



# **Formation JavaScript Front-end**

Massinissa CHAOUCHI - 2023

# Objectifs

- Développer une bonne méthode de travail
- Utiliser l'objet document
- Utiliser les événement click et scroll
- Créer des pages web dynamiques



# **Projet 0**

# **Initiation**

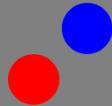
# Présentation du projet 0 - Jeu de balles

- Jeu de clique
- Il faut cliquer sur des balles qui se déplace
- A chaque clique sur une balle le score augmente de 1 point.
- Si le chronomètre arrive à zéro la partie est finie.

# Le rendu final

- J'ai cliqué 3 fois sur les balles et il me reste 3 secondes

**Score : 0**  
**Timer : 3s**



# Le rendu final **GAME OVER**

- Le chronomètre est arrivé à 0 la partie est fini et les balles ont disparues.

**Score : 3**  
**GAME OVER**

# Récupérer un élément HTML par id

- Pour récupérer un élément via son ID il faut utiliser la fonction : getElementById()

```
const elem = document.getElementById("id")
```

- Le document représente la balise <html> </html>
- On accède au élément du document avec des fonctions de l'objet document comme getElementById().
- On utilise une constante pour stocker l'élément html aussi appelé nœud.

# Modifier le contenu textuel d'un nœud

- Pour modifier le contenu d'un nœud, j'utilise l'attribut [innerText](#)

```
const elem = document.getElementById("id")  
  
elem.innerText = "Bonjour";
```



# Modifier le style d'un nœud

- Pour modifier le style d'un nœud, j'utilise l'attribut style

```
const elem = document.getElementById("id")  
//elem.style.propriétéCSS  
elem.style.color = "green";
```

# L'événement « click »

- Pour détecter le click sur un élément il faut utiliser la fonction addEventListener()

```
const elem = document.getElementById("id")

elem.addEventListener("click",function(){
    console.log("J'ai cliqué !");
})
```

# L'événement « click »

- Le premier paramètre est l'événement choisi

```
const elem = document.getElementById("id")

elem.addEventListener("click",function(){
    console.log("J'ai cliqué !");
})
```

# L'événement « click »

- Le deuxième est une fonction orpheline qui va s'exécuter lors du clique.

```
const elem = document.getElementById("id")

elem.addEventListener("click",function(){
    console.log("J'ai cliqué !");
})
```

# 0 – Jeu de balles, conception

1. Si les balles sont cliquées alors le score augmente et est affiché dans la console
2. Ensuite afficher le score dans la balise « #score »

Prérequis :

1. Variable let
2. document.getElementById()
3. elem.addEventListener()
4. elem.innerText

# Projet – 0 Ajouter un chronometre

- Créer un variable chrono qui commence à 10secondes
- Créer un intervalle qui va exécuter une fonction toutes les 1 seconde
- Dans cette fonction décrémenter de 1 le chrono et afficher le chrono
- Si le chrono est inférieur ou égal à 0 Alors l'intervalle s'arrête, le jeu affiche « GAME OVER » et les balles disparaissent.

# Supprimer un nœud HTML

- Pour supprimer un élément il faut utiliser la fonction [remove\(\)](#)

```
const elem = document.getElementById("id")
```

```
elem.remove()
```

# Exécuter une fonction à un intervalle

- Pour exécuter une fonction toutes les secondes il faut utiliser la fonction `setInterval(function(),milliseconde)`

```
let chrono = 0;
setInterval(function(){
    chrono++;
    console.log(chrono)
},1000);
```



# Exécuter une fonction à un intervalle

- Ici le deuxième paramètre vaut 1000 millisecondes soit 1 seconde

```
let chrono = 0;  
setInterval(function(){  
    chrono++;  
    console.log(chrono)  
}, 1000);
```

# Exécuter une fonction à un intervalle

- Le premier paramètre est une fonction qui augmenter le chrono de 1 à chaque seconde, C'EST UNE MONTRE ! 😊

```
let chrono = 0;  
setInterval(function(){  
    chrono++;  
    console.log(chrono)  
}, 1000);
```

# Arrêter l'intervalle

- Pour arrêter un intervalle il faut utiliser la fonction [clearInterval\(\)](#) et lui donner l'identifiant de l'intervalle retourné par la fonction setInterval()

```
let chrono = 0;
const idInterval = setInterval(function(){
    chrono++;
    if(chrono>=10){
        clearInterval(idInterval);
        console.log("10 secondes");
    }
}, 1000);
```

# Projet – 0 Ajouter un chronomètre

- Créer un variable chrono qui commence à 10secondes
- Créer un intervalle qui va exécuter une fonction toutes les 1 seconde
- Dans cette fonction décrémenter de 1 le chrono et afficher le chrono
- Si le chrono est inférieur ou égal à 0 Alors l'intervalle s'arrête, le jeu affiche « GAME OVER » et les balles disparaissent.

# 0 – Jeu de balles, conception chrono

1. Le chrono décroît de 10 à 0 puis GAME OVER
2. La balise #timer affiche le temps qui passe puis GAME OVER
3. Au GAME OVER les balles sont supprimées

Prérequis :

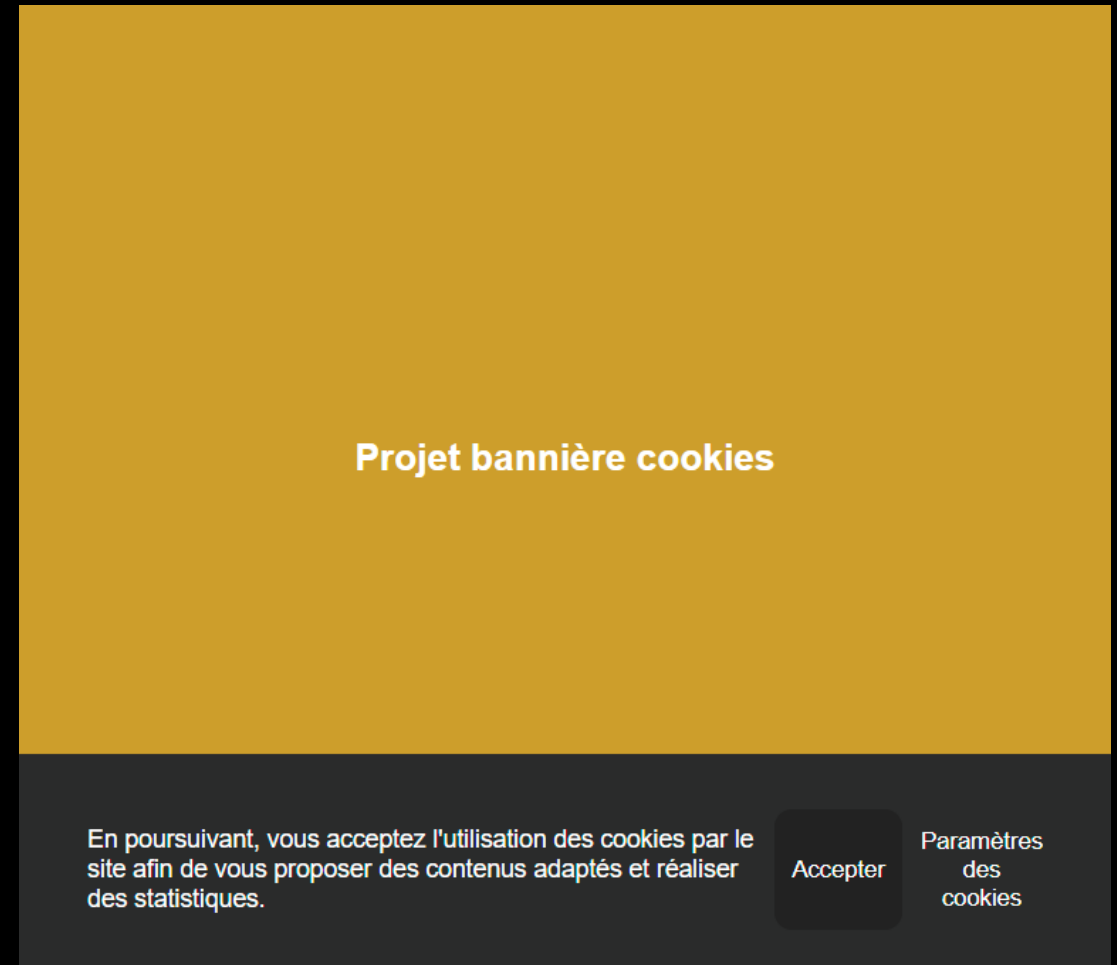
1. [setInterval\(\)](#)
2. [clearInterval\(\)](#)
3. [Opérateur --](#)

