

MODULE 2

Javascript

JAVASCRIPT : INTRODUCTION

- **Langage de programmation:** comportement, logique applicative
- **Langage interprété, script:** exécuté dans un navigateur
OS > Navigateur > Page > Javascript
- Pas d'accès au système de fichiers, ni au matériel (sécurité)
- Historiquement, un langage "**côté client**"
- Javascript "côté serveur" (NodeJS) est hors-sujet
- Nécessite l'activation de Javascript dans le navigateur (fréquent en 2014)
- Un peu d'histoire
1995 LiveScript > 1996 Javascript > 1997 ECMAScript => standardisé > **1999 ECMAScript 3** > **2009 ECMAScript 5**
- RIEN À VOIR AVEC JAVA, malgré la similitude du nom

HELLO, WORLD!

- Console du navigateur (Chrome Dev Tools, Firebug, IE F12 Tools...)

```
alert('Hello, World!');
```
- Fichier HTML

```
<script>
  alert('Hello, World!');
</script>
```
- Script externe

```
<script src="script.js" type="text/javascript"></script>
alert('Hello, World (bis)');
```

REMARQUES D'ORDRE GÉNÉRAL

- Sensible à la casse: `alert()` OK, `Alert()` NOK
- Instructions (statements)
 - Dans l'idéal: UNE SEULE par ligne, terminée par ";"
 - L'absence du ";" est pardonnée, mais pas recommandée
 - Possibilité de répartir sur plusieurs lignes des instructions plus longues
- Espaces
 - `alert('Hello, World');`
 - `alert ('Hello, World');`
 - `alert`
 (
 "Hello, World"
);
 - `alert('Hello, World');`

REMARQUES D'ORDRE GÉNÉRAL

- Commentaires
 - // commentaire
 - /* commentaire sur plusieurs lignes
*/
- Ordre d'exécution des instructions
 - Sauf indication contraire: **en séquence**
 - ➡ Importance de l'emplacement de l'inclusion des scripts dans la page (exemple de l'`alert()` dans le head ou en fin de body)

MODULE 2: JAVASCRIPT >

FONDAMENTAUX DU LANGAGE, SYNTAXES

VARIABLES

- Allocation d'un espace mémoire pour stocker une valeur

```
var titre;  
var titre='Développement Web Mobile';  
titre='Développement Web Mobile'; // à éviter  
var nom, prenom, groupe;  
var nom='Lennon', prenom='John', groupe='The Beatles';
```
- Règles pour les noms de variables:
 - pas d'espace
 - lettres
 - chiffres (mais ne peut pas commencer par un chiffre)
 - _
 - \$
 - sensible à la casse
- Les variables sont **NON TYPÉES** en Javascript
- Types: Strings, Integers, Floats, Booleans, Arrays, Functions, Objects...

OPÉRATEURS

ASSIGNEMENT

- `var age = 20;`
- `var nom = 'GUILLOU';`
- `var y = a * x + b;`

OPÉRATEURS ARITHMÉTIQUES

- `+`
- `-`
- `*`
- `/`
- `+=`
`compteur += 1;`
`compteur = compteur+1;`
- `-=`
- `*=`
- `/=`

PRÉCÉDENCE DES OPÉRATEURS

- `y = a *x + b;`
- Tableau des précédences
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

INSTRUCTIONS CONDITIONNELLES

if / else

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

COMPARATEURS

- `==`
- `===`
`var a = 5, b = '5';`
`a==b; // true`
`a===b; // false`
- `!=`
`!==`
- `>`
- `<`
- `<=`
- `>=`

OPÉRATEURS LOGIQUES

- `&&` (ET)
`expr1 && expr2`
`"A" && "B" // renvoie "B"`
- `||` (OU)
`expr1 || expr2`
`"A" || "B" // renvoie "A"`
- `!` (NON)
`!expr`
`!"A" // renvoie false`
- `%` (MODULO)
`var reste = a % b;`
- `+=, ++, -=, --` (Incrément, décrément)
 - Prefix / Postfix
`var a=5; alert(++a);`
`var a=5; alert(a++);`
- Opérateur ternaire
`condition ? valeur_si_vrai : valeur_si_faux`
`var sexe = (chromosomes == 'XY') ?`
`'M' : 'F';`

