

# JavaScript

# INTRODUCTION

# Historique

- Créé en 1995 par (et pour) Netscape
- Initialement appelé LiveScript
- Standardisé par ECMA en 1997
- Version actuelle : 1.8.5

# JavaScript

- JavaScript
  - Langage interprété par le navigateur qui a pour but de dynamiser les sites Internet
  - Langage orienté prototype (pas de classe comme en C#)
- ECMAScript
- JavaScript vs. Java

# JavaScript vs. PHP

Caractéristique	PHP	Javascript
Exécution	Exécuté sur le serveur.	Exécuté chez le client.
Nécessaire à l'exécution	Un interpréteur PHP doit être installé sur le serveur.	Tous les navigateurs possèdent un interpréteur Javascript (mais peut être désactivé).
Manipulation de fichiers	Lecture, écriture, ajout possible dans des fichiers texte et éventuellement binaire situés sur le serveur.	Totalement incapable de manipuler les fichiers.
Manipulation de BD	PHP permet d'interroger tout type de base de donnée	Impossible en Javascript.
Richesse	Plus de 2000 fonctions.	Tout au plus une centaine de fonction.
Information sur le serveur	Multitude d'informations concernant le serveur.	Impossible en Javascript.
Information sur le système du visiteur	En dehors du nom du système d'exploitation du visiteur, on ne peut rien obtenir.	En Javascript, il est possible d'établir la résolution de l'écran, ainsi que les plugins ... de l'utilisateur.
Réagir aux événements chez le client	Impossible en PHP, puisqu'il est exécuté côté serveur.	Javascript permet de réagir aux événements : (dé)chargement d'une page, validation de formulaire, clic...

# Exemple javaScript

The screenshot displays a Gmail inbox interface. At the top, there's a Google search bar and navigation icons. Below the search bar, the Gmail logo is visible, followed by a dropdown menu, a refresh button, and a 'More' button. The inbox is organized into tabs: Primary (selected), Social (50+ new), and Promotions (40 new). The left sidebar shows navigation options: Compose, Inbox (7), Starred, Important, Sent Mail, Drafts (2), and Circles. Under Circles, there are links for Notes, Personal, Travel, and More. The main inbox list shows several emails from Google+, Merced Flores, Lisa Paik, Elena Casarosa, Grace Ellington, Henri Rousseau, and Olenna Mason. The bottom section shows storage usage (0.17 GB of 15 GB used) and account activity information.

**Google** [Search Bar] [Grid Icon] [Notification Icon] [Profile Icon]

**Gmail** [Dropdown] [Refresh] [More] 1-7 of 7 [Navigation] [Settings]

**COMPOSE**

**Inbox (7)**  
Starred  
Important  
Sent Mail  
Drafts (2)  
Circles  
Notes  
Personal  
Travel  
More

**Primary** **Social** 50+ new **Promotions** 40 new +

<input type="checkbox"/>	<input type="star"/>	<input type="document"/>	<b>Google+</b>	Julia, a few Google+ posts you m	<b>Jun 9</b>
<input type="checkbox"/>	<input type="star"/>	<input type="document"/>	<b>Merced Flores</b>	Re: consultant for book - Hi Julia,	<b>Mar 18</b>
<input type="checkbox"/>	<input type="star"/>	<input type="document"/>	<b>Lisa Paik</b>	Volunteering at the Lakestone sti	<b>11/20/15</b>
<input type="checkbox"/>	<input type="star"/>	<input type="document"/>	<b>Elena Casarosa</b>	Portrait special - We'd like to annc	<b>11/20/15</b>
<input type="checkbox"/>	<input type="star"/>	<input type="document"/>	<b>Grace Ellington</b>	Volunteer opportunity - I would lik	<b>11/19/15</b>
<input type="checkbox"/>	<input type="star"/>	<input type="document"/>	<b>Henri Rousseau</b>	Niagra falls pictures - Julia, Here i	<b>11/19/15</b>
<input type="checkbox"/>	<input type="star"/>	<input type="document"/>	<b>Olenna Mason</b>	Lakestone student art exhibition	<b>11/19/15</b>

0.17 GB (1%) of 15 GB used [Terms](#) - [Privacy](#) [Manage](#)

Last account activity: 1 hour ago [Details](#)

Julia [Dropdown] [Search]

Henri Rousseau  
Hi!

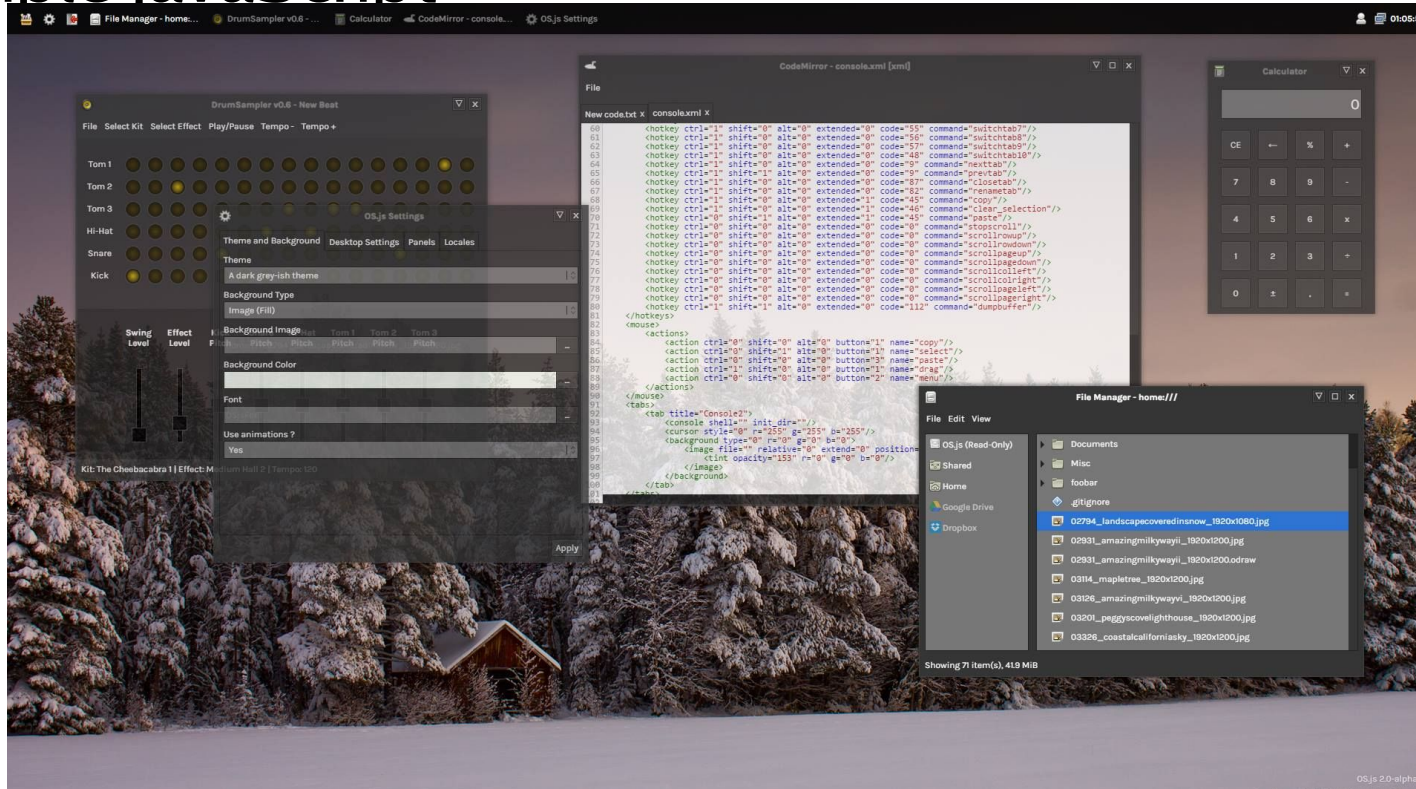
[Profile Icon] [Message Icon] [Phone Icon]

## Exemple lavaScript

The screenshot shows a Google Sheets document titled "Movies\_TechCrunch". The "Data" menu is open, and the "Filter" option is selected. A secondary menu is open for "Filter", showing options like "Create new filter view", "Filter view options", "Best movies (IMDB)", "Highest earning", and "Best movies (Rotten Tomatoes)". The spreadsheet data includes movie titles, release dates, budgets, and IMDB ratings and votes.

	A		E	F	G
1	Title		IMDB Rating		IMDB Votes
2	A Hard Day's Night	11/8/1989	100	7.6	15,291
3	Aliens	9/8/2006	100	7.5	84
4	Annie Get Your Gun	6/20/1975	100		
5	Before Sunrise	6/23/2006	100		
6	Cat on a Hot Tin Roof	6/10/1975	100		
7	Henry V	8/26/1964	100		
8	Iraq for Sale: The War Profiteers				
9	Jaws				
10	Krrish			6.1	2,735
11	Love and Death			7.6	12,111
12	Mary Poppins			7.7	34,302

# Exemple javaScript



OS.js 2.0-alpha



# Exemples d'utilisation

- Exemple de base :
  - <http://jsbin.com/dokogetebe/2/edit>
- Animations en JavaScript :
  - <http://mrdoob.com/projects/chromeexperiments/google-gra...>
  - <http://hereistoday.com/>

