Université de Bordeaux
Master Bioinformatique

Initiation au développement Web

Programmation événementielle en Javascript

Florent Grélard

florent.grelard@u-bordeaux.fr

http://www.labri.fr/perso/fgrelard/teaching.html

Version : 28 novembre 2017 (diapositives de présentation)

Objectifs

- Comprendre les différences fondamentales entre la programmation impérative et la programmation événementielle.
- Connaître les bases du langage Javascript pour la gestion de clics ou d'entrées claviers sur une page Web
- Utilisation de la bibliothèque JQuery

Partie 1: Web – Plan

Plan: Programmation événementielle

Programmation événementielle

Définition (Événement)

Un événement correspond à l'action d'un utilisateur (clic, entrée clavier) avec un élément du programme (bouton, vidéo, lien...), ou découle d'un événement interne au programme (chargement d'un fichier). Un événement induit une modification de l'état du programme.

Définition (Programmation événementielle)

La programmation événementielle est un paradigme de programmation permettant l'appel d'une fonction lors du déclenchement d'un événement.

Exemples

- Diaporama d'images sur une page Web : défilement d'images lors du clic sur un bouton.
- Masquer une section d'une page Web lors d'un clic.
- Création d'un compte client sur un site d'achat par la validation d'un formulaire HTML.

Programmation séquentielle vs. programmation événementielle (1/2)

Programmation séquentielle :

```
mareponse = 0
print("Veuillez entrer une réponse")
mareponse = input()
reponsecarre = mareponse * ma reponse
print(mareponse + "au carré ="+ reponsecarre)
```

1. L'exécution du programme est bloquée tant que l'utilisateur n'a pas entré sa réponse (fonction `input()`).
2. L'ordre de déroulement des instructions est **toujours le même**.

Programmation séquentielle vs. programmation événementielle (1/2)

Programmation événementielle :

```
#Boucle infinie en arriere plan
while (True):
    event = getEvent()
    processEvent(event)
```

1. Une boucle infinie "en arrière-plan" permet de détecter un événement (`getEvent()`) et de le traiter (`processEvent()`).
2. Le déroulement des instructions dépend de l'ordre des événements déclenchés par l'utilisateur.

Abonnement de fonctions

- La programmation événementielle repose sur l'exécution de fonctions lorsqu'un événement se produit sur élément cible
- On parle d'abonnement de la fonction à l'élément pour l'événement
- L'abonnement d'une fonction à un événement est réalisée par la « mise sur écoute » d'un élément : on parle d'`event listener`.

Langages

Résumé des langages utilisés pour la programmation Web :

- Pour le contenu des pages Web : **HTML**.
- Pour la mise en forme : **CSS**.
- Pour la programmation événementielle : **Javascript**, **PHP**.
- Pour la communication avec base de données et traitement "sécurisé" : **PHP**.

Plan: Javascript

Aperçu du langage (1/2)

Le Javascript est un langage de scripts, **interprété**.

- le code Javascript peut être inséré directement dans le fichier HTML ou mis dans un fichier séparé (extension `.js`) puis inclus dans le HTML
- les navigateurs possèdent des interpréteurs Javascript

Appel du code Javascript dans le fichier HTML (en bas du `body`) :

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

Aperçu du langage (2/2)

- Langage **orienté objet**.
  ⇒ Objet : ensemble regroupant des données et des méthodes (fonctions) agissant sur ces données.

  ⇒ Appel des méthodes :

```
objet.methode();
```

Attention : à ne pas confondre

Javascript ≠ Java

Plan: Javascript

Debuggage du code

- Des erreurs de syntaxe peuvent empêcher l'exécution du code.
  $\Rightarrow$ besoin de débugger.
  - Console dans le navigateur (raccourci sur Firefox $\rightarrow$ F12)
  - Pour afficher un message ou la valeur d'une variable dans la console (équivalent du `print` en Python) :

```
console.log("Message" + variable);
```

- Langage permissif $\Rightarrow$ utilisation fortement recommandée de `"use strict";` au début du code.

