

Institut Supérieur d'Informatique et de Multimédia de Sfax

Filière: D- LSI MM



Cours Game Graphics

Assuré par:

M. Zied LOUKIL, zied.loukil@isims.usf.tn

Mme. Souhir BOUAZIZ, souhir.bouaziz@isims.usf.tn

Plan du cours



- ▶ **Chapitre 1:** Introduction à JavaScript
- ▶ **Chapitre 2:** Programmation orientée objet à prototype avec JavaScript
- ▶ **Chapitre 3:** Graphisme avec JavaScript
- ▶ **Chapitre 4:** Autres fonctionnalités JS et HTML5
- ▶ **Chapitre 5:** Analyse de jeux 2D en JavaScript
- ▶ **Chapitre 6:** Introduction au moteur Three.js
- ▶ **Chapitre 7:** Animation des objets 3D

Cours Game Graphics

Chapitre 1: Introduction à JavaScript



Plan



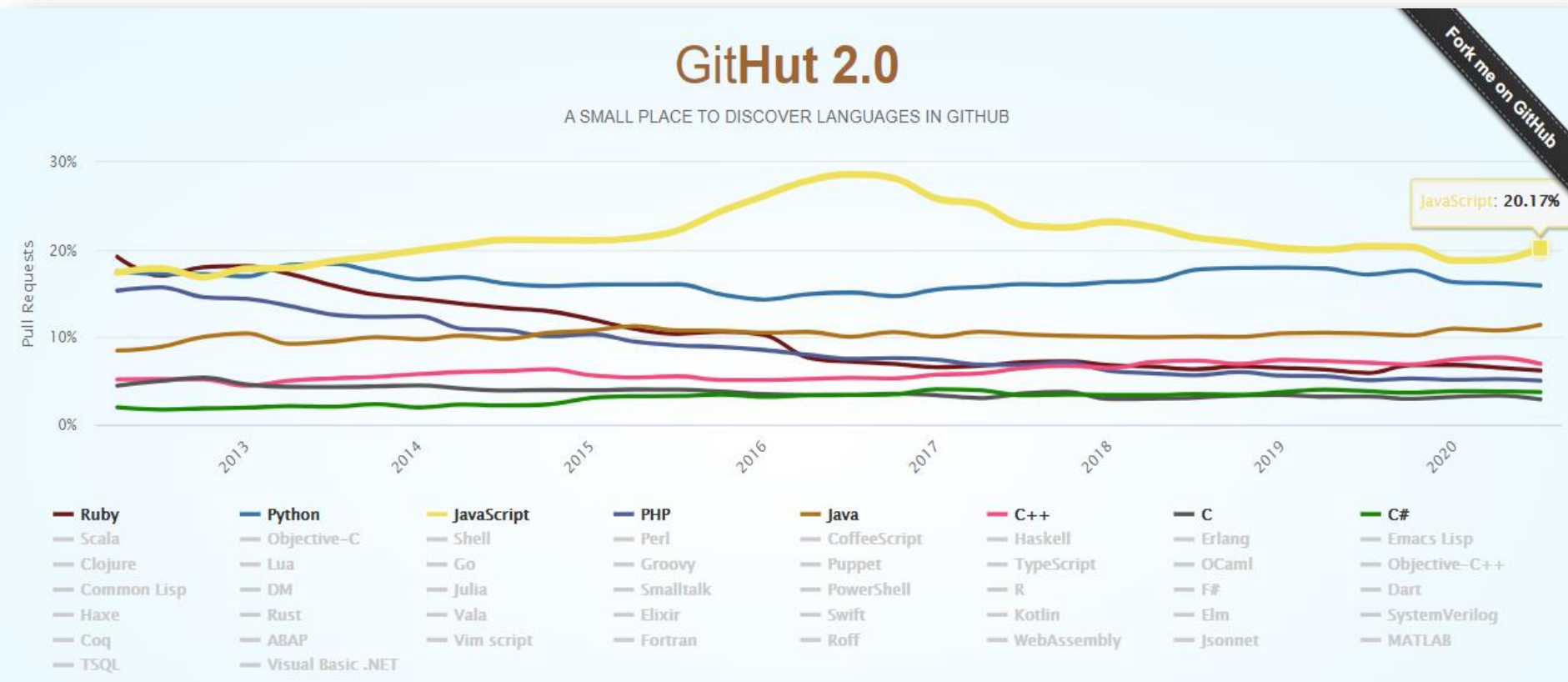
- ▶ **Introduction au langage JavaScript**
- ▶ **Insertion du JavaScript dans une page HTML**
- ▶ **Variables et typage**
- ▶ **Opérateurs**
- ▶ **Structures conditionnelles**
- ▶ **Structures itératives**
- ▶ **Les fonctions**

