

# JavaScript

Rolf Hauri

# Sommaire

- Historique
- Syntaxe
- Environnement
- Outils

# Historique

- 1995 Mosaic/Netscape
  - Mocha/Livescript 💣
- 1997 ECMA - European Computer Manufacturers Association
  - ECMA-262
- 1998 ActionScript – Flash, Adobe
- 1999 JScript – Microsoft (ECMA 262 3<sup>ième</sup> éd.)
- 2011 ECMA-262 5.1



# Historique

- Popularité des langages ...
  - Tiobe
  - PYPL (PopularitY of Programming Language index)

# Historique

- Tiobe

Position Aug 2013	Position Aug 2012	Delta in Position	Programming Language	Ratings Aug 2013	Delta Aug 2012	Status
1	2	↑	Java	15.978%	-0.37%	A
2	1	↓	C	15.974%	-2.96%	A
3	4	↑	C++	9.371%	+0.04%	A
4	3	↓	Objective-C	8.082%	-1.46%	A
5	6	↑	PHP	6.694%	+1.17%	A
6	5	↓	C#	6.117%	-0.47%	A
7	7	=	(Visual) Basic	3.873%	-1.46%	A
8	8	=	Python	3.603%	-0.27%	A
9	11	↑↑	JavaScript	2.093%	+0.73%	A
10	10	=	Ruby	2.067%	+0.38%	A
11	9	↓↓	Perl	2.041%	-0.23%	A
12	15	↑↑↑	Transact-SQL	1.393%	+0.54%	A
13	14	↑	Visual Basic .NET	1.320%	+0.44%	A
14	12	↓↓	Delphi/Object Pascal	0.918%	-0.09%	A-
15	20	↑↑↑↑	MATLAB	0.841%	+0.31%	A-
16	13	↓↓↓	Lisp	0.752%	-0.22%	A
17	19	↑↑	PL/SQL	0.751%	+0.14%	A
18	16	↓↓	Pascal	0.620%	-0.17%	A-
19	23	↑↑↑↑	Assembly	0.616%	+0.11%	B
20	22	↑↑	SAS	0.580%	+0.06%	B

# Historique

- PYPL

Position Aug 2013	Position Aug 2012	Delta in position	Programming language	Share in Aug2013	Twelve month trends
1	1		Java	27.2 %	-0.9 %
2	2		PHP	14.3 %	-0.5 %
3	3		C#	9.8 %	+0.6 %
4	6	↑↑	<u>Python</u>	9.8 %	+1.8 %
5	4	↓	C++	9.1 %	-0.6 %
6	5	↓	C	8.5 %	-1.3 %
7	7		<u>Javascript</u>	7.8 %	+1.1 %
8	8		Objective-C	5.6 %	+0.7 %
9	9		Visual Basic	3.2 %	-0.6 %
10	10		<u>Ruby</u>	2.8 %	+0.1 %
© 2013 Pierre Carbonnelle					

<https://sites.google.com/site/pydatalog/pypl/PyPL-Popularity-of-Programming-Language>

# Historique

- Quels logiciels sont utilisés?
- Quels types d'interface?
  - Lourd, léger, riche
- Quels protocoles?
- Quels types d'appareils?
- Connecté/Déconnecté?

# Sommaire

✓ Historique

ü Syntaxe

ü Environnement

ü Outils



# Syntaxe

- Typage
- Variables
- Constantes
- Commentaires
- Opérateurs
- Traitements
- Structures de données
- Objets
- Classes fournies
- Gestionnaire d'erreur

# Typage

- Number (64b, NaN)
- String (UTF-16) délimité par " ou '
- Boolean (true, false)
- Object
- Null (une seule valeur... nulle)
- Undefined (une seule valeur ... Indéfinie)

# Variables

- Définition d'une variable "locale"

```
var prix;
```

- La définition est optionnelle : les variables non définies seront automatiquement "globales"!
- Pas de constante!
  - `const` extension "propriétaire" NES 6.0

# Commentaires

// pour commenter le reste de la ligne

/\*

Pour  
un commentaire  
multi  
ligne

\*/

# Opérateurs

- Affectation
- Concaténation
- Arithmétiques
- Logiques
- Comparaison
- Spéciaux

# Opérateurs

- Affectation `[=, +=, -=, *=, /=, %=]`
- Concaténation `+`
- Arithmétiques `[+, -, *, /, ++, --, %]`
- Logiques `[&&, ||, !]`
- Comparaison `[==, !=, <, >, <=, >=, ===, !==]`
- Spéciaux
  - `this, new, instanceof, typeof, in, ? : , ...`

# Syntaxe

- ✓ Typage
- ✓ Variables
- ✓ Constantes
- ✓ Commentaires
- ✓ Opérateurs
- Traitements
  - Structures de données
  - Objets
  - Classes fournies
  - Gestionnaire d'erreur

# Traitements

- Sensible à la casse
- Séparateur d'instruction ";"
- Structures de contrôle
  - Conditionnelles, boucles, rupture
- Fonction et procédure



# Structures conditionnelles I

```
if (condition) {  
    instructionsS  
}
```

➤ { et } optionnelles... Mais rarement qu'une instruction!

```
if (condition) {  
    instructions  
} else {  
    instructions  
}
```

# Structures conditionnelles II

```
if (condition) {  
    instructions  
} else if (condition) {  
    instructions  
} else {  
    instructions  
}
```

# Structures conditionnelles III

```
switch (variable) {  
    case valeur-1: instructions  
    break;  
    ...  
    case valeur-n: instructions  
}
```

➤ Optionnel:

```
default: instructions
```

# Structures conditionnelles IV

## ➤ Opérateur conditionnel:

```
y=condition ? val-true:val-  
false;
```

## ➤ Exemple pas de frais de port si achats>100:

```
frais = achat>100 ? 0 : 20;
```

# Structures Boucles I

```
while (condition) {  
    instructionS  
}
```

➤ Boucle do while avec au moins un parcours:

```
do {  
    instructions  
} while (condition)
```

# Structures Boucles II

```
for(initialisation; condition; incr.) {  
    instructionS  
}
```

➤ Exemple:

```
for(var i=0; i<10; i++) {  
    instructions  
}
```

# Structures Boucles III

```
for (cle in tableau) {  
    instructionS  
}
```

➤ Exemple:

```
for (i in noms) {  
    instructions  
}
```

Valeurs de i: indice 0, 1, .. N ou clés associatives\*

\* Voir tableau associatif et objets

# Structures de rupture

- `break` : sort d'une boucle
- `continue` : va directement à la prochaine itération de la boucle
- `return` : sort d'une fonction
- `return x` : sort d'une fonction en retournant la valeur `x`
- `break nom` : sort du bloc labelisé\* `nom`

\*Bloc labelisé = `nom: {instructions}`



# Fonctions et procédures I

- Définition

```
// affiche un msg dans une coul:  
function afficher(msg, coul) {  
    ...  
}
```

- Appels

```
afficher("blabla", "vert");  
afficher("blabla", "rouge");
```

# Fonctions et procédures II

- Définition

```
// Calcule la surface  
function surface(forme, h, l) {  
    var res=0;  
    ...  
    return res;  
}
```

- Appel:

```
var s = surface(f, 100, 30);
```

# Syntaxe

- ✓ Typage
- ✓ Variables
- ✓ Constantes
- ✓ Commentaires
- ✓ Opérateurs
- ✓ Traitements
- Structures de données
- Objets
- Classes fournies
- Gestionnaire d'erreur

# Structures de données I

- Elles n'existent pas ...
- Implémentées par l'objet Array
- Définition, syntaxes:
  - `lst = new Array();`  
`lst[0] = "blanc";`  
`lst[1] = "bleu";`  
`// En utilisant la méthode de Array:`  
`lst.push("blanc");`  
`lst.push("bleu");`
  - `lst = new Array("blanc", "bleu");`
  - `lst = ["blanc", "bleu"];`

# Structures de données II

- Accès, méthodes:
  - `couleur = lst[0];`
  - `couleur = lst.pop(); //retire!`
  - `lst.sort(); //Trie`
  - `lst.reverse(); // En ordre inverse`
  - `lst.length; //Taille`
  - `lst.join(sep); //Transforme en chaîne en ajoutant le séparateur sep`

# Structures de données III

- Tableaux associatifs

```
– client = new Array();  
client['nom'] = 'Dumont';  
client['prenom'] = 'Sophie';
```

```
– client = { 'nom': 'Dumont', ... };
```