# Exemples d'utilisation

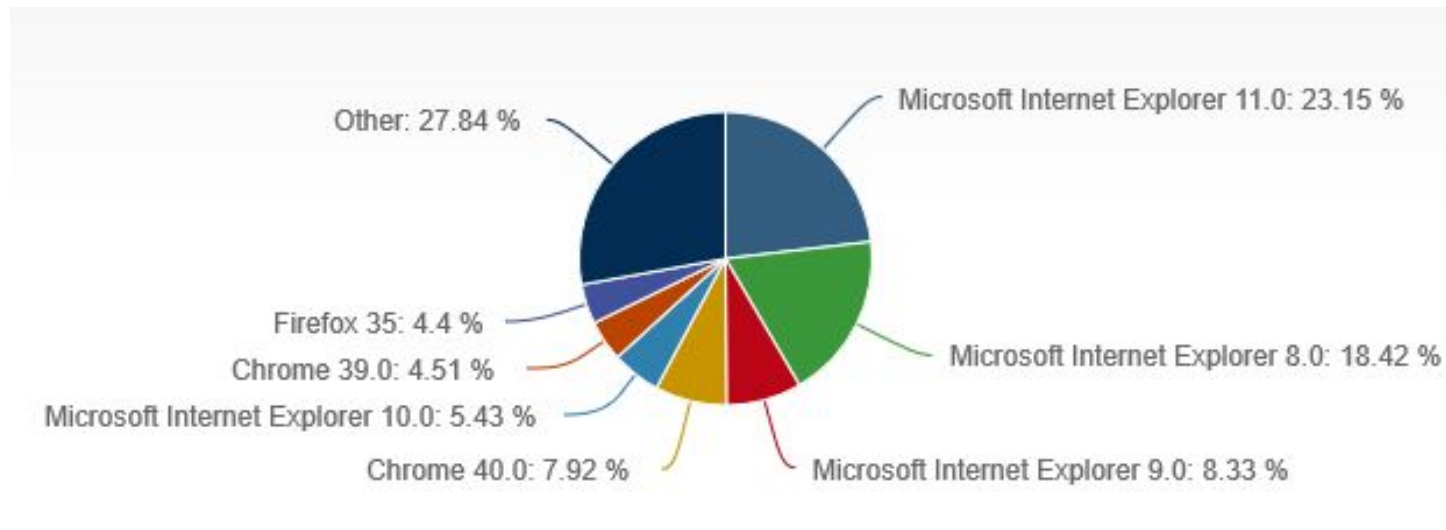
- De la 3D en JavaScript :
  - [http://shapejs.shapeways.com/creator/?li=devhome\\_main](http://shapejs.shapeways.com/creator/?li=devhome_main)
  - [http://mrdoob.github.io/three.js/examples/webgl\\_materials\\_cars.html](http://mrdoob.github.io/three.js/examples/webgl_materials_cars.html)

# Les limites du JavaScript

- Compatibilité entre navigateurs
  - Les différences sont de moins en moins importantes
- JavaScript est un langage exécuté coté client
  - Presque plus aucun internaute bloque l'exécution du JS

# JavaScript et les navigateurs

- Nécessité de vérifier la compatibilité des navigateurs



# Sources, ressources et outils

## Ressources

- JSBin : <http://jsbin.com>
- JSFiddle : <http://jsfiddle.net>
- Rubular : <http://rubular.com>
- Console Debug (F12)

## Outils

- Mozilla Developer Network
- W3Schools
- Developpez.com
- OpenClassRooms

# ÉCRIRE SON CODE

# Écrire du JavaScript

Tout comme pour le HTML et CSS, il ne faut rien de particulier pour écrire du JavaScript.

Un simple éditeur de texte suffit !

Le code est exécuté directement par le navigateur !

# Écrire du JavaScript

Cependant, il existe des éditeurs de texte proposant l'auto complétion pour ce langage :

- Eclipse (gratuit)
- Aptana Studio (gratuit)
- NetBeans (gratuit)
- Free JavaScript Editor (gratuit)
- WebStorm (payant)



# Écrire du JavaScript

- Plugin pour Sublime Text :
- SublimeCodeIntel : <http://sublimecodeintel.github.io/SublimeCodeIntel/>
- Plugin pour Notepad++ :
- Pas de plugin, l'auto complétion est là, même si elle n'est pas parfaite.

# Les commentaires

- Le JavaScript est un langage qui peut vite être incompréhensible.
- Il est important de mettre des commentaires au sein de son code !
- Commentaire sur une ligne :
- Commentaire en bloc :

// Mon commentaire

/\* Mon commentaire  
sur plusieurs  
lignes \*/

# Déboguer son code

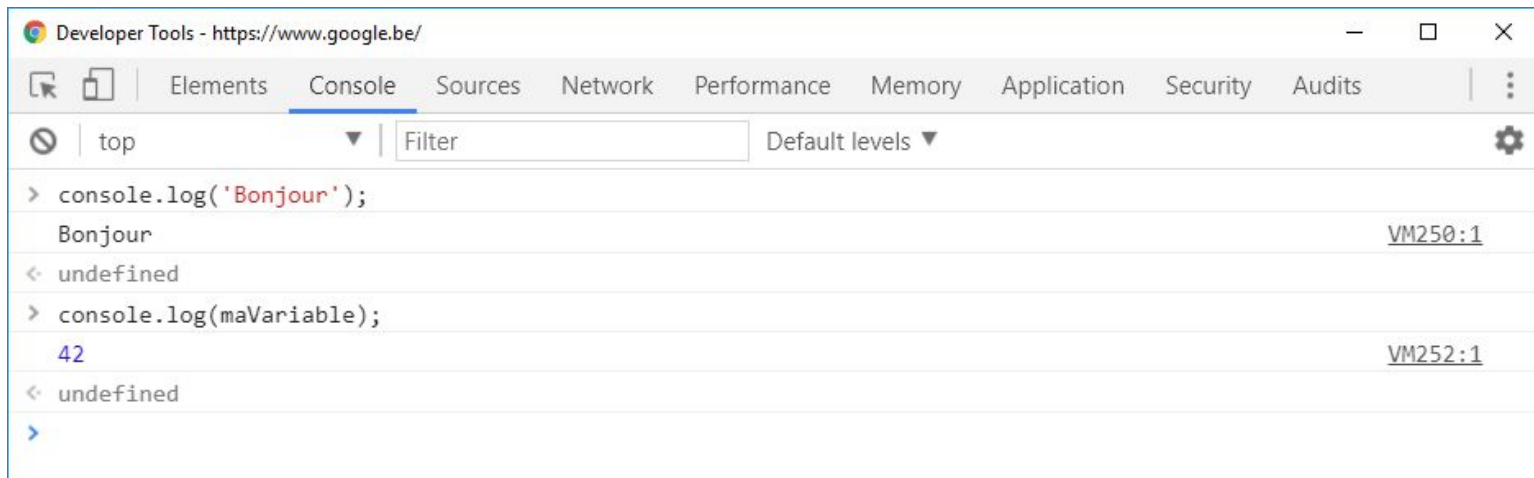
La plupart des navigateurs offrent une console JavaScript.

Celle-ci est utile pour afficher les messages d'erreur ou les messages de débogage.

# Déboguer son code

Pour afficher un message ou une variable dans la console :

```
console.log(...);
```



# EMPLACEMENT DU CODE

# Emplacement du code

Dans une page HTML:

```
<script type="text/javascript">  
    var maVariable;  
    //...  
</script>
```

# Emplacement du code

Dans un fichier externe:

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

Dans un attribut événement • A éviter!

```
<a onclick="var maVariable; //...">Accueil</a>
```

# Emplacement du code

Remarque: le code JS est exécuté par le navigateur de manière séquentielle à la lecture de la page web.

Attention à l'emplacement du code.



# Emplacement du code

```
<!DOCTYPE html>
<html>
<head>
    <meta charset=utf-8 />
    <title>Titre</title>
</head>

<body>
    <div id="01">Bienvenue</div>

<script type="text/javascript">
el = document.getElementById("01");
el.innerHTML = "Bonjour";
</script>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
    <meta charset=utf-8 />
    <title>Titre</title>
</head>

<body>

<script type="text/javascript">
el = document.getElementById("01");
el.innerHTML = "Bonjour";
</script>

    <div id="01">Bienvenue</div>
</body>
</html>
```

# La balise <noscript>

La balise <noscript>...</noscript> permet d'afficher un texte au cas où le navigateur ne supporte pas le JavaScript, ou si celui-ci a été désactivé.

```
<noscript>Votre navigateur ne supporte pas le  
JavaScript ou le JavaScript a été désactivé.</noscript>
```

# Un premier exercice

1. Créez une page HTML contenant le script JS suivant:

```
alert("Bienvenue sur mon site");
```

2. Migrer ce script dans un fichier "script.js" et faites y appel dans votre page HTML

# Solution de l'exercice

## Fichier index.html

```
<!DOCTYPE HTML>
<html>
  <head>
    <style type="text/css" media="all"></style>
  </head>
  <body>
    <h1>Un site avec du JavaScript</h1>
    <div id="contenu">
      <h2>Instructions</h2>
      <p>Grâce à JavaScript, ouvrez une boîte de dialogue au chargement
de la page affichant "Bienvenue sur mon site".</p>
    </div>
    <script type="text/javascript" src="script.js"/></script>
  </body>
</html>
```

## Fichier script.js

```
alert("Bienvenue sur mon site");
```

# CONSTANTES ET VARIABLES

# Constantes prédéfinies : NaN

Not a Number, indique que l'élément concerné n'est pas un nombre valide

- Usage:
  - Plutôt rare, NaN est renvoyé quand une fonction mathématique échoue
  - `Math.sqrt(-1)` ou lorsqu'une conversion échoue `parseInt("blabla")`

# Constantes prédéfinies : NaN

- Tester NaN:
  - `NaN === NaN;` // false
  - `Number.NaN === NaN;` // false
  - `isNaN(NaN);` // true
  - `isNaN(Number.NaN);` // true

# Constantes prédéfinies : Infinity

Infini, indique que l'élément concerné est hors des plages de valeurs définies pour un nombre

- Usage:
  - Infinity se comporte comme l'infini mathématique
  - $\text{Infinity} * \text{Infinity} = \text{Infinity}$
  - $500 / \text{Infinity} = 0$
  - $500 / 0 = \text{Infinity}$
  - $\text{Infinity}/\text{Infinity} = \text{NaN}$



# Constantes prédéfinies : Undefined

Indéfini, l'élément ne correspond à aucun type reconnu

- Tester Undefined:

- `var x;`  
`x == undefined` //true  
`typeof x == 'undefined'` //true
- `var y = 8;`  
`y == undefined` //false  
`typeof y == 'undefined'` //false

# Constantes prédéfinies

Il existe d'autres constantes reliées à des Objets Javascripts :

- Math
- Number

# Constantes de Math

- `Math.PI` → La valeur de  $\pi$  avec 15 décimal.
  - `alert(Math.PI);`                      `//Affiche 3,14159...`
- `Math.SQRT2` → la valeur de racine de 2.
  - `alert(Math.SQRT2);`                      `//Affiche 1,414...`

# Constantes de number

- Number.MAX\_VALUE
  - La plus grande valeur possible (en JS) →  $1.79 \times 10^{308}$
- Number.MIN\_VALUE
  - La plus petite valeur positive (en JS) →  $5.00 \times 10^{-324}$






# Constantes créés par l'utilisateur

Une **convention de nommage** peut être utilisée afin de spécifier que la valeur d'une variable ne doit pas être modifiée

Nom de variable en MAJUSCULE et les mots séparés par un underscore

```
var MA_CONSTANTE = "Bonjour!";
```

# Constantes créés par l'utilisateur

Support		
	IE11	Edge
	Firefox 36	
	Opera 12	
	Chrome 21	
	Safari 5.1	

En ECMAScript 6, le mot clef **const** permet de créer une constante nommée, accessible uniquement en lecture.