Introduction: Qu'est ce que JavaScript

- ▶ **HTML:** pas d'interactivité entre l'utilisateur et le site
 - Il permet une coopération avec d'autres langages Internet pour palier à ses propres limitations: JavaScript, VBScript ,Java, etc.
- ▶ **JavaScript: langage de script** incorporé aux balises HTML
 - **Script** : ensemble d'instructions permettant de réaliser une action
- ▶ Les scripts seront donc exécutés (**interprétés**) par le navigateur de l'utilisateur. Ceci sans faire appel à des ressources du côté serveur et sans avoir à attendre de téléchargements supplémentaires.



Introduction: Qu'est ce que JavaScript



https://madnight.github.io/github/#/pull_requests/2020/3

Introduction: JavaScript Vs. Java

- ▶ JavaScript est un langage de programmation **Orienté Objet à prototype**, développé par la société Netscape Corporation depuis 1995.
 - **Objet**: code et données pouvant être traités en tant qu'unités ou composants

JavaScript	Java
Langage interprété (par le navigateur au moment de l'exécution)	Langage compilé (avant l'exécution)
Code intégré au html	Code (Applet) à part du document html
N'existe pas en dehors du web	Langage à part entière
Typage faible	Typage fort (déclaration du type de variable)
Permet d'accéder aux objets du navigateur	N'accède pas aux objets du navigateur
Tous les objets sont des instances	Les classes et les instances sont deux entités distinctes
On crée un ensemble d'objets avec des fonctions qui sont des constructeurs	On instancie une classe avec des méthodes appelées constructeurs
Il est possible d'ajouter ou de retirer des propriétés dynamiquement, pour certains objets ou bien pour l'ensemble des objets.	Il est impossible d'ajouter des propriétés dynamiquement pendant l'exécution.

Introduction: Pourquoi JavaScript

- ▶ **Petites applications simples**

- ▶ calculettes, petits outils de conversions, jeu,...

- ▶ **Tests de validité des données sur les éléments de l'interface de saisie (Vérifier la cohérence des données)**

- ▶ pour vérifier qu'une valeur considérée comme obligatoire a bien été saisie, que le champ saisi correspond bien à une date,.

- ▶ **Aspects graphiques de l'interface**

- ▶ modification d'une image lors du passage de la souris, gestion de fenêtres, fabrication locale d'une page HTML

- ▶ Graphisme: animation des formes, **création des jeux web** (balise <canvas> de HTML5)

- ▶ **Gestion complète de l'interface d'une application complexe**

- ▶ Création de fenêtres, modification de menus, aide contextuelle,...

Introduction: Caractéristiques

- ▶ JavaScript est un langage **structuré**:
 - Structures conditionnelles, structures itératives, fonctions, ...
- ▶ JavaScript sait gérer les évènements principaux de la souris
 - déplacement, clicks, ...
- ▶ JavaScript sait gérer le graphisme et manier la vidéo et l'audio (+ **HTML5**)
- ▶ **Remarques:**
 - JavaScript est sensible à la casse: **bonjour()** est différent de **Bonjour()**
 - Le caractère **point-virgule** est utilisé comme séparateur de fin de ligne mais sa **présence n'est pas obligatoire** si la ligne ne comporte qu'une seule instruction.
 - Pour mettre en commentaires toute une ligne on utilise le double slash:
// commentaires
 - Pour mettre en commentaire une partie du texte (éventuellement sur plusieurs lignes) on utilise le **/*et le */**

Insertion du JavaScript dans une page HTML

Plusieurs façons d'inclure du code JavaScript dans une page HTML:

- ▶ **1^{ère} méthode:** Directement dans le code HTML entres les deux balise **<script>** et **</script>**:

```
<script>  
    /*instructions javascript*/  
</script>
```

- HTML5 considère que le langage de programmation par défaut des scripts est JavaScript.
- On peut les placer dans le corps ou dans l'en-tête de la page:
 - dans le corps (**<body>** ... **</body>**) : sont à les placer les scripts à exécuter au chargement de la page.
 - dans l'en-tête (**<head>** ... **</head>**): sont à placer les éléments qui doivent être exécutés plus tard (Exple: lors d'un événement particulier).