- Rappel: la boucle "for in" permet de parcourir les clés

# Structures de données IV

- Les tableaux associatifs sont ... des objets
- Accès par notation pointée aux propriétés:
  - `client['nom'] = 'Dumont';`
  - `client.nom = 'Dumont';`

# Les objets

- Définition d'une classe
- Instanciation
- Propriétés/Attributs
- Méthodes
- Constructeur
- Modifier une classe existante dans JS



# Les objets

- Définition d'une classe: `function`
- Instanciation: `c = new Client();`
- Propriétés/Attributs: `c.nom / this.nom`
- Méthodes: `c.x() / this.x`
- Constructeur: `function Client()`
- Modifier une classe existante dans JS
  - `laClasse.prototype.laMethode = ...`

# Les objets

- Exemple

```
function Compte(numero, nom, taux) {  
    // Attributs  
    this.numero = numero;  
    this.nom = nom;  
    this.taux = taux;  
    this.solde = 0;  
  
    // Méthodes  
    this.afficher = function () {  
        ...  
    }  
  
    this.retirer = function (montant) {  
        // Vérification  
        if (montant > this.solde) {  
            ...  
            return;  
        }  
        this.solde -= montant;  
    }  
  
    this.verser = function (montant) {  
        this.solde += montant;  
    }  
}
```

# Classes fournies (ECMA 262)

- Types de base:
  - Number, String, Date, Boolean
- Array
- Error
- Math, RegExp, JSON
- Object, Function

# Classes fournies (ECMA 262)

- Exemple String, Date:

- Utilisation

```
nom = "Dupont"; /*  
nom.toUpperCase();
```

String
length
...
charAt() indexOf() replace() search() split() substr() toLowerCase() toUpperCase() ...

Date
...
getDate() getDay() getFullYear() getHours() getMilliseconds() getMinutes() getMonth() getSeconds() getTime() ...

```
/* nom = new String("Dupont");
```

# Gestionnaire d'erreur

- Bloc Try/Catch

```
try { ... }  
catch (e) { ... e.message ... }
```

- Génération d'exception avec throw

```
throw new Error("msg d'erreur");
```

# Syntaxe

- ✓ Typage
- ✓ Variables
- ✓ Constantes
- ✓ Commentaires
- ✓ Opérateurs
- ✓ Traitements
- ✓ Structures de données
- ✓ Objets
- ✓ Classes fournies
- ✓ Gestionnaire d'erreur

# Sommaire

✓ Historique

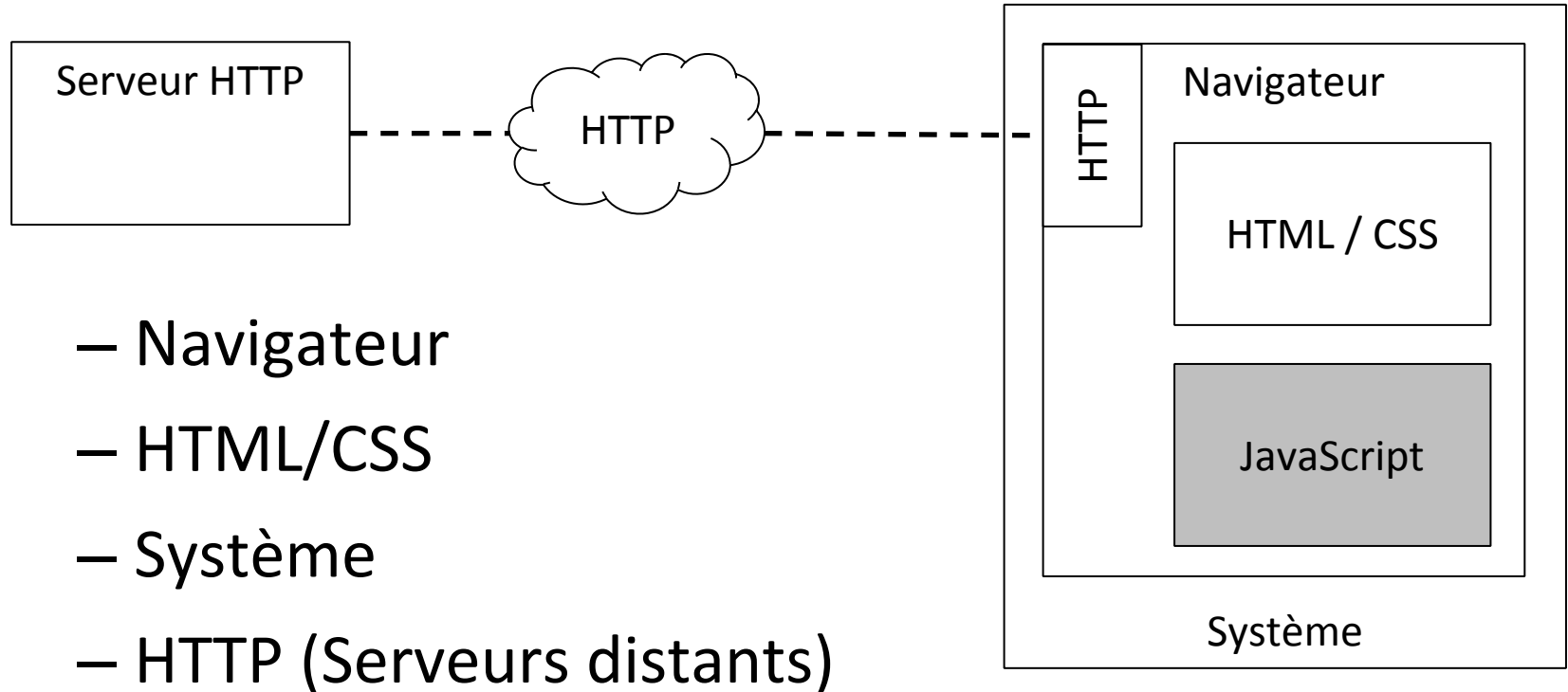
✓ Syntaxe

ü Environnement

ü Outils

# Environnement

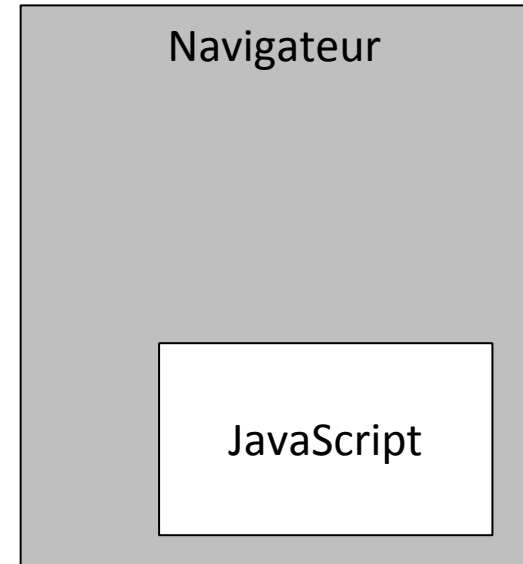
- Vue d'ensemble:





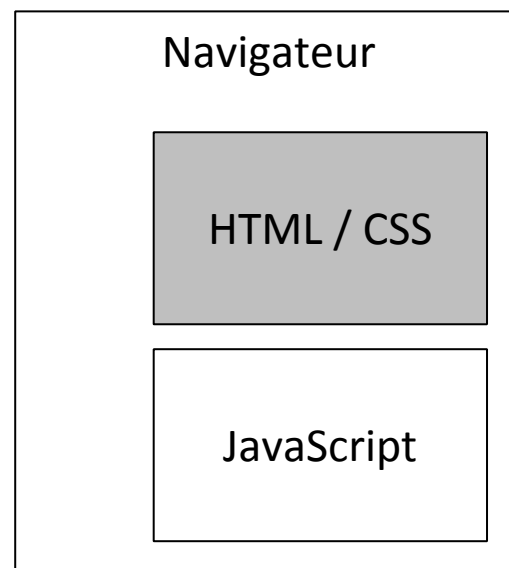
# Environnement - Navigateur

- Interpréteur JavaScript
- Syntaxe JavaScript (ECMA)
- Fourni par le navigateur
- Extensions propriétaires




# Environnement – HTML/CSS

- Accès au document HTML
  - Accès aux éléments
- Manipulation du document
  - Modification des propriétés CSS
  - Ajout/suppression
- Spécifiés dans le DOM (Domain Object Model)



# DOM

- Maintenu par le W3C (w3c.org)
- Interface programmable pour XML et HTML
- Indépendante de toute plateforme et de tout langage
- Permet d'accéder et de mettre à jour  dynamiquement le contenu, la structure et

