ISEN ALL IS DIGITAL! OUEST | yncréa

JavaScript
- Programmation événementielle -

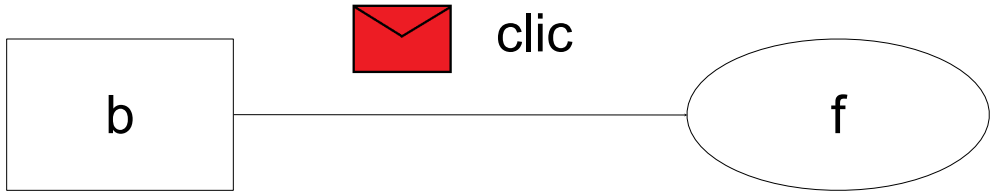
Groupe des étudiants : CIR1

Événement

- Un événement est un *message* décrivant un changement d'état d'une page web
- Les changements d'état sont la plupart du temps le résultat d'*actions de l'utilisateur*, comme :
  - le redimensionnement de la fenêtre
  - la sélection de texte
  - le survol ou le clic sur un élément
  - l'appui sur une touche du clavier
  - ...
- Mais il y a également des *événements système* comme :
  - le chargement (de la page, d'une image...)
  - la réception de données sur une connexion réseau
  - une erreur
  - …
- Le MDN référence sur cette page tous les événements existants, regroupés par catégorie. Vous verrez qu'ils sont très nombreux !

Gestionnaire d'événement

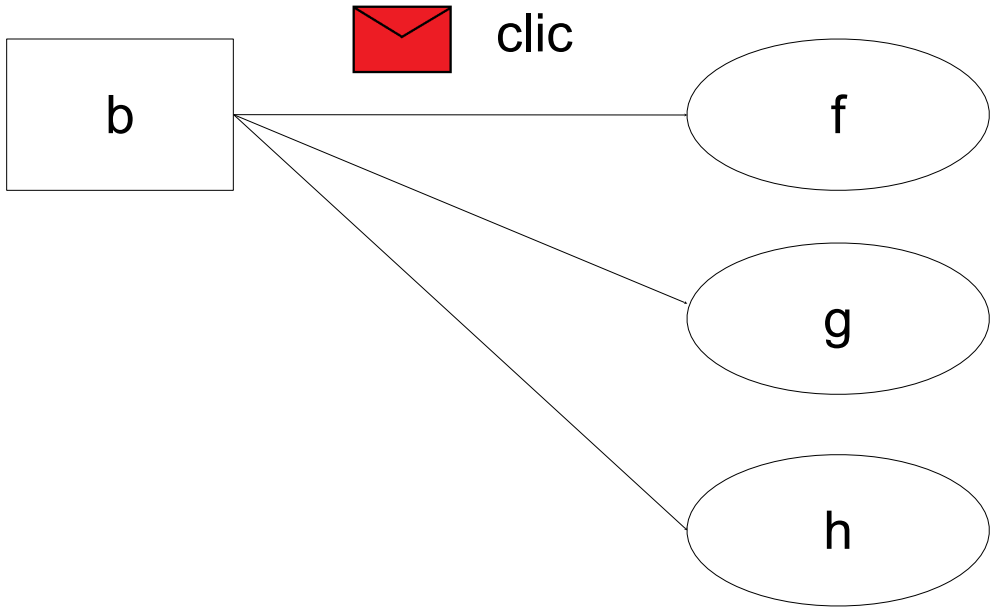
- On peut réagir à un événement en y associant une callback, appelée gestionnaire d'événement (*handler*) ou écouteur (*listener*)
- Exemple : la fonction f() écoute les clics sur le bouton b



- C'est la base de la *programmation événementielle*, qui est *asynchrone* par nature : le système attend que des événements se produisent et sollicite les gestionnaires associés.

