Semaine 11 - Interface MouseEvent et curseur personnalisé

Interface MouseEvent

L'interface **MouseEvent** représente les événements qui se produisent lors d'une interaction de l'utilisateur avec un dispositif de pointage, tel qu'une **souris**. Les principaux événements utilisant cette interface et les actions pour les déclencher sont les suivants :

Nom de l'événement	Action pour le déclencher
click	Cliquer (appuyer puis relâcher) sur l'élément HTML
dblclick	Double-cliquer sur l'élément HTML
mousedown	Appuyer (sans relâcher) sur le bouton gauche de la souris sur l'élément HTML
mouseup	Relâcher le bouton gauche de la souris sur l'élément HTML
mouseover	Déplacer le curseur dans la zone d'un élément HTML
mouseout	Déplacer le curseur en dehors de la zone d'un élément HTML
mousemove	Bouger le curseur lorsqu'il est dans la zone d'un l'élément HTML

Nous avons déjà vu que la syntaxe pour attribuer un gestionnaire d'événement de type MouseEvent est la suivante:

```
//Attribution du gestionnaire d'événement pour le clic de la souris
elementCible.addEventListener("click", fonctionSiClicBouton);
```

Lorsque l'événement sera distribué, un objet de type « *MouseEvent* » sera **obligatoirement** envoyé en paramètre au gestionnaire d'événement (ou fonction écouteur, « *listener* » en anglais). Cet objet est appelé **objet événementiel**. C'est pour cette raison que la fonction écouteur **doit préférablement** contenir dans sa signature un paramètre pour récupérer l'événement diffusé, par exemple:

```
function fonctionSiClicBouton (evenementDeTypeMouseEvent) {
……..
}
```

Rappelons que l'objet événementiel contient des informations liées à l'événement en cours de diffusion. Ces informations sont disponibles à travers des **propriétés de l'objet événementiel**. Les propriétés suivantes concernent la majorité des événements diffusés :

- **type** : Cette propriété contient le nom de l'événement diffusé sous forme de chaîne.
- **target** : La propriété target fait référence à l'objet cible de l'événement, c'est-à-dire à celui qui est à l'origine de la propagation de l'événement.

Pour un événement de souris (« *MouseEvent* »), plusieurs propriétés s'ajoutent qui permettent notamment de connaître **l'endroit où l'événement s'est produit** et, si les touches Ctrl, Shift, etc. étaient enfoncées ou non lors de l'événement.

Propriétés d'un événement MouseEvent

L'interface « *MouseEvent* » fournit, au moyen de différentes propriétés des renseignements contextuels spécifiques associés à des événements de la souris. Les propriétés faisant l'objet d'une recommandation officielle du W3C (DOM Level 3)[4] , sont les suivantes :

- **altKey** (type: Booléen). Cette propriété indique, si lors de l'événement, la touche Alt est enfoncée (true) ou non (false);
- **clientX** (type: Nombre entier). La coordonnée en X du pointeur de la souris (axe horizontal) par rapport à la fenêtre visible du navigateur;
- **clientY** (type: Nombre entier. La coordonnée en Y du pointeur de la souris (axe vertical) par rapport à la fenêtre visible du navigateur;
- **ctrlKey** (type: Booléen). Cette propriété indique, si lors de l'événement, la touche Ctrl est activée (true) ou non (false);
- **shiftKey** (type: Booléen). Cette propriété indique, si lors de l'événement, la touche Shift (Maj) est activée (true) ou non (false);
- **screenX** (type: Nombre entier). La coordonnée en X du pointeur de la souris (axe horizontal) de façon globale par rapport à l'écran physique de l'utilisateur;
- **screenY** (type: Nombre entier). La coordonnée en Y du pointeur de la souris (axe vertical) de façon globale par rapport à l'écran physique de l'utilisateur.

Notons que l'**unité de mesure** de toutes les coordonnées en **X** et en **Y** retournées est en **pixel**.

