Chapitre 18 - JavaScript avec html et css

Dans le chapitre précédent, nous avons découvert la syntaxe du langage JavaScript en faisant la correspondance entre JS et Python. Nous avons manipulé des structures conditionnelles, des boucles, des fonctions, des listes et des dictionnaires. Dans ce nouveau chapitre, l'objectif est à présent d'utiliser JS pour manipuler directement une page HTML afin la rendre interactive. Pour cela nous apprendrons :

- A récupérer un élément html d'une page web pour le placer dans une variable JS.
- A modifier le contenu ou le css de cet élément.
- A créer des évènements souris ou clavier qui entraineront l'exécution d'une fonction JS.

Les techniques présentées seront appliquées à une page web, qui permet après saisie d'un prix HT et d'un taux de TVA, de calculer le prix TTC. L'affichage de la page dans un navigateur, ses codes html et css sont donnés ci-dessous :

```html
<fieldset>
    <legend><em>Calculer le prix TTC</em></legend>
    <div class="saisie">
        <label for="ht">Prix Ht : </label>
        <input type="text" placeholder="Prix HT" id="ht">
        <label for="tva">Taux de TVA : </label>
        <input type="text" placeholder="Taux TVA" id="tva">
        <input type="button" value="Calculer" id="bouton">
        <img src="raz.png">
    </div>
    <p class="resultat">Prix TTC : <span></span></p>
</fieldset>
```

```css
fieldset {
    width:300px;
    margin:20px auto;
    border-radius: 20px;
    background-color: #c2e7bc ;
}
.saisie{
    display: grid;
    grid-template-columns: 1fr 2fr;
    gap:10px;
}
#bouton , #raz{
    justify-self: center;
    align-self:center;
    padding:5px;
}
```

1- RECUPERER UN ELEMENT HTML :

Partie Cours :

- On repère un élément html par l'intermédiaire d'un sélecteur CSS.
- Au chargement de la page, son contenu est stocké dans une variable OBJET nommée .
- En écrivant `s = document.querySelector(".saisie")`, la variable `s` récupère l'élément *html* repéré par le sélecteur Css `.saisie`

Par exemple, dans notre exemple, si on définit la variable

`s = document.querySelector(".saisie")`,

en rappelant `s` dans la console, le bloc

`<div class="saisie">` est repéré sur la page.

Cette variable `s` contient un objet qui possède ce que l'on appelle des .

La liste de tous les attributs peut être consultée dans la console :

```
» s
← ▾ <div class="saisie">
        baseURI: "http://localhost:8080/coursJsTva.html"
        childElementCount: 6
      ▸ childNodes: NodeList(13) [ #text , label , #text , … ]
      ▸ children: HTMLCollection { 0: label , 1: input#ht , length: 6, … }
      ▸ classList: DOMTokenList [ "saisie" ]
        className: "saisie"
        clientHeight: 123
```

Ces attributs permettent d'avoir accès à toutes les informations liées à ce bloc div. Par exemple :

» s.clientHeight ← 123 Donne la hauteur en px du bloc repéré.	» s.childNodes ← ▸ NodeList(13) [#text , label , #text , input#ht , #text , Donne la liste de tous les éléments **enfants** de `s` : `s.childNodes[3]` donne le contenue du 4ième élément de cette liste : » s.childNodes[3] ← ▸ <input id="ht" type="text" placeholder="Prix HT">

Remarque : Il existe d'autres méthodes que l'on peut appliquer sur l'objet *document*, et qui permettent de récupérer de la même façon :

- Un ou plusieurs éléments définis par un sélecteur de type **class** :

```
>> s = document.querySelectorAll(".saisie")
<- ▶ NodeList [ div.saisie ]
>> s[0]
<- ▶ <div class="saisie">
```

- Un ou plusieurs éléments définis par un sélecteur de type **class** :