BOUCLES

- `while(condition) { ... }`
- `do { ... } while (condition)`
- `for (initialisation variables; conditions; expressions) { ... }`
`for (var i = 0; i < 10; i++) { ... }`
- `break;`
- `continue;`

CONSOLE

- ne fait pas partie des spécifications officielles de Javascript
- `console.log()`
- `console.info()`
- `console.error()`
- `console.debug()`
- `console.warn()`

FONCTIONS

- **Fonction déclarative (function declaration statement)**

```
function functionName ( /* optionnel */ parametres ) {  
    /* optionnel */ return returnValue;  
}  
  
function message( msg ) { alert( 'Message: ' + msg); }  
  
function somme( a, b ) { return a+b; }
```

FONCTIONS

- **Expression de fonction (function definition expression)**

```
var functionName = function( /* optionnel */ parametres ) {  
    /* optionnel */ return returnValue;  
}  
  
var somme = function(a,b) { return a+b; }
```

APPEL

```
► functionName( parametres );  
  
    message('Bonjour');  
  
    var s = somme(2,3); // s = 5
```

BONNES PRATIQUES

- Définir/déclarer une fonction avant de l'appeler (hoisting des fonctions en Javascript)
- Non requis pour les fonctions déclaratives
- IMPÉRATIF pour les expressions de fonction

MATCHING DES PARAMÈTRES

- Les paramètres supplémentaires sont ignorés
- Les paramètres manquants prennent la valeur *undefined*
- Utilisation du tableau *arguments* pour gérer des paramètres dynamiques

PORTÉE DES VARIABLES

FONCTIONS ANONYMES

TYPES ET OBJETS

TABLEAUX

- Création
 - `var tab = [];`
 - `var tab = [1,2,3,4,5,'une chaine',true];`
 - `var tab = new Array();`
 - `var tab = Array();`
 - `var tab = new Array(10);` // tableau de 10 éléments
 - `var tab = new Array(1,2,3,4,5);`
// [1,2,3,4,5]
 - `var tab = new Array([10,11,12]);`
// [10,11,12]

TABLEAUX

- Écrire une valeur
 - `tab[0] = 27;`
 - `tab[12] = 'Décembre';`
- Lire une valeur
 - `var mois = tab[12];`
- Remarques
 - Les tableaux sont non typés en Javascript
 - Le premier élément du tableau est à l'index 0
 - La taille des tableaux est dynamique en Javascript

TABLEAUX IMBRIQUÉS

- `var tab = [[1,2] , [3,4]];`
- `tab[0];` // [1,2]
- `tab[1][0];` // 3

TABLEAUX

- Propriétés
 - constructor
 - prototype
 - length
- Méthodes
 - reverse()
 - join()
 - sort()
 - indexOf()
 - lastIndexOf()
 - concat()
 - pop()
 - push()
 - shift()
 - unshift()
 - slice()
 - splice()
 - toString()
 - valueOf()

NOMBRES

- `var x = 10;`
 - Entier ? Décimal ?
 - En Javascript, tous les nombres sont des décimaux de 64 bits (double)
 - `var x = x + 0.3;`

NOMBRES

- **Addition / Concaténation**
 - L'opérateur '+' additionne les opérandes, sauf si l'une d'elle au moins est une chaîne
- **NaN**
 - `var n = Number('chaîne');`
 - `var n = 50 * 'chaîne';`
 - `isNaN()`

L'OBJET MATH

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math
- `Math.round(x);`
- `Math.max(a,b,c,);`
- `Math.min(a,b,c,d,e,f);`
- `Math.random();` // [0;1[
- `Math.sqrt(9);`
- `Math.PI`
- ...

