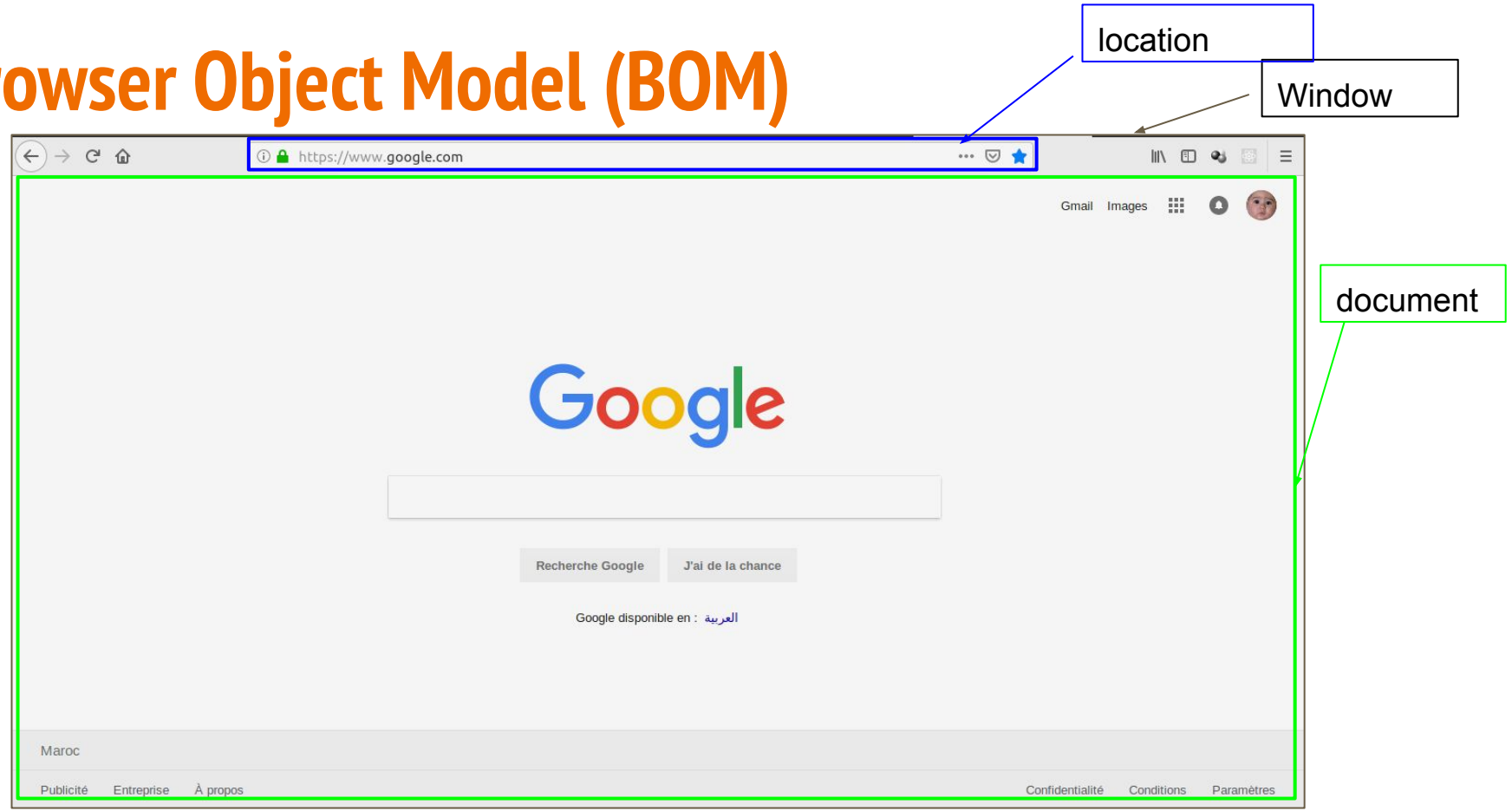
Javascript

DOM: Document Object Model

S.Gounane

GI 2019-2020

Browser Object Model (BOM)



Browser Object Model (BOM)

