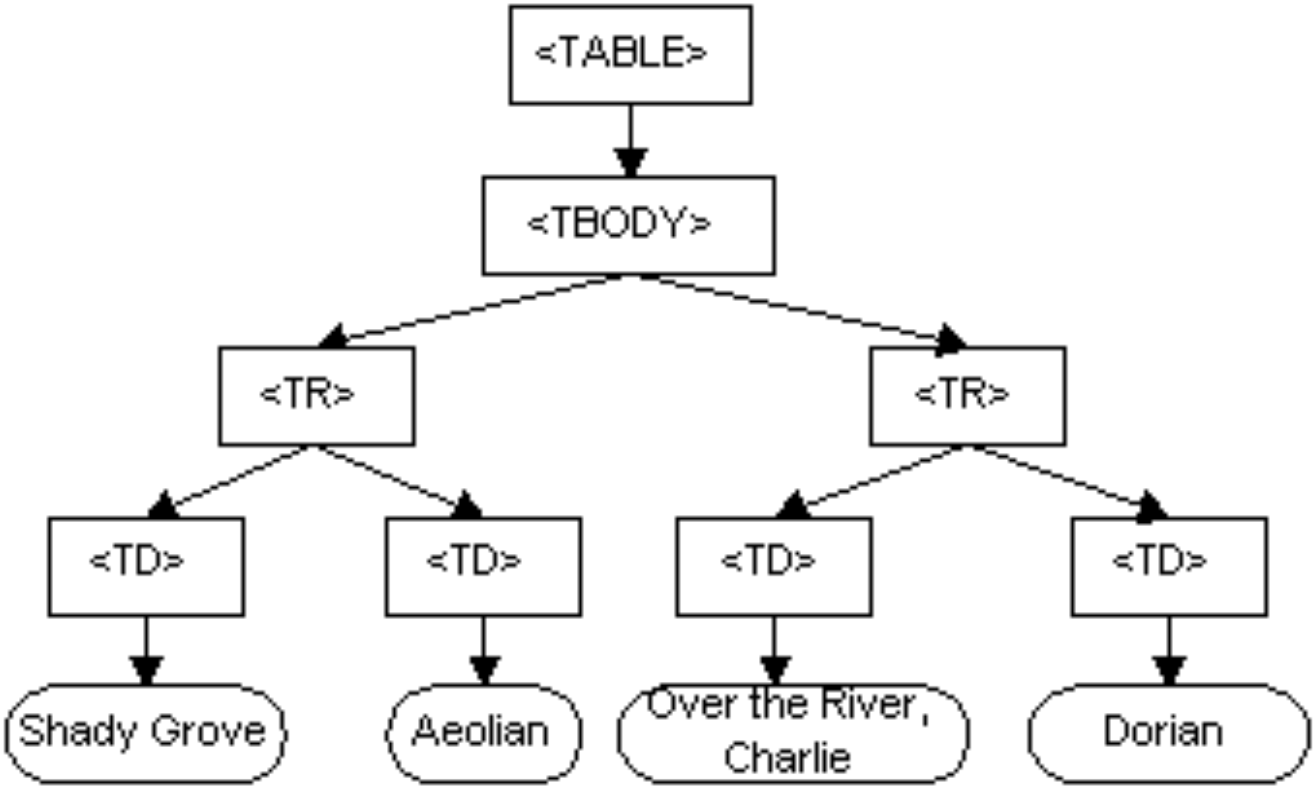
* évite les erreurs
* facilite analyse
* à faire dès l'écriture

Interprétation DOM (2)



- accéder à la deuxième cellule de la deuxième ligne du tableau (Dorian) ?

