# Javascript – Interaction avec la page

Bailly Benjamin

# Intéragir avec votre page – les différents objets

L'objet **window** est le parent « suprême » en Javascript, c'est de lui que tout part.
https://www.w3schools.com/jsref/obj_window.asp

L'objet **navigator** est l'objet qui va contenir toutes les informations du navigateur.
https://www.w3schools.com/jsref/obj_navigator.asp

L'objet **history** contient les urls visité par le visiteurs (dans une fenêtre).

L'objet **location** contient des infos sur l'url. https://www.w3schools.com/jsref/obj_location.asp

L'objet **screen** contient des infos sur l'écran du visiteur. https://www.w3schools.com/jsref/obj_screen.asp

Et le plus imporant, l'objet **document** permettra d'agir sur les élements du site web (le html) on parlera d'intéragir avec le DOM (document object model).
https://www.w3schools.com/jsref/dom_obj_document.asp

# Intéragir avec votre page  les selecteurs

Pour interagir avec les éléments du DOM il va falloir aller les sélectionner, plusieurs méthodes sont possibles :

```
let header    = document.getElementsByTagName('header'); // selection d'une ou plusieurs balises
let logo      = document.getElementById('logo'); // selection d'un id
let container = document.getElementsByClassName('container'); // selection d'une ou plusiuers classes

let h1        = document.querySelector('h1'); // super selector qui va selectionner le premier élement h1
let h1        = document.querySelector('.toto'); // super selector qui va selectionner le premier classe toto
let h1        = document.querySelector('#toto'); // super selector qui va selectionner le premier id toto

let p         = document.querySelectorAll('p');// super selector qui va selectionner tout les éléments p
let p         = document.querySelectorAll('.toto');// super selector qui va selectionner toute les classes toto
let p         = document.querySelectorAll('#toto'); // super selector qui va selectionner tout les id toto
```

# Intéragir avec votre page  tp-16-selecteur

Pour inclure nos scripts qui vont intéragir avec le DOM,  nous allons les placer avant la fin de la balise body.

- Je vous invite à chercher pourquoi placer nos balise à cette endroit et pas à un autre endroit.
- Faites un fichier js que vous incluez dans le fichier html que je vais vous donner.
- Créez et console.logguez les variables qui vont afficher : le h2, tout les « a », la div avec la class  « text ».
- Si vous utilisez querySelector et querySelectorAll constatez vous une différence ?

# Intéragir avec votre page  tp-16-selecteur

```
let h2 = document.querySelector("h2");
let allA = document.querySelectorAll("a");
let text = document.querySelector(".text");


console.dir(h2);
console.log(allA);
console.log(text);

// console.dir pour avoir toutes les infos du noeud sur un query
selector.
```

# Intéragir avec votre page  tp-17-modifier du texte

Maintenant que vous savez sélectionner des éléments vous allez pour voir les modifier.

- A l'aide de js modifier le texte du h2.
- A l'aide de js modifier le texte uniquement du deuxième a.

# Intéragir avec votre page  tp-17-modifier du texte

```
let links = document.querySelectorAll("a");

let titre = document.querySelector("h2");

titre.textContent = "titre modifier en js";

links[1].textContent="modifier en js";
```

# Intéragir avec votre page  Ajouter / Supprimer des éléments

**append =** Ajouter un élément après le dernier enfant de l'élément selectionné. Ex :

```
document.body.append("coucou");
```

**createElement =** Créer un élément, il faudra d'abord le créer et ensuite le personnaliser avant de l'ajouter au dom (append). Ex :

```
let coucou = document.createElement("div");
coucou.innerText = "coucou";
document.body.append(coucou);
```

**Prepend =** Permet d'ajouter un élément avant le premier enfant de celui-ci .
Ex :

```
document.querySelector("h2").prepend(coucou);
```

**Remove =** Vous l'avez compris, cela permet de supprimer un élément. Ex :

```
coucou.remove();
```

# Intéragir avec votre page  tp-18-Ajouter une div

- Ajouter un h1 avant la nav dans une div
- Supprimer l’image

# Intéragir avec votre page  tp-19-Changez le style

- Enlevez tout les scripts ajouté plus tôt.
- Modifiez le style du body en js pour mettre un background gradient plutôt que le bleu de base.

# Intéragir avec votre page  tp-19-Changez le style

Pour changer le style d'un élément, ici je rajoute une classe qui aura été définie dans le css. De ce fait en changeant la classe on change aussi le style :

```
document.querySelector("body").className="gradientBody";
```

Il est possible de changer directement le style d'un élément comme ceci :

```
document.querySelector('h1').style.color = '#111d5e';
```

# DOM Memento

Voici une fiche technique résumant l'ensemble des fonctions et propriétés principales liées au DOM.
Accéder aux éléments
**getElementsByTagName()** - Sélectionne tous les éléments avec la balise entre parenthèses
**getElementById()** - Sélectionne un seul élément : le premier ayant l'ID entre parenthèses
**getElementsByClassName()** - Sélectionne tous les éléments avec la classe entre parenthèses
**querySelector()** - Sélectionne un seul élément : celui avec le sélecteur entre parenthèses
**querySelectorAll()** - Sélectionne tous les éléments avec le sélecteur entre parenthèses

