Chapitre 3 :

Les objets du navigateur et les événements

Ce chapitre aborde le concept d'objet avant de se focaliser sur les objets du navigateur et enchaîner avec les événements.

I. Le modèle Objet

Le langage Javascript est un orienté objet, il permet de manipuler différents objets

I.1. Le concept d'objet

Un objet représente une entité du monde réel (voiture, personne, cahier, classe,…).

Il est caractérisé par :

- un ***état*** : collection de données (attributs/propriétés)
- un ***comportement*** décrit par des traitements (méthodes)

Ainsi, un objet regroupe les données et les moyens de traitement de données.

Exemples :

Objet 1 : Voiture Ford Fiesta

Marque : Ford
Modèle : Fiesta
Couleur : vert
Nombre de chevaux : 4
Immatriculation : 201TU8795
Nombre de roues : 4

Etat

Démarrer
Accélérer

Comportement

Objet 2 : Tableau : t = [10, 20, 30, 47, 58]

Taille : 58

Etat

Ajouter
InsérerElément
SupprimerElément
Inverser
Trier

Comportement

I.2. Création d'un objet

Un objet en javascript peut être créé :

- **De façon littérale**

Syntaxe :

```
nomObjet = {propriété1 :valeur, propriété2:valeur,…};
```

Exemple :

```
let uneVoiture = {marque:"Ford", couleur:"blanc", nbRoues: 4};
```

- **Par instanciation**

Syntaxe :

```
nomObjet = new TypeObjet ([paramètres]);
```

Exemples :

```
let tab = new Array(3,58,69,12);
let uneDate = new Date(2023,1,27);
```

I.3. Accès aux propriétés et méthodes d'un objet

L'accès aux propriétés et aux méthodes d'un objet peut se faire de différentes façons, mais la syntaxe la plus usuellement utilisée est avec la notation pointée.

Syntaxe :

```
nomObjet.propriété
nomObjet.nomMethode([paramètres]);
```

Exemples :

```
let uneVoiture = {marque:"Ford", couleur:"blanc", nbRoues: 4};
uneVoiture.couleur= 'vert';  // modifie la couleur de la voiture
let tab = [12,7,83];
alert (tab.length); // affiche 3
alert(tab.reverse()); // affiche 83, 7, 12
```

I.4. Objets personnalisés vs Objets prédéfinis en Javascript

Javascript permet, comme tout langage orienté objet, permet de définir ses propres objets, mais, il fournit, en plus une ensemble d'objets prédéfinis qu'il est directement possible d'exploiter.

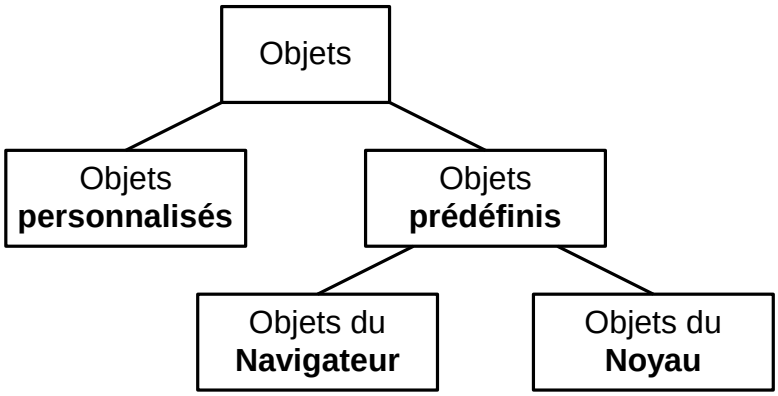


Figure 1: Classification des objets en Javascript

Les objets prédéfinis peuvent être classées en 2 catégories :

- Les objets du navigateur qui permettent de manipuler les éléments du navigateur web (window, document, navigator,…)
- Les objets du noyau qui sont des objets indépendants du navigateur et qui sont liés à la programmation en javascript (Array, String, Math, Date,…)

II. Vue d'ensemble des objets du navigateur

Javascript offre un modèle d'objet du navigateur (BOM : Browser Objet Model) qui permet d'interagir avec le navigateur.