Il est nécessaire de l'initialiser lors de sa déclaration.

```
const MA_CONSTANTE = "blablabla";
```

# Variables et typage

- JavaScript est un langage à typage dynamique
- Le type d'une variable est défini au runtime et peut être changé en cours d'exécution
- Nom des variables
  - Commence par une lettre, un underscore ou un dollar
  - Les caractères suivants sont des alphanumériques, underscores ou dollars

```
var variable = "Initialisation";  
//...  
variable = 42;
```

# Variables et typage

Formes littérales:

- Forme littérale pour un nombre entier en base décimale  
`var nombreEntier = 11;`
- Forme littérale pour un nombre réel  
`var nombreReel = 11.435;`
- Forme littérale d'une chaîne de caractères  
`var chaineCaracteres = "Une chaîne de caractères";`



# Variables et typage

Formes littérales:

- Forme littérale d'un tableau normal (object)  
*var tableau = [ "Premier élément", "Second élément" ];*
- Forme littérale d'un tableau associatif/objet  
*var tableauAssociatif = {  
    "cle1" : "valeur1",  
    "cle2" : "valeur2"  
};*

## Où se trouvent les erreurs ?

```
var mavariable = 23;  
var 3fois9 = 27;  
var troisfois9 = 27;  
var le total = 100;  
var @mail = "jean.dupont@gmail.com";  
var variable = 30+20+10;  
var phrase = Bonjour tout le monde !;  
var age = "26";  
var resultat = 3 + 3 + "3";
```

# Variables et typage

## Portée des variables

Attention au lieu de déclaration d'une variable :

```
var maVariable = "Hello";  
//...  
function maFonction(){  
    var maVariable2 = "World";  
    //...  
}
```

Accessible partout dans le code (même dans la fonction)

Accessible uniquement dans la fonction.

# Variables et typage

En javaScript, les types primitifs sont des "pseudo-objets" qui possèdent des propriétés et des méthodes

Boolean	Type dont les valeurs possibles sont true et false.
Null	Type dont l'unique valeur possible est null. Une variable possède cette valeur afin de spécifier qu'elle a bien été initialisée mais qu'elle ne pointe sur aucun objet.
Number	Type qui représente un nombre.
String	Type qui représente une chaîne de caractères
Undefined	Type dont l'unique valeur possible est undefined. Une variable définie possède cette valeur avant qu'elle soit initialisée.
Object	Type instancié grâce au mot clé new

# Variables et typage

```
var variable1;           //Undefined
```

```
var variable2 = null;    //Object
```

```
var variable3 = 12;      //Number
```

```
var variable4 = "Hello"; //String
```

```
var variable5 = true;    //Boolean
```

```
alert("Type de variable : "+(typeof variable));
```

# Variables et typage

Conversions de types primitifs en chaînes de caractères.

```
var booleen = true;  
var nombreEntier = 10;  
var nombreReel = 10.5;
```

```
booleen.toString();      // "true"  
nombreEntier.toString(); // "10"  
nombreReel.toString();   // "10.5"
```

# Variable et typage

JavaScript supporte les conversions de types primitifs en chaînes de caractères en spécifiant la base

```
var nombreEntier = 15;
```

```
nombreEntier.toString();           // "15"
```

```
nombreEntier.toString(2);          // "1111" (Binaire)
```

```
nombreEntier.toString(16);         // "f" (hexadécimal)
```

# Variables et typage

JavaScript supporte les conversions de types chaînes de caractères en nombres entiers ou réels.

```
var entier1 = parseInt("15"); //entier1 contient le nombre 15
```

```
var entier2 = parseInt("f", 16); //entier2 contient le nombre 15
```

```
var reel = parseFloat("15.5"); //reel contient le nombre réel 15,5
```



# Méthodes de détection de types

Nous pouvons utiliser l'opérateur « typeof » pour trouver le type d'une variable. Il nous retournera le type sous forme de string.

```
var prenom = "John";  
alert(typeof prenom);           // Affiche string
```

```
var ok = true;  
alert(typeof ok);               // Affiche boolean
```

```
var prenom = "John";  
alert(typeof prenom == "string"); // Affiche true
```

```
var ok = true;  
alert(typeof ok == "boolean");   // Affiche true
```

# Méthodes de détection de types

- `typeof "John"`  
//Returns "string"
- `typeof 3.14`  
//Returns "number"
- `typeof NaN`  
//Returns "number"
- `typeof false`  
//Returns "boolean"
- `typeof [1,2,3,4]`  
//Returns "object"
- `typeof {name:'John', age:34}`  
//Returns "object"
- `typeof new Date()`  
//Returns "object"
- `typeof function () {}`  
//Returns "function"
- `typeof null`  
//Returns "object"

# Méthodes de détection de types

- Par contre, l'opérateur « typeof » ne pourra pas nous indiquer si notre objet est un tableau (array) ou une date par exemple.
- Pour cela nous pourrons utiliser la propriété « constructor ».

# Méthodes de détection de types

- `"John".constructor`  
`//Returns "function String() { [native code] }"`
- `(3.14).constructor`  
`//Returns "function Number() { [native code] }"`
- `false.constructor`  
`//Returns "function Boolean() { [native code] }"`
- `[1,2,3,4].constructor`  
`//Returns "function Array() { [native code] }"`

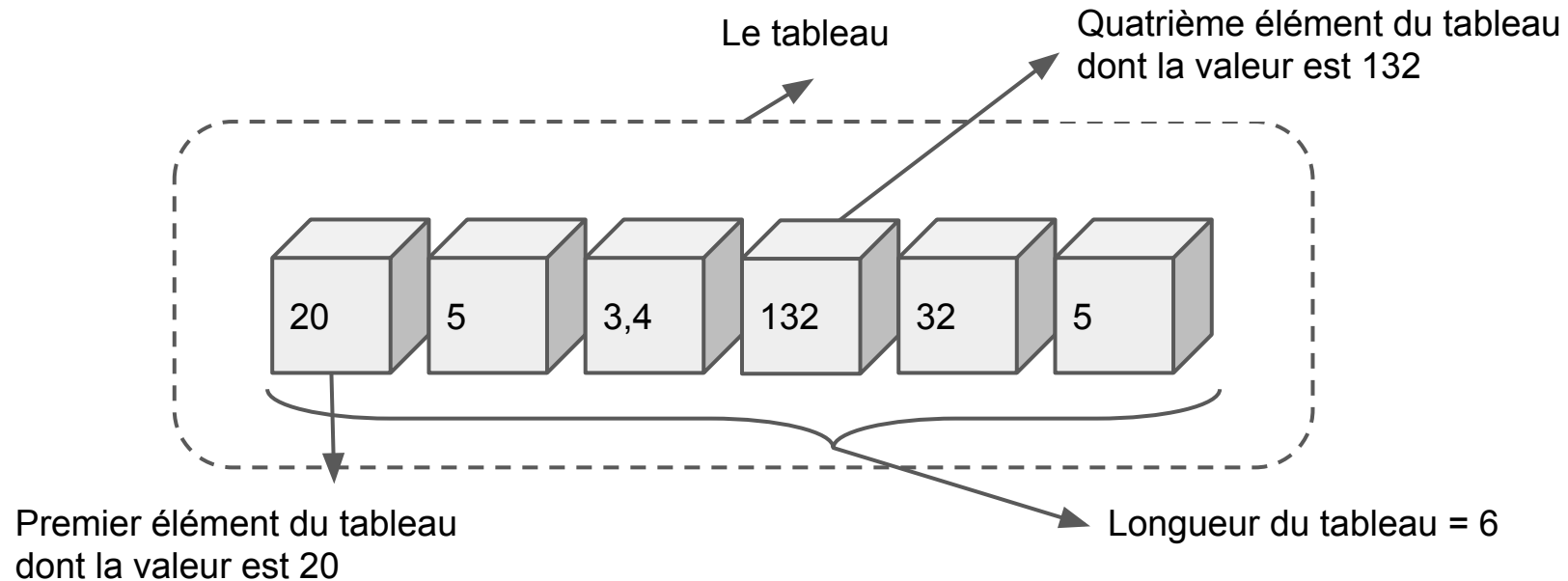
# Méthodes de détection de types

- `{name:'John', age:34}.constructor`  
`// Returns "function Object() { [native code] }"`
- `new Date().constructor`  
`// Returns "function Date() { [native code] }"`
- `function () {}.constructor`  
`// Returns "function Function() { [native code] }"`
- `"John".constructor == String`  
`// Returns "true"`

