

# JAVASCRIPT

Partie 2 - DOM

# DOCUMENT OBJECT MODEL

DOM = API communiquant entre html et javascript  
Permettant de réaliser des interactions  
Cela permet à des scripts d'examiner et de modifier  
le contenu du navigateur web

- Déplacer
- Ajouter
- Modifier
- Supprimer

# ELEMENT DE BASE «LA FENETRE»

Objet «window» est l'élément principal de votre page web.

C'est l'objet initial qui intègre tout un tas de méthodes permettant à javascript de fonctionner.

C'est le «coeur» de javascript.

Par exemple les fonctions que nous avons utilisées :

`alert()`, `console.log()`, ne sont que des méthodes de l'objet `window`.

`window.console.log() = console.log()`

`window` étant implicite, on ne le spécifie pas, cela est rajouté automatiquement par javascript lors de sa «compilation à la volée»

# PORTEE DE VARIABLE (CORRIGÉ AVEC ES6)

Une variable définie sans le mot clé «var» devient une variable global à la page  
(fenêtre)

Ce qui veux dire qu'elle devient un «attribut» de l'objet window.

N'oubliez jamais la déclaration de vos variables «var, let ,const» et maintenant  
«window» si vous souhaitez une variable globale à la page

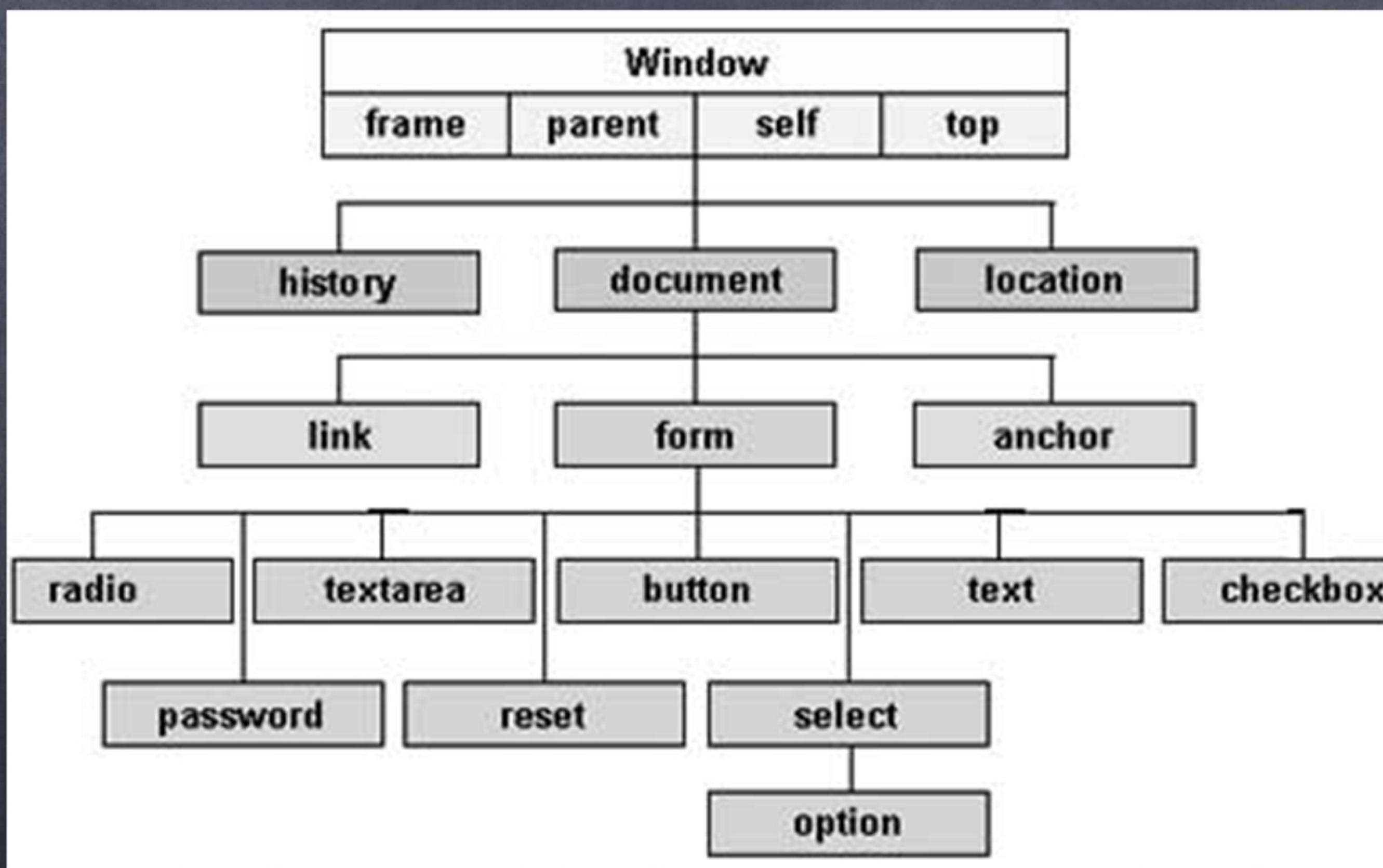
Entrez «window» dans la console de chrome

