* **Visibility :** hidden/visible… Permet de gérer la visibilité d’un élément sur la page CSS
* **Min-height** : 200vh… Veut dire que la page web va faire deux fois la taille de l’écran de l’utilisateur
* **Opacity**… Permet de gérer l’opacité de l’élément sélectionné
* **Position :** absolute ; Permet de placer un élément exactement ou on le veut sur la page. Elle est souvent utilisée avec les propriétés CSS : left, right, Bottom, top.
* **Un élément en absolute pour lui dire où il va se placer il suffit de lui injecter du top et de la left**
* **Z-index :** Permet de faire passer un élément devant ou derrière les autres. Une valeur positive fait passer l’élément devant et une valeur négative fait passer l’élément derrière.
* **Overflow :** hidden Permet de dire au body que rien de doit pouvoir casser la page(traverser)
* **Transform**… Permet d’appliquer une transformation à un élément HTML. Elle permet de modifier la position, la rotation, l’échelle et l’inclinaison d’un élément :
* Translate(x,y) : déplace l’élément sur l’axe horizontal(x) et vertical(y). Par exemple, translate (10px, 20px) déplace l’élément de 10px vers la droite et de 20px vers le bas
* Rotate(angle) : Fait tourner l’élément autour de son point central
* Scale (x, y) : modifie la taille de l’élément sur l’axe horizontal(x) et vertical(y). Une valeur escale (2,0.5) doublerait la largeur de l’élément tout en réduisant sa hauteur de moitié
* Skew(x, y) : incline l’élément selon les axes horizontal(x) et vertical (y). Par exemple, skew(20deg,10deg) inclinerait l’élément de 20 degrés horizontalement et de 10 degrés verticalement. Exemple : transform : translate(20px,30px) rotate(45deg) scale (1.5,1.5)
* **Transition** ***: background-color 1s ease ;*** Veut dire que la transition va concerner le background et va durer 1 seconde
* *Ease*
* *Linear* (Vitesse constante)
* *Ease-in* (Démarre lentement puis accélère)
* *Ease-out* (Démarre rapidement puis ralenti)
* *Ease-in-out* (Démarre lentement puis accélère et après ralentit)

1. ***LES SELECTEURS*** ils permettent de pointer sur des éléments du DOM : ça marche comme les sélecteur CSS

* QUERYSELECTOR

On peut l’utiliser pour sélectionner une classe exactement comme en CSS

**Pour modifier le style de l’élément sélectionner :**

**Document.querySelector(« nom de l’élément »).style.background= «nom de la couleur choisie »… On peut même regler le border radius**

1. ***LES EVENEMENTS***

* **addEventListener () : Cette méthode, lorsqu’elle est appelée elle s’apprête directement à gérer un évènement sur l’objet HTML où il est appelé. Elle prend deux paramètres :**

1. **L’évènement**
2. **La fonction (Une fonction qui a déjà été codée au paravent) on peut écrire FUNCTION ou encore () => {}**
3. **CLICK EVENT**

**QuestionContainer.classList.add(« nom de la classe ») : Permet d’ajouter une classe à un élément HTML (On peut l’utiliser si la classe possède des propriétés CSS qu’on aimerait aussi affecter à l’élément sélectionner)**

**Pour enlever une classe on utilise REMOVE à la place de ADD**

**On peut aussi utiliser TOGGLE : Qui veut dire que si la classe y est on la supprime et si la classe n’y est pas on l’ajoute… Ça permet de créer une sorte d’animation à chaque fois qu’on clique sur l’élément**

Priorités CSS : id>.class>baliseHTML

Donc si on injecte une classe à un style qui a été donné par un ID ça ne marchera pas

1. **LE MOUSEMOVE**

L’objet **WINDOW** est l’élément parent de l’objet DOCUMENT il concerne la fenêtre de l’utilisateur.

Le premier paramètre qu’on passe à la fonction lors d’un MouseMove est « e » (C’est un objet) il renferme toute une série de data ; tout ce qui s’est passé lors du mousemove… Même l’heure :

Window.addEventListener(« mousemove », (**e**)=> {Ce qui va se passer pendant le mousemove})

NB : Le premier paramètre qu’on met dans un eventListener récupère toutes les données de l’évènement

On a aussi :

* LE MOUSEDOWN : Ce qui va se passer lorsqu’on va faire un clic étant en bas de la page
* LE MOUSEUP : Ce qui va se passer lorsqu’on va faire un clic étant au-dessus de la page
* LE MOUSEENTER : Ce qui va se passer lorsque le curseur de la souris va survoler l’élément
* LE MOUSEOUT : Ce qui va se passer lorsque le curseur de la souris va quitter l’élément

NB : Un élément HTML ne peut pas avoir deux fois la même propriété CSS… Si on lui affecte une nouvelle propriété il va écraser la première

1. LE KEYPRESS

Avec cet évènement, on peut récupérer tout ce qui est fait sur le clavier et utiliser ça pour créer toute sorte d’évènements

1. LES FICHIERS AUDIO

En JS, il existe un objet de type audio qui permet de gérer des fichiers audios

Const audio=new Audio () ;

* Pour inclure un fichier audio, on utilise l’attribut SCR de l’objet Audio ()

Audio.src=’’nom du fichier à inclure’’

* Pour jouer un fichier audio, on utilise la méthode play () de l’objet Audio ()

Audio.play() ;

* On peut paramétrer la fonction ring au cas ou on a plusieurs fichiers audios

Audio.src=e.key+ ‘’.mp3’’

1. LE SCROLL

Le scroll en JS permet de gérer ou créer des animations lorsque l’utilisateur scroll sur la page (Il fait défiler la page du bas vers le haut et inversement) …