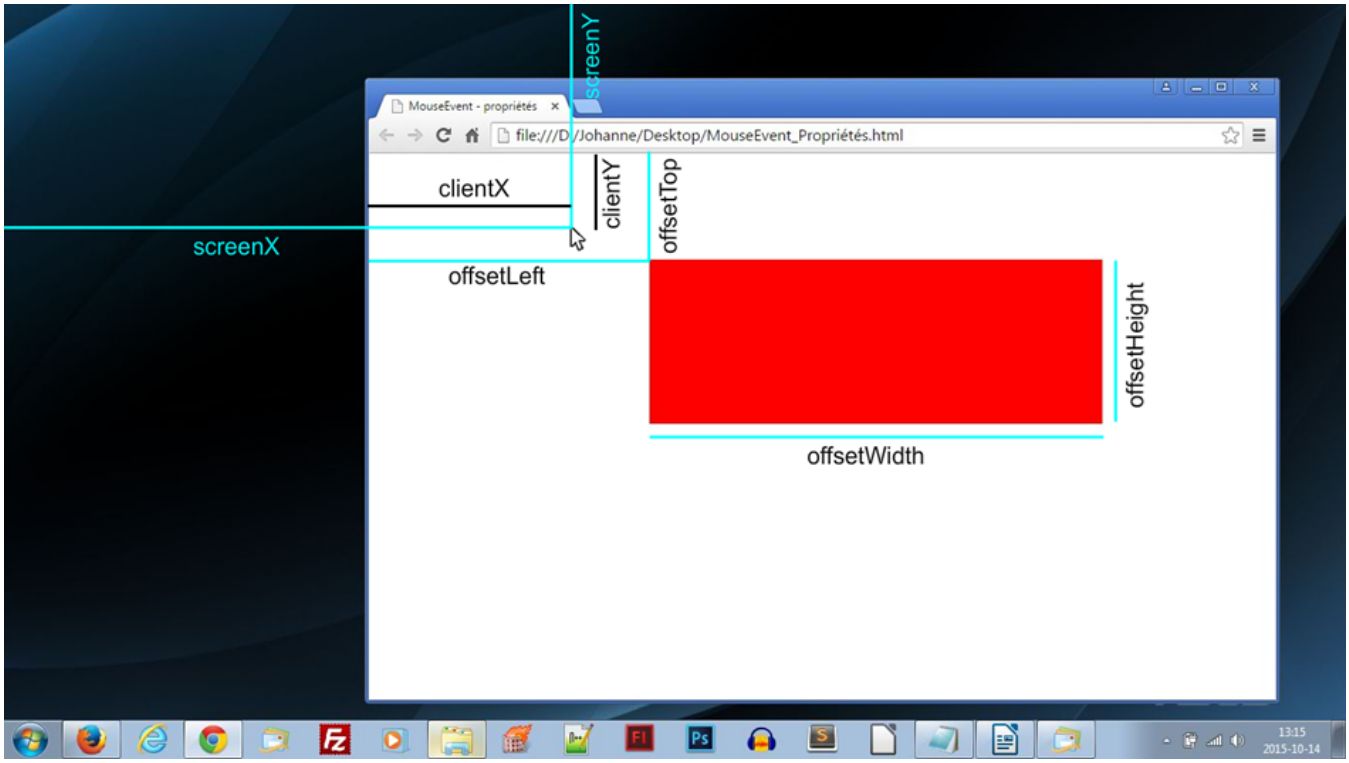
Le code pour placer un élément HTML dans la balise <body> à l'emplacement d'un clic dans la fenêtre du navigateur pourrait être le suivant :

```
//Écouteur sur le document HTML
document.addEventListener("click", placerElementHTML);

function placerElementHTML(event) {
  elementHTML.style.top = event.clientY + "px";
  elementHTML.style.left = event.clientX + "px";
}
```

La figure de la page suivante illustre les positions du pointeur de la souris (*ou d'un doigt…*) dans l'écran ou dans la fenêtre du navigateur, ainsi que d'autres valeurs qui peuvent être récupérées par programmation au sujet du dimensionnement et de la position des éléments dans une page HTML.

[4] D'autres propriétés sont en étude présentement par le W3C, comme: ***offsetX*** et ***offsetY*** (positions relatives de la souris par rapport à l'élément cible), ***pageX*** et ***pageY*** (positions de la souris par rapport au document html (ie. La mise en page de la fenêtre en tenant compte du défilement) toutefois, celles-ci ne sont pas abordées dans ce document car leur implémentation est très hétérogène entre les différents navigateurs.



NOTE IMPORTANTE : Pour positionner un élément dans la fenêtre du navigateur, ou dans l'un de ses éléments HTML parent, en fonction de la position de la souris, le conteneur parent doit avoir une **position CSS** (propriété CSS) **définie** autre que la valeur par défaut (« static »), soit : **fixed**, **relative** ou **absolute**. L'élément à placer doit quant à lui, être positionné de façon absolue avec la propriété CSS position définie à « **absolute** ».

Notons toutefois que dans le cas d'un **curseur personnalisé**, le positionnement CSS doit être **fixe (position: fixed)** afin que le bloc englobant (parent ou ancêtre) corresponde au **viewport** et que l'élément libère l'espace dans la page où il aurait normalement été situé.