CHAINES

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String
- `var s = 'une chaîne avec simples quotes';`
`var s2 = "une autre chaîne avec doubles quotes";`
- `\', \'`
- **Propriétés**
 - `length`
- **Méthodes**
 - `toUpperCase()`
 - `toLowerCase()`
 - `split(' ');` // renvoie un tableau des sous-chaînes séparées par un espace
 - `indexOf();`
 - `slice();`
 - `substr();`
 - `substring();` // préférer slice

COMPARAISON DE CHAINES

- =
- comparer avec `toLowerCase()` et `toUpperCase()`
- <, >, <=, >=, !=
(Majuscules < minuscules)

DATES

- `var today = new Date();`
- `var date1 = new Date(2014,0,1);` // 1er Janvier 2014
- `var date2 = new Date(2014,0,1,0,0,0);` // + heures, minutes, secondes
- **Méthodes**
 - `getDate()` // jour du mois: 1-31
 - `getDay()` // jour de la semaine: 0-6
 - `getHours()` // 0-23
 - `getTime()` // milli-secondes depuis le 01/01/1970
 - `setMonth()`
 - `setFullYear()`
 - `setDay()`
 - ...
- `getMonth()` // 0-11
- `getFullYear()` // YYYY

COMPARAISON DE DATES

- ATTN! L'opérateur `==` compare les objets, et non les dates représentées
- utiliser `getTime()`

TPs

- `anniversaire('1983-08-06', 2014);` // renvoie 'Mercredi 6 août 2014'
- `compareDates(date1, date2, date3, date4,);` // renvoie true si les dates sont égales

OBJETS

OBJETS INTÉGRÉS

- Number, String, Date, Array... sont des objets de Javascript
- avec des propriétés
- et des méthodes

CRÉATION D'OBJETS

- ```
var nomPersonne1 = 'GUILLLOU',
 prenomPersonne1 = 'Cédric',
 agePersonne1 = 30;
```
- ```
var personne1 = new Object(); // équivalent à var personne1 = {};  
personne1.nom = 'GUILLLOU';  
personne1.prenom = 'Cédric';  
personne1.age = 30;
```
- ```
var personne1 = { nom: 'GUILLLOU', prenom: 'Cédric', age: 30 };
```

# OBJETS

### • Propriétés

- Lire une propriété  

```
var valeurPropriété = objet.prop;
```
- Écrire une propriété  

```
objet.prop = valeurPropriété;
```

### • Méthodes

- Créer une méthode  

```
personne1.infos = function() {
 console.log(this.nom + ' - ' +
 this.prenom + ' - ' + this.age +
 ' ans');
}
```
- Exécuter une méthode  

```
personne1.infos();
```

# CONSTRUCTEUR

```
Function Personne(nom, prenom, age)
{ ... }
```

• Appel

```
var cedric = new Personne('GUILLOU',
'Cédric', 30);
```

OU

```
cedric.bonjour();
```

```
var Personne = function(nom, prenom, age)
{
 this.nom = nom;
 this.prenom = prenom;
 this.age = age;
 this.bonjour = function() {
 console.log('Bonjour ' +
 this.prenom + ' ' + this.nom);
 }
}
```

# PROTOTYPE

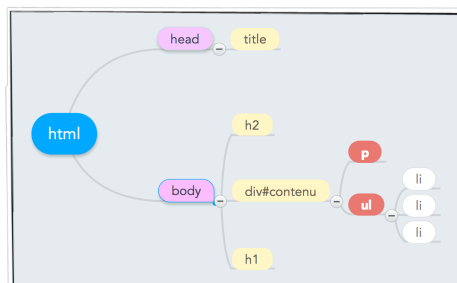
```
Personne.prototype.anniversaire = function() {
 this.age++;
 console.log('Joyeux Anniversaire! Vous avez maintenant ' + this.age + ' ans');
}

cedric.anniversaire();
```

# LE DOM (DOCUMENT OBJECT MODEL)

## LE DOM: INTRODUCTION

- **Document:** Page Web
- **Modèle:** Arbre représentant la structure et le contenu du document
- Javascript reconnaît, peut décrire, lire, et manipuler le DOM de n'importe quelle page Web



# NOEUDS

- 12 types de noeuds, dont 3 principaux
  - ELEMENT
  - ATTRIBUTE
  - TEXT
- ➡ ATTN! les contenus des éléments sont toujours des noeuds TEXT à l'intérieur des noeuds ELEMENT correspondant.