Interprétation DOM (3)

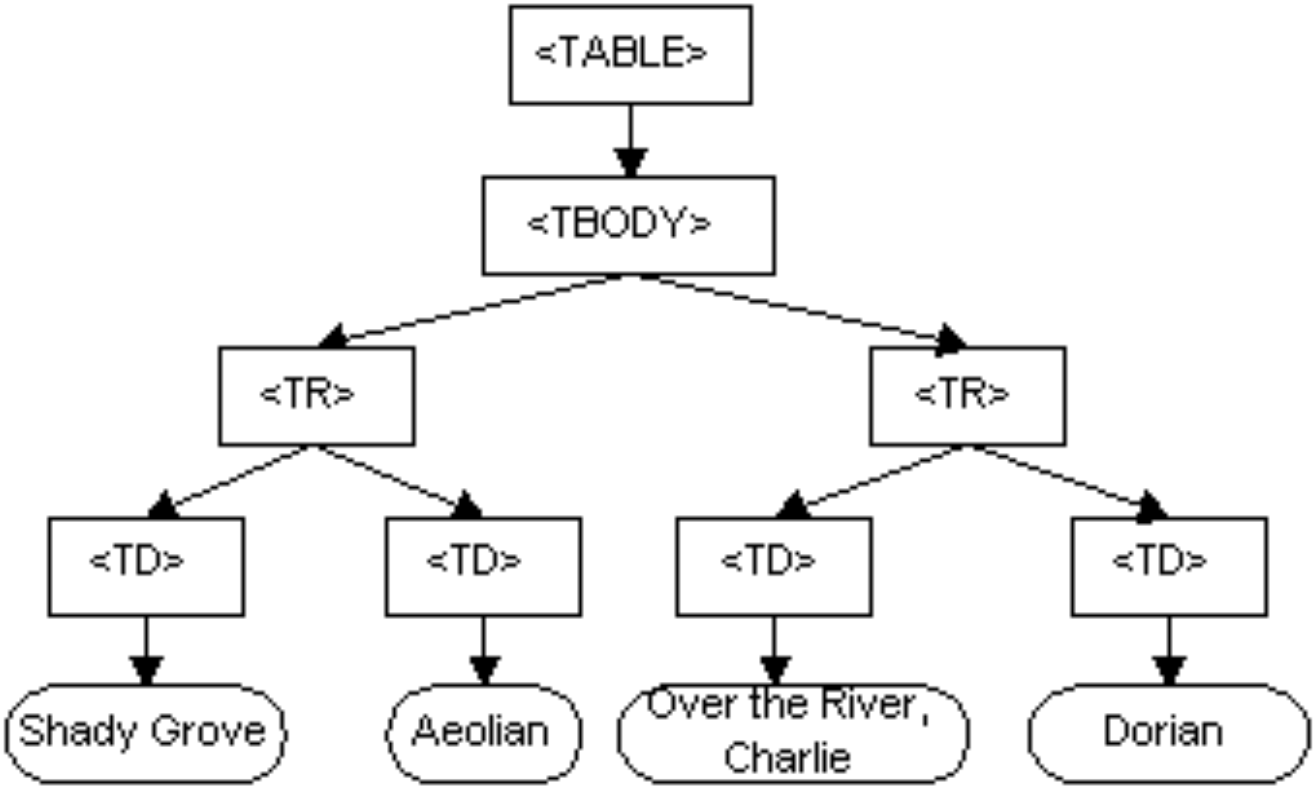


- document.table[0].tbody[0].tr[1].td[1]

Interprétation DOM (1bis)

```
<TABLE>
  <TBODY>
    <TR>
      <TD id='cell1' >ShadyGrove</TD>
      <TD id='cell2' >Aeolian</TD>
    </TR>
    <TR>
      <TD id='cell3' >Over the river, Charlie</TD>
      <TD id='cell4' >Dorian</TD>
    </TR>
  </TBODY>
</TABLE>
```

Interprétation DOM (5)