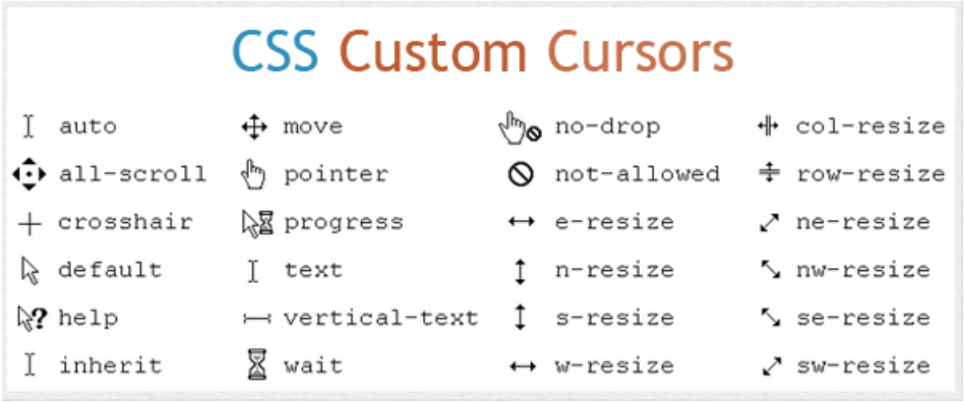
L'élément enfant sera placé dans son conteneur parent en **X** et en **Y** avec les propriétés CSS **left** et **top** définies à partir du **coin supérieur gauche du conteneur parent**. Dans certains cas, il est aussi possible d'utiliser des **transformations CSS**, notamment avec **translate(dX, dY)**.

Modifier l'apparence du curseur avec CSS

La propriété CSS **cursor** permet de modifier l'apparence du curseur lorsque le pointeur est au-dessus de l'élément HTML dans lequel cette propriété est définie. Évidemment, cela n'est pertinent que dans les navigateurs/systèmes d'exploitation dans lesquels il y a une **souris** et un curseur. Ils sont utilisés essentiellement pour l'UX pour donner à l'utilisateur un retour visuel de ce qu'il fait à l'écran.

Curseurs CSS disponibles

L'image suivante illustre différentes possibilités des curseurs CSS et les mots-clés correspondant. Il est à noter toutefois que **l'aspect** des différents curseurs peut **varier** d'un navigateur à l'autre.



Source: https://www.scmgalaxy.com/tutorials/how-to-use-cursor-property-in-css/

Curseur personnalisé avec CSS

La propriété CSS **cursor** permet également d'ajouter notre propre image en tant que curseur sur nos pages Web. La syntaxe minimale est la suivante:

- **cursor: url(url de l'image), curseur de secours;**

Par exemple:

```
/*On modifie le curseur par défaut*/
cursor: url(curseurPerso.png), pointer;
```

Il est important de définir le type de curseur à afficher lorsque le navigateur ne peut pas utiliser l'image fournie, sinon le code ne fonctionnera pas.

Formats des images

Les images peuvent être au format ***.png, *.cur**, ***.svg**. Notez toutefois que tous les navigateurs ne prennent pas en charge les fichiers ***.svg** pour les curseurs, il peut donc être judicieux de **fournir une alternative *.png ou *.cur** si vous souhaitez utiliser un curseur ***.svg**. À cet effet, la propriété **cursor** peut être définie grâce à **une ou plusieurs valeurs <url> séparées par des virgules** et suivi en dernier par un unique mot-clé obligatoire. Le navigateur essaiera de télécharger la première image indiquée et passera à la suivante s'il ne peut pas. Si aucune image ne fonctionne, il utilisera le mot-clé final. Par exemple:

```
/*On modifie le curseur par défaut avec des alternatives*/
cursor: url(curseurPerso.svg), url(curseurPerso.png), pointer;
```

Chaque <url> peut éventuellement être suivi par un **couple de nombres** séparés par un espace qui représente les **coordonnées** (abscisse puis ordonnée) **<x> <y>**. Ces coordonnées permettent de paramétrer le **point d'action** (*hotspot*) du curseur par rapport au coin en haut à gauche de l'image (position du point d'action par défaut).

Par exemple, dans le code qui suit, on place le point d'action au centre de l'image du curseur personnalisé qui a une taille de 64px X 64px:

```
/*On modifie le curseur par défaut*/
cursor: url(curseurPerso.png) 32 32, pointer;
```

Taille des images

Bien que la **spécification** ne définit aucune limite de taille pour les curseurs, chaque navigateur/agent utilisateur pourra imposer la sienne. Les images de curseur utilisées qui dépassent les limites prises en charge seront généralement ignorées. On privilégiera normalement des images aux dimensions de 32px X 32px, 64px X 64px, avec un maximum de 128px X 128px.