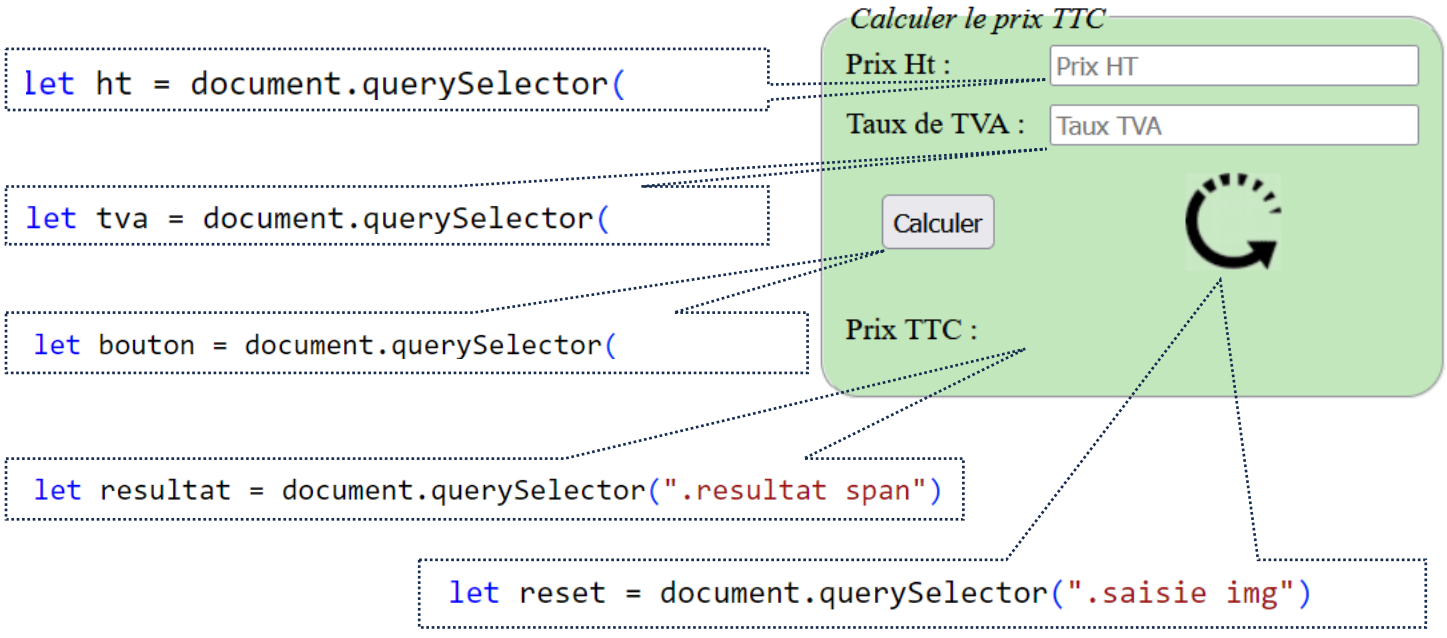
```
s = document.getElementsByClassName("saisie")
```

```
>> s
<- ▶ HTMLCollection { 0: div.saisie , length: 1 }
>> s[0]
<- ▶ <div class="saisie">
>> s.length
<- 1
```

- Un élément défini par un sélecteur de type **id** :

```
>> ht = document.getElementById("ht")
<- ▶ <input id="ht" type="text" placeholder="Prix HT">
```

Exercice : Compléter ci-dessous les lignes qui permettent de récupérer les 2 champs de saisi, le bouton, l'image et le du paragraphe de class *resultat*, respectivement dans des variables nommées ht, tva, bouton, reset, resultat :



Partie Cours : Pour récupérer ce que l'utilisateur a écrit dans un champ de saisi, on récupère la valeur de l'attribut *value* :

```
let prixHt = ht.value
let tauxTva = tva.value
```

2- Modifier un contenu de type « texte », dans un element Html :

Partie Cours :

- Pour modifier le contenu d'un élément de type « *texte* » : *textContent*

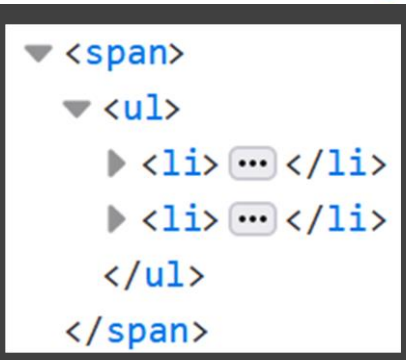
```
<span></span>
```
avec
```
resultat.textContent = "bonjour"
```

donne
```
<span>bonjour</span>
```

- Pour modifier le contenu d'un élément de type « *texte* » : *innerHtml*

```
<span></span>
```
avec
```
resultat.innerHTML = "<ul><li>bonjour</li><li></li></ul>"
```

donne :



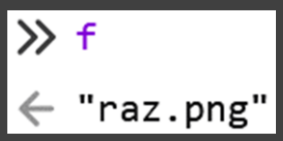
3- Recuperer ou modifier l'attribut d'un element Html :

Partie Cours :