# DOM

- Modifier le contenu:

```
document.getElementById("..").value = "Genève";
```

- Modifier le style:

```
document.getElementById("..").style.left = "120px";
```

- Modifier la structure:

```
el = document.createElement("li");
```

```
el.appendChild(document.createTextNode("Rouge"));
```

```
el.setAttribute("id", "coul_rouge");
```

```
document.getElementById("couleurs").appendChild(el);
```

# DOM

- Modifier la structure:

```
Node      insertBefore(in Node newChild,  
                        in Node refChild)  
                        raises(DOMException);  
Node      replaceChild(in Node newChild,  
                        in Node oldChild)  
                        raises(DOMException);  
Node      removeChild(in Node oldChild)  
                        raises(DOMException);  
Node      appendChild(in Node newChild)  
                        raises(DOMException);  
boolean   hasChildNodes();
```

# DOM

- Modifier la structure:

```
String      getAttribute(in DOMString name);  
Void        setAttribute(in DOMString name, in DOMString value)  
Void        removeAttribute(in DOMString name)  
Attr        getAttributeNode(in DOMString name);  
Attr        setAttributeNode(in Attr newAttr)  
Attr        removeAttributeNode(in Attr oldAttr)
```

```
Boolean hasAttribute(in DOMString name);
```

# DOM

- Parcourir un document

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <!-- Generated from data/head-home.php, ../../smarty/{head.tpl} -->
  <head>
    <title>World Wide Web Consortium (W3C)</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <link rel="Help" href="/Help/">
    <link rel="stylesheet" href="/2008/site/css/minimum" type="text/css" media="handheld, all">
    <style type="text/css" media="print, screen and (min-width: 481px)">*<![CDATA[* @import url('/2008/site/css/minimum.css');]]>*</style>
    <link href="/2008/site/css/minimum" rel="stylesheet" type="text/css" media="handheld, only screen and (min-width: 481px)">
    <meta name="viewport" content="width=device-width">
    <link rel="stylesheet" href="/2008/site/css/print" type="text/css" media="print">
    <link rel="shortcut icon" href="/2008/site/images/favicon.ico" type="image/x-icon">
    <meta name="description" content="The World Wide Web Consortium (W3C) is an international community of full-time staff, and the public work together to develop Web standards.">
    <link rel="alternate" type="application/atom+xml" title="W3C News" href="/News/atom.xml">
  </head>
  <body id="www-w3-org" class="w3c_public w3c_home w3c_javascript w3c_screen">
    <div id="w3c_container">...</div>
    <!-- Generated from data/footer.php, ../../smarty/{footer-block.tpl} -->
    <div id="w3c_footer">...</div>
    <!-- /end #footer -->
    <!-- Generated from data/scripts.php, ../../smarty/{scripts.tpl} -->
    <div id="w3c_scripts">...</div>
  </body>
</html>
```

# DOM

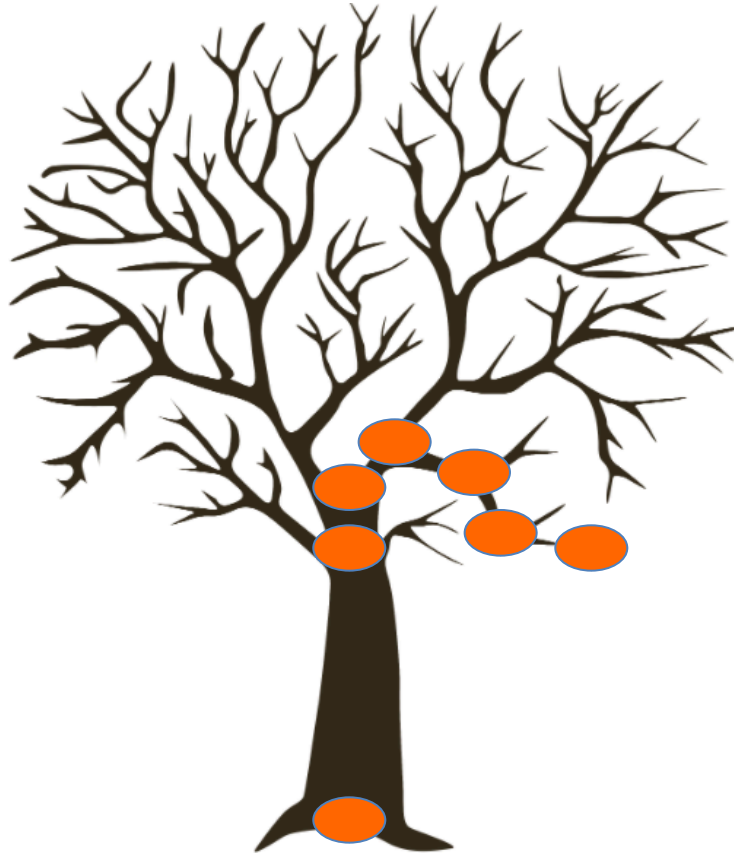
- Parcourir un document





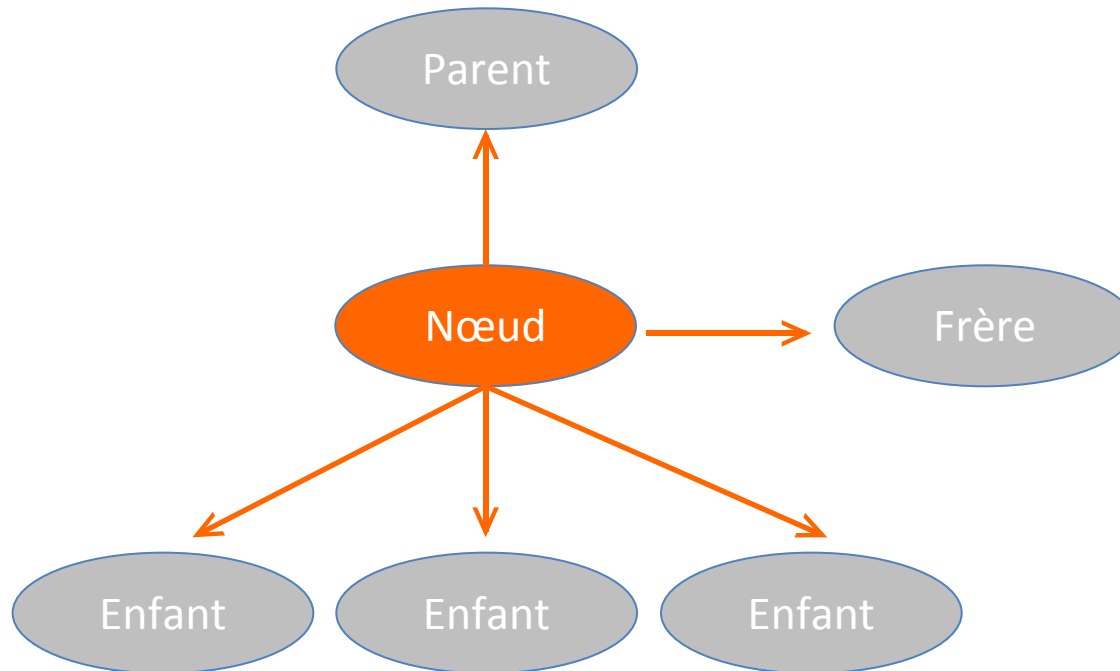
# DOM

- Parcourir un document, nœud par nœud:



# DOM

- Parcourir un document, nœud par nœud:



# DOM

- Type d'un Node (nœud):

```
// NodeType
const unsigned short    ELEMENT_NODE           = 1;
const unsigned short    ATTRIBUTE_NODE         = 2;
const unsigned short    TEXT_NODE              = 3;
const unsigned short    CDATA_SECTION_NODE     = 4;
const unsigned short    ENTITY_REFERENCE_NODE  = 5;
const unsigned short    ENTITY_NODE            = 6;
const unsigned short    PROCESSING_INSTRUCTION_NODE = 7;
const unsigned short    COMMENT_NODE           = 8;
const unsigned short    DOCUMENT_NODE          = 9;
const unsigned short    DOCUMENT_TYPE_NODE     = 10;
const unsigned short    DOCUMENT_FRAGMENT_NODE = 11;
const unsigned short    NOTATION_NODE          = 12;
```

# DOM

- Attributs d'un Node (nœud):

```
readonly attribute DOMString      nodeName;  
    attribute DOMString          nodeValue;  
                                   // raises(DOMException) on setting  
                                   // raises(DOMException) on retrieval  
  
readonly attribute unsigned short nodeType;  
readonly attribute Node           parentNode;  
readonly attribute NodeList       childNodes;  
readonly attribute Node          firstChild;  
readonly attribute Node          lastChild;  
readonly attribute Node          previousSibling;  
readonly attribute Node          nextSibling;  
readonly attribute NamedNodeMap  attributes;
```

# DOM

- Combien de Nodes?
- Combien d'attributs?