Insertion du JavaScript dans une page HTML

- ▶ **2^{ème} méthode:** En mettant le code dans un fichier (*.js)
 - On doit indiquer aux balises le nom et l'emplacement du fichier contenant le script, grâce à la propriété **src**.
 - Exemple:

- **page.html**

```
<script src="file.js"> </script>
```

- **file.js**

```
document.write("hello");
```

Insertion du JavaScript dans une page HTML

► 3^{ème} méthode: Grâce aux événements :

- Elle consiste à écrire le code directement à l'intérieur de la balise qui va être concernée par le script.
- Pour cela, une nouvelle propriété est nécessaire. Il s'agit du **gestionnaire d'événements** (event handler).

<balise *OnEvenement*="code Javascript" >

- Cette propriété est caractéristique du JS.
- Elle porte le nom de l'événement qui doit déclencher le script.
- Elle contient comme valeur le script à exécuter lors de cet événement.

➤ Exemple:

```
■ <a href="#" onclick="alert('Bonjour !');">lien</a>  
■ <a href="javascript:alert('Bonjour !');"> lien </a>  
■ 
```

Variables et typage

- ▶ Une variable pourra référencier des nombres, des chaînes de caractères, des booléens et des objets.
- ▶ Un nom de variable est composé de:
 - lettres non-accentuées, du caractère "_" et des chiffres, à condition que le 1^{er} caractère ne soit pas un chiffre.
 - Il ne doit pas faire partie des mots réservés de JavaScript (Ex: "break", "case", "char", "continue", "delete", "double", "final", "long", "new", "public" et "super").
- ▶ JavaScript est sensible à la casse.
- ▶ La déclaration des variables est optionnelle:
 - **Façon explicite:** précéder par **var** (Variables non typées)
 - **Façon implicite:** par affectation (=) (Typage dynamique)
- ▶ Si une variable n'est pas initialisée, elle prend la valeur **undefined**.

Variables et typage

► Déclaration: **exemple**

```
<script>
```

```
var date; // Déclaration sans affectation
```

```
var compteur=0; // Déclaration avec affectation
```

```
toto='coucou'; // Déclaration implicite par affectation
```

```
var prem, second; // variables séparées par des virgules
```

```
</script>
```

Variables et typage

- ▶ JavaScript possède la notion de **variable locale** (déclarée au sein d'une fonction) et **variable globale** (déclarée en dehors du corps de la fonction).
- ▶ **La portée d'une variable locale** se limite à la fonction dans laquelle elle a été déclarée.
- ▶ **La portée d'une variable globale** se limite au document dans lequel elle a été déclarée.
- ▶ Une variable globale peut être appelée au sein d'une fonction par l'intermédiaire du mot-clé **this**.

➤ Exemple:

```
//Variable globale
var x = 72;
function affiche(x){
    //Appel de la variable globale à l'aide du mot-clé this
    document.write("Valeurs : " + x + " " + this.x); }
affiche(12);
```