Syntaxe

- Généralités
- Variables, types de données
- Structures de contrôle : séquences, conditionnelles
- Fonctions
- Tableaux

Généralités

- Les lignes de code se terminent par un point virgule “;”
- Les fonctions ou les structures conditionnelles sont encadrées par des accolades

Types

- Langage faiblement typé
- Différents types pour les variables :
  - `boolean` : `true` ou `false`
    Opérateurs : `and` : `&&` ; `or` : `||`, `not` : `!`
  - `number` : décrit les nombres entiers ou réels (1, 5.2, ...)
    Opérateurs classiques pour les additions, multiplications, soustraction et division.
  - `string` : chaîne de caractères (« Hello world ! »)
    Opérateur de concaténation : +.

Types

Deux opérateurs d'égalité :

- == : teste l'égalité en forçant la conversion du type.
  Ex : '1' == 1 renvoie true

- === : teste l'égalité stricte (variables de même type et avec la même valeur).
  Ex : '1' === 1 renvoie false

Bonnes pratiques

On privilégiera l'opérateur ===.

Variables

- Les déclarations de variables se font en préfixant le nom de la variable par :
  - `var` : variable locale au sein d'une fonction.
  - `let` : variable locale, non accessible en dehors d'un bloc (boucle ou condition)
  - `const` : variable locale, non accessible en dehors d'un bloc, assignable uniquement une fois (non réassignable)

Exemple de code Javascript

Trouvez les trois erreurs de compilation :

```
1     if (condition) {
2      var one = 0;
3      let two = 0;
4      const three = "blabla";
5      three = "hello␣now";
6     }
7     console.log(one);
8     console.log(two);
9     console.log(three);
```

Exemple de code Javascript

Trouvez les trois erreurs de compilation :

```
1    if (condition) {
2     var one = 0;
3     let two = 0;
4     const three = "blabla";
5     three = "hello now"; //Erreur dans la console
6    }
7    console.log(one);
8    console.log(two); //Erreur dans la console
9    console.log(three); //Erreur dans la console
```

Fonctions

Déclaration de fonction :

```
1    function nom_fonction(parametre1, parametre2) {
2     //Code
3    }
4    var nom_fonction = function(parametre1, parametre2) { }
5    let nom_fonction = function(parametre1, parametre2) { }
```

- Des accolades encadrent le corps de la fonction
- Déclarations alternatives en récupérant un "objet fonction" (cf. lignes 4 et 5)
- L'appel à la fonction se fait par :

```
nom_fonction(2,3);
```

Boucles

- Boucle `for` :

```
1     for (let i = 0; i < 10; i++) {
2      //Code
3     }
```

- Boucle `while` :

```
1    var i = 0;
2    while (i < 10) {
3     //Code
4     i++;
5    }
```

Tableaux

Objet `Array`

1. Initialisation :

```
var tab = [];
```

2. Remplissage :

```
tab[indice] = valeur;
```

3. Parcours :

```
1  for (let i = 0; i < tab.length; i++) {
2   //Code
3  }
4  /* OU */
5  for (var index in tab) {}
```

Plan: Javascript

Pour la programmation événementielle

Le Javascript permet :

- d'agir sur les propriétés des balises d'un document HTML
- de manipuler l'arbre HTML, dit arbre DOM (Document Object Model)

$\Rightarrow$ Pages Web dynamiques dont la structure évolue lors d'événements.
$\Rightarrow$ Modification de la structure du DOM par réponse aux événements

window et document

Deux objets sont définis par défaut pour du code Javascript lié à une page Web :

- `window` : fenêtre du navigateur dans laquelle le document est chargé
- `document` : représente le DOM chargé dans la fenêtre (arbre HTML)

Exemple de méthodes :

- `window.alert()` : ouvre une boîte de dialogue (popup) dans la page Web
- `document.write()` : écrit du texte dans le document HTML

Plan: Javascript

Sélection d'élements

Afin de manipuler des éléments HTML dans le code Javascript, il est tout d'abord nécessaire de les récupérer sous forme d'objets.
La sélection d'éléments HTML peut se faire :

- par la balise
- par l'`id`
- par la `class`
- par un sélecteur CSS.