# TABLEAUX

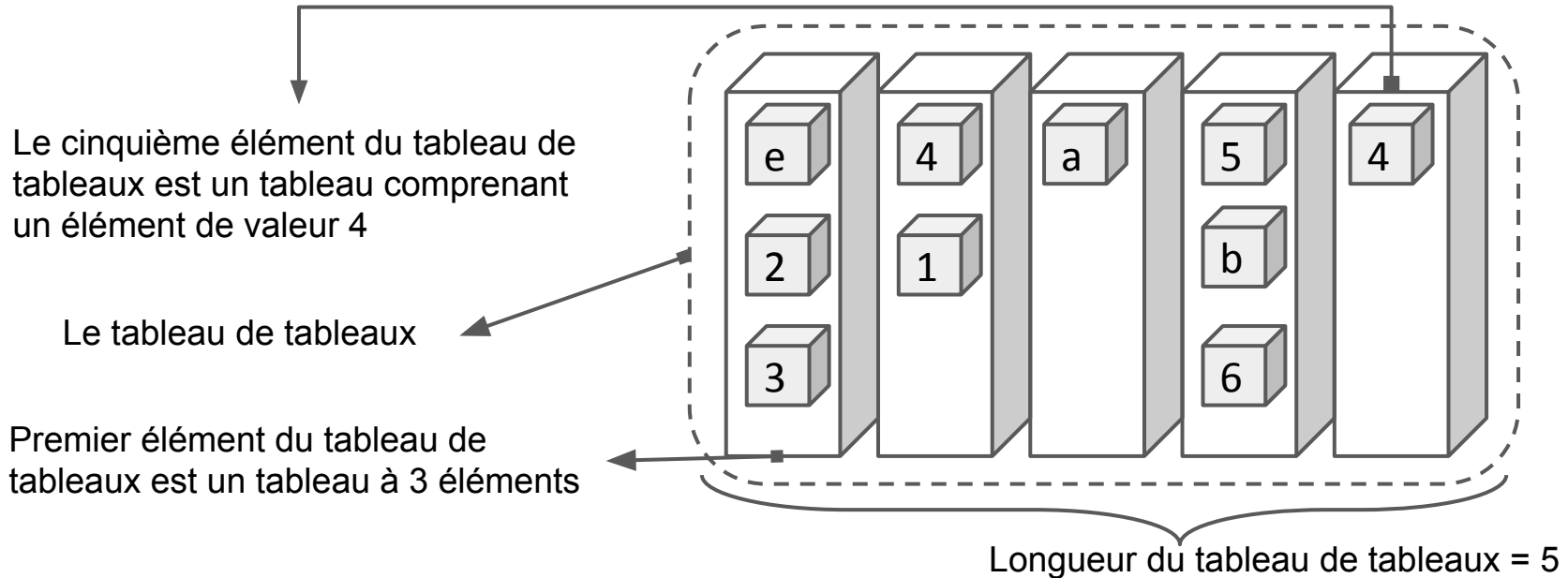
# Tableau

Ensemble de variables auxquelles on accède via un indice



# Tableau de tableaux

Ensemble de tableaux auxquels on accède via un indice





# Tableaux en JavaScript

## Tableaux multi-dimensionnels, associatifs

```
//Initialisation - Possibilité 1
var monTableau = ["Pierre","Paul"];

//Initialisation - Possibilité 2
var monTableau2 = new Array("Pierre","Paul");

//Index commence à 0
monTableau[2] = "Jacques"; //ajoute un 3ième élément
document.alert("Nom: " + monTableau[2]);

//Tableau associatif
var monTableau3 = new Array();
monTableau3["Lundi"] = "Soupe";
```

# Créer un tableau

Les tableaux sont des objets, on peut donc les créer via le mot clé new

```
var tableau = new Array();
```

On peut directement indiquer le nombre d'éléments constitutifs

```
var tableau = new Array(42);
```

# Créer un tableau

Il est aussi possible de remplir le tableau lors de sa création

```
var tableau = new Array("toto",42,mafonction());
```

Le JavaScript nous permet également de créer un tableau « à la volée », sans le mot-clé new :

```
var tableau = ["Hello",42,mafonction()];
```

# Fonctions propres aux tableaux

Il existe un certain nombre de fonctions utiles pour la manipulation des tableaux :

Fonction	
valueOf()	Renvoie la valeur primitive.
toString()	Renvoie les valeurs du tableau sous forme de string séparées par des virgules
join(*)	Idem que toString() mais on peut spécifier le séparateur
pop()	Retire le dernier élément du tableau
push()	Ajoute un élément à la fin du tableau

# Fonctions propres aux tableaux

Fonction	
shift()	Retire le premier élément du tableau, et replace bien les éléments suivants.
unshift()	Ajoute un élément au début du tableau en réarrangeant les éléments suivants.
length	Retourne le nombre d'élément du tableau
splice(a,b,el1,el2,...)	Permet d'ajouter les élément 1, 2,... à partir de l'emplacement a, en supprimant b éléments.
sort()	Permet de trier alphabétiquement le tableau (attention, comportement différent avec des nombres)

# Fonctions propres aux tableaux

Fonction	
<code>reverse()</code>	Inverse les éléments du tableau
<code>sort(function(a,b){return a-b})</code>	Permet de trier un tableau de nombres (croissant)
<code>sort(function(a,b){return b-a})</code>	Permet de trier un tableau de nombres (décroissant)
<code>concat(tab2)</code>	Permet de concaténer le tab2 au tableau
<code>slice(a,b)</code>	Découpe le tableau entre l'élément a et l'élément b

# CHAÎNES DE CARACTÈRES

# Chaînes de caractères

Une chaîne de caractères se déclare simplement entre guillemets (simples ou doubles)

Exemple :

```
var car = "Volvo";  
var car = 'Volvo';  
  
var phrase = "Je m'appelle Jean";  
var phrase = 'Il s\'appelle Jean';
```



# Chaînes de caractères

Une chaîne de caractères peut également se déclarer comme un objet via le mot clé new :

```
var phrase = new String("Hello World");
```

# Chaînes de caractères

Méthode	Paramètre	Description
charAt	Index du caractère dans la chaîne	Retourne le caractère localisé à l'index spécifié en paramètre.
charCodeAt	Index du caractère dans la chaîne	Retourne le code du caractère localisé à l'index spécifié en paramètre.
concat	Chaîne à concaténer	Concatène la chaîne en paramètres à la chaîne courante.
fromCharCode	Chaîne de caractères Unicode	Crée une chaîne de caractères en utilisant une séquence Unicode.
indexOf	Chaîne de caractères	Recherche la première occurrence de la chaîne passée en paramètre et retourne l'index de cette première occurrence.
lastIndexOf	Chaîne de caractères	Recherche la dernière occurrence de la chaîne passée en paramètre et retourne l'index de cette dernière occurrence.

# Chaînes de caractères

Méthode	Paramètre	Description
<code>match</code>	Expression régulière	Détermine si la chaîne de caractères comporte une ou plusieurs correspondances avec l'expression régulière spécifiée.
<code>replace</code>	Expression régulière ou chaîne de caractères à remplacer puis chaîne de remplacement	Remplace un bloc de caractères par un autre dans une chaîne de caractères.
<code>search</code>	Expression régulière de recherche	Recherche l'indice de la première occurrence correspondant à l'expression régulière spécifiée.
<code>slice</code>	Index dans la chaîne de caractères	Retourne une sous-chaîne de caractères en commençant à l'index spécifié en paramètre et en finissant à la fin de la chaîne initiale si la méthode ne comporte qu'un seul paramètre. Dans le cas contraire, elle se termine à l'index spécifié par le second paramètre.
<code>split</code>	Délimiteur	Permet de découper une chaîne de caractères en sous-chaînes en se fondant sur un délimiteur.

# Chaînes de caractères

Méthode	Paramètre	Description
substr	Index de début et de fin	Méthode identique à la méthode <code>slice</code>
substring	Index de début et de fin	Méthode identique à la précédente
toLowerCase	-	Convertit la chaîne de caractères en minuscules.
toString	-	Retourne la chaîne de caractère interne sous forme de chaînes de caractères.
toUpperCase	-	Convertit la chaîne de caractères en majuscules.
valueOf	-	Retourne la valeur primitive de l'objet. Est équivalente à la méthode <code>toString</code> .

# Exercices

- Chaîne : « ma formation javascript »
- Avec la chaîne ci-dessus :
  - Retourner la position de « ma »
  - Indiquer l'indice de la lettre « p »
  - Retrouver la lettre située à l'indice 21
  - Remplacer « javascript » par « Java »
  - Découper la chaîne avec le délimiteur « » (espace)
  - Inverser la chaîne de caractères (+ difficile) :  
« ma formation javascript » → « tpircsavaj noitamrof am »

# DATES

# Date

Le JavaScript nous propose un type d'objet particulier "Date"

- Exemple :

```
var today = new Date();  
alert(today);
```

- Résultat :

Sun Jan 11 2015 05:25:03 GMT+0100 (Central Europe  
Standard Time)

OK

# Date

Le constructeur Date(...) peut prendre plusieurs sortent d'arguments :

- aucun : date actuelle.
- nombre : temps 0 + n millisecondes
- 7 nombres : année, mois, jour, heure, minute, seconde et milliseconde

Note : Temps 0 = 1<sup>er</sup> Janvier 1970 00:00:00:00 UTC



# Date - Exemples

```
var d1 = new Date(); // 23/05/2017 22:40:14  
var d2 = new Date(865734983959); // 08/06/1997 03:56:23  
var d3 = new Date(2014,5,24,11,33,30,0);  
// 24/6/2014 11:33:30  
var d4 = new Date(2013,11,25); // 25/12/2013 0:00:00
```

# Date – Méthodes utiles

Méthode	Description
getFullYear()	Donne l'année en 4 chiffres
getMonth()	Donne le mois (0-11)
getDate()	Donne le jour en tant que nombre (1-31)
getDay()	Donne le jour de la semaine (0-6) ( <i>dimanche à samedi</i> )
getHours()	Donne l'heure (0-23)

# Date – Méthodes utiles

Méthode	Description
getMinutes()	Donne les minutes (0-59)
getSeconds()	Donne les secondes (0-59)
getMilliseconds()	Donne les millisecondes (0-999)
getTime()	Donne le temps absolu (millisecondes depuis le 1/1/1970)

# LES OPÉRATEURS

# Opérateurs

- Opérateurs de calcul

+ Addition

- Soustraction

\* Multiplication

/ Division

% Modulo

# Opérateurs

- Opérateurs d'affectation

= Simple affectation

+= Ajoute ce qu'il y a à droite

-= Retire ce qu'il y a à droite

\*= Multiplie par ce qu'il y a à droite

/= Divise par ce qu'il y a à droite

%= Modulo de la division entière par ce qu'il y a à droite

# Opérateurs

- Opérateurs de comparaison :  $>$ ,  $<$ ,  $>=$ ,  $<=$
- La comparaison de chaînes se fonde sur les codes des caractères (note :  $a > A$ )
- La comparaison d'une chaîne et d'un nombre, convertit automatiquement la chaîne en nombre
  - Si la conversion échoue, la comparaison s'effectue avec NaN et renvoie false

# Opérateurs

- Opérateurs d'égalité

==

=== Pas de conversion de type

!=

!==



# Opérateurs

- Opérateur de concaténation

+            Exemple : `alert("Ma " + "chaîne " + "concaténée.");`

- Opérateurs logiques

!            Négation

&&          ET logique

||          OU logique

# Opérateurs

- Opérateur conditionnel

Permet d'initialiser une variable dont la valeur se fonde sur le résultat d'une condition.

```
var maVariable = (condition) ? (si_vrai) : (si_faux);
```

# Opérateurs

- Opérateurs unaires
  - `typeof`  
Détermine le type d'une variable sous forme d'un string
  - `New`  
Création d'un objet
  - `Delete`  
Permet de retirer une propriété donnée d'un objet

# Opérateurs

## Exemple Delete

```
var Employe = {  
    age: 28,  
    nom: "John",  
    designation: "developpeur"  
}  
  
console.log(delete Employe.nom);           //renvoie true  
console.log(delete Employe.salaire);        //renvoie true
```

# Opérateurs

- Void

Souvent utilisé pour obtenir la valeur undefined avec void(0).