- document.getElementById(”cell4").innerText

Objets standards du DOM

- DOM ➔ bibliothèques d'objets standards, instanciés automatiquement par le navigateur
- Quelques propriétés standards du DOM : @javascript_innerHTML.htm
  - x.innerText ou x.innerHTML – valeur textuelle de l'objet x
  - x.nodeName – nom de l'objet x
  - x.nodeValue – valeur de l'objet x
  - x.attributes – les nœuds attributs de x
  - x.parentNode – le nœud parent de x
  - x.childNodes – les enfants de x
- Méthodes standards du DOM :
  - x.getElementsByTagName(name) – accède à tous les éléments de type name
  - x.getElementById(id) - accède à l'élément id
  - x.removeChild(node) – enlève un fils à x
  - x.appendChild(node) - insert un fils à x

HTML Dynamique

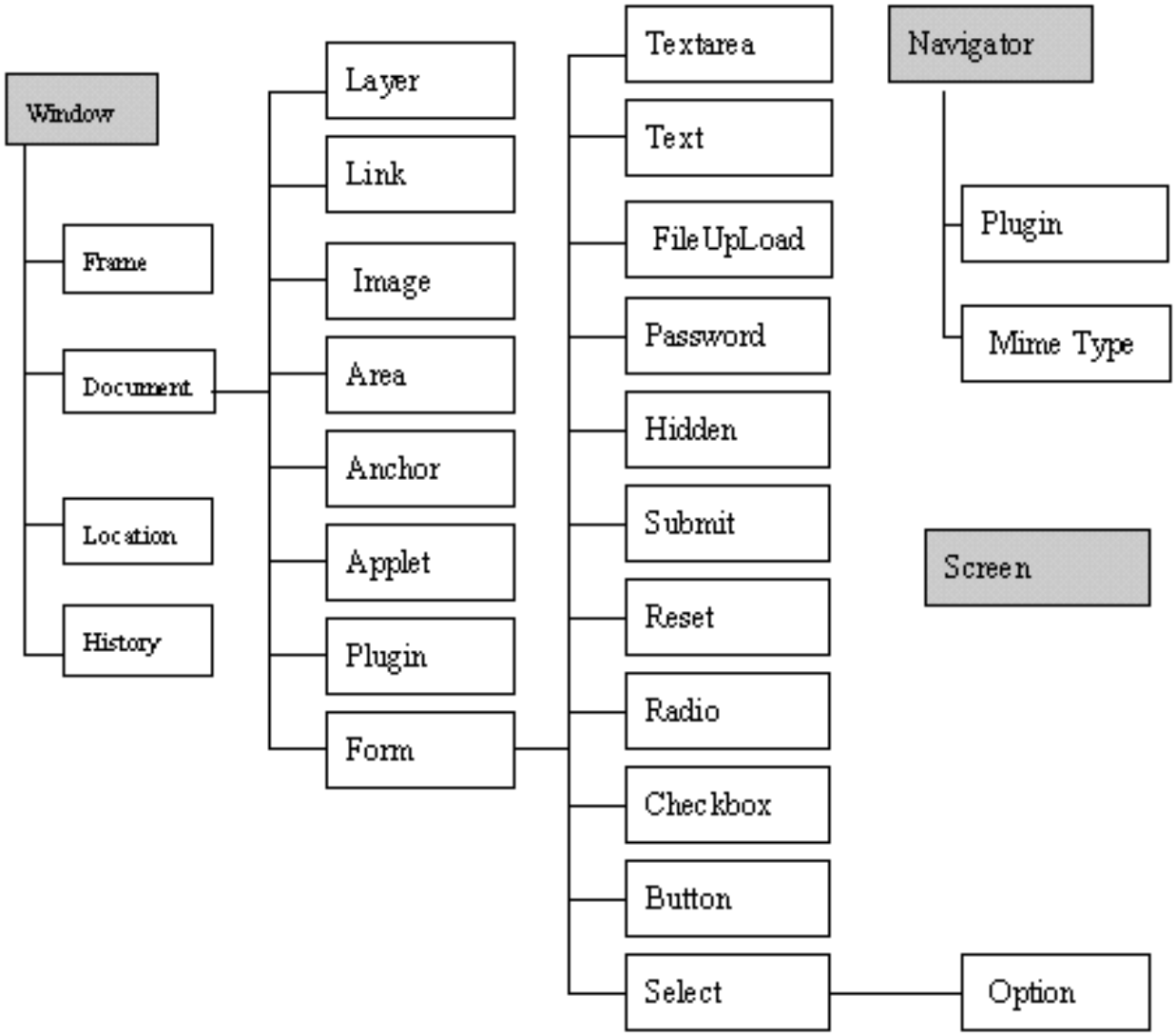
- DHTML = standard de développement qui repose sur :
  - une vue orientée objet de la page web et de ses éléments.
    - chaque élément « <tag id="id_du_tag"> blabla </tag> » est accessible de manière univoque
      - par son type et sa position dans le document ou
      - par un identifiant unique
    - la structure entière de la page peut être représentée sous la forme d'une arborescence, permettant l'accès direct (et la modification) à chacun de ses éléments.
    - formalisée par le Document Object Model, DOM
  - le recours systématique aux feuilles de style CSS pour les modifications précise de l'apparence de chacun des éléments
  - le recours à des langages de script pour la gestion des événements utilisateurs (VBscript, JavaScript...)