Sélection d'un élément par l'identité

La méthode `getElementById('id')` de l'objet document sélectionne l'unique élément dont l'id est fourni en paramètre

Exemple : changement d'image en JS :

Fichier html

```
...
<img src="image.jpg" id="
    monImage" />
...
```

Fichier javascript

```
1  var imageJS = document.
       getElementById('monImage');
2  imageJS.src="nouvelleImage.jpg"
```

Sélection de plusieurs éléments

Différentes méthodes :

- À partir du nom de la balise : `getElementsByTagName()`
- À partir de la classe : `getElementsByClassName()`
- À partir d'un sélecteur CSS : `querySelectorAll()`.

⇒ renvoient un tableau d'éléments

⇒ sont des méthodes liées à un élément HTML (ex : `document` ou élément récupéré par `getElementById()`)

Exemple : changement d'image en JS :

Fichier html

```
...
<h1 class="agauche">
<div class="agauche">
...
```

Fichier javascript

```
1  var elements = document.
       getElementsByClassName('
       agauche');
2  console.log(elements[0]);
```

Manipulation des éléments

- Les objets récupérés en Javascript ont des attributs : ce sont les mêmes que les attributs HTML.
- Ces attributs sont accessibles en lecture et en écriture

Exemple :

```
1    var imageJS = document.getElementById('monImage');
2    imageJS.alt = "Texte␣de␣description";
3    console.log(imageJS.alt);
4    imageJS.src = "nouvelleImage.png";
5    imageJS.className = imageJS.className + "laclasse";
```

Modifier le contenu d'une balise

- L'attribut `innerHTML` représente le contenu HTML d'un élément :
  - en lecture : contient les balises
  - en écriture : son contenu est interprété (les balises sont considérées comme du HTML)
- L'attribut `textContent` représente le contenu textuel d'un élément :
  - en lecture : ne contient pas les balises
  - en écriture : son contenu n'est pas interprété (les balises sont considérées comme du texte)

Différences entre innerHTML et textContent

```
<div id="exemple">
<p> Ceci est <span> mon contenu </span>
</p>
</div>
```

innerHTML :

```
1 var element = document.
      getElementById("exemple");
2 var htmlText = element.
      innerHTML ;
3 console.log(htmlText); //le
      texte contient les balises
      <p> et <span>
4 element.innerHTML = "<p>␣
      lalilou␣</p>"; // les
      balises sont interpretees
```

textContent :

```
1 var element = document.
      getElementById("exemple");
2 var htmlText = element.
      textContent ;
3 console.log(htmlText); //le
      texte ne contient pas les
      balises <p> et <span>
4 element.textContent = "<p>␣
      lalilou␣</p>"; // les
      balises sont considerees
      comme du texte
```

Manipulation du DOM

Possible de créer des éléments avec `document.createElement()`

Pour modifier l'arbre DOM :

- Ajout : `appendChild()`, `insertBefore()`
- Suppression : `removeChild()`
- Remplacement : `replaceChild()`

Plan: Javascript

Capture d'événements

Objectif : Lier une fonction à l'occurrence d'un événement sur un élément.

Le lien entre un événement et un élément se fait par l'event listener.

Exemples :

Élément (HTML) $\rightarrow$	**Evénement (JS)** $\rightarrow$	**Event listener (JS)** $\rightarrow$
Bouton	Clic	Envoi de données
Image	Clic	Changement d'image

Événements

Il existe différents types d'événements :

- actions par le clavier ou la souris : `click`, `keyup`, `mouseover` ...
- changement d'état : `change`, `focus`...
- fin ou début de chargement d'un élément sur la page : `load`.

