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.is
                                  # style.css
      window.onload = function windowReady(){
           console.log("La page est charger");
           const parent = document.querySelector("#parent");
           const enfant = document.querySelector("#enfant");
           enfant.addEventListener("click", enfantFunction);
           parent.addEventListener("click", parentFunction);
           function enfantFunction(oEvt){
               //console.log("J'ai cliquer ENFANT");
           function parentFunction(oEvt){
               //console.log("J'ai cliquer PARENT");
               console.log(oEvt.type);
               console.log(oEvt.target);
               console.log(oEvt.currentTarget);
               console.log(oEvt.clientX);
               console.log(oEvt.clientY);
      };
```

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

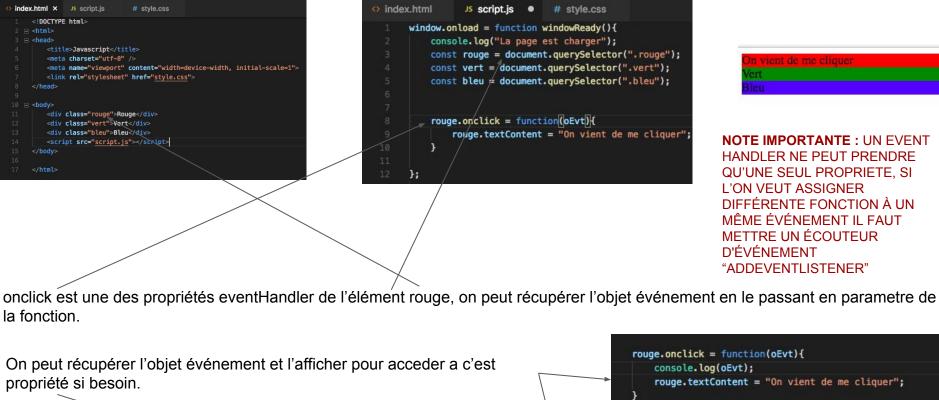
onload: f onloadReady(oEvt)
onloadeddata: null
onloadedmetadata: null
onloadstart: null
```

La page est charger <u>script.js:3</u>

On peut également lui assigner la fonction directement.

```
window.onload = windowReady;

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



METTRE UN ÉCOUTEUR D'ÉVÉNEMENT "ADDEVENTLISTENER"

script.js:9

la fonction.

rouge.onclick = function(oEvt){ On peut récupérer l'objet événement et l'afficher pour acceder a c'est console.log(oEvt); propriété si besoin. rouge.textContent = "On vient de me cliquer"; _MouseEvent {isTrusted: true, screenX: 2877, screenY: 116, clientX: 227, clientY: 19, ...} altKey: false bubbles: true button: 0 buttons: 0 MouseEvent {isTrusted: true, screenX: 2877, screenY: 116, clientX: 227, clientY: 19, ...} cancelBubble: false

cancelable: true clientX: 227 clientY: 19 composed: true

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
                                 # style.css
                 Js script.js •
      window.onload = function windowReady(){
          console.log("La page est charger");
          const rouge = document.querySelector(".rouge");
          const vert = document.querySelector(".vert");
          const bleu = document.querySelector(".bleu");
          function modifyRed(oEvt){
              console.log(oEvt);
              rouge.textContent = "On vient de me cliquer";
          function modifyblue(oEvt){
              console.log(oEvt);
              bleu.textContent = "On vient de cliquer rouge";
          // EN PASSANT DIRECTEMENT LA DECLARATION DE LA FONCTION.
          rouge.addEventListener("click", function (oEvt){
              console.log(oEvt);
              rouge.textContent = "On vient de me cliquer";
          });
          // EN PASSANT LE NOM DE LA FONCTION>
          rouge.addEventListener("click", modifyblue);
          // ON SUPRIME LE LISTENER modifyblue DE L'ELEMENT "rouge" QUAND ON CLIQUE SUR L'ELEMENT VERT
          vert.addEventListener("click", function(){
              rouge.removeEventListener("click", modifyblue);
          })
      };
```

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.

```
window.onload = function windowReady(){
    console.log("La page est charger");
    const form = document.querySelector("form");

form.addEventListener("submit",envoyerFormulaire);

function envoyerFormulaire(oEvt){
    console.log("Formulaire envoyer ");
    oEvt.preventDefault();
};
```

La propagation des événements

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

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.

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

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