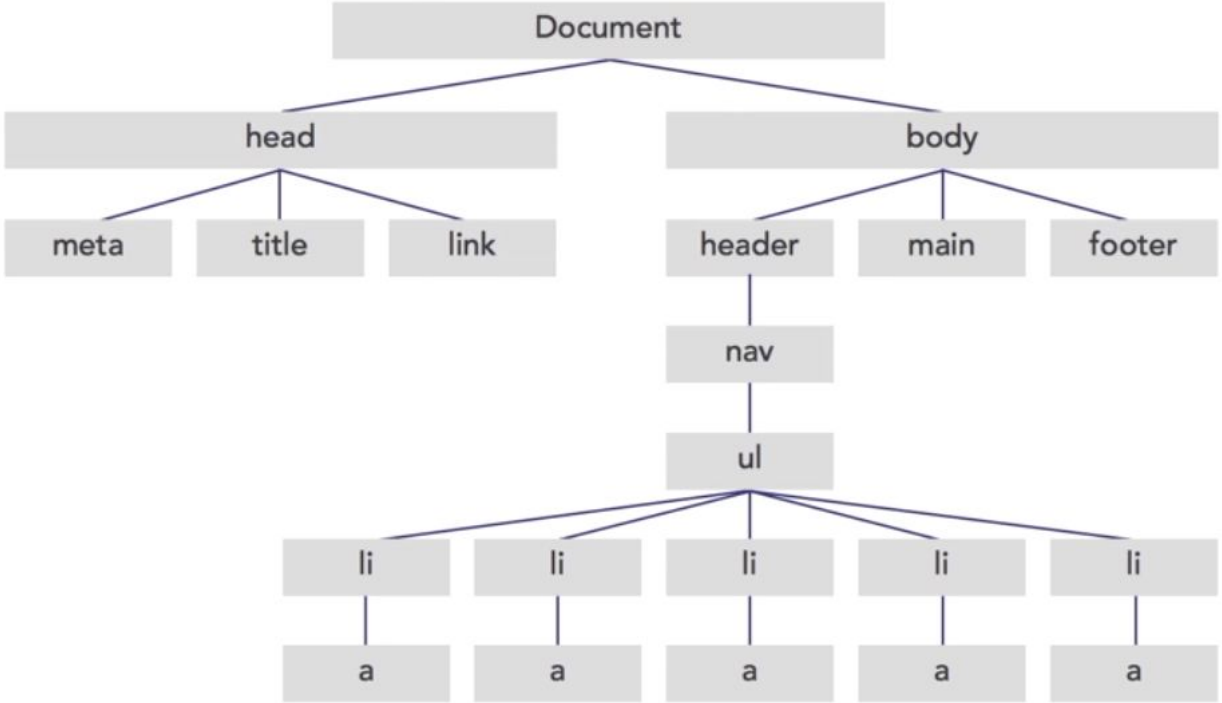
Window est l’objet du plus haut niveau dans le BOM et possède un ensemble de propriétés et méthode pour interagir avec le Browser

- window.innerWidth
- window.open()
- window.location
- Window.**document**
- **....**

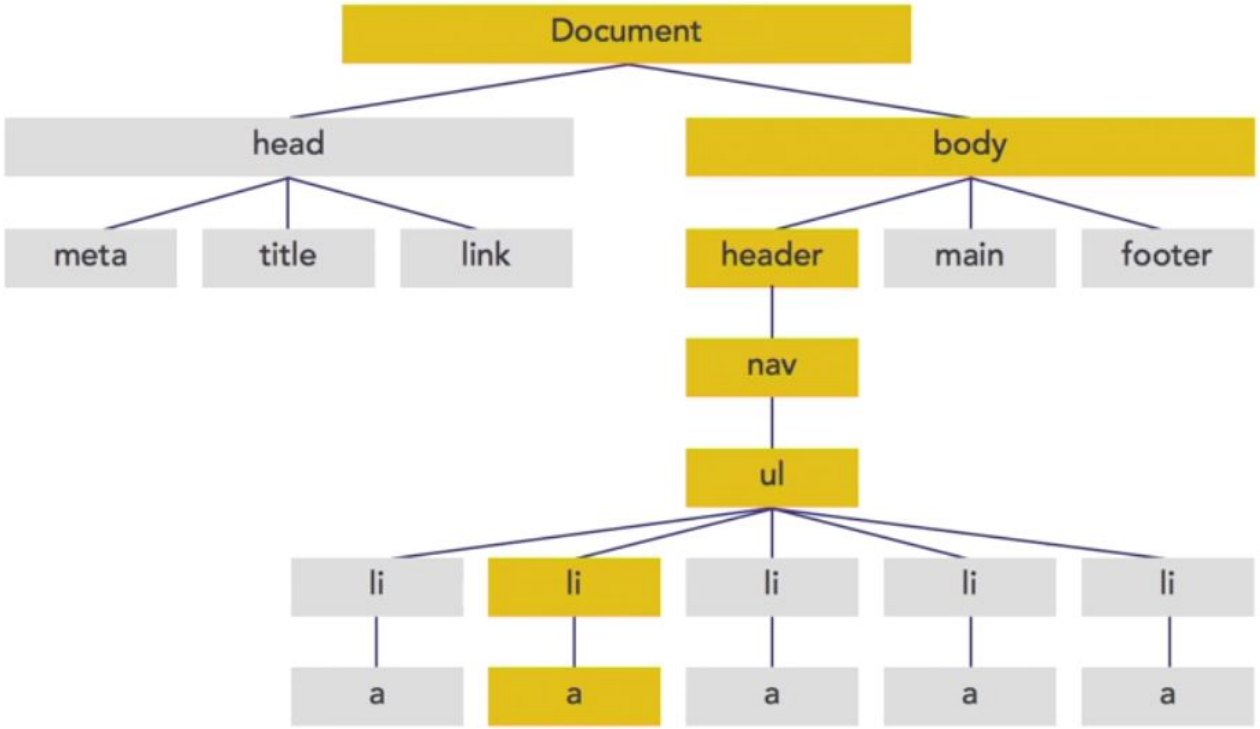
Le Document Object Model (DOM)