# **Projet 1**

## **Bannière cookie**

# Présentation du projet 1 – Bannière à cookie

- Une bannière s'affiche si l'on clique sur accepter les cookies la bannière disparaît doucement.
- Pour la faire disparaître il faudra modifier en JS la propriété CSS opacity



# Récupérer un nœud par class

- Pour récupérer un nœud via sa classe il faut utiliser la fonction [querySelector\(\)](#)

```
const elem = document.querySelector(".classname")
```



# Récupérer un nœud par class

- `querySelector` prend en paramètre un sélecteur CSS
- Et renvoi le premier élément trouvé

```
const elem = document.querySelector(".classname")
```

- *Cette fonction fonctionne donc aussi pour un id*

```
const elem = document.querySelector("#id")
```

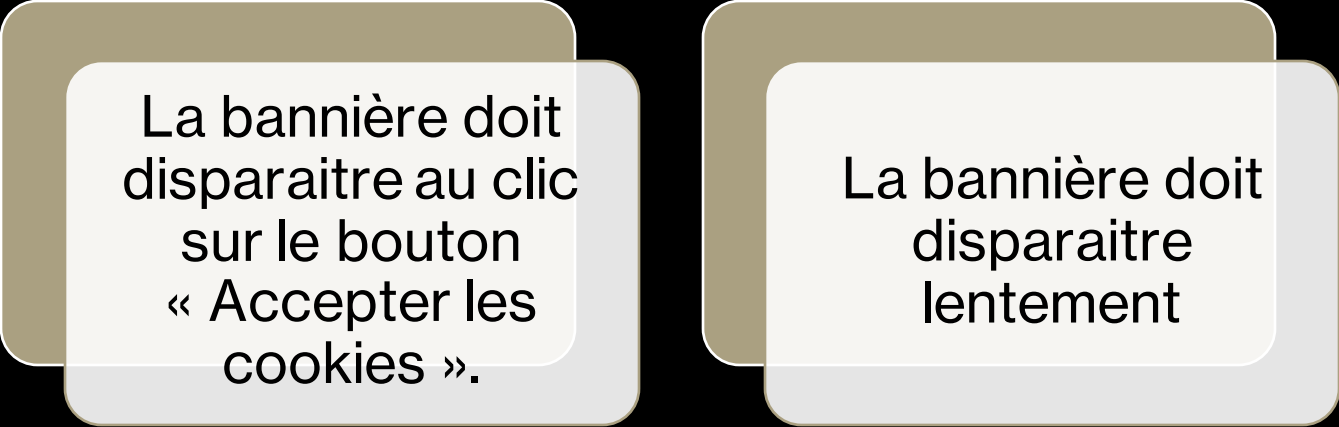
# Changer la propriété opacity d'un nœud HTML

- Je peux changer une propriété CSS en JS via l'attribut element.style

```
const elem = document.querySelector(".className")  
  
//Invisible  
elem.getElementsByClassName.opacity = 0;
```

- Si je précise en CSS une transition sur opacity l'element va effectuer un effet dit « fade-out »

# 1 – Conception Bannière cookies



La bannière doit  
disparaître au clic  
sur le bouton  
« Accepter les  
cookies ».

La bannière doit  
disparaître  
lentement

Prérequis :

1. [document.querySelector\(\)](#)
2. [elem.style.opacity](#)
3. [CSS transition](#) : opacity 1s;

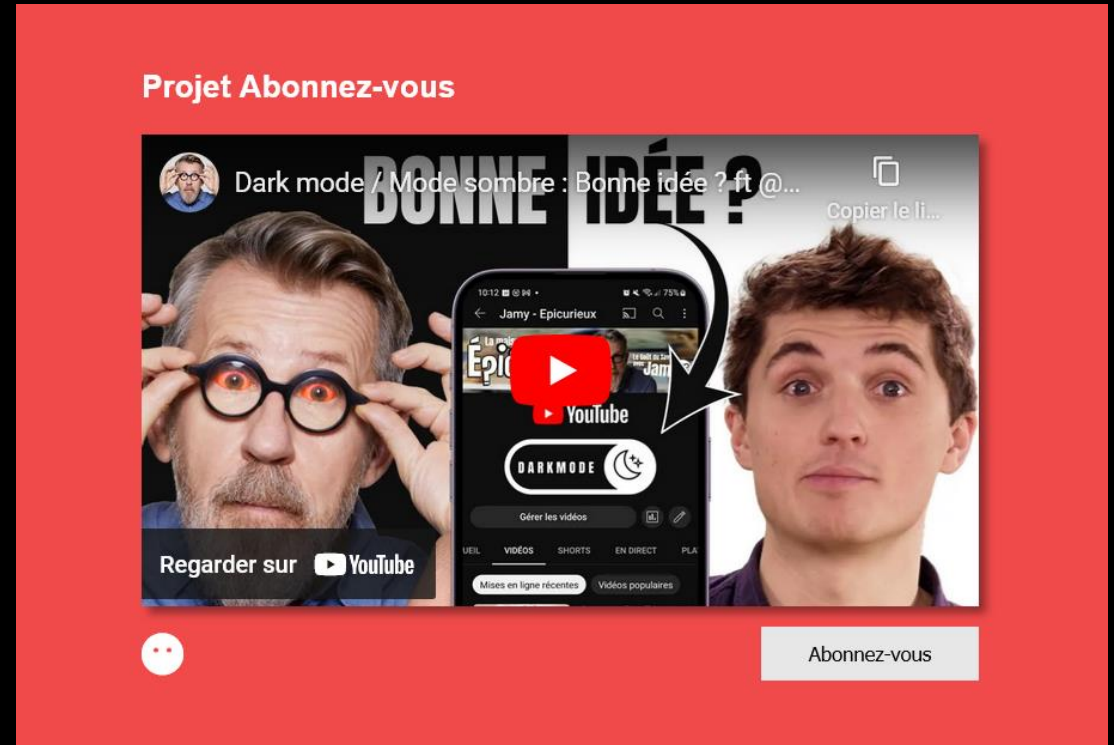


# **Projet 2**

# **Abonnez-vous**

# Présentation du projet 2 – Abonnez-vous

- Lors du clique sur le smiley, on rajoute une class css qui va changer l'apparence et la couleur du smiley
- On va utiliser FontAwesome
- Lors du clique sur abonnez-vous le texte et la couleur de fond du bouton change
- Si je reclique le texte et la couleur revient à la normal



# Rajouter une classe à un nœud

- Pour ajouter une classe à un nœud il faut utiliser la fonction `elem.classList.add()`

```
const elem = document.querySelector(".className")  
  
elem.classList.add("newClass")
```

- Cette fonction va ajouter une nouvelle classe dans l'attribut classe du nœud sélectionné.

# Supprimer une classe à un nœud

- Pour supprimer une classe d'un nœud il faut utiliser la fonction [elem.classList.remove\(\)](#)

```
const elem = document.querySelector(".className")  
  
elem.classList.remove("classInutile")
```

- Cette fonction va retirer une nouvelle classe de l'attribut « class » du nœud sélectionné si elle était présente.