DHTML = HTML + DOM + CSS + scripts

DOM et objets du navigateur

- Accès par JavaScript à 3 objets liés au navigateur et directement instanciés par JavaScript :
  - classe Screen : accès aux informations liées à l'affichage de la machine (largeur, hauteur, nombre de couleurs, …)
  - classe Navigator : accès aux informations propres au navigateur (nom, version, …)
  - classe Window : accès en lecture et écriture à tous les objets HTML (formulaires, liens, champs de saisie, images, …) par le biais de méthodes et propriétés pour interagir avec les documents HTML

Hiérarchie interne des objets



Objet Navigateur

```
<HTML>                                          @javascript_navigateur.html
<HEAD><TITLE>Objet navigateur</TITLE></HEAD>
<BODY>
  <H1>Informations diverses sur le navigateur</H1>
  <TABLE BORDER="1" CELLPADDING="10">
    <SCRIPT LANGUAGE="JavaScript">
      document.write('<TR><TH ALIGN="left">Code</TH><TD>');
      document.write(navigator.appCodeName);
      document.writeln('</TD></TR>');
      document.write('<TR><TH ALIGN="left">Nom</TH><TD>');
      document.write(navigator.appName);
      document.writeln('</TD></TR>');
    </SCRIPT>
  </TABLE>
</BODY>
</HTML>
```

Classe window (1)

- Instanciation automatique d'un objet de la classe window pour chaque fenêtre ou frame (zone) :
- Accès aux URLs visitées dans une zone (history)
- Caractéristiques de l'URL (location)
- Caractéristiques du document HTML (document) avec accès à tous les objets HTML du document:
  - exemple d'accès au premier formulaire de la page, contenant un input de type texte (name="nom")
    - document.forms[0].nom.value='mon_nom'
  - exemple d'accès au premier formulaire par son nom (name="mon_form")
    - document.mon_form.nom.value='mon_nom'

Classe window (2)

- Pour chaque objet, accès à un ensemble :
  - de propriétés (directement liées aux attributs des objets ou en supplément)
    - window.location, .history, .document, …
    - window.document.forms, .images, .title, …
    - …
  - de méthodes
    - window.alert(), .close(), .moveTo(), …
    - window.document.close(), .write(), .open(), …
    - window.document.forms[i].submit(), .reset()
    - …
- Méthodes et propriétés communes à tous les objets ou propres à chacun
  - attention : certaines méthodes sont possibles (par héritage) mais… déconseillées

Classe window (3)

```
<HTML>                                  @javascript_window.htm
<HEAD>
  <script src="my_fic.js"></script>
</HEAD>
<BODY>
  <FORM name="mon_form">
    <INPUT type="text” name="nom">
    <INPUT type="text” name="prenom">
  </FORM>
  <SCRIPT>valeurParDefaut()</SCRIPT>
</BODY>
</HTML>

FichierJavascript my_fic.js :
function valeurParDefaut(){
   window.document.mon_form.nom.value="Campion";
   window.document.mon_form.prenom.value="Jane";
   window.document.mon_form.nom.focus();
}
```