Modifier les éléments
**textContent** - Modifie le texte d'un élément
**innerHTML** - Modifie l'HTML d'un élément

Ajouter et supprimer des éléments
**createElement()** - Crée un élément
**prepend()** - Ajoute l'élément entre parenthèses avant le premier enfant
**append()** - Ajouter l'élément entre parenthèses derrière le dernier élément enfant
**appendChild()** - Ajouter l'élément entre parenthèses derrière l'élément cible (ne peut pas contenir du texte)
**insertBefore()** - Insère un élément avant l'élément cible

Modifier le style d'un élément
**style.propriété** - Modifie la propriété CSS spécifiée, par exemple : style.color = "orange"
**className** - Modifie les classes d'un élément

# Evènement

Pour interagir avec le DOM lorsque il y a un évènement provoqué par un utilisateur, comme un click ou le passe de la souris à un endroit de l'écran … Il y aura 3 possibilité :

La première ajouter l'évènement directement dans le html (à éviter aujourd'hui) : `<a onclick=`

La deuxième consiste à créer les évenements comme onclick dans le fichier js

```
a.onclick = () => {
    if(confirm('Etes-vous sûr ?')) {
      location.href=« https://www.google.fr"
    }
}
```

La troisième consiste à utiliser les « addEventListener »

# Evènement tp-20

- Créez une page html avec un lien et un bouton
- A l'aide de addEventListener créez un évènement pour supprimer le lien en cliquant dessus.
- Changez le background color du body en survolant le bouton et faites qu'il redevienne blanc quand la souris quitte le bouton.
- Créez une section avec un h1 dedans et un texte dans la section mais en dehors du h1.
- Créez un évènement au clique du h1 qui crée une alert disant que vous cliquez sur le h1 et de même pour la section. Que constatez vous ?

# Evènement  tp-20

```
let a = document.querySelectorAll("a");
let bouton = document.querySelector("button");
let body = document.querySelector('body');
let h1 = document.querySelector("h1");
let section = document.querySelector("section");


a[0].addEventListener("click", () => {
    a[0].remove();
});

a[1].addEventListener("click", () => {
    clearInterval(interval);
});

bouton.addEventListener("mouseover", () => {
    body.style.backgroundColor = "blue";
});
bouton.addEventListener("mouseout", () => {
    body.style.backgroundColor = "unset";
});
```

L’avantage d’utiliser « addEventListener » est de pouvoir supprimer un évènement avec removeEventListener().

# Evènement  tp-20

Pensez à arrêter la propagation des évènements. C’est l’héritage d’un évènement de l’enfant vers ses parents.

```
section.addEventListener("click", (e) => {
    alert("vous avez cliqué sur la section");
    e.stopPropagation();
});

h1.addEventListener("click", (e) => {
    alert("vous avez cliqué sur le h1");
    e.stopPropagation();
});
```

# Evènement  tp-20-2

Il est possible de planifier l'execution des scripts en JS :

- setInterval permet d'executer une fonctions à intervalles correspondantes à un temps donné
- setTimeout permet d'executer une fonction après un certain temps.

- Créez une fonctions permettant de générer une couleurs aléatoirement.
- Essayez de faire changer la couleur d'un bouton toutes les 2 secondes à l'aide de setInterval et de votre fonction.
- Créez un lien permettant d'arrêter ce changement de couleur.

# Evènement  tp-20-2

```
let changeButton = () =>{
    let randomRgba = () =>{

        let round = Math.round, random = Math.random, color = 255;
        let result = 'rgba(' + round(random()*color) + ',' + round(random()*color) + ','
+ round(random()*color) + ',' + random().toFixed(1) + ')';
        console.log(result);
        return result;

    }
    bouton.style.backgroundColor = randomRgba();
}
let interval = setInterval(changeButton, 2000);

a[1].addEventListener("click", () => {
    clearInterval(interval);
});
```

# Evènement Memento

Voici une fiche technique résumant l'ensemble des évènements principaux.

Les écouteurs "on" et les propriétés JavaScript
•**onfocus** - Quand l'utilisateur sélectionne l'élément
•**onchange** - Quand l'utilisateur change la valeur de l'élément
•**onclick** - Quand l'utilisateur clique sur l'élément
•**ondblclick** - Quand l'utilisateur double-clique sur l'élément
•**onkeypress** - Quand l'utilisateur appuie sur une touche du clavier dans l'élément

Les évènements avec addEventListener
•**click** - Quand l'utilisateur clique sur l'élément
•**mouseover** - Quand l'utilisateur passe avec sa souris au-dessus d'un élément
•**mouseout** - Quand l'utilisateur sort avec sa souris d'un élément
•**copy** - Quand l'utilisateur copie un élément
•**cut** - Quand l'utilisateur coupe un élément
•**paste** - Quand l'utilisateur colle un élément

Pour retrouver tous les évènements possibles avec JavaScript: https://developer.mozilla.org/fr/docs/Web/Events

# Tp-21 timer

Dans cet exercice, vous allez réaliser un Timer !