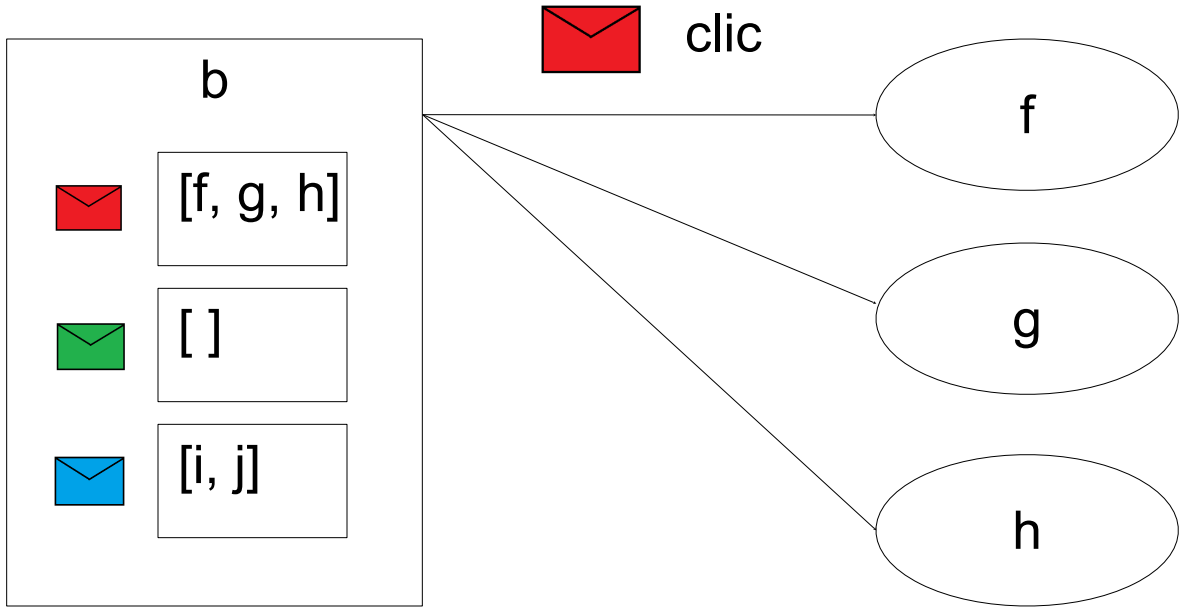
Gestionnaires d'événement

Plusieurs gestionnaires peuvent être associés (écouter) le même événement



Système de notification

- Concrètement, les gestionnaires s'*abonnent* à un événement auprès de l'émetteur
- Lorsque l'événement se produit, l'émetteur parcourt et notifie les gestionnaires



- C'est une mise en œuvre particulière d'un patron de conception appelé *observateur*, qui sera vu en programmation orientée objet

Définition

- Un événement est un message décrivant un changement d'état d'une page web

- Les changements d'état sont la plupart du temps le résultat d'actions de l'utilisateur, comme :
  - le redimensionnement de la fenêtre
  - la sélection de texte
  - le survol ou le clic sur un élément
  - l'appui sur une touche du clavier
  - ….

- Mais il y a également des événements système comme :
  - le chargement (de la page, d'une image...)
  - la réception de données sur une connexion réseau
  - une erreur
  - ….

Définition

- On peut réagir à un événement en y associant une callback, appelée gestionnaire d'événement (handler) ou écouteur (listener)

- Exemple : la fonction f() écoute les clics sur le bouton b

- C'est la base de la programmation événementielle, qui est asynchrone par nature : le système attend que des événements se produisent et sollicite les gestionnaires associés.

MISE EN PLACE

Ce qu'il ne faut pas faire

- Comme nous l'avons vu dans le cours d'introduction, il est possible d'associer du code JavaScript à certains attributs HTML, notamment les attributs `onxxx` ou `xxx` désigne un événement

- Par exemple, le code suivant permet d'appeler la fonction okClicked() au clic sur le bouton OK :

```
<button id="ok" onclick="okClicked()">OK</button>
```

- Nous rappelons que ceci est une mauvaise pratique car elle :
  - ne respecte pas la séparation structure/comportement,
  - rend le code difficile à lire, à maintenir ou à faire évoluer,
  - ne permet pas de réutiliser le code.

Mieux, mais limité

- Une autre possibilité est d'utiliser les propriétés `onxxx` des éléments HTML (variables de type HTMLElement) dans le code JavaScript
- Cette pratique ne possède plus les inconvénients mentionnés précédemment, mais ne permet d'associer qu'un seul gestionnaire à un événement
- Si on reprend l'exemple précédent :

```
function okClicked(){
  console.log("OK clicked");
}

let okButton = document.getElementById("ok");
okButton.onclick = okClicked;
```

(Rappel) pas de () ici !