II.1. Hiérarchie des objets du navigateur

Javascript offre différents objets qui peuvent être hiérarchisés comme suit (la liste des objets n'est pas exhaustive) :

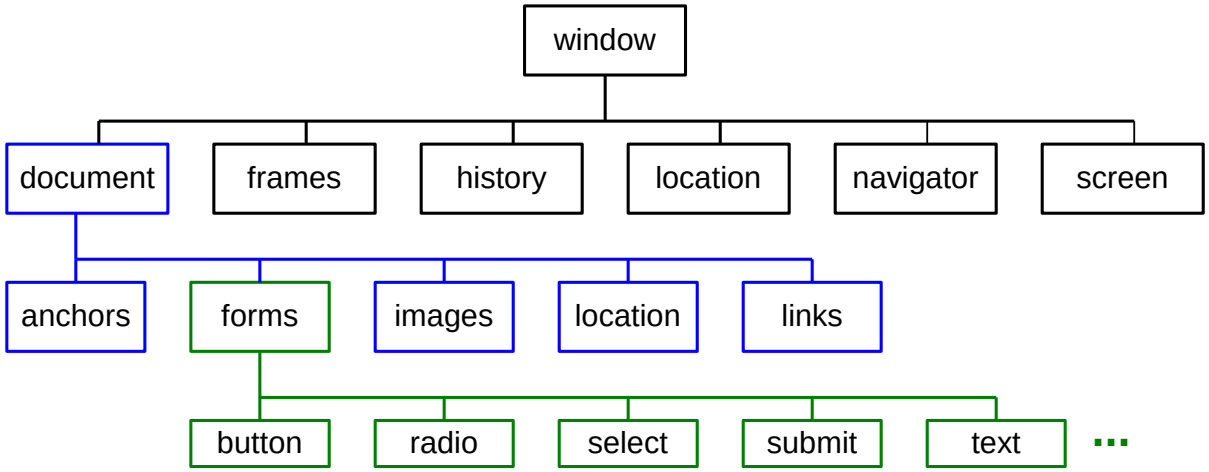


Figure 2: Hiérarchie des objets du navigateur

Les principaux objets du navigateur peuvent être synthétisés dans le tableau suivant :

Tableau 1: Description des principaux objets du navigateur

Objets du Navigateur	Description
window	Objet global qui représente la fenêtre du navigateur
document	Objet qui représente le document html affiché dans la page du navigateur
frames	Objet qui référencie la collection de frames de la page courante
history	Objet qui gère l'historique de la navigation du navigateur
location	Objet qui représente l'url de la page actuellement affichée dans la page du navigateur
navigator	Objet qui fournit des informations sur le navigateur et le système d'exploitation utilisé
screen	Objet qui fournit des informations sur la taille et la résolution de l'écran de l'utilisateur

II.2. L'objet window

L'objet ***window*** est créé automatiquement par le navigateur, il représente la fenêtre qui comporte le document HTML.

Quelques propriétés	document, length, location, name, opener, parent, self, status, top, window, …
Quelques méthodes	alert, clearTimeout, close, confirm, open, prompt, setTimeout, setInterval,…

II.2.1. La méthode open

La méthode ***open*** permet d'ouvrir une nouvelle fenêtre du navigateur.
Syntaxe :

```
window.open("url",[nom],["options"]);
```

- Le 2ème paramètre est optionnel, il comporte soit la cible (target : _blank, _self, _parent,…) soit le nom de la fenêtre.
- Les principales options possibles sont les suivantes:

Paramètre	Description	Exemple
width	Largeur de la fenêtre en pixel avec une valeur minimale de 10	width =250
height	Hauteur de la fenêtre en pixel avec une valeur minimale de 100	heigth =350
top	Position par rapport au haut de l'écran	top=150
left	Position par rapport à gauche de l'écran	left=50
popup	Indique si la fenêtre à ouvrir est une popup ou non. popup=true, popup =1, popup=yes et popup sont toutes des valeurs équivalentes.	popup=1

Exemples :

```
window.open("https://www.google.com/", "exemple", "width=450, height=300");
// Ouvre une nouvelle fenêtre d'une hauteur de 300px, d'une largeur de 450 px
window.open("EX11.html", "exo","top=40,left=20" );
// Ouvre une nouvelle fenêtre à 40 px du top de la fenêtre initiale et à 20 px
à gauche
let fen = window.open("EX11.html"); // La fenêtre est assignée à une variable
```

II.2.2. La méthode close

La méthode ***close*** permet de fermer une fenêtre
Syntaxe :

```
window.close();        // ferme la fenêtre actuelle
objetFenêtre.close();   // ferme la fenêtre référencée
```

Exemple :

```
myWindow =window.open("https://www.google.fr");
myWindow.close();  // ferme la fenêtre myWindow
```

II.2.3. La méthode setTimeout

La méthode ***setTimeout*** permet de déclencher un traitement après un temps spécifié.
Syntaxe :

```
window.setTimeout(fonction, délai);
```

Exemples :

```
function afficher(){
  alert("Ceci est un message")
}
window.setTimeout(afficher, 1000); // Appel de la fonction afficher
                                   // après 1 seconde
window.setTimeout(function(){alert('Test')}, 2000); // Affichage après 2 sec
```

II.3. L'objet document

L'objet ***document*** représente la page web chargée dans le navigateur. C'est l'élément racine de toute l'arborescence d'un document html ou xml.

Quelques propriétés	characterSet, cookie, domain, lastModified, location, referrer, title, URL,… *Tableau d'objets :* images, forms, links, applets, anchors, …
Quelques méthodes	write, writeln, getElementById, getElementsByName, getElementsByTagName, getSelection, open,…

Exemple :
Soit la page html suivante dont le code source est fourni :


```
<header>
    <img src="Images/Agence.jpg"
         alt="Logo agence" />
    <h1> Batouta Voyages </h1>
    <nav>

    <a href="Accueil.html"> Accueil </a>
    <a href="#"> Excursions </a>
    <a href="#"> Voyages </a>
    <a href="#" id="ct"> Contact </a>
    </nav>
</header>
```

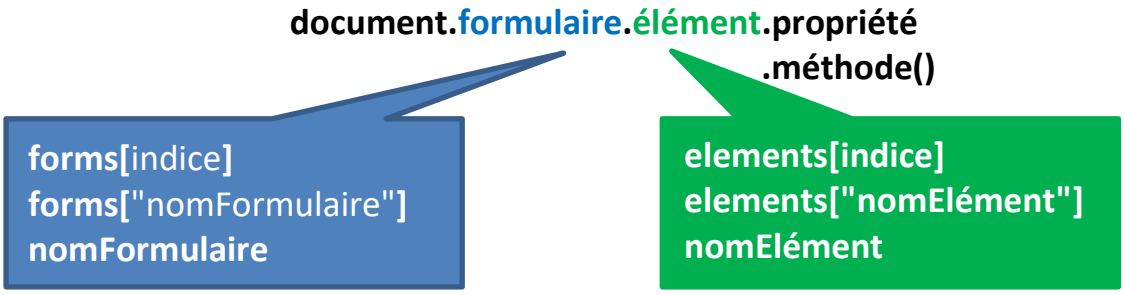
```
console.log(document.images.length); // 11
console.log(document.links.length);  // 4
console.log(document.images[0]);     //<img src="Images/Agence.jpg"
                                      alt="Logo agence" />
console.log(document.links[3].id);    // ct
console.log(document.getElementById("ct"));//<a href="#" id="ct"> Contact </a>
```

II.3.1. Accès à une propriété / méthode d'un élément d'un formulaire

L'objet *forms* est une propriété de l'objet *document* qui comporte la collection des formulaires du document.