Nous allons pouvoir réviser les notions de conditions, de fonctions, et d'intervalles.

**Voici comment va se dérouler notre Timer :**
- Un bouton s'affiche sur la page proposant de « **Lancer le décompte** »
- L'utilisateur clique sur ce bouton, qui va lancer une fonction **start()**, qui s'occupera de créer un intervalle sur la fonction **decompte()** toutes les secondes, pendant 10 secondes.
- La fonction **decompte()**, lorsqu'elle est appelée :
    - Décrémentera (retirera 1) à une variable secondes, qui contient comme valeur 10
    - Vérifiera si secondes est différent de 0, pour afficher dans la page le nombre de secondes restantes
    - Sinon, si secondes vaut 0, appellera la fonction stop(), qui aura pour but d'arrêter l'intervalle

# Tp-21 timer

```
let btn = document.querySelector('button');
let secondes = 10;
let interval;

btn.addEventListener('click', ()=>{
  start();
  disappear();
});

function disappear(){
  btn.style.display ="none";
}

function start() {
  interval = setInterval(decompte, 1000);
}

function stop() {
  clearInterval(interval);
  document.body.innerHTML += "Boum !";
}

function decompte() {

  secondes--;
  if(secondes == 0) {
    stop();
  }
  else {
    document.body.innerHTML += secondes + '<br>';
  }

}
```

# Tp-22 spoiler

Dans cet exercice, vous allez réaliser un bouton Spoiler !

Nous allons pouvoir réviser les notions de conditions, et de styles.

**Voici comment va se dérouler notre bouton Spoiler :**
- Un bouton s'affiche sur la page proposant d'afficher le message
- L'utilisateur clique sur le bouton, ce qui va activer une condition
    - La variable *hidden* vaut *true* ? Dans ce cas, on affiche le message, on change le texte du bouton en « Cacher », et on passe la variable *hidden* en *false*
    - La variable *hidden* vaut *false* ? Dans ce cas, on cache le message, on change le texte du bouton en « Afficher », et on passe la variable *hidden* en *true*

# Tp-22 spoiler

```
// Sélectionner nos éléments
let btn       = document.querySelector('button');
let div       = document.querySelector('div');
let hidden    = true;

// Cacher le message
div.className = "hidden"

// Détecter le clic
btn.addEventListener('click', () => {

  if(hidden) {
    btn.textContent = "Cacher";
    div.className = "visible";
    hidden = false;
    console.log("false");
  }
  else {
    btn.textContent = "Afficher";
    div.className = "hidden";
    hidden = true;
    console.log("true");
  }

});
```

# eval-23- Générateur de citations

Étape 1 :

- Créez le fichier HTML (attention à la sémantique des balises)
- Créez le fichier Css pour avoir un visuel le plus proche possible du mien. (font = satisfy)

# eval-23- Générateur de citations

Étape 2 :

- Créez les variables nécessaires  au projet, prenez 5 minutes pour réfléchir à un algorithme, cela vous donneras une idée de vos besoins.

- Allez chercher tout les éléments à modifier dans le DOM et stockez les également.

- Créez un tableau avec vos citations préférées ainsi que l’auteur correspondant. (une dizaine)

# eval-23- Générateur de citations

Étape 3 :

- Créez un évènement sur le clique du bouton « nouvelle citation ».
- Créez une fonction permettant d'aller piocher une citation aux hasard dans votre tableau
- Créez une fonction permettant d'actualise la citation et l'auteur.
- **Je vous laisse penser à un système pour qu'une citation ne sorte pas deux fois de suite.**

- Vous devriez avoir toutes les armes pour avancer, renseignez vous bien sur les fonctions mathématiques, elles pourront vous être utiles.
- Vous pourriez avoir besoin d'une boucle à un endroit et de connaitre la longueur de votre tableau.

J'en dis pas plus sinon c'est pas drôle, je vous donne la correction sur un zip et non dans le cours, comme ça vous avez tout.

# eval-24- Juste prix

Etape 1 - **Sélectionner nos éléments**

Etape 2 - **Cacher l'erreur**

Etape 3 - **Générer un nombre aléatoire**

Etape 4 - **Vérifier que l'utilisateur donne bien un nombre**

Etape 5 - **Agir à l'envoi du formulaire**

Etape 6 - **Créer la fonction vérifier**

Pour ceux qui ne sont pas familiers avec ce jeu, il s'agit de deviner le prix d'un produit. Ici, trois possibilités :

- C'est plus cher
- C'est moins cher
- C'est le juste prix !

Dans cette version du "Juste prix", nous générerons un prix aléatoire, et nous calculerons le nombre de coups qu'il faut à un joueur pour retrouver le juste prix.
Ainsi, il pourra se confronter à ses amis pour tenter de voir qui retrouve le bon prix avec le moins de coups possibles !
Je vous attends dans la prochaine session !

# eval-23 et 24 - Correction

Je vous envoi le zip avec le html-css et js des deux TP