Modifier l'apparence du curseur avec CSS et JavaScript

L'usage de curseur personnalisé, animé ou non, pour **améliorer l'expérience utilisateur** d'un site Web est de plus en plus fréquent sur le Web.

On voit en effet de plus en plus de sites qui utilisent des curseurs personnalisés, il y a évidemment **l'aspect esthétique**, lorsque l'aspect du curseur fait partie intégrante de la conception visuelle (*UI design),* mais aussi **pratique** et cela peut clairement entrer dans une démarche d'amélioration de l'expérience utilisateur (*UX design*) en donnant des directives additionnelles de navigation à l'utilisateur.

Par contre, n'oublions pas que ce curseur ne sera visible que sur ordinateur de bureau (*desktop*). Il ne faut donc pas baser toute sa navigation avec ce genre de technique.

Principales étapes de réalisation

Plusieurs aspects peuvent être gérés lors de la création de curseurs personnalisés avec CSS et JavaScript. **Cette note de cours s'attarde uniquement à décrire les étapes essentielles.**

Première étape: partie HTML

Dans la première étape, il s'agit simplement de créer une vide qui va être notre curseur. Dans l'exemple qui suit, on va également attribuer la classe CSS "*curseur*" à cette division:

```
    <div class="curseur"></div>
```

Deuxième étape: partie CSS

En CSS nous allons premièrement définir deux propriétés personnalisées (variables CSS). Celles-ci seront dénommées ici - - mouse-x et - - mouse-y :

```
:root {
    --mouse-x: 0px;
    --mouse-y: 0px;
}
```

Nous allons ensuite cacher le curseur CSS par défaut sur l'élément HTML <body> qui représente le contenu principal du document :

```
body {
    /*On enlève l'affichage du curseur par défaut*/
    cursor: none;
}
```

Ensuite, on va **retirer** l'élément HTML qui représentera le curseur du **flux normal** du document avec la propriété position définie à fixe (*fixed*), comme cela le curseur sera positionné relativement au bloc englobant initial formé par la zone d'affichage (viewport):

```
.curseur {
    /*Positionnement CSS: fixed*/
    position: fixed;
}
```

Enfin, on va **styliser** le curseur pour lui donner une taille et une forme. On va ensuite positionner cette forme avec les propriétés top et left dont les valeurs vont correspondre aux variables CSS déclarés ci-haut avec un léger décalage afin que le **point d'action** (*hotspot*) du curseur soit au **centre de la forme**. Le décalage correspond à la **moitié de la hauteur** de la forme et à la **moitié de sa largeur**.

```
.curseur {
    /*Positionnement CSS: fixed*/
    position: fixed;

    /*Forme du curseur*/
    height: 2.4rem;
    width: 2.4rem;
    background-color: rgba(140, 140, 140, 0.6);
    border: 1px solid #f9f9f9;
    border-radius: 50%;

    /*Positionnement et décalage pour que le centre corresponde au point d'action
      (hotspot) du curseur
    */
    top: calc(var(--mouse-y) - 1.2rem);
    left: calc(var(--mouse-x) - 1.2rem);
}
```

Enfin, on va **gérer les bogues.** En effet, comment fait-on pour cliquer sur un lien avec une div en position fixe déplacée en JavaScript. En CSS tout simplement il vous faudra mentionner la propriété « **pointer-events: none;** » , on va mettre également une valeur de **z-index** élevée pour s'assurer que le curseur personnalisé sera en avant-plan de tous les éléments de la page. Le code CSS final serait le suivant:

```
.curseur {
    /*Positionnement CSS: fixed*/
    position: fixed;

    /*Forme du curseur*/
    height: 2.4rem;
    width: 2.4rem;
    background-color: rgba(140, 140, 140, 0.6);
    border: 1px solid #f9f9f9;
    border-radius: 50%;
```

```
    /*Décalage pour que le centre corresponde au point d'action (hotspot) du
    curseur*/
    top: calc(var(--mouse-y) - 1.2rem);
    left: calc(var(--mouse-x) - 1.2rem);

    /*Enlever la détection des clics de souris et mettre en avant-plan*/
    pointer-events: none;
    z-index: 500;
}
```

Il est à noter que pour ajouter des effets plus dynamiques, il est possible d'ajouter des **transitions CSS** sur les propriétés qui seront modifiées, mais ce n'est pas une obligation.