Un formulaire comporte différents éléments : button, checkbox, radio, … L'accès à ces éléments se fait selon le principe suivant.

Principe :



Exemple :

Soit le formulaire suivant :


```
<form name="frm" action="trait.php" method="get">
  <label>Nom:</label>
  <input type="text" name ="txtNom">
  <label>Prénom:</label>
  <input type="text" name ="txtPrenom">
  <input type="submit" value =" Suivant" name="btnSuivant">
</form>
```

Pour afficher le nombre d'éléments du formulaire :

```
console.log(document.forms[0].length);     // 3
console.log(document.forms["frm"].length); // 3
console.log(document.frm.length);          // 3
```

⇨ Accès de 3 façons différentes au formulaire de la page

```
console.log(document.frm.elements[0].value);           // NomFoulen
console.log(document.frm.elements["txtPrenom"].value); // Foulen
console.log(document.frm.btnSuivant.value);            // Suivant
```

⇨ Accès de 3 façons différentes à un élément du formulaire : ici txtNom puis txtPrenom puis btnSuivant

II.3.2. Accès à un bouton radio

L'accès à un bouton radio s'effectue selon le principe suivant :

Principe :

document.formulaire.**nomBoutonRadio[indice].**propriété

Exemple :

Soit le code html suivant

```
<form name="f">
     <input type="radio" name="rb" value="etudiant" checked> Etudiant
     <input type="radio" name="rb" value="prof" > Enseignant
</form>
```

```
console.log(document.f.rb[0].value);          // etudiant
console.log(document.f.rb[1].checked);        // true
console.log(document.f.elements[0].checked); // false
```

Remarque :
La propriété **checked** renvoie si le bouton radio est coché ou non

II.3.3. Accès à l'élément select

L'accès à un bouton radio s'effectue selon le principe suivant :

Principe :

document.formulaire.**nomListe.options[indice].**propriété

Exemple :
Soit le code html suivant :

```
<form name="f">
  <select name="selDep">
    <option value="TI">TI</option>
    <option value="SEG">SEG</option>
  </select>
</form>
```

```
console.log(document.f.selDep.options.length);// 2
console.log(document.f.selDep.options[0].value);// TI
console.log(document.f.elements[0].options[1].value);// SEG
```

Remarque :
La propriété **selectedIndex** renvoie l'indice de l'option sélectionnée, si aucune option n'est sélectionnée, la propriété renvoie -1.

II.4. L'objet Image

L'objet ***image*** permet de créer une image.

Quelques propriétés	alt, complete, filesize, height, id, src, width, …
Création d'une image	`nomObjetImage = new Image([largeur, hauteur]);`
Accès aux propriétés	`nomObjetImage.propriété` `document.images[index].propriété`

Exemple 1 :

```
let monImage = new Image(250, 100);
monImage.src = "iset.jpg";
monImage.alt= "logo iset";
monImage.id= "image";
```

Exemple 2 :
On dispose d'un dossier Images qui comporte 10 images nommées respectivement img0.jpg,…img9.jpg

On souhaite que le document html affiche lors du chargement de la page l'image img0 et affiche chaque seconde l'image d'après (img1, img2,..). Une fois les 10 images défilées, c'est l'image img0 qui est à nouveau affichée et ainsi de suite.

```
<body>
  <img id="img" src="Images/img1.jpg" alt="image"
  onload="setTimeout(animerImages,delai)">

  <script>
    let delai = 1000;
    let num = 0;
    let tabImages = new Array();
    for (let i = 0; i < 10; i++) {
      tabImages[i] = new Image();
      tabImages[i].src = "Images/img" + i + ".jpg";
    }

    function animerImages() {
      document.getElementById('img').src = tabImages[num].src;
      num++;
      if (num >= 10)
        num = 0;
    }
  </script>
</body>
```