```
<div class="info" id="news">  
  <p class="notice">Nouvel élément  
dans..</p>  
</div>
```

# DOM – Gestion des événements

- Programmation événementielle
- Événements/Listeners
  - Liste des événements disponibles
- Phases et Hiérarchie
- Annulation de l'événement

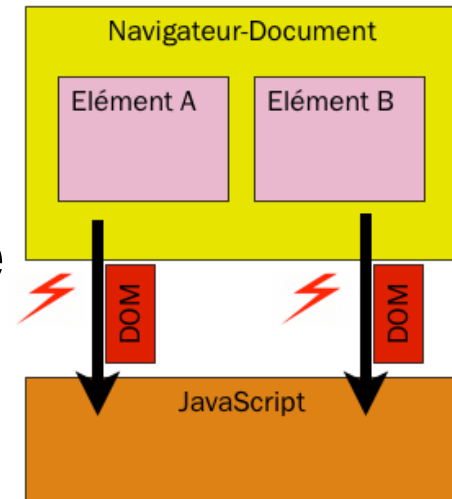
# Programmation événementielle

*En informatique, une programmation événementielle se dit d'un type de programmation fondé sur les événements. Le programme sera principalement défini par ses réactions aux différents événements qui peuvent se produire, c'est-à-dire des changements d'état de variable, par exemple l'incrémentatation d'une liste, un mouvement de souris ou de clavier.*



# DOM – Gestion des événements

- Listeners
  - Fonctions en attente passive



- Syntaxe:

