

Event


Tips :

- Pour connaître le type de l'événement.
- Pour connaître l'élément sur lequel on clique.
- Pour connaître la currentTarget c'est à dire l'élément qui exécute le code.
- Pour connaître la valeur X ou Y de la ou on clique sur l'élément.

```
<> index.html  JS script.js  # style.css
1  window.onload = function windowReady(){
2      console.log("La page est charger");
3      const parent = document.querySelector("#parent");
4      const enfant = document.querySelector("#enfant");
5
6      enfant.addEventListener("click", enfantFunction);
7      parent.addEventListener("click", parentFunction);
8
9
10     function enfantFunction(oEvt){
11         //console.log("J'ai cliquer ENFANT");
12     }
13
14     function parentFunction(oEvt){
15         //console.log("J'ai cliquer PARENT");
16         console.log(oEvt.type);
17         console.log(oEvt.target);
18         console.log(oEvt.currentTarget);
19         console.log(oEvt.clientX);
20         console.log(oEvt.clientY);
21     }
22 }
23 };
24
25
```

onload est une propriété de window, c'est un gestionnaire d'événement(EventHandler), quand l'événement se produit la fonction se déclenche.

NOTE IMPORTANTE : UN EVENT HANDLER NE PEUT PRENDRE QU'UNE SEUL PROPRIETE, SI L'ON VEUT ASSIGNER DIFFÉRENTE FONCTION À UN MÊME ÉVÉNEMENT IL FAUT METTRE UN ÉCOUTEUR D'ÉVÉNEMENT "ADDEVENTLISTENER"



```
...  
onkeyup: null  
onlanguagechange: null  
▶ onload: f onloadReady(oEvt)  
onloadeddata: null  
onloadedmetadata: null  
onloadstart: null  
onlostpointercapture: null
```


```
window.onload = function onloadReady(oEvt){  
    console.log("La page est charger");  
};
```

La page est charger

script.js:3

>

On peut également lui assigner la fonction directement.



```
window.onload = windowReady ;  
  
function windowReady(){  
    console.log("La page est charger");  
};
```

```
<? index.html x JS script.js # style.css
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Javascript</title>
5 <meta charset="utf-8" />
6 <meta name="viewport" content="width=device-width, initial-scale=1">
7 <link rel="stylesheet" href="style.css">
8 </head>
9
10 <body>
11 <div class="rouge">Rouge</div>
12 <div class="vert">Vert</div>
13 <div class="bleu">Bleu</div>
14 <script src="script.js"></script>
15 </body>
16
17 </html>
```

```
<? index.html JS script.js # style.css
1 window.onload = function windowReady(){
2     console.log("La page est charger");
3     const rouge = document.querySelector(".rouge");
4     const vert = document.querySelector(".vert");
5     const bleu = document.querySelector(".bleu");
6
7
8     rouge.onclick = function(oEvt){
9         rouge.textContent = "On vient de me cliquer";
10    }
11
12 };
```



NOTE IMPORTANTE : UN EVENT HANDLER NE PEUT PRENDRE QU'UNE SEUL PROPRIETE, SI L'ON VEUT ASSIGNER DIFFÉRENTE FONCTION À UN MÊME ÉVÉNEMENT IL FAUT METTRE UN ÉCOUTEUR D'ÉVÉNEMENT "ADDEVENTLISTENER"

onclick est une des propriétés eventHandler de l'élément rouge, on peut récupérer l'objet événement en le passant en parametre de la fonction.

On peut récupérer l'objet événement et l'afficher pour acceder a c'est propriété si besoin.

```
script.js:9
MouseEvent {isTrusted: true, screenX: 2877, screenY: 116, clientX: 227, clientY: 19, ...}
```

```
rouge.onclick = function(oEvt){
    console.log(oEvt);
    rouge.textContent = "On vient de me cliquer";
}
```

```
script.js:9
MouseEvent {isTrusted: true, screenX: 2877, screenY: 116, clientX: 227, clientY: 19, ...}
```

Quand on a besoin de faire plusieurs actions à différent endroit du code sur un événement on va utiliser "addEventListener" pour écouter l'événement quand il se déclenche.

Il y a également possibilité de passer le nom fonction en paramètre du addEventListener, dans le cas si on aurait besoin de la réutiliser à un autre endroit du code.

On peut supprimer un listener.


```
<> index.html JS script.js # style.css
1 window.onload = function windowReady(){
2   console.log("La page est charger");
3   const rouge = document.querySelector(".rouge");
4   const vert = document.querySelector(".vert");
5   const bleu = document.querySelector(".bleu");
6
7
8   function modifyRed(oEvt){
9     console.log(oEvt);
10    rouge.textContent = "On vient de me cliquer";
11  }
12
13  function modifyblue(oEvt){
14    console.log(oEvt);
15    bleu.textContent = "On vient de cliquer rouge";
16  }
17
18  // EN PASSANT DIRECTEMENT LA DECLARATION DE LA FONCTION.
19  rouge.addEventListener("click", function (oEvt){
20    console.log(oEvt);
21    rouge.textContent = "On vient de me cliquer";
22  });
23
24  // EN PASSANT LE NOM DE LA FONCTION
25  rouge.addEventListener("click", modifyblue);
26
27
28  // ON SUPRIME LE LISTENER modifyblue DE L'ELEMENT "rouge" QUAND ON CLIQUE SUR L'ELEMENT VERT
29  vert.addEventListener("click", function(){
30    rouge.removeEventListener("click", modifyblue);
31  });
32
33
34
35 };
```

La fonction `preventDefault()` permet d'empêcher le comportement par défaut d'un événement, exemple quand on valide un formulaire cela empêche à la page de se rafraîchir automatiquement.

```
1 window.onload = function windowReady(){
2     console.log("La page est charger");
3     const form = document.querySelector("form");
4
5
6     form.addEventListener("submit",envoyerFormulaire);
7
8
9     function envoyerFormulaire(oEvt){
10         console.log("Formulaire envoyer ");
11         oEvt.preventDefault();
12     }
13
14 };
```


La propagation des événements

Par défaut le code s'effectue pendant la phase de Bubbling c'est à dire remontante.



```
< index.html  JS script.js  x  # style.css
1  window.onload = function windowReady(){
2      console.log("La page est charger");
3      const parent = document.querySelector("#parent");
4      const enfant = document.querySelector("#enfant");
5
6      enfant.addEventListener("click", enfantFunction);
7      parent.addEventListener("click", parentFunction);
8
9
10     function enfantFunction(oEvt){
11         console.log("J'ai cliquer ENFANT");
12     }
13
14     function parentFunction(oEvt){
15         console.log("J'ai cliquer PARENT");
16     }
17
18
19
20
21 };
```

Si l'on veut effectuer le code pendant la phase de capture c'est à dire pendant la phase descendante il faut ajouter a notre fonction addEventListener le paramètre true.



```
< index.html  JS script.js  •  # style.css
1  window.onload = function windowReady(){
2      console.log("La page est charger");
3      const parent = document.querySelector("#parent");
4      const enfant = document.querySelector("#enfant");
5
6      enfant.addEventListener("click", enfantFunction, true);
7      parent.addEventListener("click", parentFunction, true);
8
9
10     function enfantFunction(oEvt){
11         console.log("J'ai cliquer ENFANT");
12     }
13
14     function parentFunction(oEvt){
15         console.log("J'ai cliquer PARENT");
16     }
17
18
19
20
21 };
```

stopPropagation() stop la propagation du listener au premier élément rencontrer.

```
function enfantFunction(oEvt){  
    console.log("J'ai cliquer ENFANT");  
    oEvt.stopPropagation();  
}
```