# Remplacer une classe par une autre

- J'utilise la fonction `elem.classList.replace()`

```
const elem = document.querySelector(".className")  
  
elem.classList.replace("oldClass", "newClass")
```



## 2 – Conception Abonnez-vous

Le smiley change d'humeur au clic. Il passe de sobre à sourire et inversement.

Le bouton « Abonnez-vous » doit changer de couleur et devenir « Abonné » au clique

Vous pouvez utiliser un SI lors du clique pour savoir si le texte dans le bouton est “Abonné” ou “Abonnez-vous” et faire les changements en conséquence.

Prérequis :

1. [`elem.classList.toggle\(\)`](#)
2. [`elem.innerText`](#)
3. [`Operator ===`](#)
4. Font awesome : [library](#) et [site officiel pour trouver des icones](#)



# **Projet 3**

## **Dark mode**

# Présentation du projet 3 – darkmode

- Lors du clique sur un bouton la page passe en darkmode
- Si je reclique dessus la page passe en lightmode
- Je vais avoir besoin d'appliquer une classe darkmode à tout les éléments de la page html
- Je vais avoir besoin d'un boucle for

Projet DARK MODE



Projet DARK MODE



# Récupérer un tableau d'élément par classe

- Pour ceci : `document.querySelectorAll()`

```
const elems = document.querySelectorAll(".className")
```

- *Cette fonction fonctionne comme `querySelector` sauf qu'elle renvoie tout les éléments et pas seulement le premier rencontré.*

# Effectuer une boucle form sur un élément HTML

- [forEach](#) est une fonction présente sur tout les tableaux JS, y compris le tableau reçu de `querySelectorAll()`

```
const elems = document.querySelectorAll(".className")

elems.forEach(function(elem){
  console.log(elem);
})
```

# Inverser la presence d'une classe avec `element.classList.toggle()`

- [`element.classList.toggle\(\)`](#)
  - Ajoute une classe si elle n'existe pas
  - Retirer une classe si elle existe
- Très pratique pour éviter d'avoir un if else inutile et peu élégant
- A utiliser sur tout element qui doit alterner entre deux état à chaque click par exemple.

# 3 – Conception DarkMode

- Lorsque je clique sur le switch le soleil se change en lune grâce aux classes de FontAwesome
- Lorsque je clique sur le switch tout les éléments HTML reçoivent la classe « darkmode » vous pouvez utiliser le selector CSS « \* » pour ceci.

Projet DARK MODE



Prérequis :

1. [querySelectorAll\(\)](#)
2. [forEach\(\)](#)
3. [element.classList.toggle\(\)](#)
4. [\\* selector css](#)



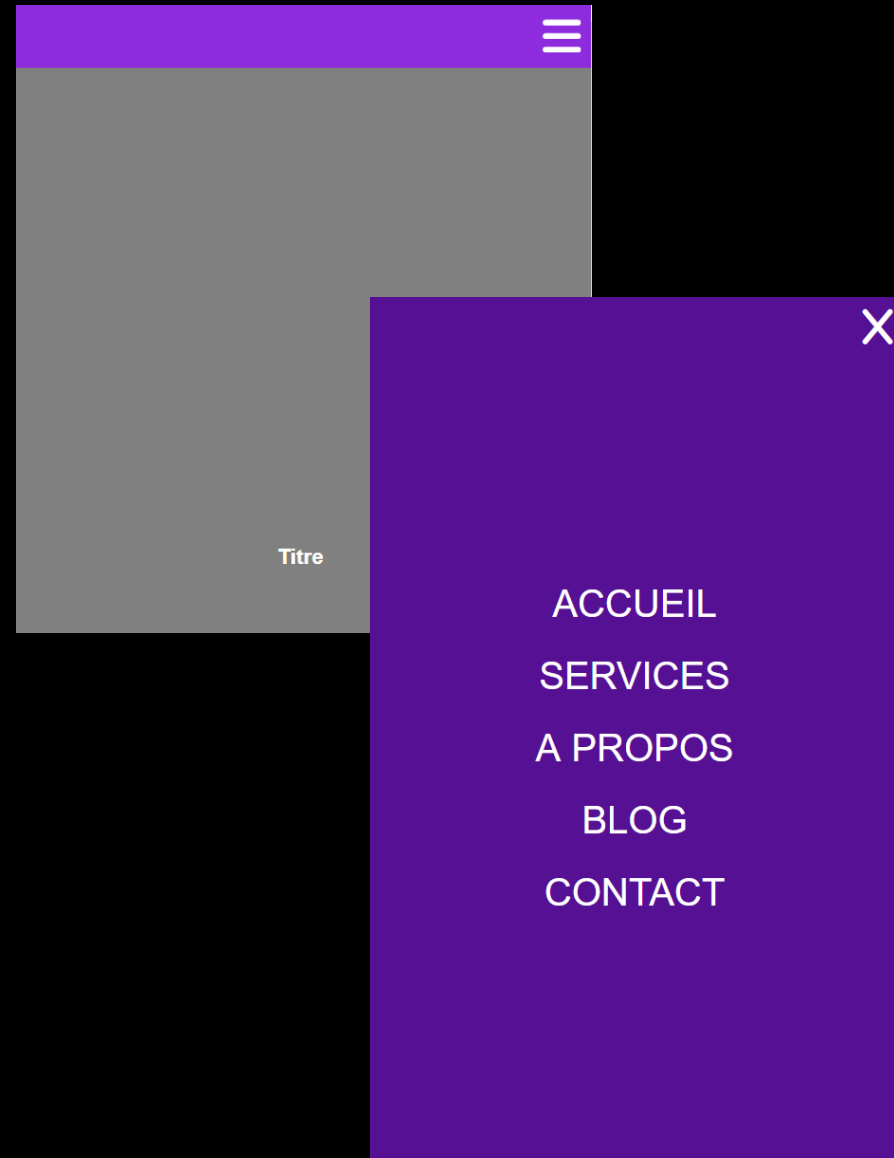
# **Projet 4**


## **Menu burger**



# 4 – Conception menu burger

- Lorsque le clique sur le burger j'applique la classe show-modal à la fenetre modal
- Lorsque je clique sur le burger j'applique la classe show-modal sur le burger et je change le burger en croix avec les classes font awesome