- Pour **récupérer** le nom du fichier image lié à une image : *getAttribute()*

```
<img src="raz.png">
```
avec
```
f = reset.getAttribute("src")
```

donne pour la variable f :



- Pour **modifier** le fichier image lié à une image : *setAttribute()*

```
<img src="raz.png">
```
avec
```
reset.setAttribute("src","gouv.png")
```

donne
```
<img src="gouv.png">
```

4- MODIFIER LE CSS D'UN ELEMENT HTML :

Partie Cours : exemple d'application

- Modifier la couleur d'arrière plan :

```
ht.style.backgroundColor="#f00"
```

- Modifier la marge intérieure :

```
ht.style.padding="5px"
```

- Modifier la bordure :

```
ht.style.borderRadius="10px"
```

Dans la console, on peut visualiser toutes les propriétés Css liées à un élément :

```
>> ht.style
← ▶ CSS2Properties(9) { "background-color" → "rgb(255, 0, 0)",
"padding-top" → "5px", "padding-right" → "5px", "padding-bottom" →
"5px", "padding-left" → "5px", "border-top-left-radius" → "10px",
"border-top-right-radius" → "10px", "border-bottom-right-radius" →
"10px", "border-bottom-left-radius" → "10px" }
```

En écrivant, toujours dans la console, par exemple :

```
>> a = ht.style.borderBottomLeftRadius
← "10px"
```

La variable « a » prend ici la valeur de la propriété Css `"border-bottom-left-radius" → "10px"`

5- EVENEMENTS SOURIS OU CLAVIER :

Partie Cours : Les évènements ci-dessous entrainent l'exécution de la fonction CALLBACK « calculer »

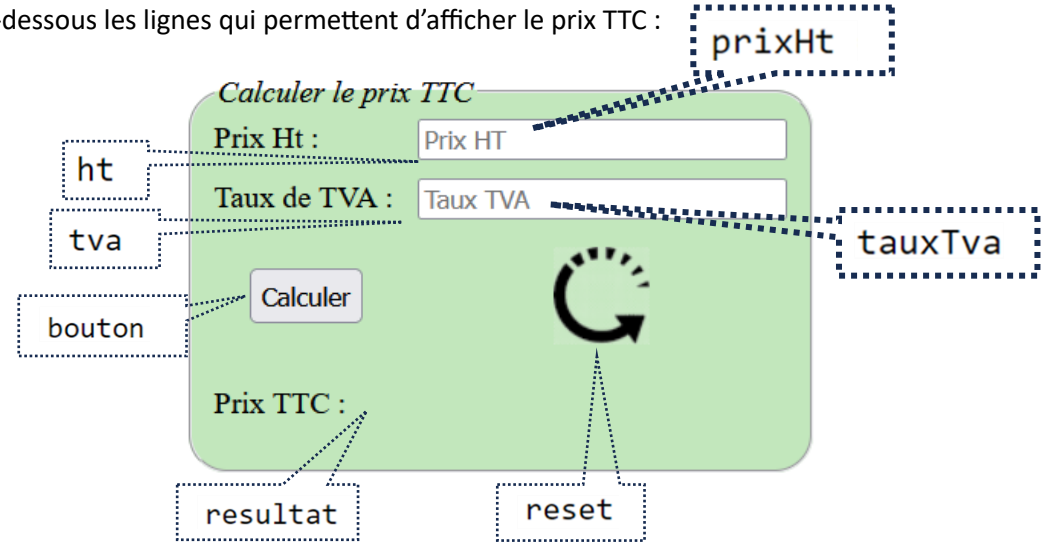
- Evènement click souris sur le bouton récupéré par la variable `bouton` :

```
bouton.addEventListener("click",calculer)
```

- Les noms des évènements souris sont : `mousemove` , `mouseover/mouseout`, `mousedown/mouseup`, `dblclick`

- Les noms des évènements claviers sont : `keydown` ou `keyup`. `document.addEventListener("keydown",calculer)`. La méthode *addEventListener* est alors appliquée sur l'objet « *document* ». Les informations données par la variable « *event* » permettent de voir laquelle des touches a été actionnée.