```
element.addEventListener(type, fct, useCapture);
```

```
// IE8
```

```
element.attachEvent(type, fct);
```

```
// Avant DOM 3.0
```

```
<input onmouseover=fct >
```



# DOM – Gestion des événements

- Types possible (événements)

- load
- unload
- click
- dblclick
- mousedown
- mouseup
- mouseover
- mousemove
- mouseout
- focus
- blur
- keypress
- keydown

- keyup
- submit

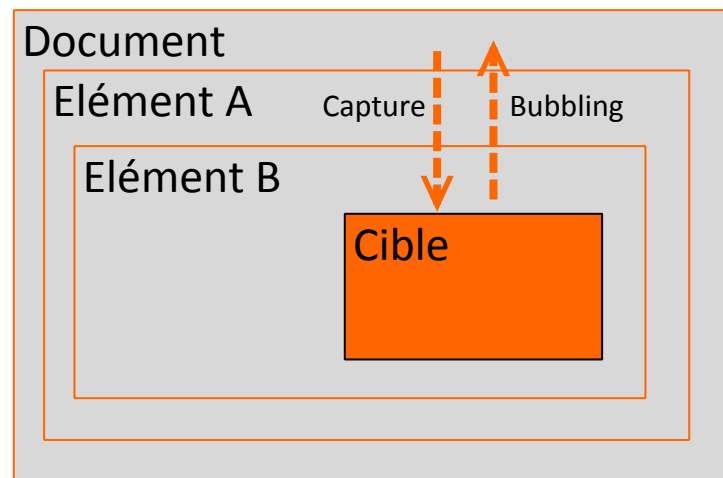
# DOM – Gestion des événements

- Phases

- Capture
- Propagation (Bubbling)

- Cible: target

- Hiérarchie: currentTarget

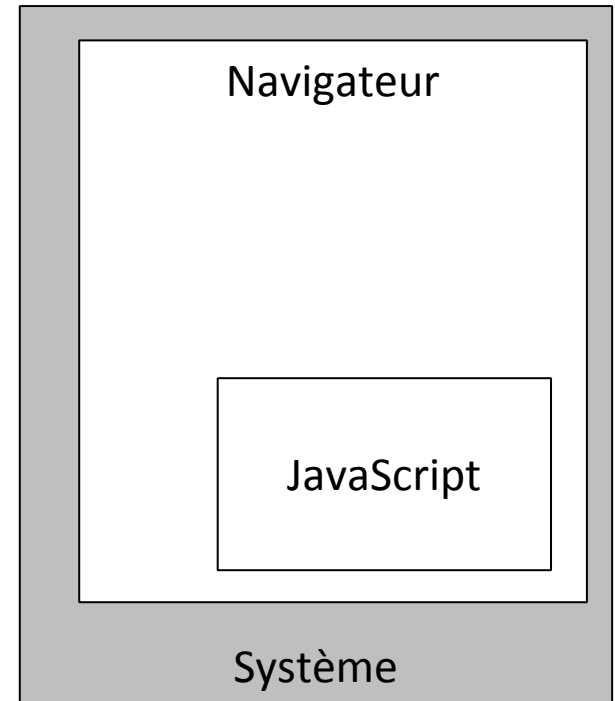


- Annulation

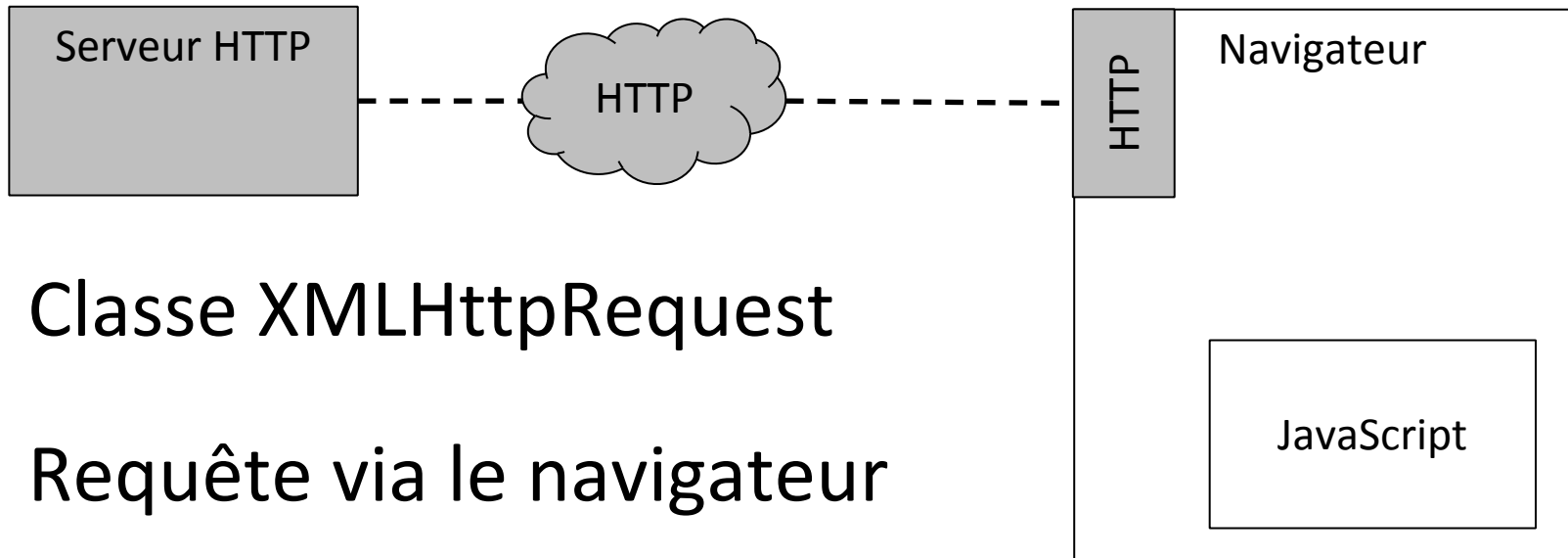
- Capture: `preventDefault()`
- Propagation: `stopPropagation()`

# Environnement - Système

- Accès au système
- Fourni par des objets JS
- Non standard! ("DOM 0")
  - Window
  - Navigator
  - Screen
  - History
  - Location

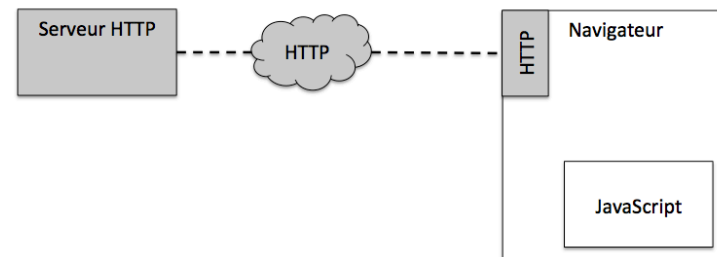


# Environnement - HTTP



- Classe XMLHttpRequest
- Requête via le navigateur
- "Limité" au serveur de la page
  - Librairie, Wrapper
- AJAX (Asynchronous JavaScript And Xml)

# Environnement - HTTP



- Classe XMLHttpRequest