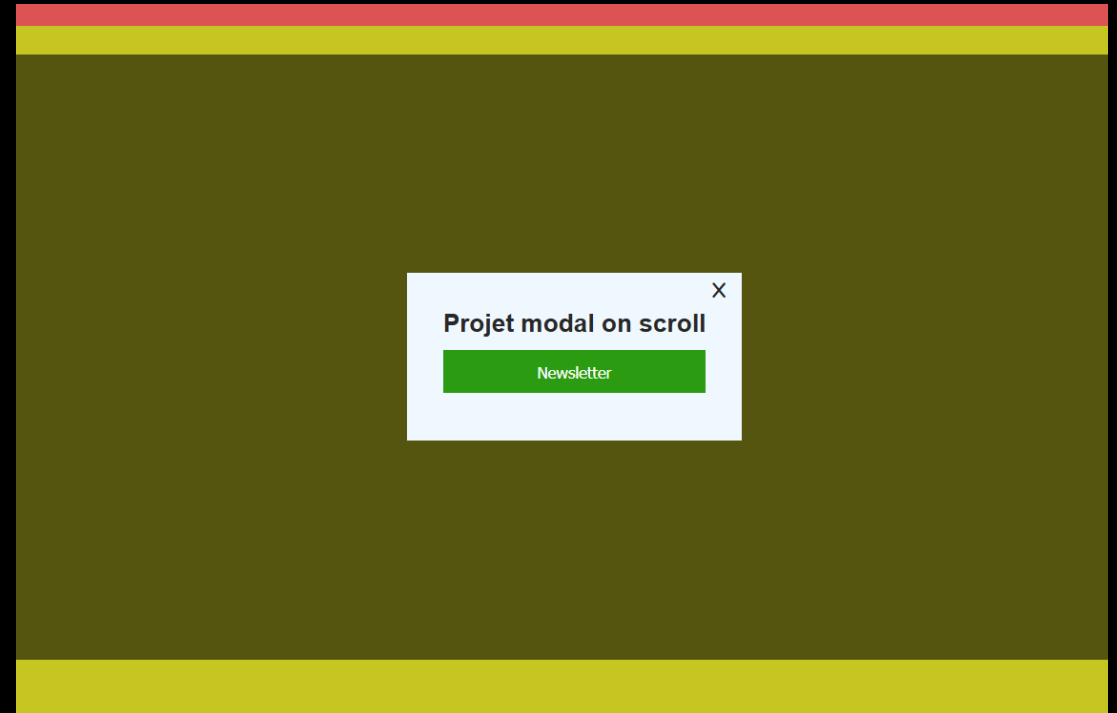


# **Projet 5**

## **Modal Scroll**

# Présentation du projet 5 – Modal Scroll

- J'écoute l'événement scroll qui se déclenche quand l'utilisateur scroll.
- Si la distance parcourue est supérieure à la position d'une section voulue j'affiche la fenêtre modale.
- A l'affichage je supprime l'écoute de l'événement scroll pour que l'événement ne se déclenche qu'une fois



# Scroll Event : « scroll »

- L'événement scroll ne se déclenche que sur un élément scrollable, le scroll de la fenêtre est géré par l'objet [window](#)

```
window.addEventListener("scroll",function(){  
    console.log(document.documentElement.scrollTop);  
});
```

- [Document.documentElement](#) renvoi la balise <html></html>
- [scrollTop](#) permet de récupérer la position du haut de la fenetre sur l'axe X.

# Scroll Event : « scroll » : test si l'utilisateur à scroller de 400px

```
✓ window.addEventListener("scroll",function(){  
    ✓ const hauteur = document.documentElement.scrollTop;  
    if(hauteur > 400){  
        console.log("400px atteint!");  
    }  
});
```

# Scroll Event : récupérer la position d'un element

- [Element.offsetTop](#) renvoie la position du bord haut de l'element

```
const hauteurSection2 = section2.offsetTop;
```

# Scroll Event : déclencher un message en scroll sur un élément

```
const html = document.documentElement;
const section2 = document.querySelector(".section2");

window.addEventListener("scroll",function(){
    const hauteur = document.documentElement.scrollTop;
    const hauteurSection2 = section2.offsetTop;

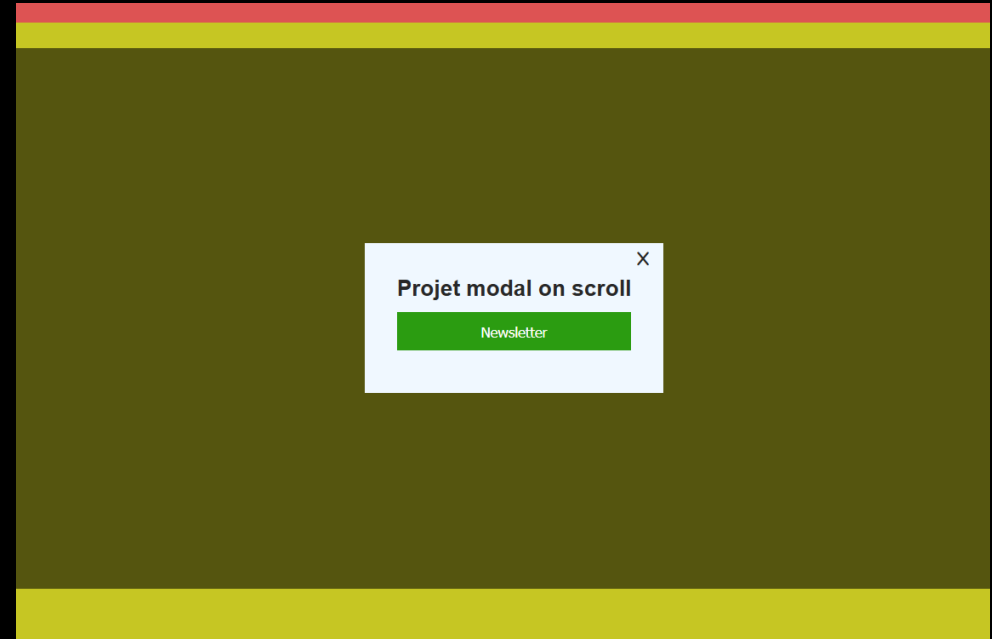
    console.log(hauteurSection2);

    if(hauteur > hauteurSection2){
        console.log("section2 atteinte!");
    }
});
```

# 5 – Conception

## Modal Scroll

- Si l'utilisateur arrive à la section 2 alors la fenêtre modal apparaît
- Si l'utilisateur clique sur la croix alors la fenêtre modal disparaît et l'écouteur d'événement « scroll » est supprimée



Prérequis :

1. [Element.removeEventListener\(\)](#)
2. [Scroll event](#)
3. [Document.documentElement.scrollTop](#)
4. [element.offsetHeight](#)





# Projet 6

# Toasts

# Présentation du projet 6 – Toast

- Les toasts sont des notifications qui disparaissent après un certain temps.
- Je vais devoir créer un `nœud(element)` en javascript
- Je vais devoir supprimer ce `nœud` après 3secondes grâce à la fonction `setTimeout()`
- Je veux aussi supprimer le toast si je clique dessus

## Projet Toast

Enregister

Votre fichier est enregistré !

Votre fichier est enregistré !

Votre fichier est enregistré !

# Création d'element en JavaScript

- La création d'element se fait en deux étapes minimum :
  1. **Créer** l'élément avec la méthode `createElement()` de l'objet document( ceci n'affiche pas l'element)
  2. **Ajouter** l'element au DOM avec la methode `appendChild()`

```
const elem = document.createElement("h2");  
container.appendChild(elem);
```

# Precision sur appendChild()

- **appendChild()** ajoute un element à l'intérieur d'une balise html conteneur,
- A l'instar d'une concatenation effectuée avec un innerHTML mais en bien plus propre.
- Ici on ajoute du text avant l'ajout de la balise newElem

```
const container = document.querySelector("section");

const newElem = document.createElement("h1");
newElem.innerText = "Titre dynamique";
container.appendChild(newElem);
```

## Avant

```
<!DOCTYPE html>
<html> event
  <head> ... </head>
  <body>
    <h1>This is a Heading</h1>
    <p>This is a paragraph.</p>
    <section>
    </section>
    <script> ... </script>
  </body>
</html>
```

## Après

```
<!DOCTYPE html>
<html> event
  <head> ... </head>
  <body>
    <h1>This is a Heading</h1>
    <p>This is a paragraph.</p>
    <section>
      <h1>Titre dynamique</h1>
    </section>
    <script> ... </script>
  </body>
</html>
```

# Suppression d'un element

## `element.remove()`

- `element.remove()` retire un element du DOM

```
const titre = document.querySelector("h1");  
titre.remove();
```

# Element.classList

- ClassList renvoi un [DomTokenList](#), qui est un tableau de string qui contient toutes les classes css de l'element.
- ClassList [hérite](#) des attributs et des méthodes de la classe DomTokenList.
- Les méthodes les plus utiles sont :

```
const titre = document.querySelector("h1");
titre.classList.add("newClass");           // Ajouter
titre.classList.remove("classInutile");    // Retirer
titre.classList.replace("sad","smile");    // Remplacer
titre.classList.toggle("className");      // Inverser l'etat de la classe
```

# setTimeout() - Executer une fonction après un temps mort

- setTimeout() execute une fonction orpheline après un certain temps, elle est cousine de la method setInterval()

```
setTimeout(function(){  
    console.log("5 secondes on passées");  
},5000)
```

- setTimeout() n'est pas bloquant.

