

Plan du cours

- Le Modèle OSI**
- Internet et le modèle OSI**
- Principe TCP/IP**
- Principe DNS**
- Routage et adressage IP**
- Internet - Intranet - Extranet**
- Principes administratifs de l'Internet**
- Les Données : HTML, SQL, XML, ...**
- Echange de données (et systèmes de gestion)**
- Langages et programmation**
- Plateformes logicielles**
- La gestion d'un projet Web**
- Annexes**

Définition

- Langage de type «script».
- Utilisation de fonctions prédéfinies par un système, appelé «noyau» («core»).
- Programmation facilitée par un environnement pré-existant.
- Exemples de langages de type script :
 - Javascript / Navigateur (Netscape, Mozilla, Safari, ...)
 - JScript / Microsoft Internet Explorer
 - PHP / PHP+webserver (Apache, IIS)
 - ActionScript / Macromedia Flash
 - Lingo / Macromedia Director

Origines

- Création en 1995 par Brendan Eich pour Netscape sous le nom «Livescript».
- Accords pour la mutualisation de développement entre Netscape et SUN.
- Le LiveScript prend l'apparence du langage Java™ de SUN et devient JavaScript.
- JavaScript 1.0 est implémenté sur Netscape Navigator 2.0
- Netscape et SUN annonce JavaScript le 4 décembre 1995 comme un langage
 - conçu pour créer des applications centrées-réseau
 - complémentaire et intégré à Java™
 - complémentaire et intégré à HTML
 - ouvert et multi-plateforme

Fonctionnalités

- Javascript est un langage interprété par le navigateur client.
- Il ajoute des fonctionnalités interactives aux pages HTML.
- Il permet au navigateur de réaliser un traitement local de données sans solliciter le serveur.

Inconvénients

- Son exécution dépend du navigateur -> problèmes de compatibilité entre les navigateurs.
- Standard contesté (enjeu commercial) -> rivalité Microsoft.
- Jusqu'à aujourd'hui, la principale difficulté de mise en œuvre est la compatibilité.

Néanmoins, standard devenu incontesté et largement utilisé.

Normalisation

- Javascript est l'objet d'une norme éditée par ECMA.
- ECMA : European Computer Manufacturers Association - Organisme pour la normalisation des technologies de l'information et de la communication. Association d'industriels basée à Genève. Fondée en 1961. L'ECMA définit des normes dans des domaines divers (support de données optiques, CD, DVD, communications sans fil, encodages, langages...)
- www.ecma-international.org
- Norme ECMA 262-3 : ECMAScript, communément JavaScript 1.5, publiée en 1999
- Normalisation en cours pour l'extension d'ECMAScript à l'environnement XML.

Normalisation = pérennité du langage.

Langage orienté objet

- Objets prédéfinis par le client : window, document, frame, image, etc...
- Création d'objet suivant des constructeurs prédéfinis : new Array()
- Manipulation d'objet par des méthodes
- Lecteur et modification des propriétés (attributs)
- Création de nouveaux constructeurs et de nouveaux objets spécifiques

Typage faible

- le type des variables est défini par défaut à la première attribution
- le type d'une variable est adapté à son contexte d'utilisation
- le type d'une variable peut être modifié par une instruction

Insertion au langage HTML

- **Balise** `<script>`

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

```
<script type="text/javascript"><!--  
.....  
// --></script>
```

- **Commentaire** `<script>`

```
// ligne de commentaire  
  
/* Début d'un bloc de commentaires  
.....  
    Fin du bloc commentaire */
```

Insertion au langage HTML

- Appel externe `<script>`

```
<script type="text/javascript" src="myScript.js"></script>
```

- Insertion de la balise `<script>` dans l'en-tête `<head>`

- Définition d'une fonction

```
<html>
<head>
<script type="text/javascript"><!--
  function myFunction {
    .....
  }
// --></script>
</head>
<body>
...
<a href="javascript:myFunction()">Clic here</a>
</body>
</html>
```

Insertion au langage HTML

- Appel externe `<script>`

```
<script type="text/javascript" src="myScript.js"></script>
```

- Insertion de la balise `<script>` dans l'en-tête `<head>`

- Définition d'une fonction

```
<html>
<head>
<script type="text/javascript"><!--
  function myFunction {
    .....
  }
// --></script>
</head>
<body>
...
<a href="javascript:myFunction()">Clic here</a>
</body>
</html>
```

Opérateurs

- Standards (Java, C)

+ - * /

% (modulo)

myVar++ (incrémentation)

myVar-- (décrémentation)

&& (ET logique)

|| (OU logique)

! (NON logique)

== (égal - comparaison)

!= (différent - comparaison)

<= (inférieur ou égal - comparaison)

>= (supérieur ou égal - comparaison)

< (strictement inférieur - comparaison)

> (strictement supérieur - comparaison)

Affectations

- Standards (Java, C)

`var myVar = 0;` (A noter le ; est une fin d'instruction)

`myVar = 3;`

`myVar += 5;` (Ajoute 5 à la variable -> nouvelle valeur 8)

`myVar -= 5;` (Ôte 5 à la variable -> nouvelle valeur 3)

`myVar *= 5;` (Multiplie par 5 la variable -> nouvelle valeur 15)

Bien différencier affectation et comparaison

affectation : =

comparaison : ==

Structure de condition if

```
if (expression1) {  
    .....  
} else if (expression2) {  
    .....  
} else {  
    .....  
}
```

```
if (myVar == 0) {  
    .....  
} else if (myVar > 0) {  
    .....  
} else {  
    .....  
}
```

Structure de condition switch

```
switch (expression){  
  case label :  
    .....  
    break;  
  case label :  
    .....  
    break;  
  ...  
  default :  
    .....  
}
```

Structure itérative boucle for

```
for ([initial-expression]; [condition]; [increment-expression]) {  
    .....  
}  
  
for (k=1; k<=100; k++) {  
    .....  
}
```

Structure itérative boucle for..in

```
myArray = new Array("poisson", "poulet", "boeuf");  
for (k in myArray) {  
    .....  
}
```

Structure itérative boucle while et do...while

```
var k = 1;
while (k<=100) {
    .....
    ++k;
}
```

```
var k = 1;
do
    .....
    ++k;
while (k<=100);
```