Troisième étape: partie JavaScript

Dans la partie JavaScript, il faut d'abord récupérer le **sélecteur CSS :root** pour avoir accès aux variables CSS déclarées. Nous allons ensuite attribuer un gestionnaire d'événement "**mousemove**" sur le document qui va venir gérer les valeurs des variables **- -mouse-x** et **- -mouse-y**, pour ajuster la position du curseur au fil du déplacement du pointeur caché dans la fenêtre du navigateur.

```
//Récupérer la pseudo-classe :root
let leRoot = document.querySelector(":root");
//Écouteur sur le document pour l'événement de souris "mousemove"
document.addEventListener("mousemove", deplacerCurseur);
/**
 * Fonction permettant de déplacer le curseur à l'endroit du
 * pointeur de la souris dans l'écran
 */
function deplacerCurseur(event) {
    //On change ici les valeurs des variables CSS déclarées
    leRoot.style.setProperty("--mouse-x", event.clientX + "px");
    leRoot.style.setProperty("--mouse-y", event.clientY + "px");
}
```

Optimisation

Il est possible d'optimiser l'effet d'animation du déplacement du curseur en utilisant la propriété CSS **will-change**. Cette propriété fournit une indication au navigateur sur la propension d'un élément à changer (afin que le navigateur puisse mettre en place les optimisations nécessaires avant que l'élément change vraiment). Ce type d'optimisation permet d'augmenter la réactivité de la page en effectuant des calculs en prévision du changement.

Comme cette **propriété** est éventuellement **coûteuse en calculs**, elle **doit être utilisée avec parcimonie**.

Dans notre exemple, puisque ce sont les propriétés top et left qui sont modifiées indirectement avec le code JavaScript, on ajoutera dans le sélecteur CSS, la commande suivante:

```
.curseur {
    /*...*/
```

```
    /*Le navigateur doit être à l'affût des changements de propriétés du curseur
  will-change: top, left;
}
```

Éviter les conflits avec les périphériques mobiles

Certains événements de souris ne sont pas détectés sur les périphériques mobiles, notamment "*mouseover*" et "*mousemove*". Afin d'éviter un éventuel conflit, notamment en enlevant l'affichage du curseur par défaut dans le code CSS, on va trouver un moyen d'enlever le curseur et de mettre les gestionnaires d'événements "*mousemove*" sur la fenêtre du navigateur uniquement si le **mécanisme principal de saisie du périphérique a la capacité de survoler les éléments**.

En CSS, la requête qui nous permet de détecter cela est la suivante: **@media(hover: hover)** (Voir: https://developer.mozilla.org/fr/docs/Web/CSS/@media/hover . Nous allons donc gérer cette requête en JavaScript avec la méthode **window.matchMedia()** dont la syntaxe est la suivante:

- **window.matchMedia("(requeteSousForneDeChaîne)").matches**

La méthode va retourner une valeur booléenne (true ou false) si la requête est vérifiée. Alors, dans le code JavaScript on va ajouter une fonction gererCurseur() dont le code sera le suivant:

```
//L'élément HTML qui agira S'IL Y A Lieu comme curseur.
let leCurseur = null;

//Gérer ou NON l'apparence du curseur selon le dispositif de pointage de
l'utilisateur
gererCurseur();

function gererCurseur() {
    leCurseur = document.querySelector(".curseur");
    // Vérifier avec une requête média la capacité
    //du mécanisme de saisie principal à survoler les éléments
    if (window.matchMedia("( hover: hover )").matches) {
        //On enlève l'affichage du curseur par défaut
        document.querySelector("body").style.cursor = "none";
        //Écouteur sur le document pour l'événement de souris "mousemove"
        document.addEventListener("mousemove", deplacerCurseur);
    } else {
        //On met l'affichage du curseur par défaut
        document.querySelector("body").style.cursor = "auto";
        //On enlève la balise du curseur de l'affichage
        leCurseur.style.display = "none";
    }
}
```

Quelques références utiles

- Modzilla (Octobre 2022). Cursor - CSS : Feuilles de style en cascade | MDN. Repéré à: https://developer.mozilla.org/fr/docs/Web/CSS/cursor
- Modzilla (Octobre 2022). MouseEvent - Reference Web API | MDN. Repéré à : }https://developer.mozilla.org/fr/docs/Web/API/MouseEvent
- Modzilla (Septembre 2022). Position - CSS: feuilles de style en cascade | MDN. Repéré à : https://developer.mozilla.org/fr/docs/Web/CSS/position
- Modzilla (Octobre 2022). Will-change - CSS: feuilles de style en cascade | MDN. Repéré à https://developer.mozilla.org/fr/docs/Web/CSS/will-change