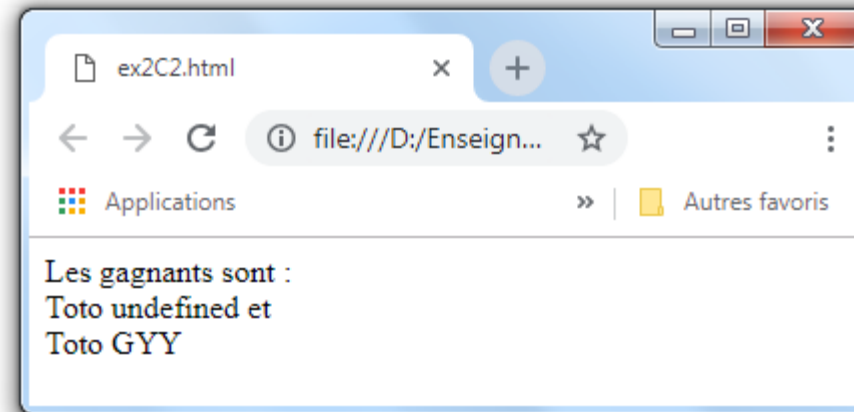
Variables et typage

► Exemple:

```
<HTML> <head>
<script>
  var Nom; // Déclaration sans affectation
  var Prenom='Toto' ;// Déclaration avec affectation
  function affiche()
  { var Nom='GY';
    document.write(Prenom + ' ' + Nom);
  }
</script> </head>
<body>
  Les gagnants sont : <br/>
  <script>
    document.write(Prenom + ' ' + Nom + ' et <br/>');
    affiche()
  </script> </body> </HTML>
```

Variables globales

Variable locale



Variables et typage

► Typage:

- lors de l'initialisation ou l'affectation
- **dynamique**: changement possible du type d'une variable selon la valeur

► Le type des variables n'est pas précisé. En revanche, il existe :

➤ trois types de données simples:

- les chaînes de caractères (**String**)
- les nombres (**Number**)
- les booléens (**Boolean**)

➤ un type pour les données structurées : les objets (**Array, Date, ...**)

➤ les fonctions: (**function**)

➤ types particuliers : **null, undefined**

► Nombres particuliers : **Infinity, -Infinity, NaN** (Not a Number)

► Type dominant : **String**

- `A=23 ; B="45" ; C=A+B` //contient "2345" (Conversion de type implicite)

Variables et typage

► Typage: exemple

```
<script >
```

```
var monNombre= new Number(); // Déclaration typée sans affectation
```

```
var e = new Number(2.71828); // Déclaration typée avec affectation
```

```
var maChaine= new String(); //Déclaration de chaîne
```

```
var toto = new Boolean(true); //Déclaration de booléen
```

```
var noms = new Array("Pierre", "Paul", "Jacques"); //Déclaration de tableau
```

```
noms[1] = "Toto";
```

```
noms[3] = "Dupont";
```

```
for(var indice in noms)
```

```
    document.write(indice , " -> " , noms[indice], "<br/>");
```

```
</script>
```

```
0 -> Pierre  
1 -> Toto  
2 -> Jacques  
3 -> Dupont
```

Variables et typage

- ▶ Même si JavaScript gère de façon transparente les changements de type des variables, il est parfois nécessaire de forcer la conversion du type:
 - **parseInt()** permet de convertir une variable en nombre entier
 - **parseFloat()** permet de convertir une variable en nombre décimal

▶ Exemple:

```
<script >
var a = "123";
var b = "456";
document.write(a+b,"<br/>"); //Affiche 123456
document.write(parseInt(a)+parseInt(b),"<br/>"); // Affiche 579
</script>
```

Opérateurs

► Arithmétique:

► Binaire:

+	Addition	5+4 ; Comp + 1 . '+' opère aussi sur les chaînes, c'est un opérateur de concaténation: 'bon'+`jour`==`bonjour`
-	Soustraction	2-3; rebours -1
*	Multiplication	2*3; a*b
/	Division	2/3; a/b . Attention: nombre/0 rend null
%	Modulo	13 % 5 ; Reste dans la division euclidienne 13%5 = 3

► Unaire:

-	Opposé	-monSolde
++	Incrémentation	i++ ; équivaut à i = i+1 ++i ; l'incrément est fait avant d'utiliser la valeur de i
--	Décrément	k-- ; équivaut à k = k-1 --k ; la décrémentation est faite avant d'utiliser la valeur de k

Opérateurs

► Comparaison:

<	Inférieur strict	<code>X < 2; nbre < 100</code>
>	Supérieur strict	<code>X > 2 ; compteur > x</code>
>=	Supérieur large	<code>y >= 5</code>
<=	Inférieur large	<code>Y <= 5</code>
==	Egal (identique à)	<code>X == 3; j == i;</code>
!=	Différent	<code>X != y</code>

- Un opérateur supplémentaire de comparaison "**===**": Il compare la valeur aussi bien que le type
- **Exemple:** `var x=12; var y='12'; var z1= (x==y); var z2= (x===y);`
 - On a `z1= true` et `z2 = false` car x et y ont la même valeur mais ne sont pas du même type

Opérateurs

► Logique:

&&	ET	Bool1 && bool2 <i>true</i> uniquement dans le cas : <i>true</i> && <i>true</i>
	OU	Bool1 bool2 <i>false</i> uniquement dans le cas : <i>false</i> <i>false</i>
!	NON	! Bool1 ! true = false ; ! false = true

► Affectation :

➤ L'affectation simple se fait par '='

+=	Addition	x+=y signifie x=x+y
-=	Soustraction	x-=y signifie x=x-y
=	Multiplication	x=y signifie x=x*y
/=	Division	x/=y signifie x=x/y
%=	Modulo	x%=y signifie x=x%y

Structures conditionnelles

- Structure conditionnelle généralisée: **if, else if, else:**

```
if (condition_1)
{
    instructions_1; // code exécuté si la condition_1 est remplie
}
else if (condition_2)
{
    instructions_2;
}
:
else
{
    instructions_n;
}
```

Structures conditionnelles

► Structure à choix multiples: **switch**:

```
switch(variable)
{
    case val1:
        instructions_1;
        break ; // le break pour indiquer qu'on a fini les instructions pour ce cas
    case val2:
        instructions_2;
        break ;
    :
    default :
        instructions_n;
        break ;
}
```


Structures itératives

- ▶ Structure itérative complète: la boucle **for**:

```
for (valeur initiale ; condition ; progression)
{
    instructions;
}
```

```
for ( var x in tab)
{
    instructions;
}
```

- ▶ Structure itérative à condition d'arrêt: la boucle **while**:

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

Structures itératives

- ▶ Structure itérative à condition d'arrêt: la boucle **do while**:

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

- ▶ L'instruction **break**: permet d'interrompre une boucle for ou while ou do while.
- ▶ L'instruction **continue**: permet de sauter à l'itération suivante dans une boucle (sans sortir de celui-ci comme le fait break)

Exercice

- ▶ Deux nombres entiers n et m sont qualifiés d'amis, si la somme des diviseurs de n est égale à l'entier m et la somme des diviseurs de m est égale à l'entier n
 - on ne compte pas comme diviseur le nombre lui-même et 1
- ▶ Exemple : les nombres 48 et 75 sont deux nombres amis puisque :
 - Les diviseurs de 48 sont : $2 + 3 + 4 + 6 + 8 + 12 + 16 + 24 = 75$
 - Les diviseurs de 75 sont : $3 + 5 + 15 + 25 = 48$
- ▶ Ecrire une page HTML intitulé « Nombres amis » qui permet de saisir deux entiers n et m et puis vérifier s'ils sont amis ou non

Les fonctions

► Deux types de fonctions:

- Les fonctions prédéfinies (exemple: **alert()**, **parseInt()**,)
- Les fonctions utilisateurs:

```
function Nom_fonct(argument_1, argument_2, ..., argument_n)
{
    liste d'instructions;
}
```

NB:

- ✓ Le noms de fonction doit respecter les règles de nommage des variables
- ✓ Arguments non typés
- ✓ Nombre d'arguments non fixé lors de la déclaration
- ✓ Pour exécuter une fonction, il suffit de faire appel à elle:
Nom_fonct (arg1, arg2, ..., arg_n);
- ✓ Une fonction peut retourner une valeur en ajoutant le mot-clé **return**.

Les fonctions

- ▶ Autre manière de création de fonction: en utilisant une **variable**.

var nomF = function(arg1, arg2) {...}

- ▶ **Exemple:**

```
var operation = new Array();
operation["add"] = function(x,y){ return x+y; };
operation["soustr"] = function(x,y){ return x-y; };
operation["mult"] = function(x,y){ return x*y; };
operation["div"] = function(x,y){ return x/y; };
operation["moy"] = function(x,y){ return (x+y)/2; };
var a = parseFloat( prompt("Premier nombre ?") );
var b = parseFloat( prompt("Deuxième nombre ?") );
var fct = prompt("Fonction à appliquer ?");
var resultat = operation[fct](a,b);
alert("Resultat : " + resultat);
```

Les fonctions

- ▶ Si on a **function f(x,y){...}** et qu'on fait appel à **f** de cette manière: **f(5);**
 - deux variables locales, **x** et **y**, vont être créées
 - la variable **x** va être initialisée avec la valeur 5
 - la variable **y** ne sera pas initialisée: **undefined**.
- ▶ Pour chaque fonction, il est associé un **tableau** contenant tous les arguments.
- ▶ **Exemple:** `function addition() {...};`
 - En appelant `addition(21, 4, 15)`, on aurait:
`addition.arguments = new Array(21, 4, 15);`
 - On pourrait ainsi faire des opérations sur les arguments quel que soit leur nombre.