Classes prédéfinies JS

- Date :
  - var nom_objet = new Date()
  - nom_objet.getDate(), nom_objet.getDay()...
- Image :
  - chargée mais pas affichée, doit venir remplacer le contenu d'un <IMG>
  - src, heigth, width, complete
- Option :
  - pour les <SELECT> des formulaires
  - crée des options supplémentaires qui peuvent ensuite être associées à des select
- Math :
  - 1 seule instance par page
  - accès aux fonctions : abs(), ceil(), random(), sqrt()...
- String et Array :
  - new optionnel
  - accès aux fonctions : replace(), indexOf(), search()... et concat(), sort(), join()...

JavaScript et CSS

- Possibilité de redéfinir des élément de style CSS directement par le code JavaScript
- Principe : utilisation de la propriété « style » des objets associés aux balises HTML :

```
var e = document.getElementById("divMenu");
e.style.visibility="hidden“;
```

- Précaution : usage des "-" interdit dans les scripts JavaScript
  - certaines propriétés de style doivent être transformées : text-align → textAlign
  - règle : le "-" est omis et la lettre suivante est mise en majuscule (CamelCase)

Conclusion

- Une dimension supplémentaire au HTML
- La majorité des objets HTML et des styles CSS modifiables en direct par l'utilisateur :
  - !!! DHTML !!!
- Attention à l'ergonomie, à la sécurité et à la charge CPU du client
- Standardisation du DOM simplifie beaucoup de choses...

- Et au cas où … <NOSCRIPT>...</NOSCRIPT>

DBWEB 2 : Architecture Web Dynamique Licence 1ère Année

Fabrice Lefèvre, CERI, 2021

JavaScript évènements

Fabrice Lefèvre

Fabrice.Lefevre@univ-avignon.fr

2021

Gestion des événements (1)

- Objet – événement ➔ action JavaScript
- Sous forme d'une série d'instructions ou d'une fonction
- Chaque objet possède une propriété par évènement possible qui pointe sur une séquence d'instructions JavaScript
- Manière de faire :
  - En modifiant directement la **propriété associée à l'évènement** de l'objet :
    - objet.onEvent=alert("Il s'est passe quelque chose !");
    - ex : window.document.links[0].onClick=alert("Un lien cliqué");

Gestion des événements (2)

- Attributs de gestion des boutons de la souris :
  - onClick et onDblClick permettent la prise en compte d'un clic simple ou double,
  - onMouseDown et onMouseUp détectent si un bouton de la souris a été enfoncé ou relaché.
- Attributs de gestion des déplacements de la souris :
  - onMouseover et onMouseout détectent si la souris passe sur l'élément courant, ou bien le quitte.
- Attributs de gestion du clavier :
  - onKeyPress, onKeyDown et onKeyUp détectent si l'utilisateur a appuyé puis relaché, simplement appuyé, ou relaché une touche du clavier.

Gestion des événements (4)