Event listener (1/2)

- « Mise sur écoute » des événements sur un élément.
- Méthode `addEventListener()` réalise l'abonnement d'une fonction à un événement pour un élément.

Utilisation de la méthode :

```
objet.addEventListener(typeEvenement, fonctionDeclenchee);
```

- `objet` : objet ciblé (ex : `document` ou objet récupéré par `getElementById()`).
- `typeEvenement` : chaîne de caractères désignant l'événement concerné.
- `fonctionDeclenchee` : fonction appelée (ex : changement image, ou contenu d'une balise)

Event listener (2/2)

Alternative à `addEventListener()` pour définir un abonnement à une fonction :

```
objet.onevent = fonctionDeclenchee;
```

Exemples :

```
1  document.onkeyup = fonctionDeclenchee;
2  //Equivalent a
3  document.addEventListener("keyup", fonctionDeclenchee);
```

Capture d'événements : exemple (1/2)

HTML :

```
...
<p id="monParagraphe"> Si vous cliquez sur ce bouton,
    vous ne me verrez plus jamais :(
</p>

<button id="monBouton" type="submit"> Envoyer </button>
...
```

Javascript :

```
1  var changerMonParagraphe = function() {
2   var texte = document.getElementById("monParagraphe");
3   texte.textContent = "Mon nouveau texte tout propre tout net";
4  }
5
6  var bouton = document.getElementById("monBouton");
7  bouton.addEventListener("click", changerMonParagraphe);
8  //Equivalent a : bouton.onclick = changerMonParagraphe;
```

Capture d'événements : exemple (2/2)

Au début :

Si vous cliquez sur ce bouton, vous ne me verrez plus jamais :(

Envoyer

Après clic sur le bouton :

Mon nouveau texte tout propre tout net

Envoyer

Abonnements de plusieurs fonctions

Sur un même élément on peut avoir :

- plusieurs abonnements pour différents événéments
- plusieurs abonnements pour le même événément

Méthodologie et bonnes pratiques :

1. Définir une fonction `setupListeners` chargée de mettre en place tous les abonnements
   $\Rightarrow$ facilite la maintenance de votre code
2. Appeler la fonction `setupListeners` lorsque la page HTML est complètement chargée.
   $\Rightarrow$ `window.addEventListener("load", setupListeners)`

Objet event

- Un objet `event` est créé pour chaque événement
- Le type d'objet `event` varie selon l'événement
- Cet objet possède des attributs qui informent sur la nature de l'événement :
  - `type` : le type d'événement (clic, ...)
  - `clientX`, `clientY`, `screenX`, `screenY`, `pageX`, `pageY` : coordonnées par rapport à la partie visible de la page - l'écran - ou l'ensemble de la page.
  - `key` : information sur la touche appuyée
  - `target` : cible de l'événement

Objet event : exemple (1/2)

```
1  var changerCouleur = function(event) {
2   var texte = document.getElementById("monParagraphe");
3   if (event.key === "r") {
4    texte.style.color = "red";
5   }
6   else if (event.key === "g") {
7    texte.style.color = "green";
8   }
9   else if (event.key === "b") {
10   texte.style.color = "blue";
11  }
12 }
13
14 var setupListeners = function() {
15  document.addEventListener("keyup", changerCouleur);
16 }
17
18 window.addEventListener("load", setupListeners);
```

Objet event : exemple (2/2)

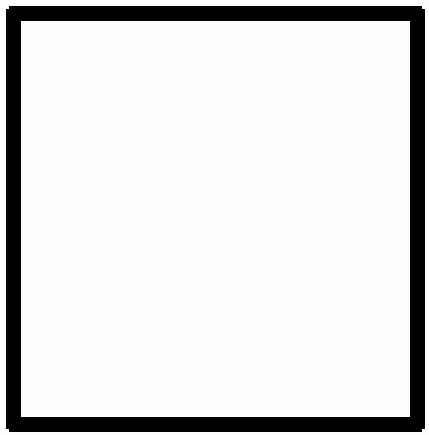
Résultat :

Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.