Dans le cadre d'une URI qui est évaluée, le résultat remplace le contenu de la page, sauf si la valeur renvoyée vaut undefined.

```
<a href = "javascript:void(0);"> cliquer (sans effet) </a>
```

# Opérateurs

- ++var

```
var x = 3;
```

```
y = ++x;
```

Résultat : y = 4, x = 4

- var++

```
var x = 3;
```

```
y = x++;
```

Résultat : y = 3, x = 4

# Exercice

- Calcul de la TVA

Écrire un programme qui :

1. Demande à l'utilisateur un prix unitaire hors taxe d'un livre
2. Demande à l'utilisateur la quantité de livre
3. Calcule et affiche le prix total TTC de la commande, en utilisant une TVA de 21%

Pour interagir avec l'utilisateur, vous utiliserez les fonctions d'entrée/sortie `prompt()` et `alert()`.

# STRUCTURE DE CONTRÔLE ET EXCEPTIONS



# Condition : if... else...

- Syntaxe

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

# Condition : switch

- Syntaxe

```
switch( variable ) {  
    case valeur :  
        //...  
        break;  
    default :  
        //...  
}
```

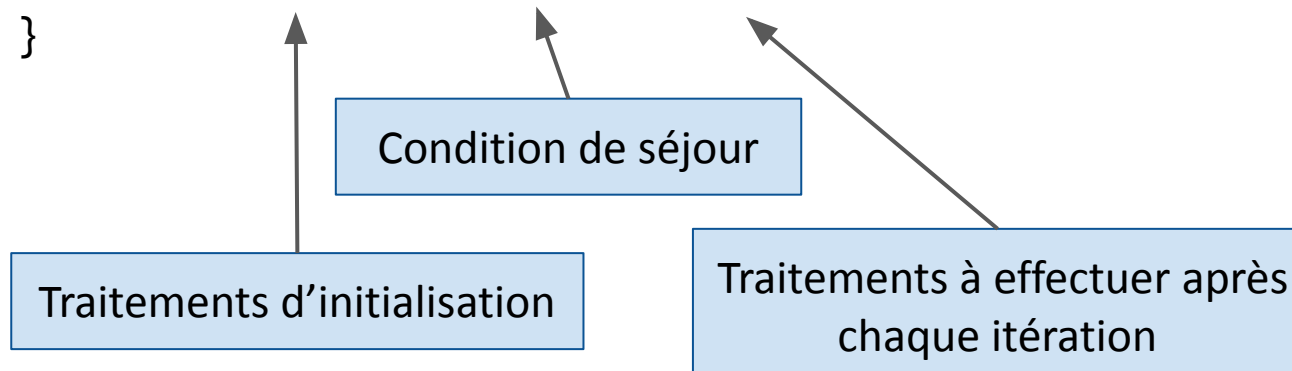
# Exemple de switch

```
switch (expr) {  
  case "Pommes":  
    console.log("Pommes : 0.32 € le kilo.");  
    break;  
  case "Cerises":  
    console.log("Cerises : 3.00 € le kilo.");  
    break;  
  case "Mangues":  
  case "Papayes":  
    console.log("Mangues et papayes : 2.79 € le kilo.");  
    break;  
  default:  
    console.log("Désolé, nous n'avons plus de " + expr + "");  
}
```

# Boucle : for

- Syntaxe

```
for( var cpt=0; cpt<10; cpt++ ) {  
    //...  
}
```



# Boucle : foreach

- Syntaxe

```
for( variable in structure ) {  
    //...  
}
```

- Exemple

```
var tab = ["a", "b", "c", "d"];  
for( var cle in tab ) {  
    alert(cle + ": " + tab[cle]);  
}
```

# Boucle : while

- Syntaxe

```
while( condition de séjour) {  
    //...  
}
```

- Exemple

```
var nombre = 0;  
while( nombre < 10 ) {  
    //...  
    nombre++;  
}
```

# Boucle : do .. while

- Syntaxe

```
do {  
    //...  
} while( condition de séjour);
```

- Exemple

```
var nombre = 0;  
do {  
    //...  
    nombre++;  
} while( nombre < 10 );
```

# Break / Continue

Le mot clé break permet de sortir d'une boucle.

Le mot clé continue permet de sortir d'une itération.

/!\ Il faut utiliser ces mots clés avec parcimonie /!\



# Exercice

- Utilisez l'objet Date et des structures conditionnelles, écrivez un programme qui affiche le jour de la semaine.

Exemple : « Bonjour, nous sommes lundi! »

# Exercice

- Réalisez un programme qui permet d'afficher, dans la console, la structure suivante à l'aide d'une boucle :

```
"A "  
"AA "  
"AAA "  
"AAAA "  
"AAAAA "  
"AAAAAA "  
"AAAAAAA "  
"AAAAAAA "  
"AAAAAAA "  
"AAAAAAA "
```

# FONCTIONS

# Fonctions

- Déclaration de la fonction

```
function nomFonction(arg1, arg2)
{
    var c = 10;
    var result;
    result = (arg1 + arg2) * c;
    return result;
};
```

Nom de la fonction

Signature de la fonction

Variable locale

Corps de la fonction

# Fonctions

- Appel de la fonction

```
var a = 1;  
var b = 2;  
var res;  
res = nomFonction(a, b);
```

# Fonctions : les paramètres

- Paramètres formels: Paramètres utilisés dans le corps de la fonction
  - Ex: arg1, arg2 sur le premier slide
- Paramètres effectifs: Variables utilisées lors de l'appel d'une fonction
  - Ex: a, b sur le deuxieme slide

# Votre première fonction !

- Créez une fonction « inverser(chaine) » qui effectuera une inversion des caractères d'une chaîne et affichera le résultat en console et en alerte.

Pour rappel : Inverser les caractères se fait en 3 étapes !

# Exercice sur les fonctions

```
function maFonction(a, b, c, d, e, f){  
    var res;  
    res = a + b;  
    res = res * d;  
    e = f;  
    res = res - e;  
    return res;}  
var a = 2;  
var b = 3;  
var c = 4;  
var d = 5;  
var e = 6;  
var f = 7;  
var g = maFonction(f, e, d, c, b, a);
```

Que valent g et e après l'appel de la fonction?



# Fonctions : les arguments

- Arguments :

Type simple et string passés par valeur

Type complexe passé par référence

Tous les arguments sont optionnels

# Valeur VS Référence

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

```
function emptyMe(arg1)
```

```
{
```

```
    arg1.value = "";
```

```
}
```

```
</script>
```

```
...
```

```
<input type="text" value="Howdy" onchange="emptyMe(this)">
```

Objet

OK!

---

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

```
function emptyMe(arg1)
```

```
{
```

```
    arg1 = "";
```

```
}
```

```
</script>
```

```
...
```

```
<input type="text" value="Howdy" onchange="emptyMe(this.value)">
```

Valeur

KO!

# Fonctions anonymes

- Permet de créer des « variables » contenant des fonctions.

```
var x = function (arguments) {  
    // Le code de votre fonction anonyme  
};  
  
x(arguments);
```

# Fonctions anonymes

- Les fonctions anonymes ont également une utilité pour isoler son code.

```
(function () {  
    // Code isolé ...  
})();
```

# Gestion des exceptions

Il est possible de « tester » du code en utilisant la structure try/catch.

Le navigateur va essayer d'exécuter le code situé dans le bloc try. Le bloc catch contient le comportement à avoir lorsqu'une erreur est lancée.

# Gestion des exceptions

Le mot clé throw permet de gérer ses propres messages d'erreurs personnalisés.

Throw lance une exception. Cette exception est soit une chaîne de caractères, soit un chiffre ou encore un booléen.

```
<p>Please input a number between 5 and 10 : </p>
```

```
<input id='demo' type='text' />  
<button type='button' onclick="myFonction()">Test Input</button>  
<p id='message'></p>
```

```
<script type="text/javascript">  
    function myFonction() {  
        var message, x;  
        message = document.getElementById('message');  
        message.innerHTML = "";  
        x = document.getElementById('demo').value;  
        try {  
            if(x == "") throw 'Empty';  
            if(isNaN(x)) throw 'Not a number';  
            x = Number(x);  
            if(x < 5) throw 'Too low';  
            if(x > 10) throw 'Too high';  
        }  
        catch(err) {  
            message.innerHTML = 'Input is : ' + err;  
        }  
    }  
</script>
```