# DOCUMENT

Sous ensemble le plus utilisée en javascript de l'objet window

Document représente la page Web (le code html)

Entrez «document» dans la console de chrome  
(Voir arbre du DOM)



# CRÉATION PAGE DE TEST

```
<div id="container">
    <p class="paragraphe1">Bonjour à tous</p>
    <p class="paragraphe2">Ceci est ma page de <a href="#" title="contact">contact</a> </p>
    <p class="paragraphe3">J'aime</p>
    <ul id="liste">
        <li>Netflix</li>
        <li>L'argent</li>
        <li>Les voitures</li>
        <li>La programmation</li>
    </ul>
</div>
```

Ce sera notre base de travail pour débuter la manipulation le dom

Notion de «noeud» / «node» : Chaque éléments est un noeud (une liaison)

Notion «parent» : div#container est le parent de p.paragraphe1

Notion «enfant» : le contenu est l'enfant de li, li est l'enfant de ul#liste, ul#liste est l'enfant de div#container, div#container est l'enfant de body

A noter : p.paragraphe2 , contient combien d'enfants?

# NAVIGATION DOM

getElementById()

Permet d'accéder à un élément par son ID

```
console.log(document.getElementById('container'));
```

getElementsByName()

Avec un S à element signifie un «tableau de»

```
var li = document.getElementsByTagName('li');
console.log(li);

for (var i = 0; i < li.length ; i++) {
    console.log('Elm ' + (i + 1) + ' : ' + li[i]);
    console.log('Elm ' + (i + 1) + ' : ' + li[i].innerHTML);
}
```

# NAVIGATION DOM

getElementsByName()

Permet de récupérer les éléments sous forme de tableau disposant d'un attribut «name» (formulaire champs)

getElementsByClassName()

Permet de récupérer les éléments sous forme de tableau en fonction des class  
Possible d'indiquer plusieurs class en utilisant un espace

```
var p = document.getElementsByClassName('paragraphe1');
console.log(p);

for (var i = 0; i < p.length ; i++) {
    console.log('Elm ' + (i + 1) + ' : ' + p[i].innerHTML);
}
```

# QUERYSELECTOR (NON COMPATIBLE IE8 & <) + FIREFOX 2-3

querySelector et querySelectorAll

Plus moderne ces fonctionnalités permettent de travailler avec le dom en ne fournissant qu'une chaîne de caractère de type CSS  
(similitude dans le fonctionnement avec Jquery)

```
var query = document.querySelector('div#container ul#liste>li'),
    queryAll = document.querySelectorAll('div#container ul#liste>li');

console.log(query.innerHTML);

console.log(queryAll.length);
console.log(queryAll[0].innerHTML + ' - ' + queryAll[1].innerHTML);
```

# EDITION JS

getAttribute() : Récupère un attribut

setAttribute() : Modifie un attribut

Ces méthodes fonctionnent également avec getElementById(),...

```
var query = document.querySelector('div#container p.paragraphe2 > a');

console.log(query.getAttribute('href'));
query.setAttribute('href', 'http://www.google.fr');
console.log(query.getAttribute('href'));
console.log(document.querySelector('div#container p.paragraphe2').getAttribute('class'));
```

Attention :

Si vous utilisez setAttribute('class','...'), cela re-initialise les classes sur la balise.