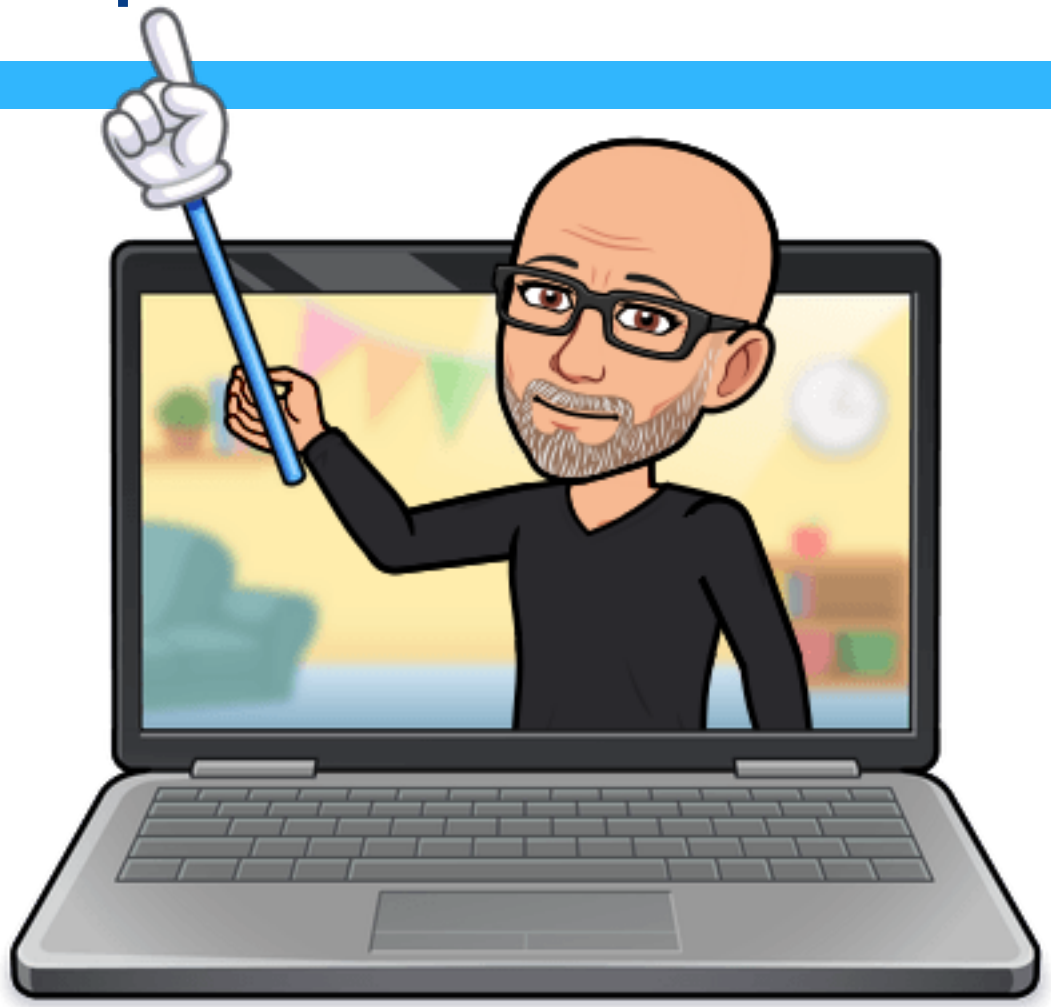
onFocus/ Blur	Récupération ou perte de focus (=passage de la souris)	onLoad/ Abort	Chargement d'une image ou son interruption
on[Dbl]Click	Clic de la souris	onUnload	Déchargement d'un document
onMouseDown/ Move/ Out/ Up	Bouton souris enfoncé / mouvement de la souris / retrait de la souris d'un objet / souris sur un objet / bouton relâché	onChange/ Reset/ Select	Changement/annulation de valeurs saisies/Sélection de texte dans un champ de saisie
OnDragDrop	Drag & Drop d'un objet	onSubmit	Soumission d'un formulaire
onMove/ Resize	Déplacement/redimensionnement d'une fenêtre	onError	Erreur chargement image ou exécution JS
onKeyDown/ Press/ Up/	Touche enfoncée/pressée/relâchée		

Gestion des événements (5)

```
<HTML>
<HEAD>
</HEAD>
<BODY>
...
  <INPUT type="button" id="myButton" />
...
  <SCRIPT>
    function afficheDial(message){alert(message);}
    document.getElementById("myButton").onclick=function(){
afficheDial("exemple 1") };
  </SCRIPT>
...
</BODY>
</HTML>
```

Utilisation du DOM

En pratique



Gestion des événements (1BIS)