DOM

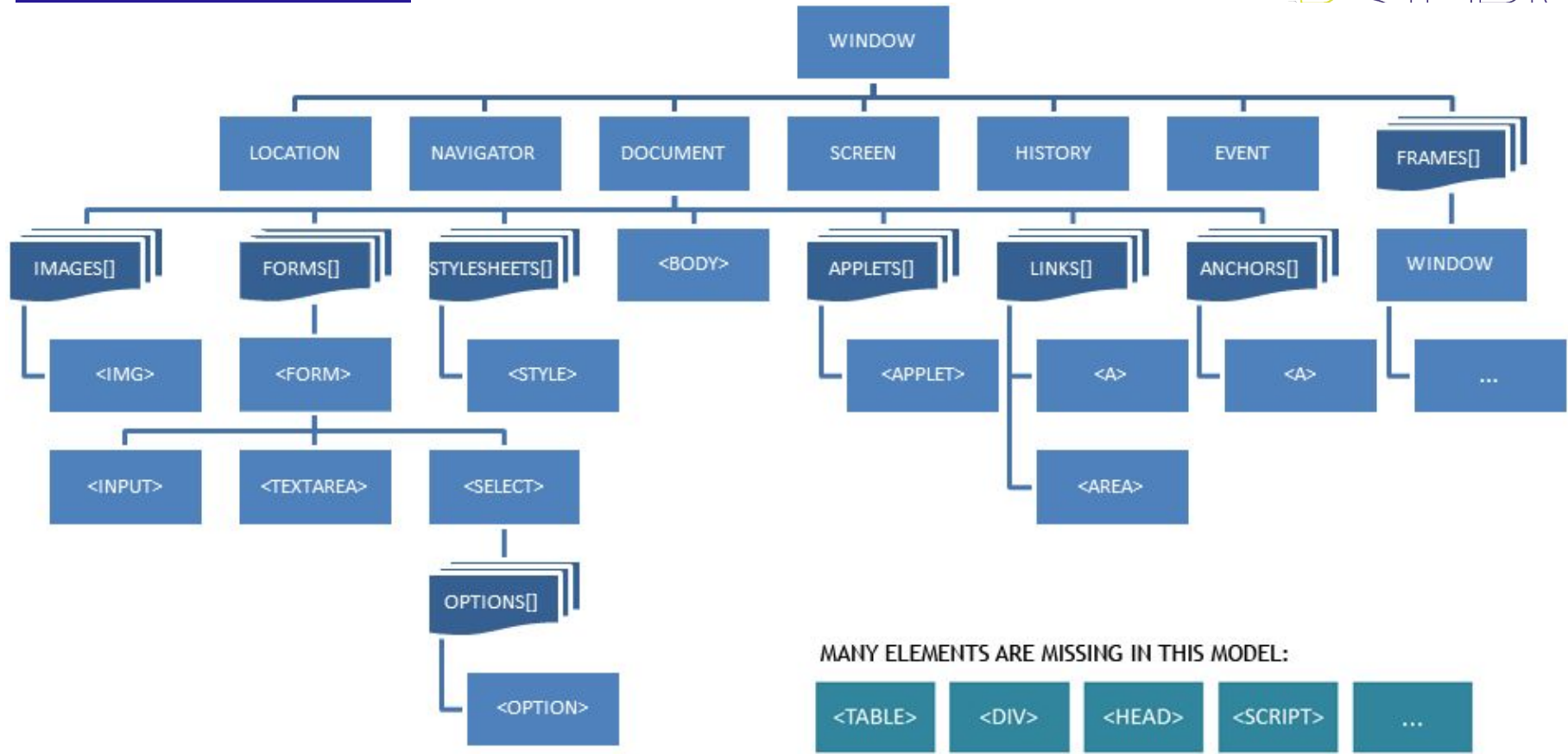


# Document Object Model

L'API du DOM (Document Object Model) permet d'accéder à une page Web et de manipuler son contenu, sa structure ainsi que ses styles

DOM présente un document sous la forme d'un arbre de noeuds

[https://developer.mozilla.org/en-US/docs/DOM/DOM\\_Reference](https://developer.mozilla.org/en-US/docs/DOM/DOM_Reference)



# Objets du DOM

- window : Représente la fenêtre du navigateur où le code HTML est affiché.
- navigator : Objet en lecture seule. Représente les caractéristiques du navigateur utilisé.
- screen : Objet en lecture seule permettant d'accéder aux caractéristiques de l'écran affichant la page.

# Objets du DOM

- location : Cet objet possède les caractéristiques de la page actuelle (url, protocole, ...)
- document : C'est l'objet qui contient toute notre page HTML. Cet objet possède une très grande quantité de propriétés et fonctions.

# Afficher les propriétés d'un objet

```
function afficherProprietes(obj){  
    for(prop in obj) {  
        var valeur = obj[prop];  
        document.getElementById("info").innerHTML +=  
            "<br>" + prop + ":" + valeur;  
    }  
}  
afficherProprietes(window);
```

# Nommer les objets

- Pour manipuler les différents objets d'une page Web (paragraphes, divs, images, ...), il faut pouvoir les différencier.
- On utilise très souvent l'id pour nommer un élément.
- Pour rappel : l'id d'un élément doit être unique au sein de la page.

# DOM Document

L'objet Document est l'élément racine d'un document (page Web, document XML, ...)

Il hérite des méthodes et propriétés de l'objet Noeud

# DOM Document - Méthodes

- `write(string)`

Écris dans la page. À n'utiliser que lors du chargement de la page ! Écrase tout ce qui se trouve déjà dans la page.

Paramètres :

string : STRING

Return :

/

Exemple :

```
document.write("Hello World");
```



# DOM Document - Méthodes

- getElementById(elementID)

Retourne l'élément ayant l'attribut id spécifié

Paramètres:

elementID: STRING

Return:

ELEMENT (OBJET !!)

Exemple :

```
var maDiv = document.getElementById("demo");
```

# DOM Document - Méthodes

- `getElementsByTagName(tagName)`

Retourne une collection d'éléments ayant le nom spécifié

Paramètres :

`tagName: STRING`

Return :

`NODELIST (OBJET !)`

Exemple :

```
var nodes = document.getElementsByTagName("h1");
```

# DOM Document - Méthodes

- createElement(nodeName)

Retourne l'élément créé. Ne le place pas dans la page.

Paramètres :

nodeName: STRING

Return :

ELEMENT

Exemple:

```
var titre = document.createElement("h1");
```

# DOM Node

Représente un noeud dans un document HTML

Il existe 12 types de noeuds HTML, dont:

- Element
- Attr
- Text
- Comment

<https://developer.mozilla.org/en-US/docs/Web/API/Node.nodeType>

# DOM Node - Propriétés

- attributes

Retourne une collection (NamedNodeMap) contenant les attributs d'un noeud.

- childNodes

Retourne une collection (NodeList) contenant les noeuds enfants d'un noeud.

# DOM Node - Propriétés

- firstChild

Retourne le premier noeud enfant d'un noeud.

- lastChild

Retourne le dernier noeud enfant d'un noeud.

- parentNode

Retourne le noeud parent.

# DOM Node - Propriétés

- nodeName

Retourne le nom d'un noeud, c-à-d.:

- Le nom de la balise pour les noeuds Element
- Le nom de l'attribut pour les attributs
- ...

- textContent

Retourne le texte d'un noeud et de ses descendants.

# DOM Node - Propriétés

- `previousSibling`

Retourne le noeud précédent du niveau identique du noeud cible.

- `nextSibling`

Retourne le noeud suivant du niveau identique du noeud courant.

- `nodeValue`

Retourne la valeur d'un noeud.



# DOM Node - Propriétés

- `nodeType`

Retourne le type d'un noeud, c-à-d.:

- 1 pour les noeuds Element
- 2 pour les attributs
- 3 pour le text
- ...
- [http://www.w3schools.com/jsref/prop\\_node\\_nodetype.asp](http://www.w3schools.com/jsref/prop_node_nodetype.asp)

# DOM Node - Méthodes

- appendChild(node)

Ajoute un noeud en tant que dernier enfant

Paramètres :

node: NODE

Return :

NODE

Exemple:

```
var node=document.getElementById("myList2").lastChild;  
document.getElementById("myList1").appendChild(node);
```

# DOM Node - Méthodes

- insertBefore(newNode, existingNode)

Ajoute un noeud juste avant le noeud enfant spécifié

Paramètres :

newNode: NODE

existingNode: NODE

Return :

NODE

# DOM Node - Méthodes

- removeChild(node)

Supprime le noeud enfant spécifié

Paramètres :

node: NODE

Return :

NODE

Exemple :

```
var list=document.getElementById("myList");  
list.removeChild(list.childNodes[0]);
```

# DOM Element

- Manipuler les attributs

Méthode	Paramètre	Description
getAttribute	Identifiant de l'attribut	Référence un attribut d'un élément en utilisant son identifiant.
hasAttribute	Identifiant de l'attribut	Détermine si un attribut est présent pour un élément.
removeAttribute	Identifiant de l'attribut	Supprime un attribut pour un élément.
setAttribute	Identifiant de l'attribut ainsi que sa valeur	Crée un attribut ou remplace un attribut existant d'un élément.

# DOM Element - Propriétés

- `previousElementSibling`

Retourne le noeud élément précédent du niveau identique du noeud courant

- `nextElementSibling`

Retourne le noeud élément suivant du niveau identique du noeud courant

# DOM HTML Element

- innerHTML

Cet attribut permet d'accéder ou de remplacer complètement le contenu d'un élément par celui spécifié dans une chaîne de caractères.

# DOM HTML Element

- `querySelector(cssSelector)`

Retourne le premier élément satisfaisant le selecteur CSS

Paramètres :

CSSselector : STRING

Return :

ELEMENT

Exemple :

```
var element = document.querySelector("h1");
```



# DOM HTML Element

- `querySelectorAll(cssSelector)`

Retourne une collection d'éléments satisfaisants le sélecteur CSS

Paramètres :

CSSselector : STRING

Return :

NODELIST

Exemple :

```
var elements = document.querySelectorAll("h1");
```

# LES TIMERS

# Les timers

Les timers nous permettent d'exécuter des actions après un certain délai ou d'exécuter des actions toutes les "n" secondes.

- `setTimeout(fonction, compte à rebours)`
- `setInterval(fonction, compte à rebours)`

# Les timers

- `setTimeout`

Prend deux paramètres:- la fonction à exécuter

- le délai en millisecondes

Le délai est un compte à rebours qui permettra de déclencher la fonction.

Une fois l'exécution de la fonction lancée, le compte à rebours n'est pas déclenché à nouveau.

# Les timers

- `clearTimeout()`

Prend en paramètre le timer (généralement stocké dans une variable).

Permet l'arrêt du compte à rebours avant le déclenchement de la méthode.

# Les timers

- `setInterval`

Prend deux paramètres:- la fonction à exécuter

- le délai en millisecondes

Le délai est un compte à rebours qui permettra de déclencher la fonction.

Ici la différence c'est que le compte à rebours se déclenchera en boucle.

# Les timers

- `clearInterval()`

Prend en paramètre le timer (généralement stocké dans une variable).

Permet l'arrêt du compte à rebours.

# Exercice

- Affichez l'heure actuelle (heure : minutes : secondes) dans le titre de la fenêtre de votre navigateur en utilisant `setTimeout` et ensuite `setInterval`.
- Affichez la date et l'heure sur votre page web.

**Mardi 25 Avril**

**13:16:32**



# MANIPULER LE CSS

# Manipuler le CSS

S'il est possible de manipuler le code HTML, il est également possible de modifier le CSS d'une page web !

```
document.getElementById("toto")  
    .style.backgroundColor = "red";
```

# Manipuler le CSS

La manipulation du CSS se fait à l'aide de l'attribut style qui est propre aux éléments.