## ACCÉDER AUX ÉLÉMENTS DU DOM

- `getElementById('id');`
  - `document.getElementById('id');`
  - `rootElement.getElementById('id');`
- `getElementsByTagName('tagName');`
- `getElementsByClassName('classe');`
- `element.childNodes();`
- `document.forms['formName']`  
`['formElementName']`

## LIRE ET MODIFIER DES ÉLÉMENTS EXISTANTS

- `myElement.getAttribute('attributeName');`  
ex : `content.getAttribute('align');`
- `myElement.setAttribute('align', 'left');`
  - L'attribut est créé s'il n'existe pas au préalable
- `innerHTML`
- `nodeType`
- `.value`
- `.checked`
- `.selectedIndex`
- `.options[x].selected`
- `.style`
  - `.display`
  - `.color`
  - `.left`
  - `.fontWeight`
  - ...
- ➡ convention de nommage camelCase à partir des propriétés CSS.  
`background-color` > `backgroundColor`

## CRÉER DES ÉLÉMENTS

- `createElement('tagName');`  
ex : `document.createElement('li');`
- `createTextNode('');`
- `parent.appendChild(childElement);`
- `parent.insertBefore(newElement, existingElement);`

# LES ÉVÈNEMENTS

- Exemples
  - › `<button onclick="alert('clic');" >`
  - › `element.onclick = function() {`  
    ...  
};
- Attendre le chargement de la page
  - › `window.onload()`
  - `window.onload = function() { ... };`

## PRINCIPAUX ÉVÈNEMENTS

- onload
- onclick
- onmouseover
- onblur
- onfocus
- onchange
- onkeypress
- onkeydown
- onkeyup
- onsubmit
- ...
  - ➡ `return false` ou `e.preventDefault()` pour interrompre le comportement par défaut (soumission de formulaire, suivi d'un lien...)

## RÉAGIR AUX ÉVÈNEMENTS: LES LISTENERS

- Les évènements sont émis automatiquement en Javascript
- Utilisation des listeners pour les intercepter et y réagir
- Différentes méthodes
  - › `<button onclick="alert('clic');">Afficher une alerte</button>`
  - › `element.eventName = function() { ... }; // eventName: onclick, onblur, ...`
  - › `document.addEventListener('click', myFunction / * handler */ , false)`
- AVANTAGES: plusieurs handlers pour un même évènement
- INCONVÉNIENTS: pas supporté par tous les navigateurs

(IE8-)  
`document.attachEvent('onclick', myFunction);`

L'utilisation d'une librairie comme jQuery peut s'avérer très utile!!

## LES TIMERS

- `setTimeout(callbackFunction, timeout_ms);`
- `setInterval(callbackFunction, interval_ms);`
- `clearTimeout`
- `clearInterval`

# DEBUGGING

- Messages d'erreur (systèmes) dans la console
- Inspection des éléments de la page
  - F8, F10, F11, MAJ+F11
  - Édition d'un point d'arrêt
  - Watch expressions
- Inspection des scripts, points d'arrêts (ex. Chrome Dev Tools)

# BONNE PRATIQUES

- Lisibilité
  - Noms significatifs
  - Code clair
  - Code commenté
- Consistance
  - Conventions de nommage(variables, fonctions, objets)
    - camelCase conseillé
    - Objets: 1ère lettre en majuscule  
var monObjet = new MonObjet();
  - Définir les fonctions avant de les appeler
  - Point-virgule à la fin de chaque instruction
- Blocs
  - indentation
  - toujours utiliser un bloc, même pour une seule instruction
- ...

# BIBLIOTHÈQUES JAVASCRIPT

- Générales
  - jQuery
  - MooTools
  - ...
- Spécialisées
  - Lightbox
  - Scriptaculous
  - jQuery UI
  - Modernizr
  - ...
- Attention aux dépendances
- Attention aux performances

# JQUERY

- Téléchargement: **jquery.com**
- ou CDN (Content Distribution Network)
- Appel du script dans la page Web  
<script type="text/javascript" src="jquery.js"/>

# SÉLECTEURS

- `$('#alert')` // ou `jQuery(...)`
- `$('.alert')`
- `$( 'p' )`
- `$( 'li.menu-item' )`
- `$( 'li.menu-item:first' )`

# MÉTHODES

- `addClass()`
- `removeClass()`
- `toggleClass()`
- `hide()` / `show()`
- `fadeIn()` / `fadeOut()`
- `slideIn()` / `slideOut()`
- ...