- Le DOM est une API qui s'utilise avec les documents HTML, et qui va nous permettre, via le JavaScript, d'accéder au code HTML d'un document.
- C'est grâce au DOM que nous allons pouvoir:
    - ***modifier des éléments HTML (afficher ou masquer un par exemple),***
    - ***ajouter des éléments HTML***
    - ***déplacer des éléments HTML***
    - ***Supprimer des éléments HTML***

Le Document Object Model (DOM)



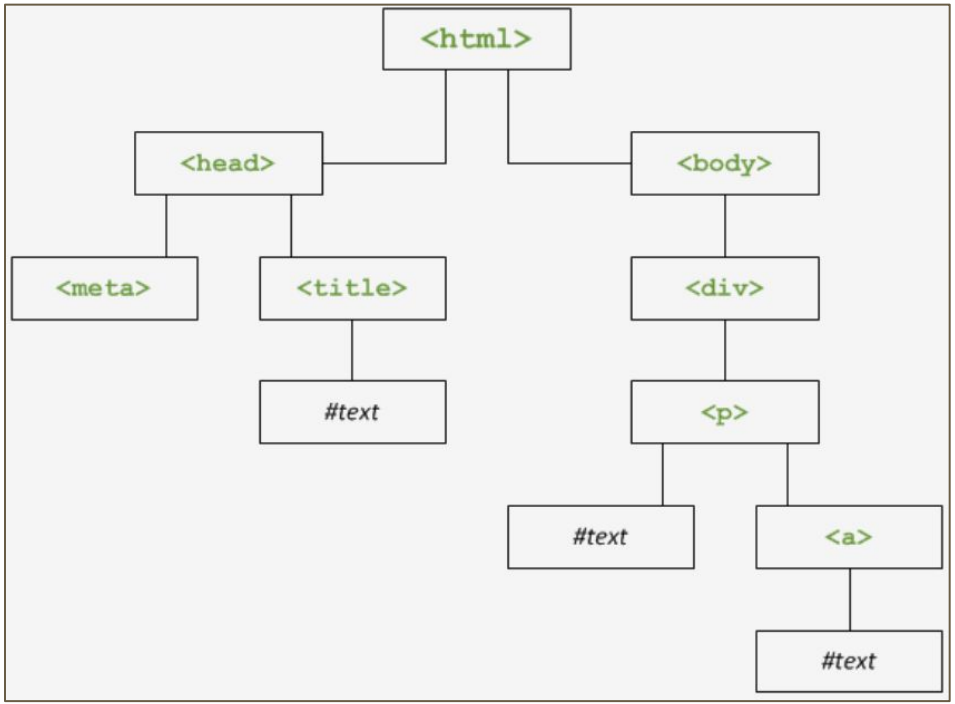
Le Document Object Model (DOM)



Le Document Object Model (DOM)

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Le titre de la page</title>
</head>

<body>
    <div>
        <p>Un peu de texte <a>et un lien</a></p>
    </div>
</body>
</html>
```



Accéder aux éléments

- document.body
- Document.title
- document.URL
- document.getElementById(“#id”),
- document.getElementsByTagName(“tagName”)
- document.getElementsByClassName(“.class”)
- document.getElementsByName(“name”)
- ...

Accéder aux éléments

document.querySelector("css selector")

=> Retourne Le premier élément qui vérifie le sélecteur css spécifier

document.querySelectorAll("css selector")

=> Retourne Tous les éléments vérifiant le sélecteur CSS

Ces deux méthodes sont les plus utilisées

Accéder aux éléments

1. querySelector()
2. querySelectorAll()

```html
<div id="menu">

    <div class="item">
        <span>Élément 1</span>
        <span>Élément 2</span>
    </div>

    <div class="publicite">
        <span>Élément 3</span>
        <span>Élément 4</span>
    </div>

</div>

<div id="contenu">
    <span>Introduction au contenu de la page...</span>
</div>
```