# EDITION JS

innerHTML : Gérer le contenu HTML d'un élément

```
//ON CIBLE .PARAGRAPHE2
var query = document.querySelector('div#container p.paragraphe2');
//ON AFFICHE SON CONTENU HTML
console.log(query.innerHTML);
//ON RAJOUTE DU CONTENU A LA SUITE
query.innerHTML += " <u>en plus</u> ";
console.log(query.innerHTML);
```

Vous rencontrerez parfois

innerText : Spécifique à IE pour récupérer du texte brut

textContent : Tout navigateurs sauf < ie9

Le plus simple étant d'utiliser innerHTML

innerHTML est standardisé

Toujours utiliser du standardisé cela évite de gérer des conflits futurs

# NAVIGUER DANS LA PAGE

parentNode, récupère élément parent

```
let elm = document.getElementsByClassName("paragraphe2")[0];
console.log(elm);
//ELEMENT PARENT
console.log(elm.parentNode);
```

firstChild,LastChild : récupère le premier et dernier élément

```
let elm2 = document.getElementById("liste");
console.log(elm2);
//PREMIER ELEMENT / DERNIER ELEMENT
console.log(elm2.firstChild.innerHTML,elm2.lastChild.innerHTML);
```

Etrange? La valeur renvoie undefined pourquoi? solution?

# NAVIGUER DANS LA PAGE

Les espaces / retours lignes sont considérés comme des éléments

solution1 : Bricolage : on supprime les espaces

```
<ul id="liste2"><li>Netflix</li><li>L'argent</li><li>Les voitures</li><li>La programmation</li></ul>
</div>
<script>

let elm3 = document.getElementById("liste2");
console.log(elm3);
//PREMIER ELEMENT / DERNIER ELEMENT
console.log(elm3.firstChild.innerHTML,elm3.lastChild.innerHTML);
```

solution2 : on utilise firstElementChild / lastElementChild (non compatible <i>ie9</i>)

IL faudra utiliser un polyfill

```
console.log(elm2);
//PREMIER ELEMENT / DERNIER ELEMENT
console.log(elm2.firstElementChild.innerHTML,elm2.lastElementChild.innerHTML);
```

# NAVIGUER DANS LA PAGE

nodeValue/data: renvoie le contenu d'un noeud textuel

Childnodes liste les noeud d'un elements sous forme de tableau

nodeType permet de cibler le type de noeud

```
let elm = document.getElementsByClassName("paragraphe2")[0];
console.log(elm);
//Node value récupère un contenu textuel, s'applique sur un noeud textuel
//elm.lastChild = Lien . firstchild.data Le contenu textuel
console.log(elm.firstChild.nodeValue,elm.lastChild.firstChild.data);

//childNodes renvoie un tableau listant les noeuds
elm.childNodes.forEach(x=>(x.nodeType==1)?console.log(x.firstChild.data):console.log(x.data));
//Equivalent
for(var i=0; i<elm.childNodes.length; i++){
    if(elm.childNodes[i].nodeType === 1){
        console.log(elm.childNodes[i].firstChild.data);
    }else{
        console.log(elm.childNodes[i].data);
    }
}
```

# NAVIGUER DANS LA PAGE

NodeType, nodeName, nodeValue  
association

Node type	nodeName returns	nodeValue returns
1 Element	element name	null
2 Attr	attribute name	attribute value
3 Text	#text	content of node
4 CDATASection	#cdata-section	content of node
5 EntityReference	entity reference name	null
6 Entity	entity name	null
7 ProcessingInstruction	target	content of node
8 Comment	#comment	comment text
9 Document	#document	null
10 DocumentType	doctype name	null
11 DocumentFragment	#document fragment	null
12 Notation	notation name	null