Objet event : exemple (2/2)

Résultat :

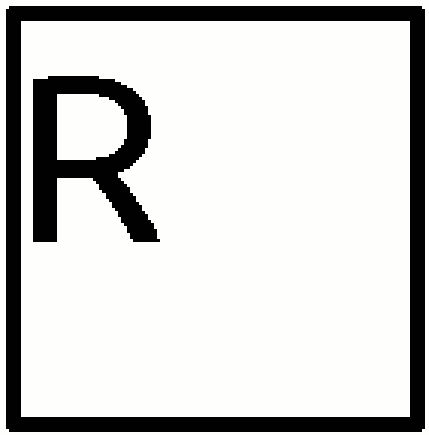
Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Objet event : exemple (2/2)

Résultat :

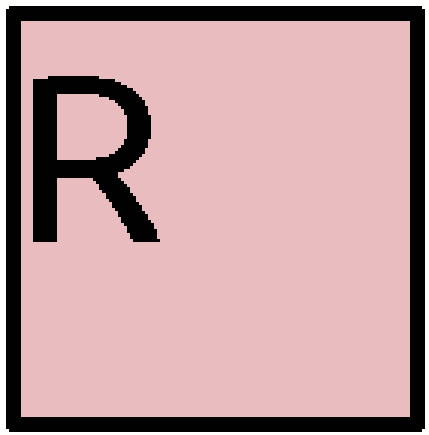
Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Objet event : exemple (2/2)

Résultat :

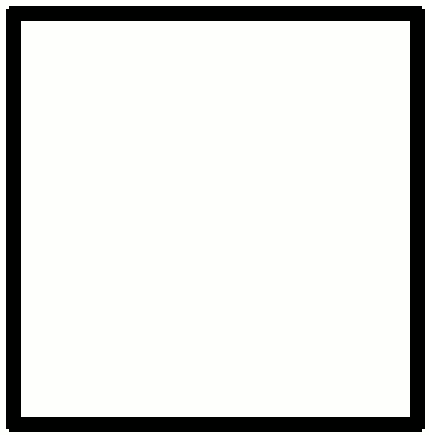
Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Objet event : exemple (2/2)

Résultat :

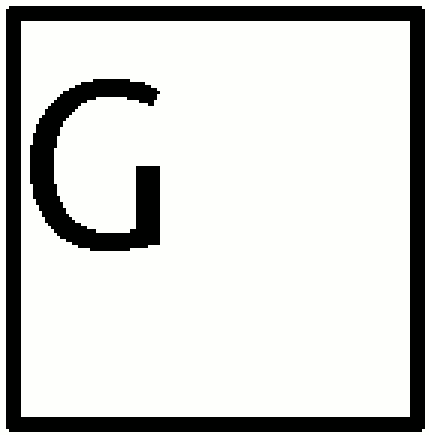
Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Objet event : exemple (2/2)

Résultat :

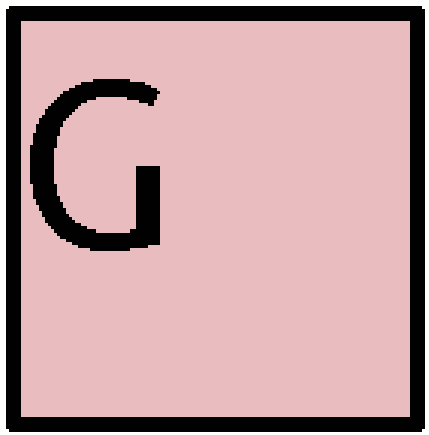
Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Objet event : exemple (2/2)