Exercice : Compléter ci-dessous les lignes qui permettent d'afficher le prix TTC :

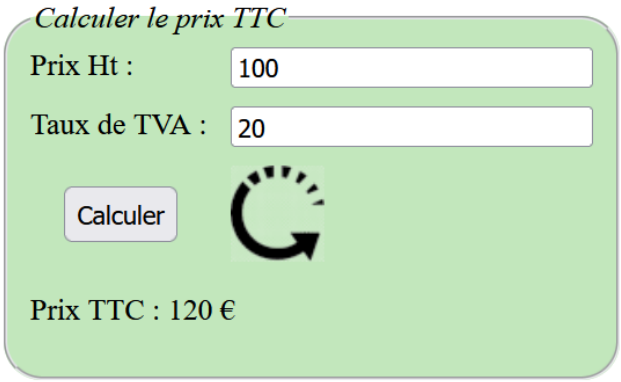



```
<script>
    let ht = document.querySelector("#ht")
    let tva = document.querySelector("#tva")
    let bouton = document.querySelector("#bouton")
    let reset = document.querySelector(".saisie img")
    let resultat = document.querySelector(".resultat span")
    ht.value = ""
    tva.value = ""

    function calculer(event){
        console.log(event,this)
        let prixHt =
        let tauxTva =
        let prixTtc = parseFloat(prixHt) * (1+parseFloat(tauxTva) / 100)
        resultat.textContent =
    }
    bouton.addEventListener("click",calculer)

</script>
```

Exemple d'exécution :



Que se passe-t-il dans la console ? :

Lorsque l'évènement est déclenché :

- un objet nommé « *event* » est créé. Il fournit des informations intéressantes qui peuvent être utilisées dans la fonction.

- Une variable nommée « this » est créée. Elle contient l'élément html sur lequel l'évènement a été est créé.

```
click { target: input#bouton , buttons: 0, clientX: 475, clientY: 141, layerX: 475, layerY: 141 }
<input id="bouton" type="button" value="Calculer">
```

On peut rajouter d'autres évènements afin d'améliorer le fonctionnement :

```
bouton.addEventListener("click",calculer)

document.addEventListener("keydown",calculer)

tva.addEventListener("change",calculer)

ht.addEventListener("change",calculer)
```

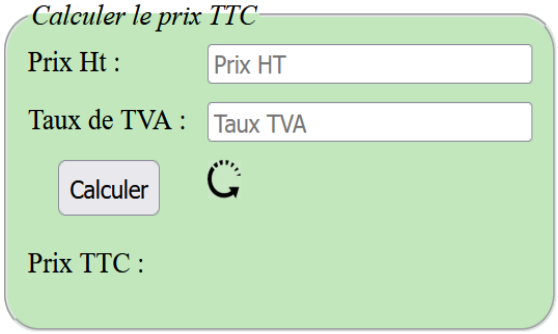
Objet « event » après une action sur la touche "a" du clavier :

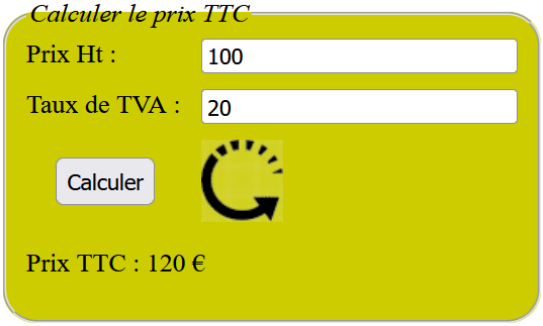
```
keydown { target: body , key: "a", charCode: 0, keyCode: 65 }
```

6- EXERCICE :

Compléter le code Js de la page suivante afin :

- d'avoir au départ une largeur d'image de 20px,
- d'obtenir après calcul du prix TTC, l'élément <fiedset> qui change de couleur (#CCCC00) et la largeur de l'image qui augmente à 50px,
- de créer un évènement click sur l'image qui permette de revenir à l'état initial : couleur, taille image et champs de saisies vident.





```
<script>
    let ht = document.querySelector("#ht")
    let tva = document.querySelector("#tva")
    let bouton = document.querySelector("#bouton")
    let reset = document.querySelector(".saisie img")
    let resultat = document.querySelector(".resultat span")
    let fieldset =
    ht.value = ""
    tva.value = ""
    reset.

    function calculer(event){
        console.log(event,this)
        let prixHt = ht.value
        let tauxTva = tva.value
        let prixTtc = parseFloat(prixHt) * (1+parseFloat(tauxTva) / 100)
        resultat.textContent = String(prixTtc) + " €"
        fieldset.
        reset.

    }
    function raz(event){
        ht.
        tva.
        resultat.textContent = ""
        fieldset.
        reset.
    }

    bouton.addEventListener("click",calculer)
    reset.

</script>
```