# 6 – Conception

## Projet Toast

- Si l'utilisateur clique sur le bouton « enregistrer » un toast est créé et affiché.
- Si l'utilisateur clique sur un toast, le toast est supprimé.
- 3 secondes après l'affichage d'un toast, il est supprimé.



## Prérequis

[Document.createElement\(\)](#)

[Element.appendChild\(\)](#)

[Element.remove\(\)](#)

[Element.classList.add\(\)](#)

[Element.innerHTML](#)

[setTimeout\(\)](#)



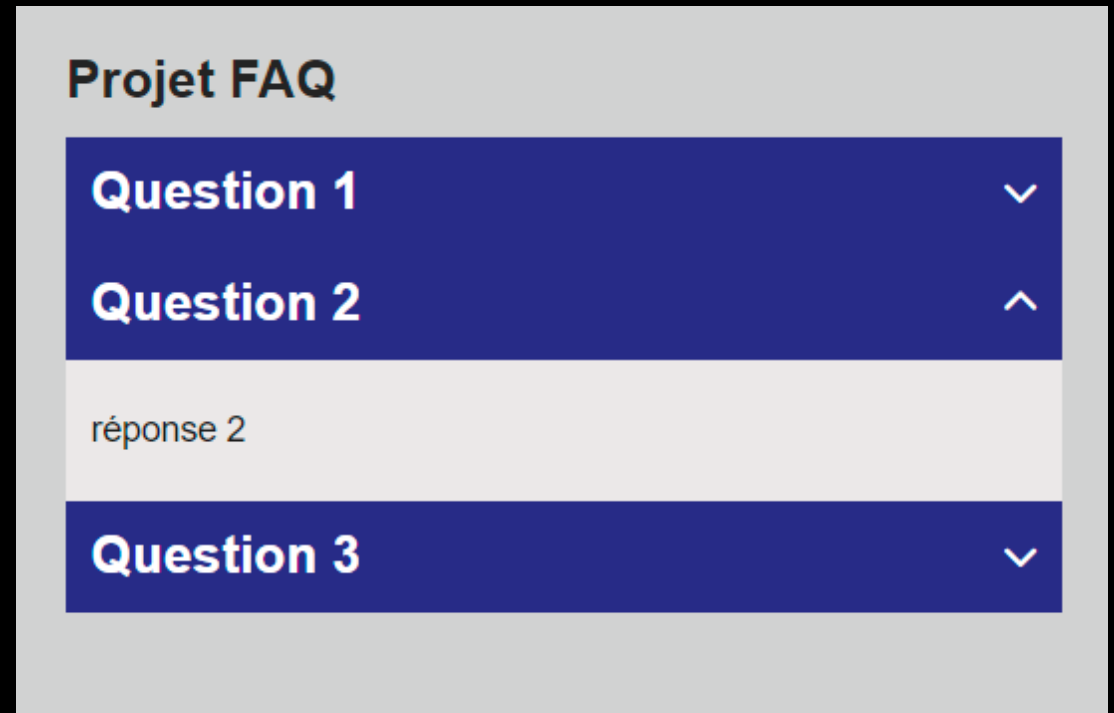


# Projet 7

## FAQ

# Présentation du projet 7 – FAQ

- Lors d'un clic sur la question je récupère la réponse correspondante avec l'attribut [nextElementSibling](#)
- Lors d'un clic sur la question j'applique la classe « show-reponse » à la classe « reponse »
- Au clic j'inverse également les chevrons



# Accéder au balises enfants

## firstElementChild, lastElementChild & childNodes

- Ces fonctions permettent de récupérer le premier et le dernier element d'un container d'élément.

```
const container = document.querySelector(".container");  
container.firstElementChild;  
container.lastElementChild;
```

- ChildNodes renvoi une tableau des balises enfants d'un container

```
const container = document.querySelector(".container");  
const childNodes = container.childNodes;  
console.log(childNodes);|
```

# Accéder aux balises adjacentes

Element.nextElementSibling renvoi l'element suivant

```
const firstElem = document.querySelector("li");  
const secondElem = firstElem.nextElementSibling;  
const thirdElem = secondElem.nextElementSibling;
```

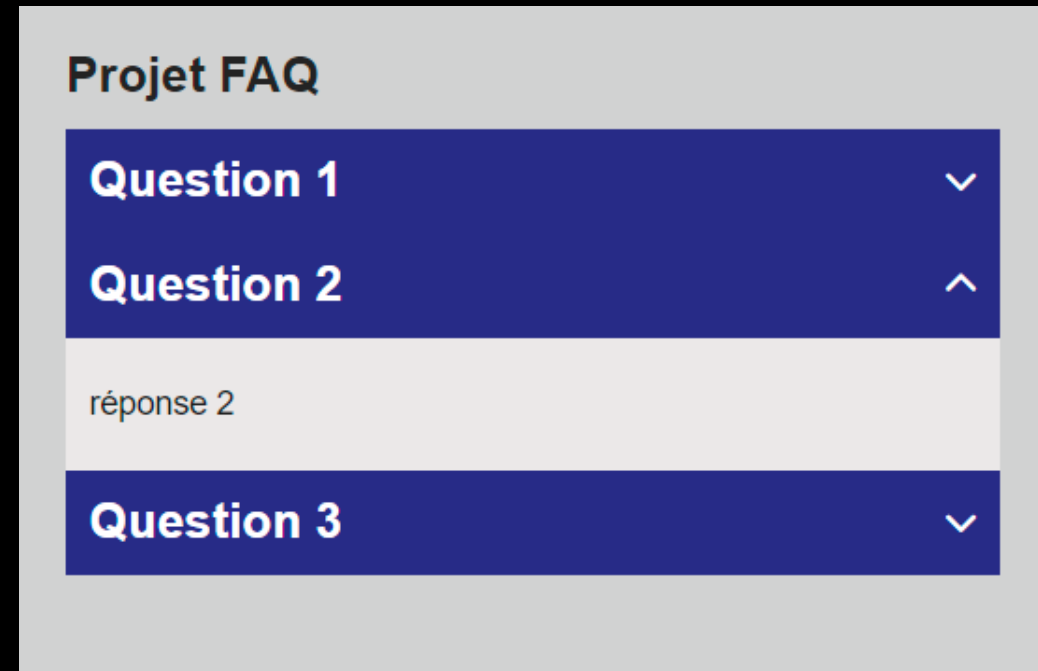
Element.previousElementSibling renvoi l'element précédent

```
const dernierElem = document.querySelector("ul").lastElementChild;  
const avantDernierElem = dernierElem.previousElementSibling;
```

# 7 – Conception

## Projet FAQ

- Si l'utilisateur clique une question une réponse apparaît et les chevrons s'inversent



### Prérequis

`Elem.nextElementSibling`

`Parent.lastElementChild`

`querySelectorAll()`

`classList.toggle()`

`forEach()`



# **Projet 8**

## **Fiche produit**

# Présentation du projet 8 – Fiche produit

- Lorsque je clique sur une vignette, j'inverse les attributs src de la grande image et de la vignette cliquée.
- Lorsque je clique sur le bouton AJOUTER j'incrmente un compteur de produits de 1.
- Si le compteur est plus grand que 0 j'affiche le message indiquant le nombre de produits dans le panier en accordant en nombre le mot produit.



## Chaise scan

99€

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesentium, nobis. Magna aliqua ut enim ut blanditiis nesciunt illum, vel ut voluptates cupiditate sint a aliquam in corporis.