L'attribut style possède toutes les propriétés CSS, il faut néanmoins faire attention aux changement de nom dans certain cas.

# Manipuler le CSS

element.style

```
.background = "#f3f3f3 url('img.png') no-repeat right top";  
.backgroundColor = "red";  
.display = "none";  
.fontSize = "35px";  
.position = "absolute";  
.top = "210px";  
.left = "100px";
```

# Manipuler le CSS

Le seul « problème », c'est que le style CSS est ajouté dans le code HTML. Il s'agit de style in-line !

Essayez de modifier le CSS d'un élément, remarquez que le style s'applique avec l'attribut `style="..."`

# Limites

Essayez le code suivant (sur un élément stylisé) :

```
var elem = document.getElementById("toto");  
elem.style.color = "red";  
console.log(elem.style.color);
```

Quel est le message affiché en console ?

# Limites

Le JavaScript ne sait lire que les propriétés CSS inline !

Il faudra utiliser la fonction `getComputedStyle()` pour lire le CSS externe.

```
//Fonctionnement :  
var elem = document.getElementById("toto");  
var styles = getComputedStyle(elem);  
var color = styles.color;
```

```
//Pour faire plus rapide :  
var color = getComputedStyle(document  
    .getElementById("toto")).color;
```

# Limites

Pour récupérer le positionnement, il faut s'y prendre autrement. On utilise les propriétés offset.

- `offsetWidth` → `width+padding+border` !
- `offsetHeight` → `height+padding+border` !
- `offsetLeft`
- `offsetTop`
- `offsetParent` → contient l'objet de l'élément parent par rapport auquel est positionné l'élément actuel.



# LES ÉVÉNEMENTS

# Événements

Les objets du DOM peuvent réagir à des événements. L'exemple le plus commun d'événement est le clic sur un élément.

- Exemple

- html

```
<button onclick="afficherText()">Cliquer ici</button>  
<p id="info"></p>
```

- JS

```
function afficherText() {  
    document.getElementById("info").innerHTML = "Hello World";  
}
```

# Événements

- Ajouter une action à un élément pour un événement :

```
document.getElementById("toto").onclick = maFonction;
```

```
// OU
```

```
document.getElementById("toto")  
.addEventListener("click",maFonction);
```

# Événements

Si l'on veut passer des arguments à la fonction qui est lancée, il faut passer par une fonction intermédiaire :

```
document.getElementById("toto").onclick = function (e){  
    maFonction(arg1, arg2);  
}
```

# Événements

Descriptif	addEventListener(...)	Ajout direct
Le contenu d'un champ change	change	onchange
Click de souris sur un élément	click	onclick
Double click sur un élément	dblclick	ondblclick
Si une erreur apparaît lors du chargement de la page, d'une image...	error	onerror
L'élément reçoit le focus	focus	onfocus
L'élément perd le focus	blur	onblur

# Événements

Descriptif	addEventListener(...)	Ajout direct
Une touche est pressée	keydown	onkeydown
Une touche est relâchée	keyup	onkeyup
Une touche de caractère est pressée	keypress	onkeypress
L'élément est chargé	load	onload
L'utilisateur sort de la page	unload	onunload
Le bouton de la souris est pressé	mousedown	onmousedown
Le bouton de la souris est relâché	mouseup	onmouseup
La souris est bougée	mousemove	onmousemove

# Événements

Descriptif	addEventListener(...)	Ajout direct
La souris survole un élément	mouseover	onmouseover
La taille de l'élément est réajustée	resize	onresize
Du texte est sélectionné	select	onselect

# Exercice

- Créez un programme qui permet d'ajouter un élément à une liste lorsque l'on clique sur un bouton.

Rappels utiles :

```
document.createElement(TypeElement); // Créer un élément
element.innerHTML = "chaîne";         // Modifier le contenu
element.appendChild(element);         // Ajouter à la suite
```



# Exercice

- En reprenant le code précédent, permettre à l'utilisateur d'écrire l'élément à l'aide d'un input.

Faites attention à ce que le champ soit rempli ;-)

## Exercice

### Descriptif Produit

Chaise	25	Ajouter au panier
Table	150	Ajouter au panier
Meuble TV	250	Ajouter au panier

### Mon panier

Chaise	25
Chaise	25
Table	150
Chaise	25
Chaise	25

Prix Total : 250

- Créez le programme permettant d'ajouter des articles dans un panier. Il faut pouvoir calculer le total des achats en temps réel.

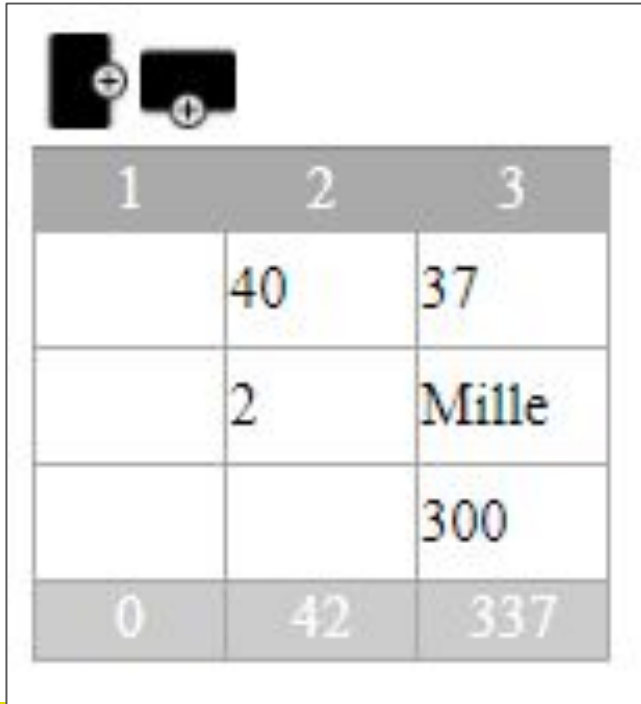
# Exercice

- Reprenez le programme du panier.
  - Modifiez l'affichage du panier pour que celui-ci soit agrégé.

Chaise	25	4
Table	150	1
Prix Total :		250

- Ajoutez la possibilité de supprimer un article du panier.  
Il faudra bien sûr recalculer le total.

# Exercice Supplémentaire



1	2	3
	40	37
	2	Mille
		300
0	42	337

- Créez une page web contenant un tableau dynamique.
- Des lignes et des colonnes peuvent être ajouter des boutons.
- Du contenu peut être ajouter par l'utilisateur dans les cellules.
- La dernière ligne (tfoot) contient la somme de la colonne.

# LES FORMULAIRES

# Objectifs des formulaires

- La création d'un formulaire se fait via la balise HTML <form>
- Les formulaires sont utilisés pour récolter des informations des utilisateurs
- 2 problèmes :
  - Comment faire transiter les données ?
  - Comment traiter les données reçues ?

# Comment faire transiter les données ?

- 2 méthodes:
  - GET: Fait transiter les données via l'adresse de la page
    - <https://www.monsite.com/page.php?cidReset=true&cidReq=B11B15>
    - Limite à 255 caractères
  - POST: Fait transiter les données via la requête HTTP
    - Permet de faire transiter un plus gros nombre de caractères
- Définit avec l'attribut method
  - `method="get"`
  - `method="post"`

# Comment faire transiter les données ?

- Il faut envoyer la requête contenant les données du formulaire (envoyé par GET ou POST) à un script qui pourra les traiter (ex. page contenant du PHP)
- Définit avec l'attribut action
  - ex: action="/register.php"



# Exemple de formulaire

## Déclaration d'un formulaire

```
<form name="MyForm" method="get" action="./register.php">  
  <input type="text" name="nom" />  
  <input type="text" name="prénom" />  
  <input type="submit" />  
</form>
```

<input type="text" value="Jean"/>	<input type="text" value="Dupont"/>	<input type="submit" value="Envoyer"/>
-----------------------------------	-------------------------------------	--

# La balise <input>

- L'élément le plus important d'un formulaire est la balise <input>
- <input> est utilisé pour recueillir l'information de l'utilisateur
- <input> peut être utilisé de nombreuses manières différentes en fonction de la valeur de son attribut type

# La balise <input>

- Champs de texte

```
<input type="text" name="prenom">
```

Prénom:

- Mot de passe

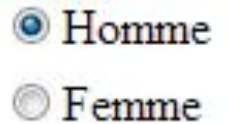
```
<input type="password" name="pwd">
```

```
<input type="submit" value="S'inscrire">
```

# La balise <input>

- Bouton radio

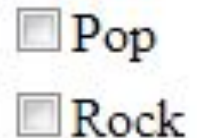
```
<input type="radio" name="sexe" value="male">Homme  
<br>  
<input type="radio" name="sexe" value="female">Femme
```



**ATTENTION : La valeur de "name" doit être la même pour tous les radio boutons!**

- Checkbox

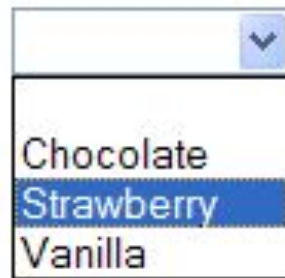
```
<input type="checkbox" name="type_musique" value="pop">Pop  
<br>  
<input type="checkbox" name="type_musique" value="rock">Rock
```



# La balise <select>

- Select

```
<select name="voiture">  
  <option value=""></option>  
  <option value="choco">Chocolate</option>  
  <option value="straw" selected>Strawberry</option>  
  <option value="vanilla">Vanilla</option>  
</select>
```



```
<optgroup label="Europe">  
  <option value="straw" selected>Strawberry</option>  
  <option value="vanilla">Vanilla</option>  
</optgroup>
```

# La balise <textarea>

- Zone de texte

```
<textarea rows="10" cols="30">
```

Ceci est un texte déjà inscrit dans la zone de texte.

```
</textarea>
```

Commentaires:



# L'attribut value

Lorsque le formulaire est envoyé, ce sont les valeurs des attributs “value” qui sont envoyés au serveur lié au nom de l'input (name)

Ecrire dans un “input” de type texte modifie son attribut “value”

# La balise <label>

- Label

Permet de donner un libellé à un input

Peut être lié avec les attributs for et id

```
<label for="firstname">Prénom</label>  
<input type="text" id="firstname" name="prenom">
```



# La balise <datalist>

- Datalist

- La balise datalist spécifie une liste d'options prédéfinies pour un input
- Utile pour de l'autocomplétion (affichage d'une dropdown list)
- La datalist est liée à l'input via les attributs list et id

```
<input list="browsers">  
<datalist id="browsers">  
  <option value="Internet Explorer">  
  <option value="Firefox">  
  <option value="Chrome">  
  <option value="Opera">  
  <option value="Safari">  
</datalist>
```

# Nouveaux types d'input – HTML5

- color
- date
- datetime-local
- email
- month
- number
- range
- search
- tel (utile pour mobile, clavier adapté)
- time
- url
- week

Attention à la compatibilité avec les navigateurs !