II.5. L'objet « location »

L'objet ***location*** fournit des informations sur l'URL de la fenêtre actuelle

Quelques propriétés	host, href, protocol, …
Quelques méthodes	assign, reload, replace,…

Exemples :

On suppose qu'on accède à notre cours placé sur la plateforme moodle à l'URL suivante: https://iset.uvt.tn/course/view.php?id=6052

```
console.log(location.host);     //iset.uvt.tn
console.log(location.protocol); //https:
location.reload(); // Recharge la page courante
```

II.6. L'objet « history »

L'objet ***history*** permet de manipuler l'historique du navigateur

Quelques propriétés	length, state …
Quelques méthodes	back, forward, go, pushState, replaceState,…

Exemples :

```
alert(history.length);   // affiche le nombre d'éléments dans l'historique
                         //de la session
history.back() ;  // renvoie la page précédente dans l'historique
                  // de la navigation
```

II.7. L'objet « navigator »

L'objet ***navigator*** donne des informations sur l'application exécutant le script.

Quelques propriétés	cookieEnabled, language, mimeTypes, platform, plugins, …
Quelques méthodes	javaEnabled(), plugins.refresh(), registerProtocolHandler(),…

Exemples :

```
console.log(navigator.cookieEnabled); // affiche un booléen qui indique si
                                      // l'ajout de cookie est pris en compte
console.log(navigator.language);  // (fr-FR par exemple): indique la langue
                                  // préférée de l'utilisateur
```

III. Les événements

Un objet graphique (bouton, formulaire, case à cocher) peut répondre à différents événements (clic sur un bouton, survol d'une image, redimensionnement d'une fenêtre,…).
La réponse à un événement se traduit par l'exécution d'un traitement donné qualifié de gestionnaire d'événements.

III.1. Les principaux événements et les objets qui leurs sont associés

Les évènements commencent en javascript par le préfixe **on**, ils peuvent être liés à différents déclencheurs tels que :

- La souris : onmousedown, onmouseover, onmouseenter onmousemove,…
- Le clavier: onkeypress, onkeydown, onkeyup, ..
- Les clics: onclick, ondblclick
- …

Il est primordial de noter qu'un même événement ne peut pas être appliqué à tous les composants graphiques. A titre d'exemples :

- Une fenêtre peut être redimensionnée (onresize) alors qu'une liste déroulante ne peut pas l'être
- Un formulaire peut être soumis (onsubmit) alors qu'une image ne peut pas l'être

Le tableau qui suit synthétise les principaux événements, leur rôle ainsi que les éléments HTML auxquels ils peuvent s'appliquer.

Tableau 2: Synthèse des principaux événements, leur description et les éléments HTML qui leur sont associés

Evénement	Description	Principaux éléments HTML
onabort	Le chargement d'une ressource a été interrompu	img
onblur	Un élément perd le focus	**Toutes les balises html sauf** <base>, <bdo>, , <head>, <html>, <iframe>, <meta>, <param>, <script>, <style>, <title> et <img>
onchange	Un élément perd le focus et sa valeur a changé depuis l'acquisition du focus	input, select, textarea
onclick	Un clic s'applique sur un élément	**Toutes les balises html sauf** <base>, <bdo>, , <head>, <html>, <iframe>, <meta>, <param>, <script>, <style> et <title>
oncontextmenu	Le bouton droit de la souris est cliqué	
ondblclick	Un double clic s'applique sur un élément	
onerror	Une erreur de chargement de l'image ou de la page se produit	img, object, script, link
onfocus	Un élément reçoit le focus	**Toutes les balises html sauf** <base>, <bdo>, , <head>, <html>, <iframe>, <meta>, <param>, <script>, <style>, <title> et <img>
oninput	La valeur d'un élément change	textarea et input avec type= color, date, datetime, email, month, number, password, range, search, tel, text, time, url, week
onkeydown	La touche du clavier est enfoncée	**Toutes les balises html sauf** <base>, <bdo>, , <head>, <html>, <iframe>, <meta>, <param>, <script>, <style> et <title>
onkeypress	La touche est pressée et cette touche produit un caractère	
onkeyup	La touche du clavier est relâchée	
onload	La ressource est ses ressources dépendantes dont chargées	body, frame, iframe, img, link, script, style
onmousedown	Le bouton de la souris est enfoncé	**Toutes les balises html sauf** <base>, <bdo>, , <head>, <html>, <iframe>, <meta>, <param>, <script>, <style> et <title>
onmousemove	Le pointeur de la souris est déplacé et passe sur l'objet	
onmouseover	La souris survole l'événement	
onmouseout	La souris quitte l'élément	
onreset	Le formulaire est réinitialisé	form
onresize	Le document est redimensionné	body
onselect	L'élément a été sélectionné	textarea et input avec type = file, text ou password
onsubmit	Le formulaire est soumis	form