# NAVIGUER DANS LA PAGE

nextSibling : noeud suivant

previousSibling : noeud précédent

Attention à l'espace et retour ligne

Utilisez : nextElementSibling, previousElementSibling

```
let elm = document.querySelector('p.paragraphe2');
console.log(elm,elm.nextSibling,elm.previousSibling);
console.log(elm,elm.nextElementSibling,elm.previousElementSibling);
```

# CREATION ELEMENT

createElement, createTextNode, appendChild...

```
<script>
    let paragraphe = document.createElement('p');
    let paragrapheTexte = document.createTextNode("Mon premier paragraphe");

    //DEFINIR ATTRIBUT
    paragraphe.setAttribute('id','entete');
    //AUTRE FACON QUI N'ECRASE PAS
    paragraphe.id = 'Autre';
    paragraphe.className = 'Exemple de class';
    paragraphe.style.color = '#FF0000';
    paragraphe.style.border = '3px solid black';

    document.querySelector('body').appendChild(paragraphe);
    paragraphe.appendChild(paragrapheTexte);

    //CSS
    paragraphe.style.cssText = "font-size:18px;font-weight:bold;text-transform:uppercase";
    paragraphe.setAttribute('style',"font-size:18px;font-weight:bold;text-transform:uppercase");
</script>
```

# CREATION ELEMENT

On ne peut pas faire ça car il y a passage par référence, newDiv1 est newDiv2

Passage par valeur = La variable est copié à un instant T

Passage par référence : Son emplacement mémoire est copié, quand A est modifié B est impacté

```
var newDiv1 = document.createElement('div');
var texte = document.createTextNode('bonjour');
var newDiv2 = newDiv1; // On tente de copier le <div>

console.log(newDiv1,newDiv2);

document.querySelector('body').appendChild(newDiv1);
newDiv1.appendChild(texte);
console.log(newDiv1,newDiv2);
```

CloneNode true copie les enfants / False juste élément courant

Cela permet de créer un nouvelle élément identique au premier avec un emplacement mémoire différent.

```
var autreDiv = newDiv1.cloneNode(true);
document.querySelector('body').appendChild(autreDiv);
```

# MANIPULER ELEMENT

```
<div>
    <p id="test">texte avec <a href="#">un lien</a></p>
</div>
```

Remplacer un contenu (utile)

```
var lien = document.querySelector('a');
var nouveauTexte = document.createTextNode('nouveau texte de lien');
lien.replaceChild(nouveauTexte, lien.firstChild);
```

Supprimer et déplacer

```
let ancien = lien.parentNode.removeChild(lien);
//document.querySelector('body').appendChild(ancien);
document.body.appendChild(ancien);
```

insertBefore (1.élément à ajouter,2.élément avant lequel votre contenu sera inséré)

insertAfter n'existe pas on n'utilise appendChild

```
var paragraphe = document.querySelector('p#test');
var info = document.createElement('span'),
spanText = document.createTextNode('super span!');

info.appendChild(spanText);
//S'applique sur l'élément parent
paragraphe.insertBefore(info, paragraphe.lastChild);
```

# EVENEMENTS

Les événements permettent de lancer une fonction (ou plusieurs) en fonction d'une action sur un élément contenu dans la page

click	Cliquer (appuyer puis relâcher) sur l'élément
dblclick	Double-cliquer sur l'élément
mouseover	Faire entrer le curseur sur l'élément
mouseout	Faire sortir le curseur de l'élément
mousedown	Appuyer (sans relâcher) sur le bouton gauche de la souris sur l'élément
mouseup	Relâcher le bouton gauche de la souris sur l'élément
mousemove	Faire déplacer le curseur sur l'élément
keydown	Appuyer (sans relâcher) sur une touche de clavier sur l'élément
keyup	Relâcher une touche de clavier sur l'élément
keypress	Frapper (appuyer puis relâcher) une touche de clavier sur l'élément
focus	« Cibler » l'élément
blur	Annuler le « ciblage » de l'élément
change	Changer la valeur d'un élément spécifique aux formulaires ( <code>input</code> , <code>checkbox</code> , etc.)
input	Taper un caractère dans un champ de texte ( <a href="#">son support n'est pas complet sur tous les navigateurs</a> )
select	Sélectionner le contenu d'un champ de texte ( <code>input</code> , <code>textarea</code> , etc.)