# ÉVÈNEMENTS

- `.click()`  
  
`$( 'h2' ).click( function() {  
 $( this ).text( 'nouveau titre' );  
});`
- `.mouseover()`
- `.keypressed()`
- ...
- `$(document).ready(function() {...});`  
  
**AVANTAGE** par rapport à `window.onload`: peut apparaître plusieurs fois.

# JAVASCRIPT ET HTML5

- `getElementsByClassName()`
- `<video></video>`
  - `.play()`
  - `.pause()`
  - `.currentTime`
  - évènements: `play`, `pause`, `ended`, ...
- Stockage hors ligne
  - `localStorage`
  - `IndexedDB`
  - `(WebSQL)`

# SUPPORT NAVIGATEUR

- Inutile de détecter un navigateur ou une version (à éviter)

```
if(navigator.userAgent.indexOf('Netscape')...

if(navigator.appName == 'Microsoft Internet Explorer'...
```

- Tester les fonctionnalités nécessaires

```
if(document.getElementsByClassName (
```

- Modernizr

- Télécharger sur **modernizr.com**

- Appeler le script dans <head>

- Utilisation

```
• if (Modernizr.video) {
 // HTML5 video
} else {
 // Flash par exemple
}
```

```
• Modernizr.audio
```

```
• Modernizr.canvas
```

```
• Modernizr.localStorage
```

```
• Modernizr.websockets
```

```
• ...
```

# EXPRESSIONS RÉGULIÈRES

- Exemple

```
var regExp1 = /terme/; // ou regExp1 = new RegExp('Terme');
```

```
var uneChaine = 'Cette chaine contient-elle le terme?'
```

```
if (regExp1.test(uneChaine) {
 alert('oui');
}
```

# CRÉATION DE MODÈLES (PATTERNS)

```
• var re1 = /^terme/; // au début
```

```
• var re2 = /terme$/; // à la fin
```

```
• var re3 = /ter+me/; // une fois ou plus
 // terme, terrme,
 // terrrrrrrrrrrrme...
```

```
• var re4 = /ter*me/; // zéro ou plus
 // teme, terme,
 // terrrrme...
```

```
• var re5 = /ter?me/; // zero ou une fois
 // teme, terme
```

```
• var re6 = /termelmot/; // OU
```

```
• var re7 = /te.e/; // caractère quelconque
```

```
• var re8 = /\werme/; //alphanumérique ou _
```

```
• var re9 = /\bterme/; // délimitation de mot
```

```
• var re10 = /[tg]erme/; // caractères autorisés
 // terme OK, germe OK,
 // verme KO
```

# EXEMPLE 2: EMAIL

```
• /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/
```

ATTN! Expression non parfaite



# AJAX

- Asynchronous Javascript And XML
- Pur Javascript, pas une technologie à part entière
- Fonctionnement
  - Créer et envoyer une requête au serveur
  - Attendre la réponse (asynchrone)
  - Traiter les données reçues
- Fonctionnalités AJAX de jQuery

## EXEMPLE

```
var req; // la requête

// vérification pour la compatibilité
if (window.XMLHttpRequest) { // Firefox, Safari, Chrome...
 req = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE
 req = new ActiveXObject("Microsoft.XMLHTTP");
}

// préparation à la réception de la réponse
req.onreadystatechange = function() {
 console.info("changement d'état de la requête ->", req.readyState);
 if (req.readyState === 4) {
 console.log(req.responseText);
 }
}

// configurer la requête
req.open("GET", "data.json", true);

// envoyer la requête
req.send(null);
```

## EXERCICES

## TPS

- Compte-à-rebours
- Slideshow
- Réagir aux redimensionnement
  - window.onresize
  - window.innerWidth
  - document.documentElement.clientWidth
  - document.body.clientWidth
- Création d'accordéon avec jQuery et jQuery UI
  - <http://jquery.com/download/>
  - <http://jqueryui.com/>
- "Parsing Sauvage"

## QUIZZ