- Objet – événement ➔ action JavaScript
- Sous forme d'une série d'instructions ou d'une fonction
- Chaque objet possède une propriété par évènement possible qui pointe sur une séquence d'instructions JavaScript
- **Deux** manières de faire :
  - en modifiant directement les propriétés de l'objet :
    - objet.onevent=fonction_js;
    - ex : window.document.links[0].onClick=alert("Un lien cliqué");
  - directement dans la balise :
    - <balise onEvent="code_js">
    - ex : Click ici

Gestion des événements (5BIS)

```
<HTML>
<HEAD>
  <SCRIPT>
    function afficheDial(message){
      alert(message);
     }
  </SCRIPT>
</HEAD>
<BODY>
...
  <INPUT type="button” onClick="afficheDial('voici un exemple
d\'appel de fonctions definie dans le header!')" />
...
</BODY>
</HTML
```

Insertion dans les balises HTML

Gestion des événements (6)

AVIGNON UNIVERSITÉ

@javascript_events.htm

```
<HTML>
<HEAD></HEAD>
<BODY>
  <p onMouseover="this.style.color='blue'”
onMouseout="this.style.color='black'”>Ceci est un texte dont la
couleur va changer au passage de la souris.</p>

  <p style="position:absolute; left:10px; top:230px; width:400px"
onMouseover="this.style.left='100px';this.style.top='300px';this.styl
e.width='600px';this.style.textAlign='center'"
onMouseout="this.style.left='10px';this.style.top='230px';this.style.
width='400px';this.style.textAlign='left'">Exemple de texte à
géométrie variable.</p>
</BODY>
</HTML>
```

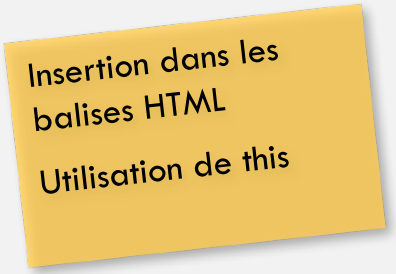
onMouseover peut être remplacé par la pseudo-classe CSS hover

Insertion dans les balises HTML (1)

- Principe : ajout d'attributs associés à du code JavaScript directement dans les balises HTML
  - <balise onEvénement="code JavaScript">... </balise>
  - Gestion d'événements liés aux actions de l'utilisateur:
    - clic de souris (ex "onClick")
    - modification de valeurs saisies par l'utilisateur dans un champ
    - soumission d'un formulaire
    - ...
- Conseillé pour l'appel de fonctions uniquement !
- Astuce : dans ce cas l'objet "balise" sur laquelle on se trouve est accessible dans le code JavaScript par la variable "this"

Insertion dans les balises HTML (2)

```
<HTML>
<HEAD>
  <SCRIPT>
    function afficheDial(message){
      alert(message);
    }
  </SCRIPT>
</HEAD>
<BODY>
...
  <INPUT type="button" value="avant" onClick="this.value='apres';
afficheDial('voici un exemple d\'appel de fonctions definie dans le
header!')" />
...
</BODY>
</HTML
```



DBWEB 2 : Architecture Web Dynamique Licence 1ère Année

Fabrice Lefèvre, CERI, 2021

JavaScript exemples En pratique

Fabrice Lefèvre

Fabrice.Lefevre@univ-avignon.fr

2021

Quelques exemples

Quelques exemples illustrant les capacités de Javascript :

- Changement de proprietés @javascript_journuit.html
- Menu avec ancres @javascript_ancres.html
- Technique du rollover @javascript_roll-over.html
  @javascript_clignotant.html
- Combinaison formulaire HTML @js_radiocheckbox.html

Attention : comportements différents selon les réglages des navigateurs (eg pour ouverture de fenêtres)

Conseils méthodologiques

- Pour cette année préférer partir d'exemples tout fait plutôt que d'écrire complétement une solution
  - beaucoup de détails du langage sont encore complexes à ce niveau (malgré apparente simplicité)
- Recourir aux outils de "debugging" disponibles sur les navigateurs
  - Par exemple Firebug avec Firefox
  - Pas simples mais permettent de débloquer quelques situations