Pratique recommandée

- Les éléments HTML possèdent une méthode `addEventListener(event,listener)` qui permet d'associer le gestionnaire `listener` (fonction) à l'événement `event` (chaîne de caractères)

- Concrètement, `listener` est ajouté à la liste des gestionnaires abonnés à `event`

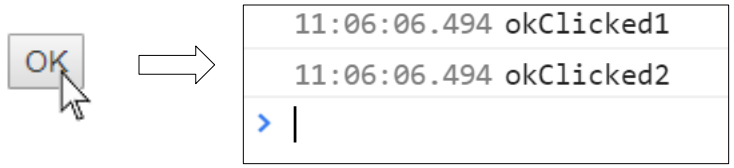
- Si on reprend l'exemple précédent :

```
function okClicked(){
  console.log("OK clicked");
}

let okButton = document.getElementById("ok");
okButton.addEventListener("click", okClicked);
```

Écrire "xxx" pour un événement xxx, et pas "onxxx" !

Exemple avec plusieurs gestionnaires




```
function okClicked1(){
    console.log("okClicked1");
}

function okClicked2(){
    console.log("okClicked2");
}


let okButton = document.getElementById("ok");
okButton.addEventListener("click", okClicked1);
okButton.addEventListener("click", okClicked2);
```

Note : on peut désabonner un gestionnaire à un événement avec removeEventListener()

EXPLOITATION DES ÉVÉNEMENTS

Événement en argument des gestionnaires

- Les gestionnaires ont un paramètre qui modélise l'événement qui a eu lieu. Ils sont donc de la forme `function(e)`, où `e` est de type Event

- On y trouve des informations très générales, par exemple :
  - l'émetteur de l'événement : `e.target`
  - le type d'événement : `e.type`

Exemple

```
function printInnerHTML(event){
    let targetInnerHTML = event.target.innerHTML;
    console.log("Target inner HTML : " + targetInnerHTML);
}

function printEventType(event){
    let eventType = event.type;
    console.log("Event type : " + eventType);
}


let okButton = document.getElementById("ok");
okButton.addEventListener("click", printInnerHTML);
okButton.addEventListener("click", printEventType);
```

Sous types d'Event

Les événements fournis en argument des gestionnaires sont en fait des sous-types d'Event : ils possèdent toutes les propriétés d'Event auxquelles s'ajoutent des propriétés spécifiques (selon la nature de l'événement). Par exemple :

- MouseEvent est associé aux événements souris. On pourra par exemple y trouver le bouton appuyé, les coordonnées du clic, etc.

- KeyboardEvent est associé aux événements clavier. On pourra par exemple y trouver la valeur de la touche, si elle est maintenue enfoncée, si elle a été combinée à ALT, SHIFT ou CTRL, etc.

Exemple

```
function printButton(mouseEvent){
    let button = mouseEvent.button;
    console.log("Mouse button : " + button);
}

function printClientCoordinates(mouseEvent){
    let x = mouseEvent.clientX;
    let y = mouseEvent.clientY;
    console.log("Client coords : ("+x+","+y+")");
}


let okButton = document.getElementById("ok");
okButton.addEventListener("click", printButton);
okButton.addEventListener("click", printClientCoordinates);
```

ANNEXES

Annuler le comportement par défaut

- Les événements ont une méthode preventDefaut() qui permet d'annuler le comportement par défaut associé à certains éléments HTML*.

- L'exemple classique est la soumission d'un formulaire :
  - par défaut, l'activation d'un bouton de type « submit » provoque l'envoi des données du formulaire au serveur
  - on peut annuler ce comportement si les données du formulaires ne sont pas valides

** A noter que ceci n'est possible que si l'événement est annulable, ce que l'on peut vérifier en accédant à sa propriété* `cancelable`

LISTE DES EVENTS POSSIBLE

Vous pouvez consulter la liste complète des events possible sur :

https://developer.mozilla.org/fr/docs/Web/Events