Les fonctions

► Exemples:

```
function carre(nombre)
{
    return nombre * nombre;
}
```

```
var Tva= function(montant, taux)
{ return montant * taux; };
```

```
function somme()
{
    var argv= somme.arguments;
    var argc= somme.arguments.length;
    var result= 0;
    for (var i = 0 ; i < argc; i++) {
        result+= argv[i];
    }
    return result;
}
```



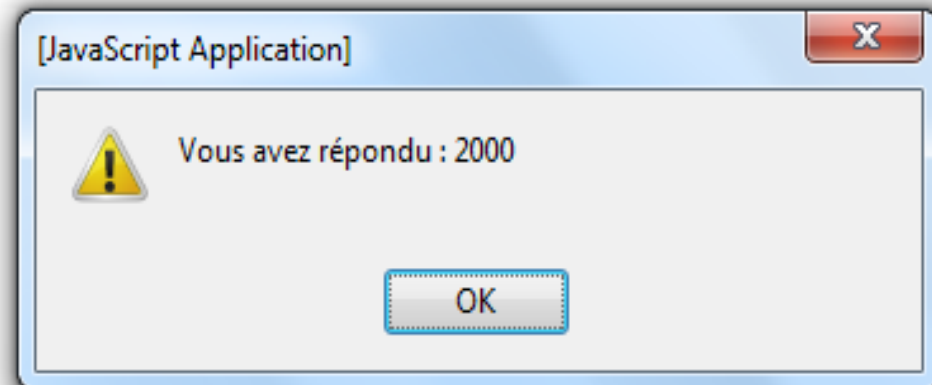
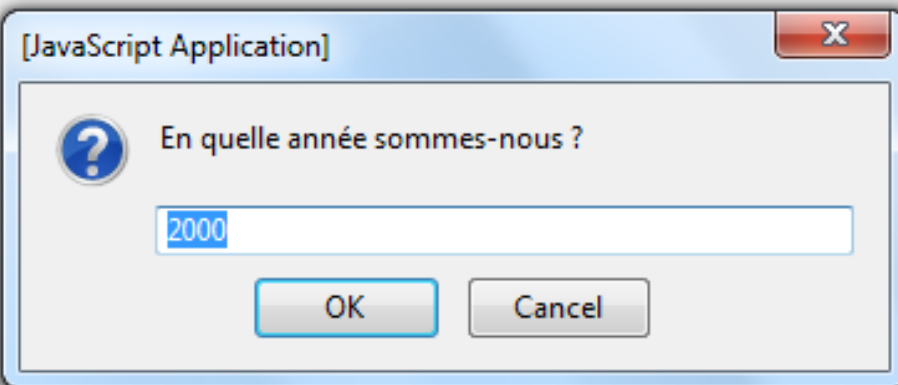
somme(1,2,3) retourne 6
somme(2) retourne 2

Les fonctions

► Quelques fonctions utiles:

- **prompt(message, valeur_defaut)** : crée une fenêtre de dialogue permettant la saisie et dans laquelle le message est affiché et 2 boutons (**OK** et **Annuler**)
- **alert('message')**: crée une fenêtre de dialogue dans laquelle le message est affiché
- **Exemple:**