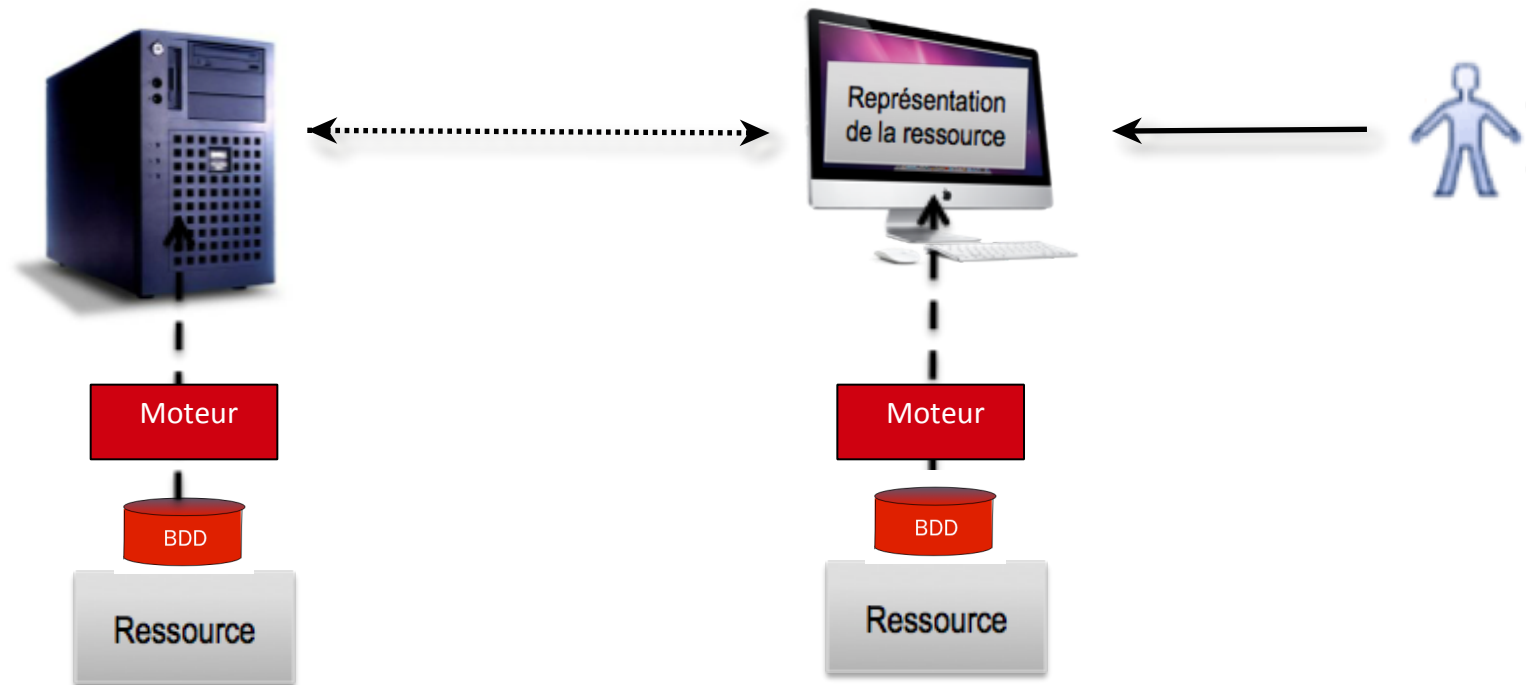
XMLHttpRequest
status statusText readyState onreadystatechange responseText responseXML ...
open(method, url, optional boolean async) send(optional data) ...

## readyState:

Value	State	Description
0	UNSENT	<code>open()</code> has not been called yet.
1	OPENED	<code>send()</code> has not been called yet.
2	HEADERS_RECEIVED	<code>send()</code> has been called, and headers and status are available.
3	LOADING	Downloading; <code>responseText</code> holds partial data.
4	DONE	The operation is complete.

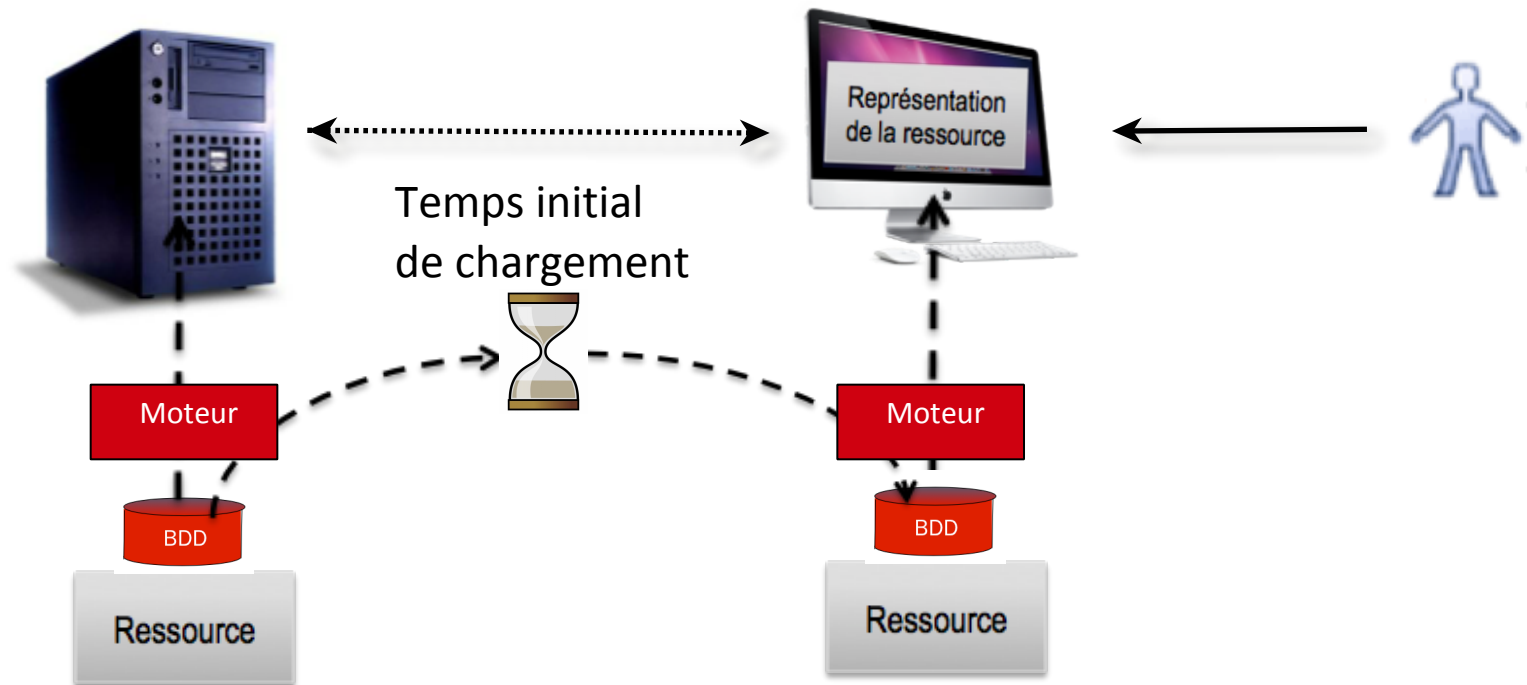
# Environnement - HTTP

- Architecture "Sans AJAX"



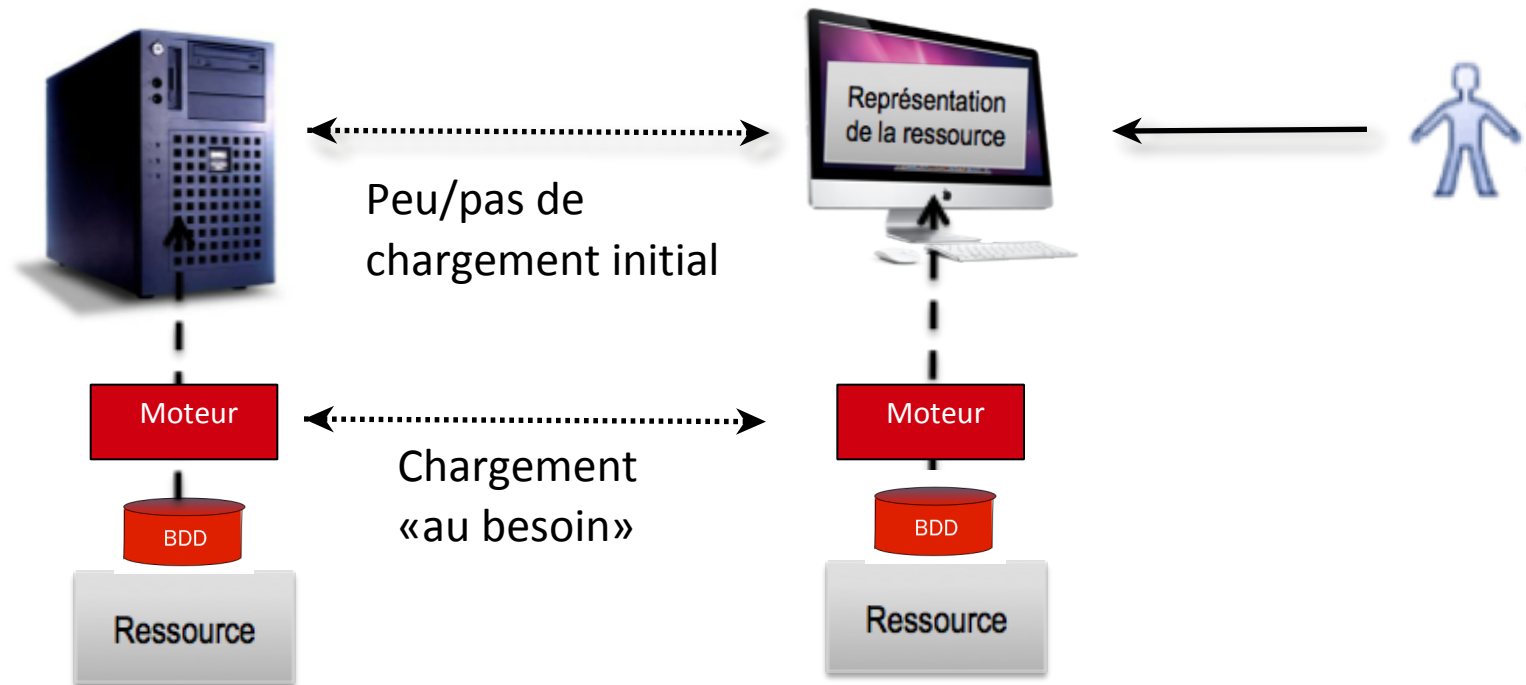
# Environnement - HTTP

- Temps de chargement initial des données



# Environnement - HTTP

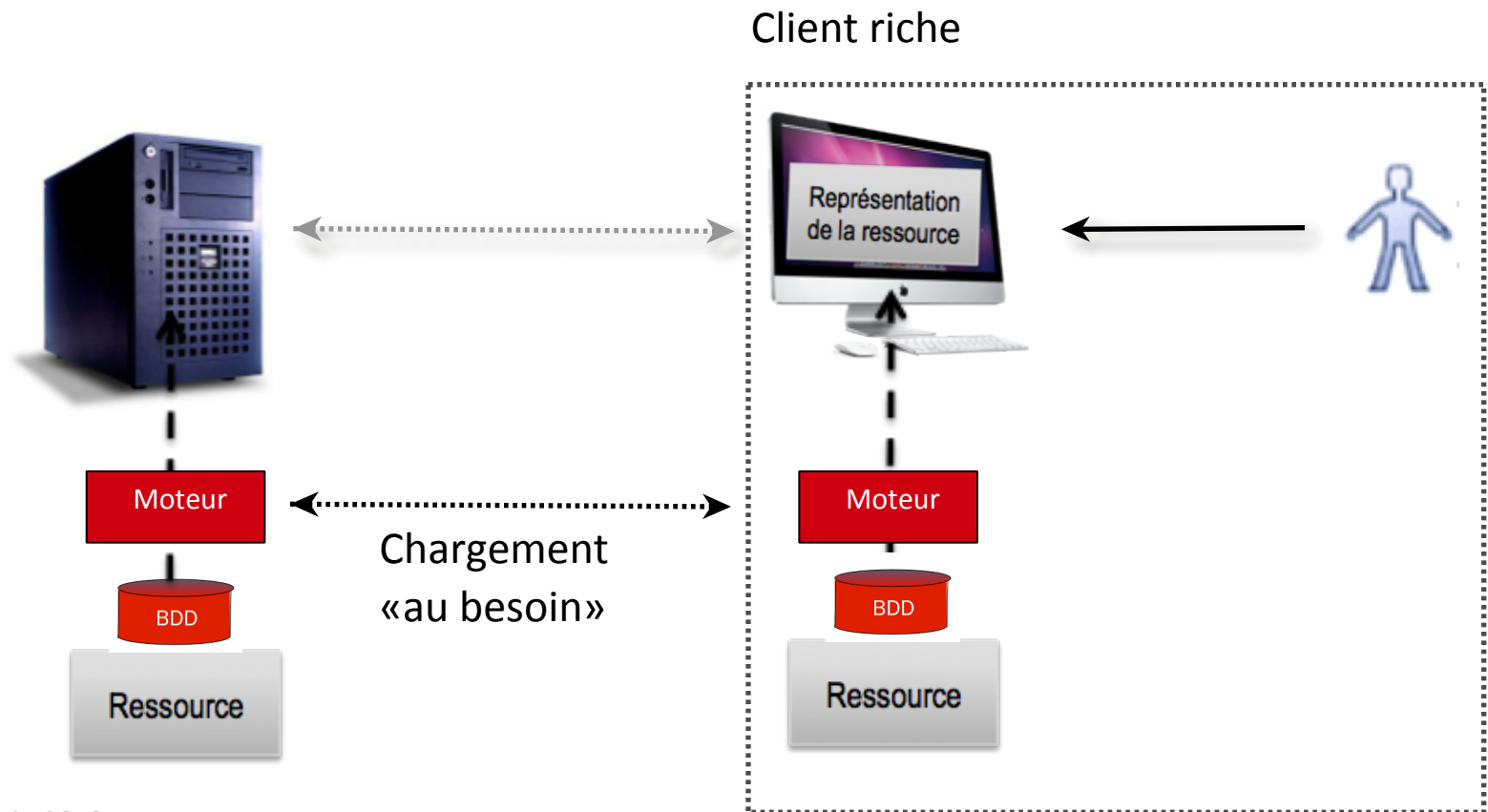
- AJAX: chargement "selon besoin"





# Environnement - HTTP

- Requêtes asynchrones et transparentes

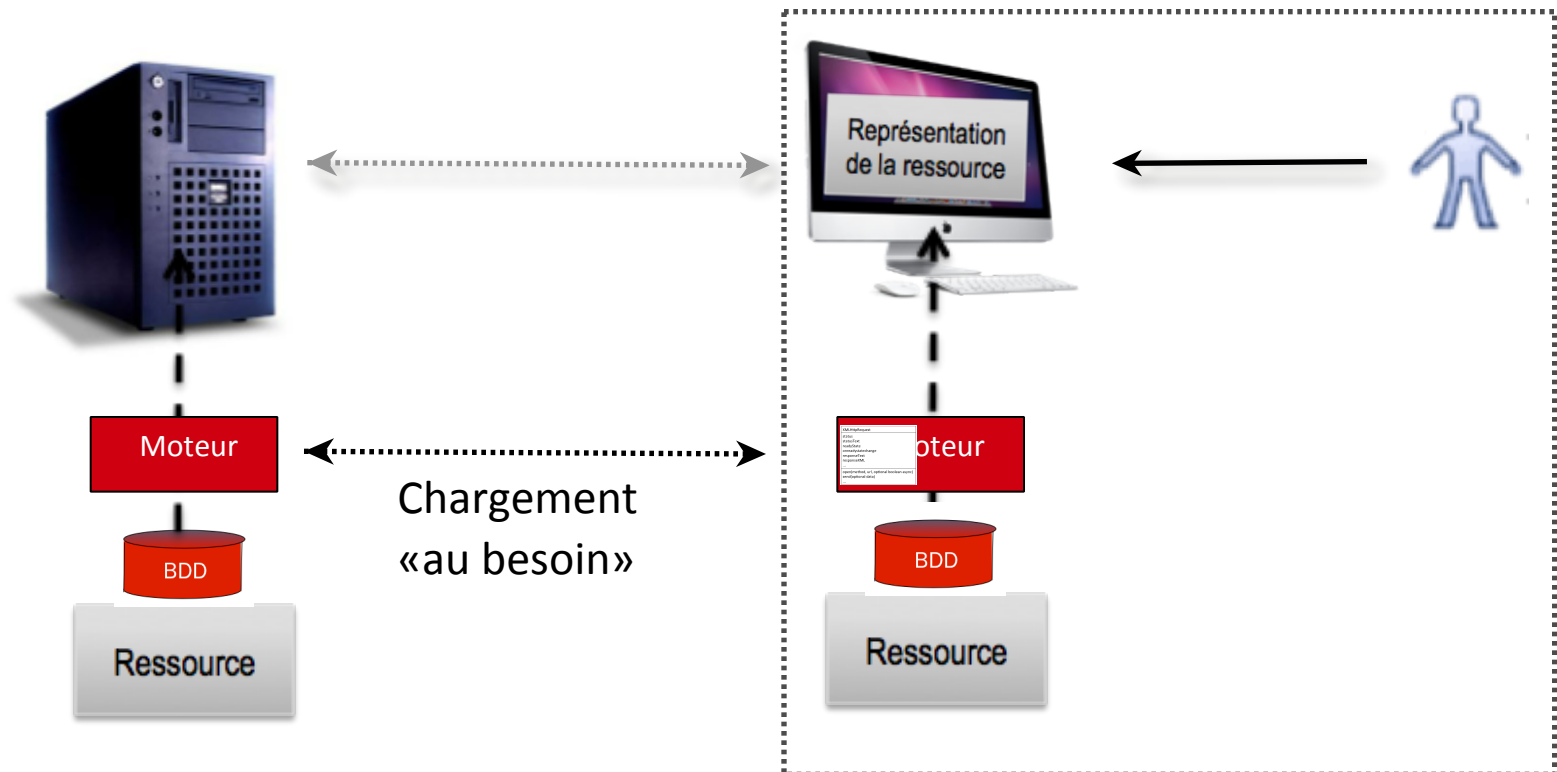


# Environnement - HTTP

- La classe XMLHttpRequest

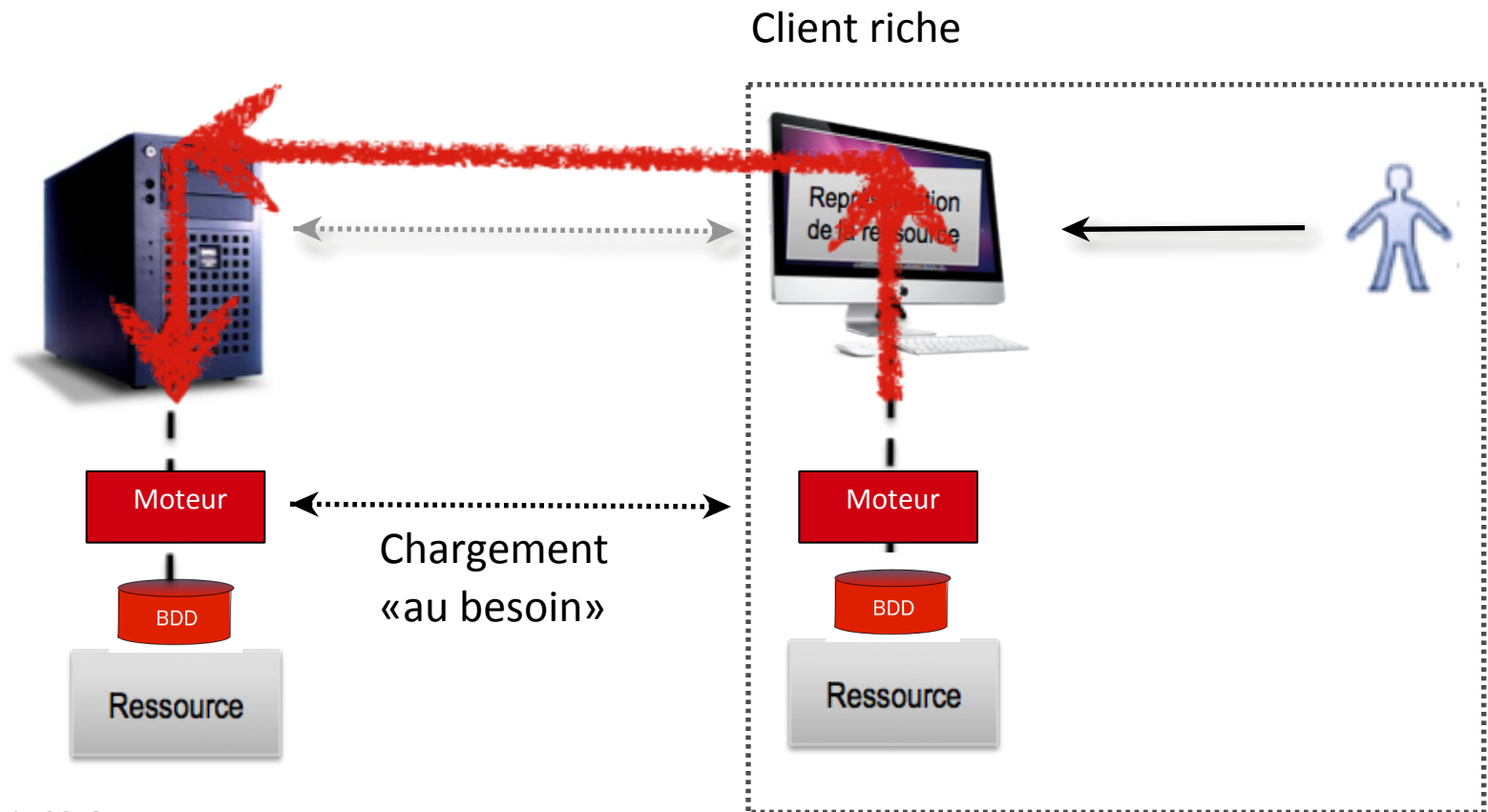
XMLHttpRequest
status
statusText
readyState
onreadystatechange
responseText
responseXML
...