```js
var query = document.querySelector('#menu .item span'),
    queryAll = document.querySelectorAll('#menu .item span');

alert(query.innerHTML); // Affiche : "Élément 1"

alert(queryAll.length); // Affiche : "2"
alert(queryAll[0].innerHTML + ' - ' + queryAll[1].innerHTML); // Affiche : "Élément 1
- Élément 2"
```

Accéder aux attributs d'un éléments

Les attributs des élément HTML peuvent êtres accède en lecture/écriture ou en lecture seule

1. element.atributes
2. element.innerHTML
3. element.outerHTML
4. element.clientHight
5. element.className
6. element.classList
7. element.id
8. ...

Accéder aux attributs d'un éléments

Pour modifier les attributs en lecture seul on utilise de methodes:

1. element.classList.add("une nouvelle classe")
2. element.classList.remove("une classe existante")
3. element.classList.contains("une classe")
4. element.haseAttribute(attribut)
5. element.getAttribute(attribut)
6. element.setAttribute(attibut, value) // modifier ou ajouter un attribut
7. element.removeAttribute(attribut)

Exp: ***document.querySellector("a").setAttribut("target","_blank")***

Accéder aux éléments

getAttribute() et setAttribute()

```
<body>
    <a id="myLink" href="http://www.un_lien_quelconque.com">Un lien modifié
dynamiquement</a>

    <script>
        var link = document.getElementById('myLink');
        var href = link.getAttribute('href'); // On récupère l'attribut « href »

        alert(href);

        link.setAttribute('href', 'http://www.siteduzero.com'); // On édite l'attribut
« href »
    </script>
</body>
```

Accéder aux éléments (Elenment.className)

```
3  <head>
4      <meta charset="utf-8" />
5      <title>Le titre de la page</title>
6      <style>
7          .blue {
8              background: blue;
9              color: white;
10         }
11     </style>
12 </head>
13 <body>
14     <div id="myColoredDiv">
15         <p>Un peu de texte <a>et un lien</a></p>
16     </div>
17     <script>
18         document.getElementById('myColoredDiv').className = 'blue';
19     </script>
20 </body>
```

Accéder aux éléments (Elenment.classList)

```
var div = document.querySelector('div');
// Ajoute une nouvelle classe
div.classList.add('new-class');
// Retire une classe
div.classList.remove('new-class');
// Retire une classe si elle est présente ou bien l'ajoute si elle est absente
div.classList.toggle('toggled-class');
// Indique si une classe est présente ou non
if (div.classList.contains('old-class')) {
    alert('La classe .old-class est présente !');
}
// Parcourt et affiche les classes CSS
var result = '';
for (var i = 0; i < div.classList.length; i++) {
    result += '.' + div.classList[i] + '\n';
}
alert(result);
```

Ajout d’un élément au DOM

1. Creer l’element <= document.createElement()
2. Créer le noeud texte de cet élément <= document.createTextNode()
3. Ajouter le noeud texte à l'élément <= document.appendChild()
4. Ajouter l'élément au DOM <= document.appendChild()

Exercice:

Ajouter un élément <***caption>….</caption>*** a l'élément <***figure ..> …</figure>***

```
9       <figure class="mafig">
10          <img class="monImg" src="img.jpeg" alt="un texte
            ">
11      </figure>
```

Ajout d'un élément au DOM

```
1  const fig=document.querySelector(".mafig");
2  const img=fig.querySelector(".monImg");
3  var altTxt=img.getAttribute("alt");
4  var capElmt=document.createElement("figcaption");
5  var capTxt=document.createTextNode(altTxt);
6  capElmt.appendChild(capTxt);
7  fig.appendChild(capElmt);
8  console.log(fig);
```

Une nouvelle methode : .append()

```
1  const fig=document.querySelector(".mafig");
2  const img=fig.querySelector(".monImg");
3  var altTxt=img.getAttribute("alt");
4  var capElmt=document.createElement("figcaption");
5  capElmt.append(altTxt);
6  fig.append(capElmt);
```

Style CSS Inline

Avec l’attribut style on peut ajouter n’import quel propriété CSS à n'importe quel élément.

- ***Element.style;*** => **uniquement** le **Inline** CSS de l'élément {attribut:“value”, …. } mais pas les autre style définie dans des fichier css au dans le head.
- ***Element.style.color=”blue”;***
- ***Element.style.backgroundColor=”yellow”;*** //**backgroundColor** pas **background-color**
- ***Element.style.cssText=”color: blue; backgound-color: yellow; …”***
- ***Element.style.setAttribute(“style”, ”color: blue; backgound-color: yellow; …”);***

Style CSS Inline

Les style CSS inline remplace les style definie dans les feuille de style.

Dans la plupart des cas il vaut mieu définir des règles CSS et gérer les class avec javascripte

Exercice

1. Créer une horloge en utilisant une image SVG à l'aide d’un éditeur de graphiques vectoriel.
2. Créer un fichier css pour modifier son apparence.
3. Écrire un code javascript pour animer cette horloge.