onunload	Le document ou une ressource associée est déchargée	body

III.2. Appel et définition des gestionnaires d'événements

Une fois les événements introduits, nous passons à l'implémentation des traitements qui seront déclenchés suite à un événement. Ces traitements sont qualifiés de « Gestionnaires d'événements »

Un gestionnaire d'événement peut être appelé soit

- Dans une balise HTML
- Dans le code javascript

III.2.1. Gestionnaire d'événement défini dans une balise

Il est possible de définir un gestionnaire d'événement comme suit

Syntaxe :

```
<Balise onEvenement = "code javascript ou appel fonction" />
```

Exemple 1:

On souhaite qu'un clic sur un bouton Afficher affiche dans une boîte de dialogue le message « Clic sur un bouton »

```
<form name="frm">
  <button name ="btn" onclick="afficher()">Afficher</button>
</form>
<script>
  function afficher(){
    alert("Clic sur un bouton");
  }
</script>
```

Si le code du gestionnaire d'événement se limite à une ou 2 instructions, il est possible d'intégrer directement le code comme suit :

```
<form name="frm">
  <button name ="btn" onclick="alert('Clic sur un bouton')"> Afficher
  </button>
</form>
```

Exemple 2:

On souhaite que la couleur du **background** d'un bouton vire au **rouge** lorsque la souris survole le bouton et vire au **bleu** en le quittant

onmouseover

onmouseleave

```
<form name="frm">
  <input type="button" name="btn" value="Afficher"
  onmouseover="document.frm.btn.style.background='red';"
  onmouseleave="document.frm.btn.style.background= 'blue'" />
</form>
```

Dans ce dernier cas, il est possible de noter que :

- l'événement est déclenché par *btn*
- Le traitement porte sur *btn*

⇨ Il est possible d'optimiser le code en utilisant la référence **this**

❖ La référence this

this renvoie une référence à l'élément sur lequel le gestionnaire d'événement a été lancé.

Exemple :

On souhaite reprendre le même exemple que précédemment en utilisant la référence this.

```
<form name="frm">
  <input type="button" name="btn" value="Afficher"
  onmouseover="this.style.background='red';"
  onmouseleave="this.style.background= 'blue'" />
</form>
```

III.2.2. Gestionnaire d'événement défini dans le code javascript

Le gestionnaire d'événement peut être défini complètement dans le javascript de 2 façons :

- En rattachant l'événement à l'objet qui déclenche l'événement
- En utilisant un écouteur d'événements

Ces 2 façons vont être présentées dans la suite.

❖ 1ère méthode : Attachement de l'événement à un objet

L'attachement se fait selon la syntaxe suivante

Syntaxe :

```
cible.onEvenement = fonction
```

La **cible** représente l'objet qui prend en charge l'événement : çà peut être l'élément html, le document ou la fenêtre.