AJOUTER AU PANIER

Vous avez 6 produits dans votre panier

# Précision sur la portée d'une variable : let ou var.

- Une variable **let** sera accessible dans son bloc **bloc d'exécution uniquement** : les accolades d'un if else, while, for, switch
- Alors qu'une variable **var** sera accessible dans son contexte d'exécution : la fonction.
- Toutes variables **let** ou **var** déclarer dans le script en dehors de tout bloc d'exécution ou fonction sera globale au script ENTIER.
- Les portées d'un **const** et d'un **let** sont les mêmes.

```
let moi = "Massi"; // Variable globale au script
var lui = "Lounes"; // Variable globale au script

if(moi === "Massi"){
    let age = 23; //Variable local au bloc d'exécution {...}
    var nom = "CHAOUCHI"; // Variable global au script
}

console.log(nom);
console.log(age); // !ReferenceError: age is not defined!

function hello(){
    /**
     * Les variables moi et lui sont gloables au script
     * donc connu de tous.
     * La variable nom à été déclarée avec var en dehors
     * d'une fonction, elle est donc globale au script.
     */
    console.log("Hello "+moi);
    console.log(nom)
    console.log("Hello "+lui);
    /**
     * Variable local au contexte d'exécution
     * la fonction hello uniquement.
     */
    var phrase = "Au revoir";
}

hello();
console.log(phrase); // !ReferenceError: phrase is not defined!
```



# Modifier l'attribut d'un element

- Pour modifier des attributs comme src ou href il faut utiliser la paire de fonctions
  - [Element.getAttribute\(\)](#)
  - [Element.setAttribute\(\)](#)

```
//<a href="www.site.com">mon lien</a>  
const element = document.querySelector("a");  
const elemAttr = element.getAttribute("href");  
element.setAttribute("href", "www.new-site.com");
```

# L'opérateur ternaire

[Voir la documentation mdn](#)

- L'opérateur ternaire est un opérateur à **trois opérandes** qui permet le **test d'une expression** et le retour d'une valeur si l'expression est VRAI ou d'une autre si l'expression est FAUSSE.
- Il est utilisé en raccourcis à un if else.
- Syntaxe :

```
retour = expression?retour1:retour2;
```

# L'opérateur ternaire : exemple avec et sans.

## Sans

```
/**
 * Sans opérateur ternaire
 */
const prix = 120;

let prixTTC = null;
if(prix >= 100){
    prixTTC = prix *0.70 // Réduction de 30%
}
else{
    prixTTC = prix;
}
```

## Avec

```
/**
 * Avec opérateur ternaire
 */
const prix = 120;
const prixTTC = prix>=100?prix*0.70:prix;
```

Ici prixTTC est égal à 84 car prix est supérieur à 100.

# 8 – Conception

## Projet fiche produit

- Si l'utilisateur clique sur le bouton ajouter, je lui indique combien de produit il a ajouté jusqu'à maintenant dans un message en dessous du bouton.
- Si l'utilisateur clique sur une vignette j'affiche la vignette dans la grande photo et j'affiche la grande photo dans la petite vignette.



## Prérequis

Let variable

Element.getAttribute()

Element.setAttribute()

Opérateur ternaire

Opérateur ++

# **Projet 9**

## **Articles de blog**

# Objet Javascript (simple)

- Un objet c'est la compilation de plusieurs informations dans une structure de donnée appelée **objet**.
- [Voir la MDN !](#)

```
const eleve = {  
    prenom: "Mathieu",  
    nom: "Dubois",  
    age: 16,  
    classe: 721  
}  
console.log(eleve);
```

# Accéder au attribut d'un objet

- J'accède au attributs (variables) d'un objet avec l'opérateur property accessor `'.'`

```
console.log(eleve.nom);  
console.log(eleve.classe);  
console.log(eleve.prenom);  
console.log(eleve.age);
```

# Présentation du projet 9 – Articles de blog

- Je veux afficher des articles de blog
- Je vais utiliser une base de données sous la forme d'un tableau d'objet javascript( à la place de SQL)
- Je vais parcourir ma bdd et récupérer les objets représentent mes articles
- Créer dynamiquement les articles à partir des infos de ma bdd.

## Projet Articles

### SEO, les bonnes pratiques

Mollit ut mollit esse exercitation nisi ut labore velit anim pariatur sit deserunt anim. Dolore consequat aliquip esse elit culpa aliqua. Consectetur mollit irure minim incididunt nulla non. Ad sunt mollit aliqua minim fugiat et minim commodo. Anim proident incididunt veniam Duis cupidatat irure eu. Elit nulla nisi ea laborum mollit excepteur enim ut Lorem. Cupidatat minim consectetur mollit in ut consectetur est Duis do sint cillum nisi.

#SEO

### Bien, les bonnes pratiques

Mollit ut mollit esse exercitation nisi ut labore velit anim pariatur sit deserunt anim. Dolore consequat aliquip esse elit culpa aliqua. Consectetur mollit irure minim incididunt nulla non. Ad sunt mollit aliqua minim fugiat et minim commodo. Anim proident incididunt veniam Duis cupidatat irure eu. Elit nulla nisi ea laborum mollit excepteur enim ut Lorem. Cupidatat minim consectetur mollit in ut consectetur est Duis do sint cillum nisi.

#JS

### Content, les bonnes pratiques

Mollit ut mollit esse exercitation nisi ut labore velit anim pariatur sit deserunt anim. Dolore consequat aliquip esse elit culpa aliqua. Consectetur mollit irure minim incididunt nulla non. Ad sunt mollit aliqua minim fugiat et minim commodo. Anim proident incididunt veniam Duis cupidatat irure eu. Elit nulla nisi ea laborum mollit excepteur enim ut Lorem. Cupidatat minim consectetur mollit in ut consectetur est Duis do sint cillum nisi.

#PHP



# Template string

- Les templates string sont des string qui utilisent les caractères `` plutôt que les " " ou ' ';
- Une template string peut contenir une variable sans effectuer de concatenation.

```
const prenom = "Massinissa";  
const nom = "CHAOUCHI";  
const age = 23;  
const phrase = `Bonjour je suis ${prenom} ${nom} et j'ai ${age} ans.`;
```

# Template String : exemple réel avec du HTML

```
const perso = {
  nom: "Artorias",
  vitalite : 500,
  pdv : 200,
  manaMax: 1300,
  mana: 600,
  photo : "http://unsplash.it/500/500" //photo aléatoire
};

const fichePerso = `
  <div>
    
    <h2>Nom : ${perso.nom}</h2>
    <label for="vie">Vie:</label>
    <progress id="vie" max="${perso.vitalite}" value="${perso.pdv}"> 70% </progress>
    <label for="mana">Mana:</label>
    <progress id="mana" max="${perso.manaMax}" value="${perso.mana}"> 70% </progress>
  </div>
`;
document.body.innerHTML+=fichePerso;
```

# Template String : exemple réel avec du HTML

Création de l'objet personnage

```
const perso = {  
  nom: "Artorias",  
  vitalite : 500,  
  pdv : 200,  
  manaMax: 1300,  
  mana: 600,  
  photo : "http://unsplash.it/500/500" //photo aléatoire  
};
```

# Template String : exemple réel avec du HTML

Création du html de la fiche de perso

```
const fichePerso = `

![icon du personnage](${perso.photo})

## Nom : ${perso.nom}</h2><label for="vie">Vie:</label><progress id="vie" max="${perso.vitalite}" value="${perso.pdv}"> 70% </progress><label for="mana">Mana:</label><progress id="mana" max="${perso.manaMax}" value="${perso.mana}"> 70% </progress></div>`;


```

# Template String : exemple réel avec du HTML

Concatenation de la div dans le innerHTML

```
document.body.innerHTML+=fichePerso;
```

# Concatenation de html avec innerHTML

- `element.innerHTML` contient une string qui est le code HTML de l'element.
- `element.InnerText` ne renvoi que le texte ignorant le HTML.
- `innerHTML` nous permet grace à l'opérateur `+=` de rajouter un element dans un container sans utiliser `document.createElement()` et `container.appendChild()`.