Résultat :

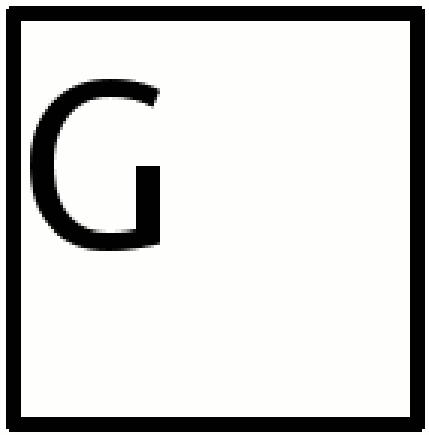
Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Objet event : exemple (2/2)

Résultat :

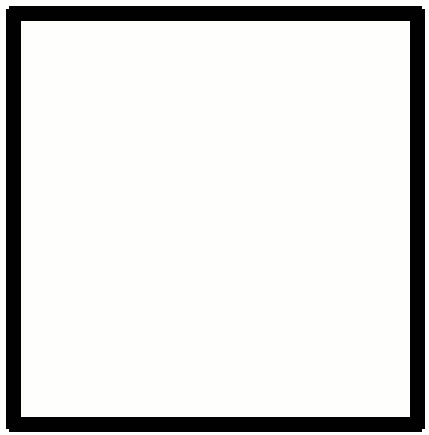
Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Objet event : exemple (2/2)

Résultat :

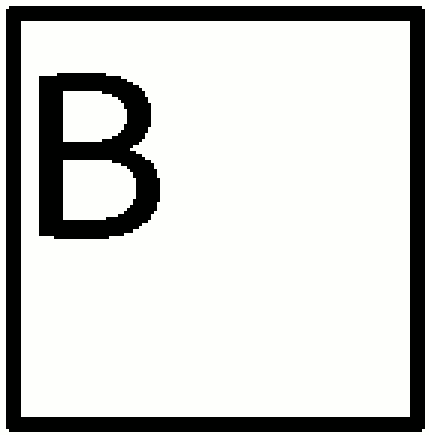
Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Objet event : exemple (2/2)

Résultat :

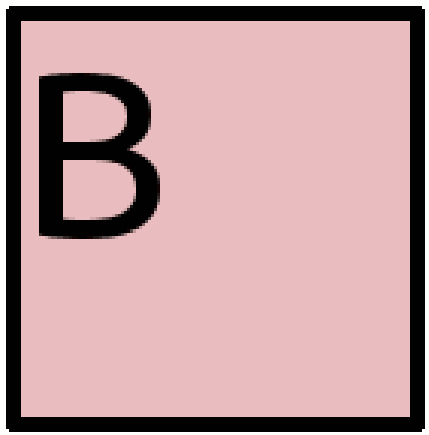
Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Objet event : exemple (2/2)

Résultat :

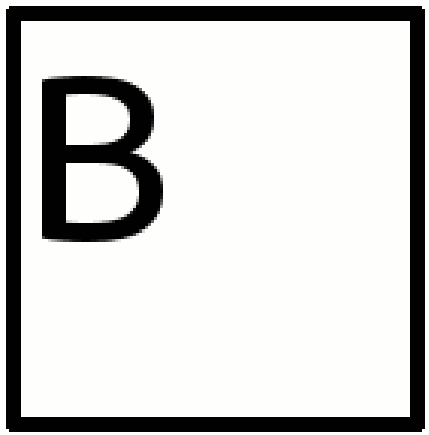
Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Objet event : exemple (2/2)

Résultat :

Essayez les touches "R", "G", et "B" pour avoir des étoiles dans les yeux.



Plan: Introduction à JQuery

Sources et références

Sources et références

Références :

- Cours de Jean-Christophe Routier (Université Lille 1) : `http://www.fil.univ-lille1.fr/~routier/enseignement/licence/tw1/spoc/#chap8`