# Attributs de formulaire – HTML5

Nouveaux attributs pour <input>:

- formaction (input)
- formmethod (input)
- formnovalidate
- formtarget
- height et width
- list
- autocomplete
- autofocus
- checked (checkbox, radio)
- disabled
- form (permet de désigner plusieurs forms)

# Attributs de formulaire – HTML5

Nouveaux attributs pour <input>:

- placeholder (exemple d'input de l'élément)
- required
- value
- ...
- min et max (nombre, date)
- maxlength (nombre caractère max.)
- multiple (email, file)
- pattern (regexp)

# Objet Form du DOM

JavaScript permet de travailler sur les formulaires via un objet prédéfini Form

Il existe un objet Form pour chacun des formulaires d'un document HTML

# Objet Form du DOM

- Pour accéder aux formulaires

```
<form name="formName" method="get" action="./register.php">  
    ...  
</form>
```

```
document.forms[i]; //i est la position du formulaire
```

```
document.forms["formName"]; //formName est le nom du formulaire  
document.formName;
```

# Objet Form du DOM

```
var f = document.forms[i];    //f est un objet form

f.elements[];    //retourne un tableau contenant les éléments du formulaire
f[];

f.reset();        //vide le formulaire

f.submit();       //soumet le formulaire

var att = f[0].attributeName; //récupère la valeur d'un attribut d'un input
f[0].attributeName = att;     //change la valeur d'un attribut d'un input
f[0].focus();    //positionne le focus clavier sur l'élément spécifié
```

# Exercice (1)

- Créez un formulaire simple : Nom, Prénom.
  1. À l'aide du JavaScript, proposez une complétion automatique du formulaire (Jean, Dupont) lorsque l'on appuie sur un bouton.
  2. Ajoutez un bouton qui remet à zéro tous les champs du formulaire.

Conseil : Placez ces boutons hors du formulaire.

Rappels utiles :

```
<button id="monBouton" >Auto-complétion</button>
```

```
var f = document.formName;           // Accéder au formulaire
var valeur = f.champ.value;           // Récupérer la valeur d'un champ
f.champ.value = "valeur";             // Insérer une valeur dans un champ
document.getElementById("id");        // Accéder à un élément de la page
```



## Exercice (2)

- Ajoutez un champ Code Postal au formulaire précédent.

À l'aide du JavaScript, faites les vérifications nécessaires sur les champs.

→ Un code postal est un nombre de 4 chiffres (entre 1000 et 9999).

Vérifiez si les champs sont bien remplis, et si le code postal est conforme.  
Lorsque le formulaire est valide, un message s'affiche sur la page.

Rappels utiles :

```
var longueur = element.length; // Renvoie la longueur d'une chaîne de caractères
var nbr = parseInt(variable); // Changer une chaîne de caractères en nombre
Element.innerHTML = "chaîne de caractères" // Ecrire dans un élément
```

# LES EXPRESSIONS RÉGULIÈRES

# Les expressions régulières

Une expression rationnelle ou expression régulière ou encore RegEx est en informatique une chaîne de caractères que l'on appelle parfois un motif et qui décrit un ensemble de chaînes de caractères possibles selon une syntaxe précise.

- Objectifs:

Retrouver des patterns, des motifs, des structures dans une chaîne de caractères.

# Exemple d'application

- Une expression régulière va nous permettre d'extraire toutes les adresses mail.

"Voici un petit texte qui contient plusieurs adresses email. Par exemple en voici une première: **toto@hotmail.com**. Si je vous présente cette adresse: **jules@gmail.com** vous pourrez également conclure qu'il s'agit d'une adresse email. Cependant **tata@msn** n'est pas une adresse valide. Il y a donc 2 emails dans cet exemple."

# Structure d'une expression régulière

/monExpressionRégulière /options

Exemple dans le code :

```
var regex = /[a-z]/gi;  
str = "ma chaîne de CarActèRes";  
str.match(regex);
```

# Un outil pratique

Pour créer, tester et comprendre les expressions régulières :

<https://regex101.com/#javascript>

# Exprimer une chaîne de caractères

/chainedecaractères/options

- Exemple :

Regex = /bonjour/i

"**B**onjour, comment allez-vous ?"

Regex = /allez/i

"Bonjour, comment **allez**-vous ?"

Regex = /ou/gi

"Bon**j**our, comment allez-vous ?"

# Exprimer une classe de caractères

- Les crochets [ ] sont utilisés pour exprimer une classe de caractères

[xyz]	N'importe lequel des trois : x, y, ou z
[^xyz]	N'importe quel caractère sauf x, y, ou z
[a-z]	N'importe quel caractère de a à z minuscule
[^a-z]	N'importe quel caractère sauf une lettre minuscule de a à z
[a-fm-w]	N'importe quel caractère de a à f ou de m à w inclus
[^0-9A]	N'importe quel caractère sauf les chiffres de 0 à 9 ou la lettre A majuscule



# Manipuler les RegEx

- Allez sur le site : <https://regex101.com>
- Dans la zone de test, entrez ces phrases :
  - J'apprends le javascript
  - J'apprends le java
  - J'apprends le JavaScript
  - J'apprends le JAVASCRIPT

Écrivez l'expression régulière qui mets en évidence le mot « javascript », peu importe la casse.

# Comprendre une RegEx

- RegEx : `/gr[ioa]s/`

Quelles sont les possibilités de recherche avec cette regex ?

# Autres opérateurs

Caractère	Description/utilisation
.	Sélectionne n'importe quel caractère
\	Caractères spéciaux
^	Ne sélectionne que les occurrences en début d'input
\$	Ne sélectionne que les occurrences en fin d'input
*	Sélectionne le(s) caractère(s) entre 0 et n fois
+	Sélectionne le(s) caractère(s) entre 1 et n fois
?	Sélectionnes le(s) caractère(s) entre 0 et 1 fois
x(?=y)	Sélectionne x seulement s'il est suivi de y
X(?!y)	Sélectionne x seulement s'il n'est pas suivi de y

# Autres opérateurs

Caractère	Description/utilisation
x y	Sélectionne x ou y (les deux)
x{n}	Sélectionne les occurrences de n x.
x{n,m}	Sélectionne les occurrences de n à m x.
\s	Sélectionne les espaces vides
\d	Sélectionne les caractères numérique (identique à [0-9])
\w	Sélectionne tous les caractères alphanumérique et les underscores

# Manipulons encore les RegEx

- Écrivez une RegEx qui permet de sélectionner une URL.
- Une URL se compose de :
  - `http://www.monsite.codePays`
  - Le code pays se compose de 2 à 3 lettres.
- Trop facile ? ;-)

Essayez celle-ci : `http(s)://(www.)mon(.)site.codePays`

# Parenthèse capturantes ou non

- Les parenthèse permettent de capturer un sous-motif d'un motif

Exemple : Récupérer gmail et hotmail dans l'exemple précédent

```
[a-zA-Z0-9._-]+@([a-zA-Z0-9._-]{2,})\.[a-z]{2,4}
```

- Pour utiliser des parenthèses non capturantes : (?: ..... )

# Recherche Gourmande VS Non Gourmande

- Par défaut, une RegEx recherchera la chaîne la plus longue possible
- Cependant, dans de nombreux cas, on est intéressé par l'occurrence la plus petite
- Exemple : Capturer une URL dans un ensemble de balises HTML

```
/<a href="(.)"/>/
```

→ Valide pour : `<a href="www.mon-adresse.be">Mon site</a>`

→ Quid de : `<a href="www.mon-adresse.be"><strong class="web">Mon site</strong></a>`

# Recherche Gourmande VS Non Gourmande

- Solution: `/<a href="(.*?)"/`
- Le point d'interrogation indique que la recherche doit s'arrêter dès qu'une chaîne correspondante (minimum) est trouvée



# Exercice

- Pour être sûr de bien comprendre les RegEx, construisez les RegEx capables de :
  1. Trouver tous les mots commençant par « fin ».
  2. Trouver tous les mots « cash » uniquement s'ils sont directement suivis de « flow(s) » (sans espace entre)
  3. Trouver une RegEx capable d'extraire 800 millions et 15 milliards, c'est-à-dire :  
[nombre] mill\*\*\*

# Créer un objet RegEx

```
var maRegex = /regex/flags; //Notation à privilégier  
var maRegex = new RegExp('regex', 'flags');
```

- Optionnels
- Peuvent prendre les valeurs:
  - i : non-sensible à la casse (maj/min)
  - g : sans le flag "g" la RegEx arrête l'analyse dès la découverte de la première occurrence
  - m : Multi-ligne. Affecte l'utilisation de ^ et \$ (recherche sur la ligne plutôt que sur les chaînes de caractères séparément)

# Les méthodes d'un objet RegEx

- `reg.test(str)` : renvoie `true` si `str` vérifie la regex, `false` sinon
- `reg.exec(str)` : applique la regex à la chaîne, renvoie le résultat
- `str.match(reg)` : applique la regex à la chaîne, renvoie le(s) résultat(s)
- `str.replace(reg, str2)` : remplace le(s) sous-chaîne(s) vérifiant la regex par `str2` et renvoie le résultat
- `str.search(reg)` : renvoie la position de la première sous-chaîne vérifiant la regex
- `str.split(reg)` : pour "découper" une chaîne.

# Derniers exercices

- Dans l'exercice <http://jsbin.com/batogusexi/1/edit> , vérifiez que l'utilisateur a bien renseigné une adresse mail valide lorsqu'il clique sur le bouton « Valider ».
- Idem 1, mais faites cette vérification en temps réel.
- Dans l'exercice <http://jsbin.com/qadulivere/1/edit> , remplacez tous les montants en € en \$



Merci pour votre attention.

CONSULTING **B**STORM