# 9– Conception

## Projet articles de blog

- Si l'utilisateur clique sur le bouton ajouter, je lui indique combien de produit il a ajouté jusqu'à maintenant dans un message en dessous du bouton.
- Si l'utilisateur clique sur une vignette j'affiche la vignette dans la grande photo et j'affiche la grande photo dans la petite vignette.

### Projet Articles

#### SEO, les bonnes pratiques

Mollit ut mollit esse exercitation nisi ut labore velit anim pariatur sit deserunt anim. Dolore consequat aliquip esse elit culpa aliqua. Consectetur mollit irure minim incididunt nulla non. Ad sunt mollit aliqua minim fugiat et minim commodo. Anim proident incididunt veniam Duis cupidatat irure eu. Elit nulla nisi ea laborum mollit excepteur enim ut Lorem. Cupidatat minim consectetur mollit in ut consectetur est Duis do sint cillum nisi.

#SEO

#### Bien, les bonnes pratiques

Mollit ut mollit esse exercitation nisi ut labore velit anim pariatur sit deserunt anim. Dolore consequat aliquip esse elit culpa aliqua. Consectetur mollit irure minim incididunt nulla non. Ad sunt mollit aliqua minim fugiat et minim commodo. Anim proident incididunt veniam Duis cupidatat irure eu. Elit nulla nisi ea laborum mollit excepteur enim ut Lorem. Cupidatat minim consectetur mollit in ut consectetur est Duis do sint cillum nisi.

#JS

#### Content, les bonnes pratiques

Mollit ut mollit esse exercitation nisi ut labore velit anim pariatur sit deserunt anim. Dolore consequat aliquip esse elit culpa aliqua. Consectetur mollit irure minim incididunt nulla non. Ad sunt mollit aliqua minim fugiat et minim commodo. Anim proident incididunt veniam Duis cupidatat irure eu. Elit nulla nisi ea laborum mollit excepteur enim ut Lorem. Cupidatat minim consectetur mollit in ut consectetur est Duis do sint cillum nisi.

#PHP

## Prérequis

Objet

Template string vo vf

innerHTML

+= operator

forEach



# **Projet 10**

# **Formulaire**



# Présentation du projet 10 - Formulaire

- Je dois annuler le comportement par défaut du formulaire.
- Je dois récupérer le contenu des champs.
- Je dois vérifier si les champs sont valides(nombres de lettres)
- Je dois vérifier si le champ email est valide avec une regex.
- Je dois placer mes données ainsi triées dans un objet data
- Je dois console.log cet objet

Projet contact form

Prenom

Nom

Email

Votre message

ENVOYER

# L'evenement "submit"

- L'evenement submit s'active lors de click sur le bouton type="submit" ou l'appuie sur la touche ENTER.

```
const form = document.querySelector("form");  
  
form.addEventListener("submit",function(){  
    console.log("submitted!");  
});
```



Le problème c'est que si je soumetts mon form, rien ne s'affiche. Car par défaut, lors du clique, le formulaire charge la page demandée dans l'attribut action du form. Il nous faut donc annuler cette effet indésirable en JS.

# Annuler le rechargement de la page à la soumission du form

- Utiliser la fonction `event.preventDefault()` présente dans le paramètre de la fonction callback de `addEventListener()`

```
const form = document.querySelector("form");

form.addEventListener("submit",function(event){
    event.preventDefault();
    console.log("submitted!")
});
```

# Récupérer le contenu d'un champ

- Utiliser l'attribut value de l'element HTMLInputElement

```
const form = document.querySelector("form");
const champ = document.getElementById("idChamp");

form.addEventListener("submit",function(event){
    event.preventDefault();
    const champValue = champ.value;
    console.log("submitted!");
});
```

# Eviter les champs vides

- Pour éviter de prendre un champ rempli d'espace pour un champ rempli on supprime les caractères espace avant et après la string `Element.value`.
- On utilise la fonction `trim()` directement sur `Element.value` pour effectuer ce traitement en une seule ligne.

```
const form = document.querySelector("form");
const champ = document.getElementById("idChamp");

form.addEventListener("submit", function(event){
    event.preventDefault();
    const champValue = champ.value.trim();
    console.log("submitted!");
});
```

# Connaitre la taille d'une string

- String.prototype.length permet de récupérer la taille d'une string.

```
const champ = "Massinissa";  
console.log(champ.length) //10
```

# Vérifier un email avec les regex

- Le regex est un langage de traitement de chaîne de caractère.
- Il est plutôt complexe à lire je vous propose donc :
  - [Le site regex101 pour apprendre à créer des regex.](#)
  - La fonction `isValidEmail` que vous pouvez utiliser dans vos projets pour tester un email valide. A copier coller diapo suivante.

# Vérifier un email avec les regex

```
/**
 * @description Renvoi true si email est un email valide, renvoi false sinon.
 * @param {email} email
 * @returns boolean
 */
function isValidEmail(email){
    const emailFormat = /^\\w+([\\.-]?\\w+)*@\\w+([\\.-]?\\w+)*\\.\\w{2,3}+$/; //
    Création d'un objet RegExp
    if (emailFormat.test(email))
    {
        return true;
    }else{
        return false
    }
}
```



# 10 – Conception Formulaire

- Je dois afficher les champs écrits dans le formulaire lors de l'envoi.
- Les champs doivent être mis en forme dans un objet nommé data.
- Le script JS doit vérifier la validité de chaque champ en vous basant sur les messages d'erreurs déjà présent dans le html sous la classe error.
- Les messages d'erreur doivent être affichés en cas d'erreur.

Projet contact form

Prenom

Nom

Email

Votre message

## Prérequis

Objet

If else

Event.preventDefault()

Submit event

string.trim()

IsValidEmail()