open(method, url, optional boolean async)
send(optional data)
...

Client riche

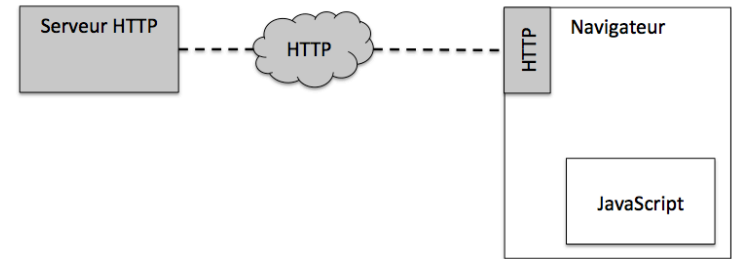


# Environnement - HTTP

- L'échange HTTP passe par le navigateur!

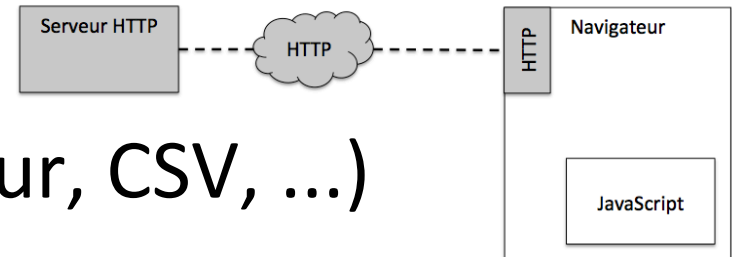


# Environnement - HTTP



- Format des données
- Texte brut
  - Structure avec séparateur
- XML (le "X" d'AJAX...)
- JSON
  - JavaScript Object Notation
  - Encapsulation/sérialisation/dé-sérialisation
    - Object → Chaine-JSON → Object
  - [www.json.org](http://www.json.org)

# Environnement - HTTP



- Texte brut (avec séparateur, CSV, ...)

123: Dupont: Jean: FR; IT; ALL

345: Dupont: Martine: FR

→ Séparateurs?

→ Langues associée à Martine?

→ Quel est le premier champ?

→ Qui sont ces personnes?

→ Comment ajouter des téléphones (type, num.)?

# Environnement - HTTP

- *Extensible Markup Language (XML) (français : « langage extensible de balisage ») est un langage informatique de balisage générique. Il sert essentiellement à stocker/transférer des données de type texte Unicode structurées en champs arborescents. Ce langage est qualifié d'extensible car il permet de définir les balises des éléments*