## Event spécial formulaire

submit	Envoyer le formulaire
reset	Réinitialiser le formulaire

# EVENEMENTS

```
<button id="buttonEvent">C'est un bouton, cliquez dessus!</button>

<script>
    document.querySelector('#buttonEvent').addEventListener('click',function essai(){
        alert(this.nodeName.toLowerCase() + ' : ' + this.innerHTML);
        this.style.backgroundColor = "#FF0000";
        this.style.color = "white";
        this.style.fontSize = "20px";
        this.removeEventListener('click',essai);
    })
</script>
```

addListener : Ajout un évènement à écouter

1. L'évènement
- 2.L'action

This : Variable défini ciblant l'élément courant ici <button>

On peut s'en servir comme un élément du dom

On peut modifier facilement l'élément courant.

removeEventListener : supprime un évènement

# OBJET EVENT / EVENEMENT

```
let x = 0;
document.querySelector('#buttonEvent').addEventListener('click',function(e){
    //ENSEMBLE DES METHODES
    console.log(e);
    //TYPE EVENEMENT
    console.log(e.type);
    //CHANGEMENT CONTENU
    e.target.innerHTML = "Changement" + x++;
    //POSITION DANS LA FENETRE
    console.log("x : " + e.clientX + "| y : " + e.clientY);
})
```

L'objet e : type event  
Permet de récupérer des éléments de l'évènement en cours.  
type, cible, position, id, contenu, ...

# OBJET EVENT / EVENEMENT

```
let x = 0;
document.querySelector('#buttonEvent').addEventListener('click',function(e){
    //ENSEMBLE DES METHODES
    console.log(e);
    //TYPE EVENEMENT
    console.log(e.type);
    //CHANGEMENT CONTENU
    e.target.innerHTML = "Changement" + x++;
    //POSITION DANS LA FENETRE
    console.log("x : " + e.clientX + "| y : " + e.clientY);
})
```

L'objet e : type event

Permet de récupérer des éléments de l'évènement en cours.  
type, cible, position, id, contenu, ...

e.preventDefault(); permet d'annuler l'action par défaut de certains événements (click, submit, reset, etc).

# OBJET EVENT / EVENEMENT

```
<p>
  <input id="field" type="text" />
</p>

<table>
  <tr>
    <td>keydown</td>
    <td id="down"></td>
  </tr>
  <tr>
    <td>keypress</td>
    <td id="press"></td>
  </tr>
  <tr>
    <td>keyup</td>
    <td id="up"></td>
  </tr>
</table>
```

```
var field = document.getElementById('field'),
  down = document.getElementById('down'),
  press = document.getElementById('press'),
  up = document.getElementById('up');

//APPUIE SUR LA TOUCHÉ | MAJ DEFAUT
document.addEventListener('keydown', function(e) {
  down.innerHTML = e.keyCode + " " + String.fromCharCode(e.keyCode);
});

//COMBINAISON DE TOUCHÉ | Que ce qui écrit du caractère
document.addEventListener('keypress', function(e) {
  press.innerHTML = e.keyCode + " " + String.fromCharCode(e.keyCode);
});

//RELACHE LA TOUCHÉ | MAJ DEFAUT
document.addEventListener('keyup', function(e) {
  up.innerHTML = e.keyCode + " " + String.fromCharCode(e.keyCode);
});
```

e.keyCode retourne le code ASCII, string.fromCharCode renvoie le caractère.  
formCharCode est une méthode native de l'objet javascript String.  
Pourquoi on utilise document. et non field. ??  
Créez une div 10px\*10px et faites la bouger en utilisant les flèches du clavier

