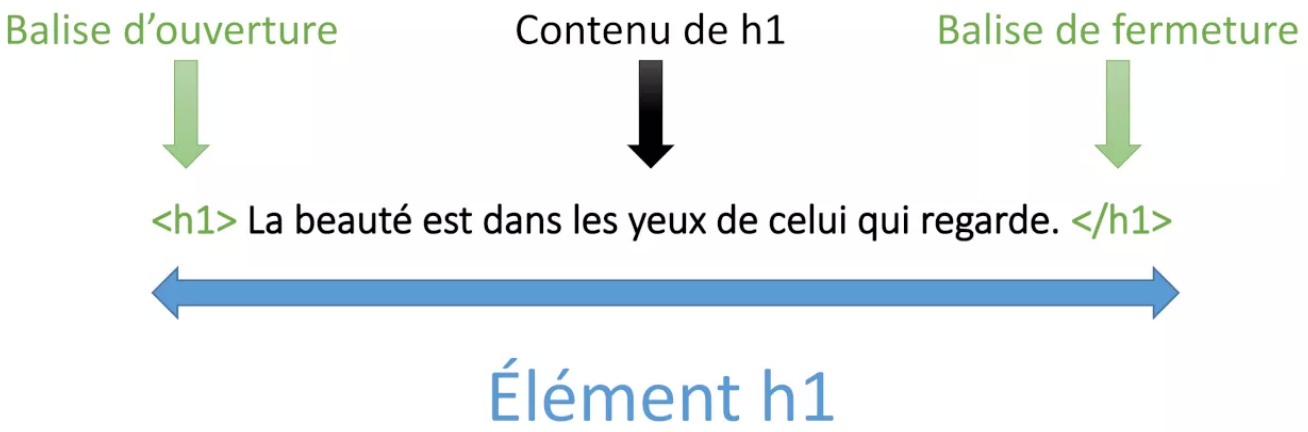
Les sélecteurs JavaScript

Le langage JavaScript nécessite de sélectionner certains éléments de la page Web pour en modifier le contenu, la mise en forme, ou y greffer des événements. Plusieurs techniques de sélection sont à notre disposition.

Avant tout chose, il convient de clarifier le vocabulaire, car les *balises* et les *éléments* ne désignent pas tout à fait la même chose :



Supposons le code HTML suivant :

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <title>Sélecteur JavaScript</title>
    <meta charset="utf-8">
</head>

<body>
    <p id="pres">Cher Guillaume,</p>
    <p class="texte">J'adore le JavaScript.</p>
    <p class="texte">Voilà, c'est dit.</p>
</body>

</html>
```

Pour sélectionner le premier élément <p>, on peut utiliser la « traditionnelle » méthode *getElementById* :

La méthode getElementById()

La méthode *getElementById()* sélectionne un élément par la valeur de son attribut *id*. Etant donné que le duo *id= »valeur »* est unique dans une page, le JavaScript peut agir sur un endroit précis sans risque de confusion.

Syntaxe

```
document.getElementById("nom")...
```

- *document* représente la page HTML, plus précisément l'intérieur de la fenêtre du navigateur, sous la barre d'adresse.
- *nom* désigne la valeur de l'attribut *id* d'une balise d'ouverture située quelque part dans la page.

Pour colorer en rouge le texte du premier élément <p>, on peut écrire :

```
document.getElementById("pres").style.color = "red";
```

Pour y écrire un texte :

```
document.getElementById("pres").innerText = "Bonjour à tous";
```

Et y greffer un évènement :

```
document.getElementById("pres").addEventListener("click", function () {
    alert("vous avez cliqué !");
});
```

La méthode *getElementById* est donc bien pratique, à condition d'avoir positionné au préalable un id dans le code HTML. Si vous ne souhaitez pas alourdir votre code HTML en mettant des id partout, optez pour ces autres méthodes :

La méthode getElementsByTagName()

La méthode *getElementsByTagName* sélectionne un ou plus éléments grâce au nom de la balise. Notez le « s » à *Elements*, car les balises peuvent être évidemment multiples dans une page. Cette méthode génère une liste de balises qui se gère comme une variable de type tableau. Pour colorer le texte du premier élément <p>, on mentionne l'indice [0] :

```
document.getElementsByTagName("p")[0].style.color = "red";
```

Pour le deuxième élément <p>, l'indice [1] :

```
document.getElementsByTagName("p")[1].style.color = "red";
```

Pour l'ensemble des éléments <p>, on peut écrire une boucle *for* avec la propriété *length* qui renvoie la taille du tableau. Dans cet exemple : *document.getElementsByTagName(« p »).length* est égal à 3 :

```
for (let i = 0; i < document.getElementsByTagName("p").length; i++) {
    document.getElementsByTagName("p")[i].style.color = "red";
}
```

ou de manière plus concise :

```
const p = document.getElementsByTagName("p");

for (let i = 0; i < p.length; i++) {
    p[i].style.color = "red";
}
```

La méthode getElementsByClassName()

De la même manière, on peut sélectionner une ou plusieurs classes avec la méthode *getElementsByClassName*. Là encore, cette méthode se gère à la manière d'un tableau.
Le code ci-dessous va colorer en rouge tous les éléments <p> qui ont la classe « texte » :

```
const texte = document.getElementsByClassName("texte");

for (let i = 0; i < texte.length; i++) {
    texte[i].style.color = "red";
}
```

La méthode querySelector()

La méthode *querySelector()* utilise un **sélecteur CSS**, ce qui est très pratique quand vous connaissez le langage CSS. S'il y a des éléments multiples, elle sélectionnera le premier.

Ce code colore en rouge le premier élément <p> :

```
document.querySelector("p").style.color = "red";
```

Ce code colore en rouge la première classe *texte*, donc le deuxième élément <p> :

```
document.querySelector(".texte").style.color = "red";
```

Ce code colore en rouge l'élément <p> dont l'attribut id a la valeur *pres* :

```
document.querySelector("#pres").style.color = "red";
```

La méthode querySelectorAll()

La méthode *querySelectorAll()* sélectionne plusieurs éléments en utilisant des **sélecteurs CSS**. Cette méthode génère une *nodelist* qui se gère comme une variable de type tableau. Si l'on souhaite colorer le troisième élément <p>, on écrit ceci :

```
document.querySelectorAll("p")[2].style.color = "red";
```

Si je souhaite colorer tous les éléments <p>, on peut écrire :

```
for (let i = 0; i < document.querySelectorAll("p").length; i++) {
    document.querySelectorAll("p")[i].style.color = "red";
}
```

ou de manière plus concise :

```
const ListeP = document.querySelectorAll("p");

for (let i = 0; i < ListeP.length; i++) {
    ListeP[i].style.color = "red";
}
```

On peut aussi utiliser la boucle *forEach* qui évite l'usage de la propriété *length* :

```
const ListeP = document.querySelectorAll("p");
ListeP.forEach(function (element) {
    element.style.color = "red";
});
```