* Exemple d’une barre de navigation
* On peut cacher la barre au départ (Cacher en haut) avec un top : -50px et utiliser le scroll pour la faire apparaitre à un moment donner.
* Pour ce faire, nous allons utiliser les pixels… Nous allons donc dire à la machine : Dès que l’utilisateur scroll tel nombre de pixels, fait apparaitre la barre sinon fait la disparaitre.
* Pour faire ça, on peut utiliser le paramètre ‘’e’’ et accéder à l’attribut : e. scrollY qui contient le nombre de pixels que l’utilisateur a scroller.
* Ou alors on peut utiliser window.scrollY qui contient également le nombre de pixels que l’utilisateur a scroller.
* On met donc un IF dans la fonction de l’évènement qui va vérifier si l’utilisateur a parcouru un certain nombre de pixels et faire apparaitre la barre de navigation

1. LES FORMULAIRES

* L’évènement qu’on utilise sur les inputs est : SUBMIT

Exemple: inputName.addEvenListener(‘’submit’’,function(e){console.log(e)});

* Pour récupérer ce qui est stocké dans l’input, on utilise :

inputName.addEvenListener(‘’submit’’,function(e){

Let pseudo=e.target.value;

console.log(e.target.value)});

NB : Ceci marche pour tous les éléments INPUT d’un formulaire

NB : On peut accéder à une balise de cette manière : document.querySelector(‘’form >div’’) ;

NB : On peut afficher une variable dans une chaine de caractère de cette manière :

: document.querySelector(‘’form >div’’).innerHTML=`Le nom est : ${nom}` ;

NB : textContent ne peut pas interpréter les balises… Il faut souvent utiliser innerHTML

* CAS DES CHECKBOX
* On peut utiliser l’ID de l’input pour savoir s’il a été coché ou pas

Exemple: if (nomInput.checked) alert (‘Il a coché’) … Ceci renvoie TRUE si le Checkbox a été coché et FALSE sinon… ON A PAS BESOIN DE CREER UNE VARIABLE POUR CA, LE JS CONNAIT DEJA LES CHECKBOX

Pour récupérer les données d’un formulaire en JS, il faut toujours penser à utiliser un preventDefault () pour se prémunir du comportement par défaut du navigateur (qui est de changer de page à chaques fois qu’on soumet le formulaire)

* Pour récupérer l’évènement d’un checkbox (Savoir si il a été coché ou pas), il faut le mettre dans un gestionnaire d’évènement qui va appeler la fonction de vérification et vérifier si le bouton a été coché

1. LOAD EVENT
2. getElementsByTagName

Permet de sélectionner tous les éléments qui ont la même classe dans une page HTML… C’est un peu comme querySelectorAll

NB : A part addEventListener, on peut aussi utiliser ONCLICK de cette manière :

Document.body. onclick=function(){Alert(‘’Il a fait un click’’) ;

} Mais ce n’est pas recommandé car avec le ON on ne peut pas donner plusieurs évènements à la même balise HTML… Mais avec addEventListener c’est possible de donner plusieurs évènements à la même balise HTML

1. BUBLING & USECAPTURE

Il faut savoir que de base, le **addEventListener** est paramétré en mode BUBLING ce qui s’signifie qu’il s’exécute à la fin. Si on veut qu’il s’exécute au début, on le paramètre en USECAPTURE

* Pour le faire, on ajoute un 3e paramètre à la fonction addEventListener(true) pour le mettre en USECAPTURE
* FALSE c’est pour le mettre en BUBLING

1. LE BOM (Browser Object Model)

Le BOM est au-dessus de tout… En fait il représente le WINDOW

Il contient l’objet : WINDOW, on a par exemple :

* **Window.innerHeight** : Qui permet de connaitre la hauteur de notre fenêtre
* **Window.scrollY** : Qui permet de connaitre le scroll sur l’axe des Y
* **Window.open**(‘’http://google.com’’, ‘’Cours JS’’,’’height=600 ’’, ‘’width=800’’): d’ouvrir une nouvelle fenêtre. On peut l’utiliser pour faire une redirection
* **Window.close()** Permet de refermer la fenêtre (par exemple lorsque l’utilisateur soumet le formulaire)

1. Les évènements adossés à WINDOW :

* **Alert**
* **Confirm** : Permet de créer une sorte de pop-up qui demande à l’utilisateur si il est sur de l’option qu’il vient de choisir
* **Prompt.**
* **setTimeOut : Il pend deux paramètres :**

1. **La fonction à exécuter**
2. **Le temps d’arrêt avant d’exécuter le contenu de la fonction (en milliseconde)**

* **setInterval : Permet d’exécuter un bout de code de manière continue et à un interval choisi… La méthode prend deux paramètres :**

1. **Le bout de cote à exécuter**
2. **L’interval entre chaque exécution(en miliseconde)**

**NB : setInterval a un inconvénient c’est qu’il ne s’arrete jamais… Pour l’arreter, on fait comme ca :**

* **On affecte setInterval dans une variable (avec tout son contenu)**
* **On appelle la fonction clearInterval() et on lui passe en paramètre la variable qui a reçu setInterval**

**NB : Pour savoir à chaque fois sur quoi on a cliquer, onpeut utiliser l’évènement click sur le body, passer (e) en paramètre et afficher… e.target qui représente l’élément sur lequel on a cliqué**

**NB : On peut également utiliser e.target pour enlever un élément de la page. Pour faire cela, on utilise : e.target.remove()**

1. **L’objet LOCATION**

* **Location.href : Permet de savoir ou on est exactement**

**NB :**

* **On peut enlever un attribut à un élément HTML avec la fonction element.removeAttribute(‘’nom\_de\_attribut’’)**
* **On peut ajouter un attribut à un élément HTML en utilisant la fonction element.setAttribute( )**