<html>

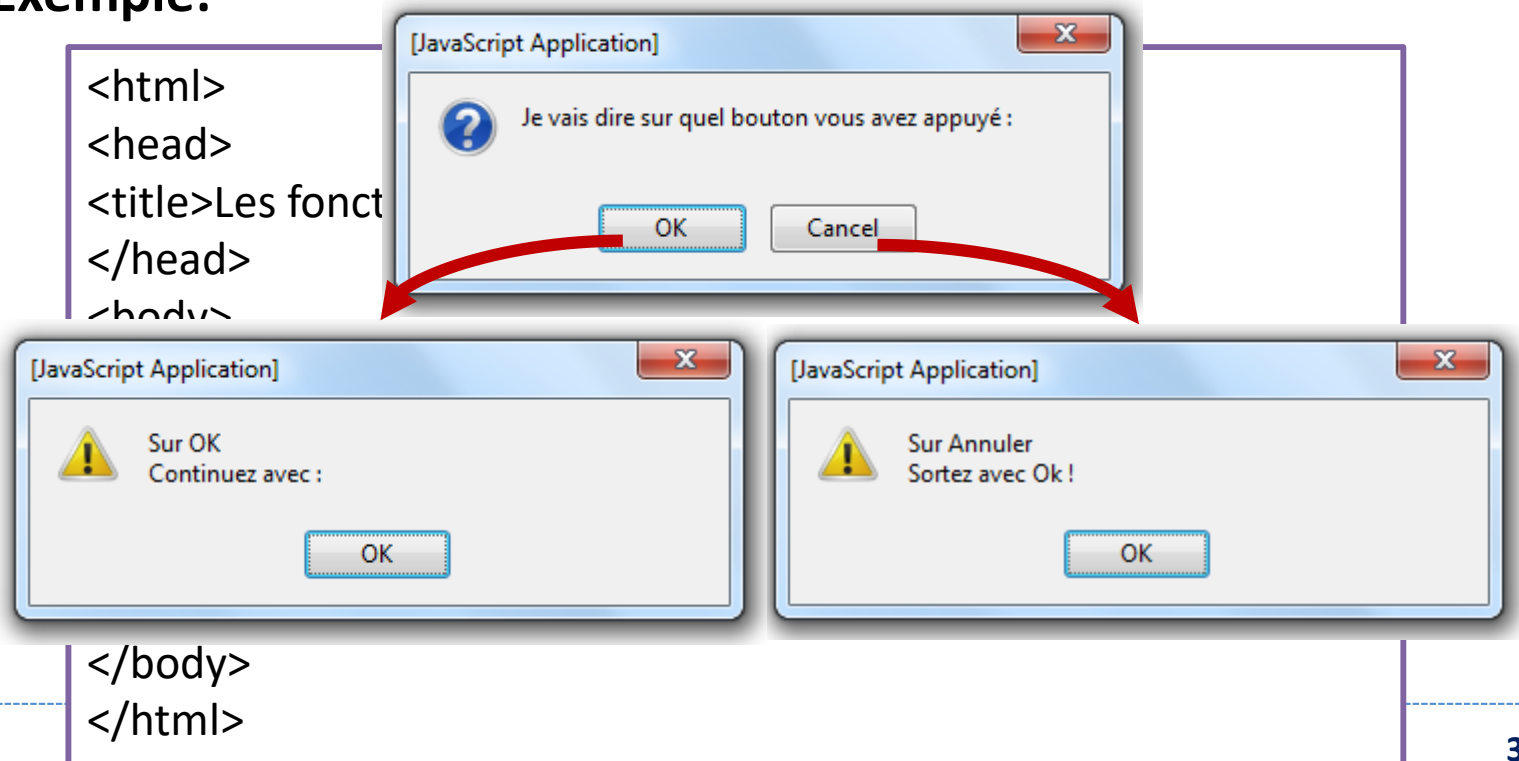


Les fonctions

► Quelques fonctions utiles:

- **confirm(message)**: crée une fenêtre de dialogue pour confirmer une action (choix entre 2 boutons: **OK** et **Annuler**). Elle permet d'envoyer une information et de recevoir un booléen.

► Exemple:



Les fonctions: Exercices

► Exercice 1:

- Ecrire une page HTML faisant apparaître les nombres premiers compris entre 0 et 100.
- **N.B.** Si un nombre n n'est pas premier un de ses diviseurs est inférieur à \sqrt{n}
- Utiliser une fonction ***EstPremier(n)*** qui permet de déterminer si l'entier n est premier. Elle permet de retourner ***true*** si n est premier et ***false*** sinon.

Les fonctions: Exercices

► Exercice 2:

- Ecrire une page HTML permettant d'afficher les nombreux chanceux formés de deux chiffres.
- **N.B.** Un entier n est appelé **nombre chanceux** si $(n+a+a^2)$ est premier pour tout a tel que $0 \leq a < (n-1)$
- Créer en JavaScript une fonction ***EstChanceux(n)*** qui permet de déterminer si l'entier n est chanceux. Elle permet de retourner ***true*** si n est un nombre chanceux et ***false*** sinon.