```
<livres>
  <livre id="123">
    <ISBN>ISBN-01</ISBN>
    <titre>Titre-01</titre>
    <auteurs>
      <auteur>...</auteur>
      <auteur>...</auteur>
    </auteurs>
  </livre>
  <livre id="456">
    ...
```



# Environnement - HTTP

- XML

→ Quel est le champ "123"?

→ Qui sont ces personnes?

→ Ratio **information**/structure?

```
<clients>
  <client id="123">
    <nom>Dupont</nom>
    <prenom>Jean</prenom>
    <langues>
      <lg>FR</lg>
      <lg>IT</lg>
      <lg>ALL</lg>
    </langues>
  </client>
  <client id="345">
    <nom>Dupont</nom>
    <prenom>Martine</prenom>
    <langues>
      <lg>FR</lg>
    </langues>
  </client>
</clients>
```

# Environnement - HTTP

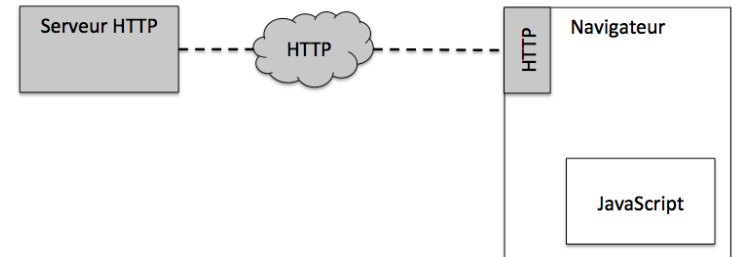
- JSON (JavaScript Object Notation) est un format de données textuel, générique, dérivé de la notation des objets du langage ECMAScript. Il permet de représenter de l'information structurée. Créé par Douglas Crockford, il est décrit par la RFC 4627 de l'IETF.*

```
{livres:[{"id":"123","titre":"Titre-01","auteur":  
[...]} , {"id":"456","titre":"Titre-  
02","auteur":[...]}
```





# Environnement - HTTP



- JSON

```
[{"id": "123", "nom": "Dupont", "prenom": "Jean", "langues": ["FR", "IT", "ALL"]}, {"id": "345", "nom": "Dupont", "prenom": "Martine", "langues": ["FR"]}]
```

→ Comment générer la chaine (en PHP)?

```
json_encode($obj);
```

→ Comment instancier les objets (en JS)?

```
obj = JSON.parse(str);
```

# Sommaire

- ✓ Historique
- ✓ Syntaxe
- ✓ Environnement

ü Outils

# Outils

- Editeurs
- Référence
- Interpréteurs
- Débogueurs
- Librairies
- Tests, Beautifier, Générateur de documentation, ...

# Editeurs

- Editplus
- Smultron
- Dreamweaver
- NetBeans
- Eclipse
- ...

# Références

## Références officielles

- <http://www.ecma-international.org/publications/standards/Ecma-262.htm>
- <http://www.w3.org/DOM/>

## Fournisseurs

- <https://developer.mozilla.org/fr/docs/JavaScript>
- [http://msdn.microsoft.com/fr-fr/library/ie/yek4tbz0\(v=vs.94\).aspx](http://msdn.microsoft.com/fr-fr/library/ie/yek4tbz0(v=vs.94).aspx)
- <http://code.google.com/p/v8/>

## Tiers

- <http://www.w3schools.com>
- <http://www.javascriptkit.com/jsref/>
- <http://www.xul.fr/ecmascript/>

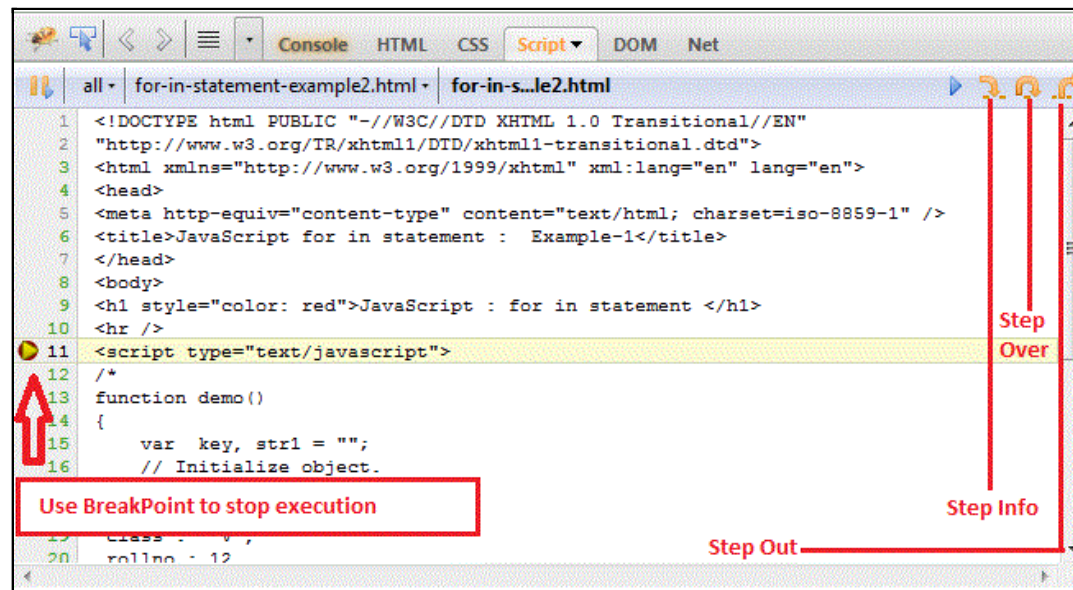
## Vos références:

# Interpréteurs

- Navigateurs
  - W3C, IE,...
  - Balise <script>
  - Fichiers externes
- Editeurs avec moteur intégré
  - Exemple: Editplus=IE

# Débogueurs

- Navigateurs
- Editeurs avec moteur intégré
- Firebug



# Librairies

- jQuery
  - [jquery.com](http://jquery.com)
- Prototype
  - [prototypejs.org](http://prototypejs.org)
- Mootools
  - [mootools.net](http://mootools.net)
- ...



# Libraries

- jQuery

```
1 | $.ajax({  
2 |   url: "test.html",  
3 |   context: document.body  
4 | }).done(function() {  
5 |   $(this).addClass("done");  
6 | });
```

# Libraries

- Prototype

```
new Ajax.Request('/some_url', {  
  method: 'get',  
  onSuccess: function(transport) {  
    var response = transport.responseText || "no response text";  
    alert("Success! \n\n" + response);  
  },  
  onFailure: function() { alert('Something went wrong...'); }  
});
```

# Librairies

- La librairie doit être disponible (chargée) dans le navigateur
- Option des librairies hébergées:

## What is the Google Hosted Libraries?

The Google Hosted Libraries is a content distribution network for the most popular, open-source JavaScript libraries. The hosted libraries provides access to a growing list of the most popular, open-source JavaScript libraries, including:

- [AngularJS](#)
- [Chrome Frame](#)
- [Dojo](#)
- [Ext Core](#)
- [jQuery](#)
- [jQuery UI](#)
- [MooTools](#)
- [Prototype](#)
- [script.aculo.us](#)
- [SWFObject](#)
- [WebFont Loader](#)

Google works directly with the key stake holders for each library effort and accepts the latest stable versions as they are released. Once we host a release of a given library, we are committed to hosting that release indefinitely.

<https://developers.google.com/speed/libraries/>

# Outils

- ✓ Editeurs
- ✓ Référence
- ✓ Interpréteurs
- ✓ Débogueurs
- ✓ Librairies
- ü Tests, Beautifier, Générateur de documentation, ...
  - JSLint, SugarTest, JSbeautifier, YUIDoc, ...

# Sommaire

- ✓ Historique
- ✓ Syntaxe
- ✓ Environnement
- ✓ Outils

# Index du support

• Historique	3	
• Syntaxe	8	
○ Typage	10	
○ Variables		11
○ Constantes	11	
○ Commentaires	12	
○ Opérateurs	13	
○ Traitements (alternatives, boucles, ..)	15	
○ Structures de données		27
○ Objets	32	
○ Classes fournies		35
○ Gestionnaire d'erreur	37	
• Environnement	39	
○ Navigateur	41	
○ HTML/CSS - DOM	42	
○ Événements	54	
○ Système	59	
○ HTTP / AJAX	60	
○ Format texte, XML, JSON	68	
• Outils	74	