string.length

nextElementSibling

ClassList.remove

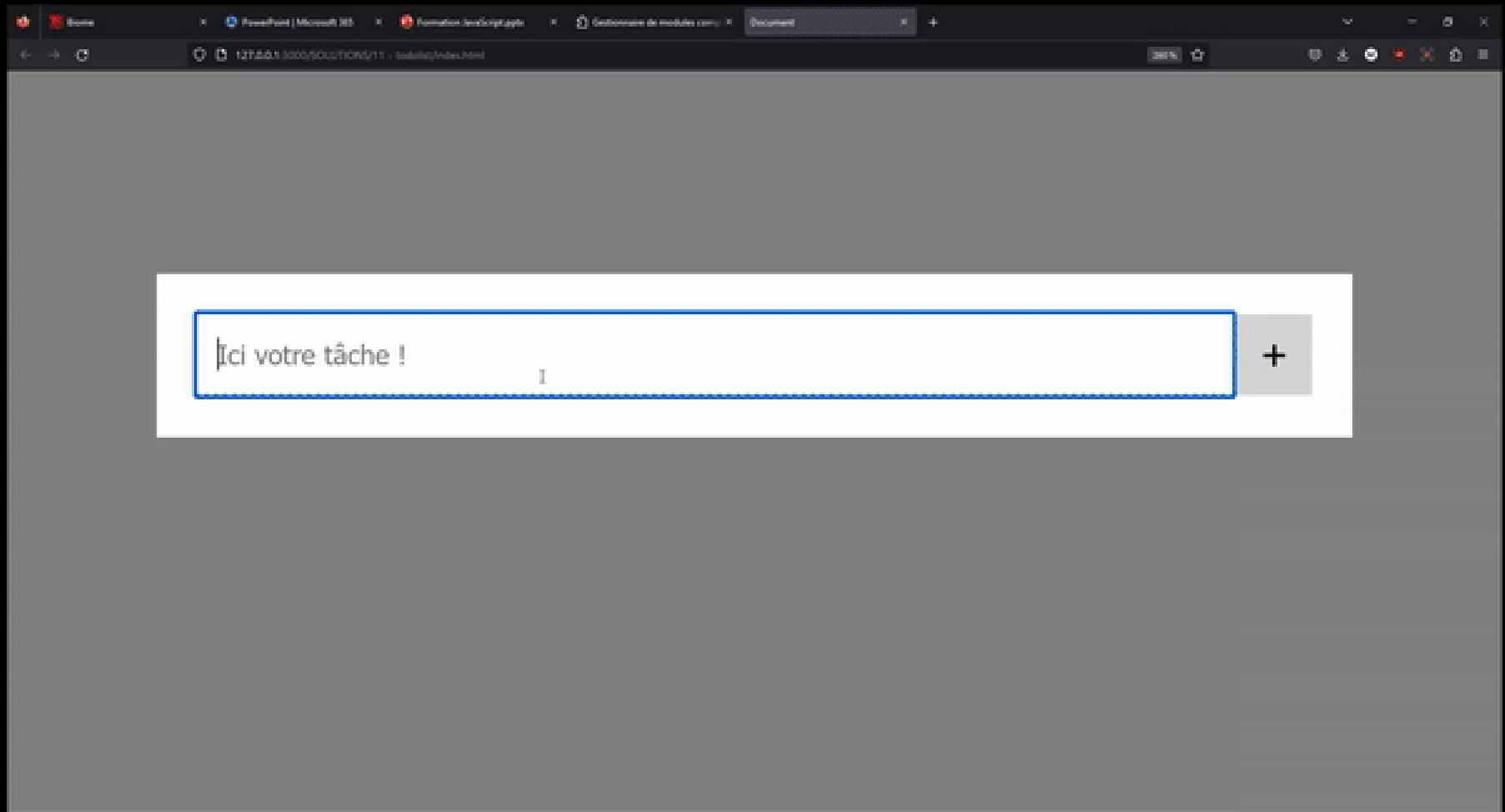
ClassList.add



# **Projet 11**

## **TODO List**

# Présentation du projet 11 – TODO List



# Récupérer l'element parent

- Element.parentElement permet de récupérer l'element parent.
- C'est très pratique lorsque l'on veut supprimer un element au clique sur un bouton qui se trouve dans l'element.

```
const btnDelete = document.querySelector(".article")  
const article = btnDelete.parentElement;
```

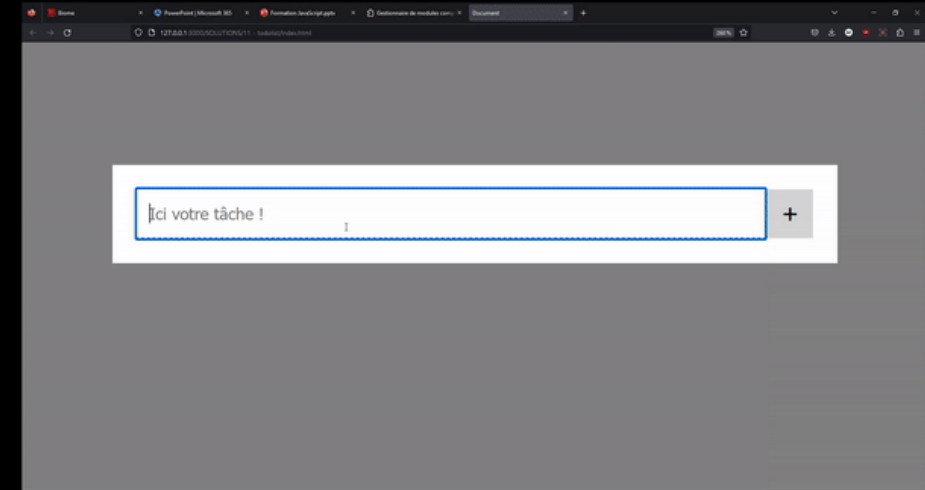
# Reset les champs du formulaire

- Pour reset les champs du formulaire il existe la fonction reset de l'element form.

```
// Je reset le contenu du formulaire pour eviter que le texte reste une  
fois la tache crée (ergonomique)  
form.reset();
```

# 11 – Conception TODO List

- Je peux ajouter une tâche avec le bouton plus ou la touche entrée
- Je peux supprimer une tâche avec le bouton supprimer (rouge)
- Je peux barrer une tâche avec le bouton archiver(vert)
- BONUS : Je n'ajoute pas la tâche si le champ de saisi est vide.



---

Prérequis

`Form.reset()`

---

`Element.ParentElement`

---

# **Projet 12**

## **Bannière localStorage**

# 12 - Présentation du projet

## Bannière localStorage

[Voir le cours sur la MDN](#)

- Cette fois si :
  - lorsque je clique sur le bouton accepter
  - j'ajoute une donnée dans le localStorage du client
    - Accept-cookies: "accept-all"
- Au chargement de ma page je vérifie si Accept-cookie existe dans le localStorage et j'affiche la bannière si il n'existe pas.



# Définir, Supprimer ou récupérer un item du localStorage

- [LocalStorage.setItem\(\)](#) : Créer un item dans le LS
- [LocalStorage.getItem\(\)](#) : Fournit l'item demander
- [LocalStorage.removeItem\(\)](#) : Supprime l'item demandé du LS
- [LocalStorage.clear\(\)](#) : Supprime tout les items du LS

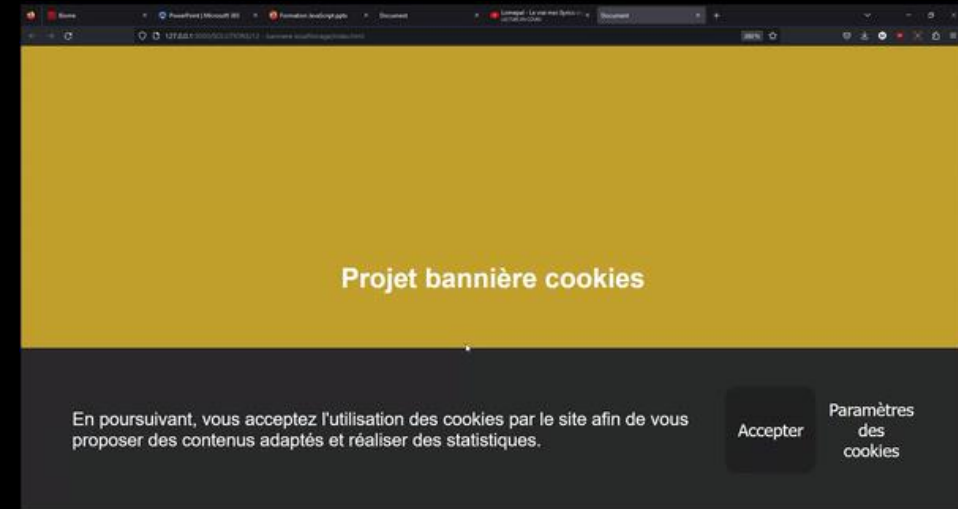
```
localStorage.setItem("user_name", "massinissa")  
const name = localStorage.getItem("user_name");  
localStorage.removeItem("user_name");  
localStorage.clear()
```

# Différence entre cookies, localStorage et sessionStorage

	cookies	localStorage	sessionStorage
Capacité	4kb	10mb	5mb
Données accessibles	Depuis toutes les fenêtres et onglets	Depuis toutes les fenêtres	Depuis l'onglet
Date d'expiration	À définir	Jamais	À la fermeture de l'onglet
Stocké dans	Le navigateur et le serveur	Le navigateur	Le navigateur
Envoyées avec les requêtes HTTP	Oui	Non	Non

# 12 – Conception Bannière cookies

- Je peux ajouter une tâche avec le bouton plus ou la touche entrée
- Je peux supprimer une tâche avec le bouton supprimer (rouge)
- Je peux barrer une tâche avec le bouton archiver(vert)
- **ASTUCE** : Rendez-vous dans l'onglet dev> Stockage pour voir le localStorage
- **BONUS** : Je n'ajoute pas la tâche si le champ de saisi est vide.



Prérequis

# localStorage