

# LA GESTION DES ÉVÉNEMENTS

# UN ÉVÉNEMENT ?

- Permet de déclencher une fonction selon une action enclenchée :
  - soit par un utilisateur lorsqu'il interagit avec un élément HTML
  - (avec la souris, le clavier ...au survol, au clic,...)
  - Soit de manière autonome(chargement d'une page...)



# LISTE DES ÉVÉNEMENTS

- Les événements sont identifiés selon leur type :

Catégorie	Effet	type
Événement window	Chargement Chargement de page	'load' 'unload' 'pageshow' 'pagehide'
événement souris	Clic  déplacer	'click' 'dblclick' 'mousedown' 'mouseup' 'mouseover' 'mouseenter' 'mouseout'
Événement clavier	Touche pressée	'keypress'
Événement focus	Focus d'attention	'focus' 'blur' (sur la cible) 'focusin' 'focusout' (bouillonne)



# LISTE DES ÉVÉNEMENTS

- Certains sont spécifiques au formulaire

Catégorie	Effet	type
Événement formulaire	<form>	'submit' 'reset'
	<u>&lt;input&gt;</u> , <u>&lt;select&gt;</u> et <u>&lt;textarea&gt;</u>	'change'
	<u>&lt;input&gt;</u> de type 'text' et <u>&lt;textarea&gt;</u>	'select'

- liste exhaustive de tous les événements (plus de 130 événements standards) :

<https://developer.mozilla.org/en-US/docs/Web/Reference/Events>



## UTILISATION – SANS LE DOM

- Il est possible d'utiliser les événements directement dans la balise, il faut dans ce cas là, ajouter le préfixe 'on' devant l'événement.
- Exemples :

```
<span onclick="alert('Hello !');">Cliquez-moi !</span>
```

Tout le code javascript doit être écrit entre les guillemets

```
<span onclick="alert('Vous avez cliqué sur' + this.innerHTML);">Cliquez-moi !</span>
```

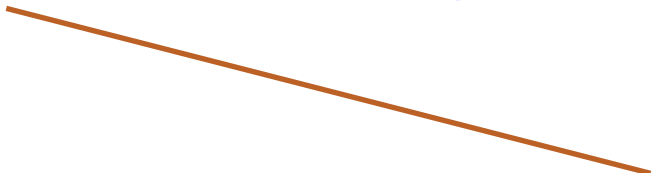
Le mot-clé 'this' renvoie à l'élément qui a déclenché l'événement, ici 'span'



# UTILISATION SANS DOM – CAS DES LIENS

- 'return false' empêche le lien d'aller vers la page ou l'ancre indiquée dans href

```
<a href="#" onclick="alert('Vous avez cliqué !'); return false;">Cliquez-moi !</a>
```



Préférer l'utilisation de l'élément **<button>** dans le cas où le lien n'est utilisé que comme déclencheur

- L'utilisation sans le DOM est possible mais reste limitée.



# UTILISATION AU TRAVERS DU DOM – DOM-0

- DOM-0, implémentée par Netscape. Méthode ancienne mais qui peut toujours être utilisée.
- L'événement est toujours préfixé par 'on', mais n'est plus présent dans la balise, sa valeur est une fonction. A l'aide de méthode(getElementById, querySelector...), on sélectionne l'élément dans le DOM

```
<script>
```

```
var element = document.getElementById('clickme');  
  
element.onclick = function() {  
    alert("Vous m'avez cliqué !");  
};
```

```
</script>
```



# UTILISATION AU TRAVERS DU DOM – DOM-2

- DOM-2 :
- Utilisation d'un écouteur d'événement.
- On positionne l'écouteur avec la méthode `addEventListener()`, 3 paramètres
  - Le nom de l'événement (sans le 'on')
  - La fonction à exécuter
  - Un booléen qui précise la phase(capture ou bouillonnement)

```
<script>
```

```
var element = document.getElementById('clickme');  
element.addEventListener('click', alert("Vous m'avez cliqué !"));
```

```
</script>
```





# DÉROULEMENT D'UN ÉVÉNEMENT

- Quels que soient les événements, ils suivent le MÊME schéma :
- 1 – l'événement est modélisé par un objet de type **Event** → objet **event**
- 2 – l'objet **event** va jusqu'à l'élément cible de cet événement, en partant de la racine de l'arbre (*capturing phase*)
- 3 – s'il y a un traitement associé à cet événement pour cette cible, il est exécuté
- 4- l'objet **event** remonte dans l'arbre (*bubbling phase*) et est traité par les nœuds qui écoutent cet événement



```
<div id="container">
  <span id="contenu">Du texte !</span>
</div>
```

True : phase de capture  
Affichage de 1 puis de 2

```
<script>
```

```
  var container = document.getElementById('container'),
      contenu = document.getElementById('contenu'),
```

```
  container.addEventListener('click', function() {
    alert("L'événement du div vient de se déclencher.");
  }, true);
```

```
  contenu.addEventListener('click', function() {
    alert("L'événement du span vient de se déclencher.");
  }, true);
```

---

```
  container.addEventListener('click', function() {
    alert("L'événement du div vient de se déclencher.");
  }, false);
```

```
  contenu.addEventListener('click', function() {
    alert("L'événement du span vient de se déclencher.");
  }, false);
```

```
</script>
```

false : phase de bouillonnement  
Affichage de 2 puis de 1  
Par défaut



# L'OBJET EVENT

**cible.addEventListener( typeEvenement, action ) ;**  
**function action(event) {...}**

- → chaque fois que **typeEvenement** se produit sur l'objet **cible**, l'écouteur exécute la **fonction action** (à qui on fait toujours passer un paramètre : c'est l'objet **event** qui contient toutes les informations sur l'événement qui vient de se produire).
- L'objet Event n'est donc accessible que lorsqu'un événement est déclenché

```
<script>
  element.addEventListener('click', function(e) {
    alert(e.type);
  });
</script>
```

Affiche le type de  
l'objet Event, ici  
click

Récupère une  
référence de  
l'objet Event  
(e ou autre)

# FONCTIONNALITÉS DE L'OBJET EVENT

- **target** : récupère l'élément de l'événement actuellement déclenché (élément sélectionné)

```
<script>
  var clickme = document.getElementById('clickme');

  clickme.addEventListener('click', function(e) {
    e.target.innerHTML = 'Vous avez cliqué !';
  });
</script>
```

Récupère l'élément d'id  
'clickme'

- **currentTarget** : récupère l'élément à l'origine de l'événement déclenché, et non pas ses enfants.(l'élément en cours)



# FONCTIONNALITÉS DE L'OBJET EVENT

- **clientX et clientY** : récupère la position du curseur
- **key** : propriété pour récupérer la valeur de la touche

```
document.addEventListener('keypress', action1);  
function action1(evt) {window.alert(evt.key);}
```



Événement  
keypress

- → Supprimer l'écouteur :  
cible.removeEventListener(type, action) ;
- → Annuler la propagation de l'événement :  
event.stopPropagation() ;