Exemple :

On souhaite qu'un clic sur un bouton Afficher affiche dans une boîte de dialogue le message « Clic sur un bouton »

```
<form name="frm">
  <button name ="btn">Afficher</button>
</form>
<script>
  document.frm.btn.onclick = function(){ alert("Clic sur un bouton");}
</script>
```

❖ 2ème méthode : Utilisation de l'écouteur eventListener

Cette méthode est particulièrement intéressante lorsque les événements sont rattachés dynamiquement. La syntaxe est comme suit :

Syntaxe :

```
cible.addEventListener('typeEvénement', listener [,options])
```

Avec :

- *cible* : Objet qui prend en charge l'événement (Elément Html, document, window)
- *typeEvénement* : click, mouseover, keypress,… (sans le préfixe **on**)
- *listener* : fonction java (ce qui sera traité dans ce cours) ou objet qui implémente l'interface EventListener
- *options* : ce sont des caractéristiques supplémentaires de l'écouteur qui ne seront pas traités dans ce cours

Exemple :
On souhaite qu'un clic sur un bouton Afficher affiche dans une boîte de dialogue le message « Clic »

```
  <form name="frm">
    <button id ="bt">Afficher</button>
  </form>
  <script>
    document.getElementById("bt").addEventListener('click',
function(){alert('clic')})
  </script>
```

III.3. L'objet event

L'objet **event** fournit des informations sur l'événement qui s'est produit. Il fournit plusieurs attributs et méthodes dont il est possible de mentionner :

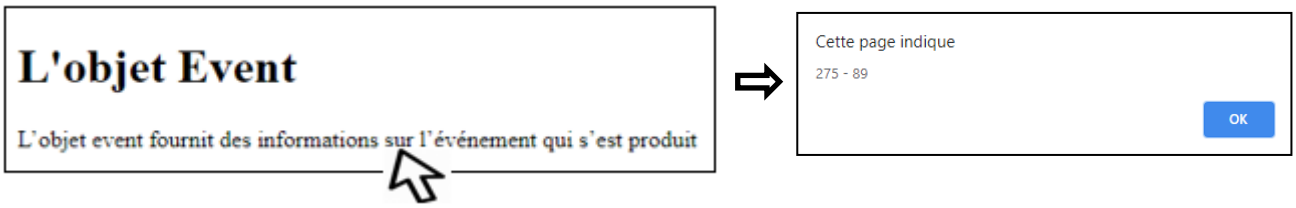
Attribut/Méthode	Description
clientX	Fournit la position horizontale du pointeur de la souris par rapport à la fenêtre
clientY	Fournit la position verticale du pointeur de la souris par rapport à la fenêtre
defaultPrevented	Renvoie un booléen qui indique si un événement a été annulé ou non
key	Fournit la touche pressée lorsqu'une touche du clavier est activée
target	Fournit l'élément (objet) enfant le plus profond survolé lors du déclenchement de l'événement.
preventDefault()	Annule un événement si celui-ci est annulable empêchant ainsi que l'action par défaut se produise. *Exemples :* Annuler la soumission d'un formulaire, empêcher lors d'un clic sur un lien la redirection vers une URL, annuler la pression sur une touche,…

Exemple 1:
On souhaite qu'en pressant un caractère dans une zone de texte, un message d'alerte affiche le caractère tapé, si celui-ci correspond au symbole « ! », l'évènement doit être annulé.

```
<form name="frm">
 <input type="text" name="txt" onkeypress="afficher(event)" />
</form>
<script>
    function afficher(event) {
        if (event.key != "!")
            alert('le caractère introduit est ' + event.key)
        else
            event.preventDefault();
    }
</script>
```

Exemple 2:

On souhaite qu'en cliquant avec la souris dans un paragraphe, les coordonnés de la position pointée s'affiche dans une boîte d'alerte.




```
<p onmousedown="alert (event.clientX + ' - '+ event.clientY);">
    L'objet event fournit des informations sur l'événement qui s'est produit
</p>
```