# DELEGATION EVENT

```
<ul id="maListe">
<li class="element" data-texte="foo">Item 1</li>
<li class="element" data-texte="bar">Item 2</li>
<li class="element" data-texte="baz">Item 3</li>
<li class="element" data-texte="toto">Item 4</li>
<li class="element" data-texte="titi">Item 5</li>
<li class="element" data-texte="tata">Item 6</li>
<li>Item 7</li>
<li>Item 8</li>
<li>Item 9</li>
<li class="element" data-texte="42">Item 10</li>
</ul>

<div id="parent"></div>
<button id="ajout">Ajouter un élément</button>
```

```
document.getElementById('maListe').addEventListener('click', function(e){
  var initElem = e.target;
  if(initElem.className != 'element'){ // Si l'élément n'est pas un de ceux à traiter
    return;
  }
  alert(initElem.dataset.texte);
});

document.getElementById('ajout').addEventListener('click', function(){
  document.getElementById('parent').innerHTML = '<span id="enfant">Elément enfant ajouté</span>';
} );
document.getElementById('parent').addEventListener('click', function(e){
  var initElem = e.target;
  if(initElem.id == 'enfant'){
    alert('Vous avez cliqué !');
  }
});
```

Lorsque l'on souhaite associer des événements similaires à différents éléments. Par exemple, imaginons que l'on souhaite afficher un texte initialement masqué en cliquant sur des items d'une liste ordonnée. Plutôt que de définir autant d'événement que d'éléments dans la liste, on pourra n'en utiliser qu'un seul placé sur la balise `<ul>`

Lorsque l'on veut prévoir des gestionnaires d'événements sur des éléments n'étant pas encore présents dans la page. Ajouté dynamiquement

# TRIGGER

```
<div id="myDiv1" onclick="console.log('Clic sur myDiv1 détecté')">Mon premier DIV</div>
<div id="myDiv2" onclick="console.log('Clic sur myDiv2 détecté')">Mon second DIV</div>
<div id="myDiv3" onclick="console.log('Clic sur myDiv3 détecté')">Mon dernier DIV</div>

<script>
var evt=new Event("click");
document.getElementById("myDiv1").dispatchEvent(evt);

document.getElementById("myDiv2").click();
</script>

<script>
document.addEventListener('hey', function(e)
{
    console.log(e.type + ' ' + e.detail.user); // hey stackoverflow
});
document.dispatchEvent(new CustomEvent('hey', {'detail': {'user': 'm2i'}}));
</script>
```

Les triggers permettent de simuler le démarrage d'un évènement par exemple un click ou de créer des évènements personnalisés à certains moments du programme

# FORMULAIRE

```
document.getElementById('form_connexion').addEventListener('submit', function(e){  
    var regnombre = /^[a-zA-Z]{2,}$/;  
    //ANNULE LA VALIDATION  
    e.preventDefault();  
    //CONTIENT LE FORMULAIRE  
    console.log(this);  
    //ACCEDER A L'ELEMENT  
    console.log(this.email.value);  
    //AUTRE MOYEN  
    console.log(document.forms["form_connexion"].elements["email"].value);  
    //TESTER SI VALIDE MAIL  
    if(regnombre.test(document.forms["form_connexion"].elements["email"].value)) { console.log("plus de 2");}  
    //SI CASE COCHE  
    if(this.elements['remember'].checked){console.log("coché");}  
    //VERIFIER SI BOUTON RADIO  
    document.forms["form_connexion"].elements["optradio"].forEach(function(elm){  
        if(elm.checked){console.log('ok')}  
    });  
    //SELECT INDEX / VALEUR  
    console.log(document.forms["form_connexion"].elements["sel1"].selectedIndex,  
    document.forms["form_connexion"].elements["sel1"].value );  
    //VALIDER FORMULAIRE  
    this.submit();
```

# FORMULAIRE

## EXERCICE

Créez une formulaire incluant du responsive design

Champs :

- email
- password
- case à cocher «j'accepte les cookies»
- 3 boutons radios avec le même «name» [Oui, non, aucun avis]
- Un menu déroulant : [Premiere option «choisir une valeur», 1,2,3,4]

But :

Faire en sorte que ce formulaire soit correctement rempli avant validation

- Email non vide et valide (regex)
- Password entre 6 et 8 caractères et non vide
- Case à cocher, coché
- Un des boutons radios sélectionné
- Une valeur du menu déroulant autre que la première (vide)

Une fois que tout est ok, soumettre le formulaire sinon bloquer le formulaire et afficher un message d'erreur correspondant à l'erreur en question (div fond de couleur rouge)

Exemple : Email non valide