Modifier des éléments du DOM

Les classes de noeuds du `DOM`

Les éléments du `DOM` ont de nombreuses classes.

L'ensemble des éléments ont la classe `Object` qui permet d'utiliser toutes les méthodes des objets `JavaScript`.

L'ensemble des éléments ont également la classe `EventTarget` qui permet d'utiliser des événements comme nous le verrons.

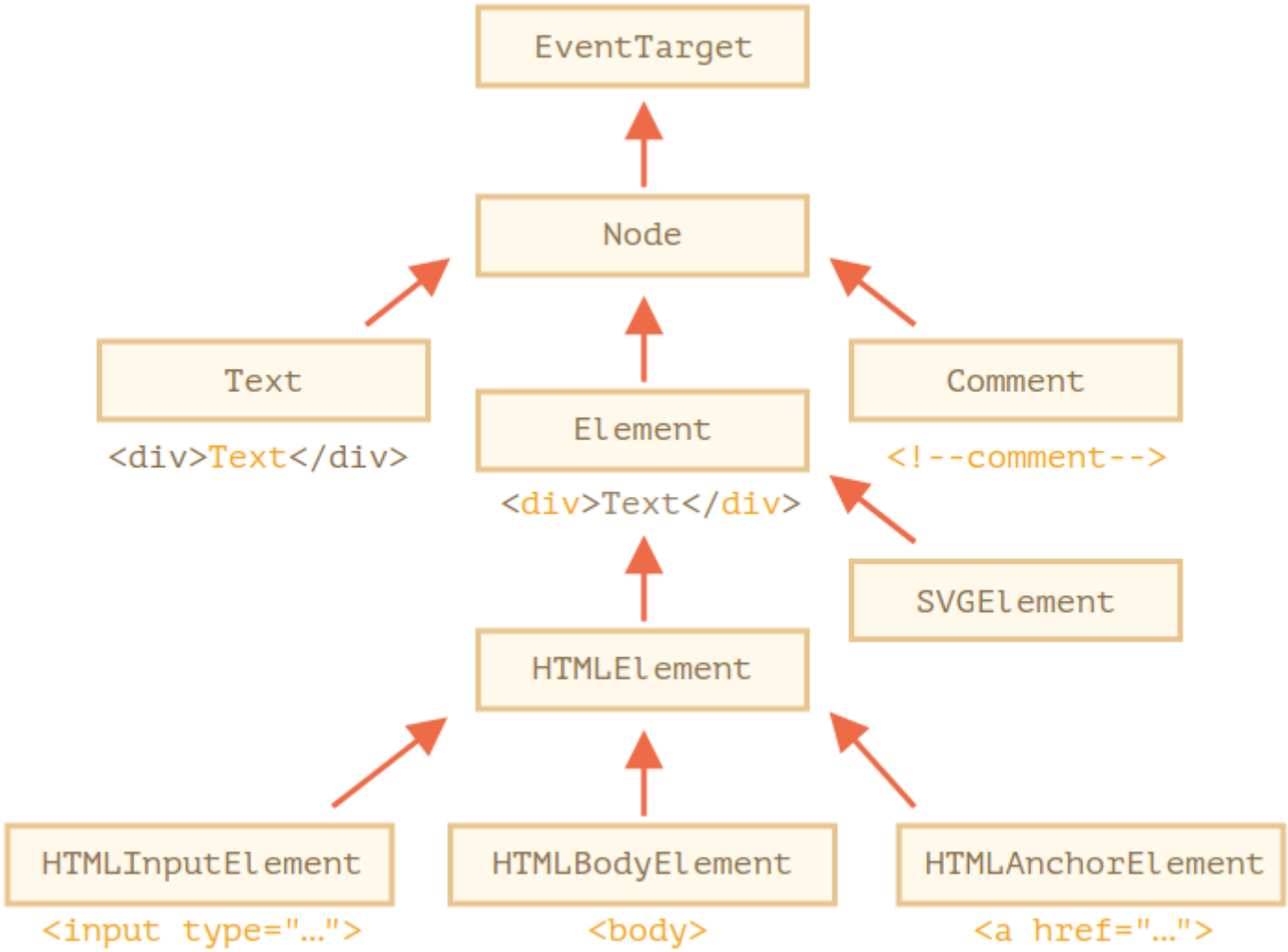
La classe `Node` est aussi commune à tous les éléments et elle hérite de la classe `EventTarget`. Cela signifie que tous les éléments `Node` ont également toutes les propriétés de la classe `EventTarget`.

Cette classe a des propriétés que nous avons vues, comme `parentNode`, `nextSibling` et `childNodes`.

La classe `Element` est la classe qui permet d'utiliser des propriétés comme `children`, `querySelector()` ou `previousElementSibling`.

Ensuite, chaque élément a une classe spécifique contenant les propriétés utiles pour celui-ci.

Par exemple `HTMLAnchorElement` va contenir un ensemble de propriétés nécessaires pour gérer les ancres.



Retenez que les noeuds du DOM sont tous des objets JavaScript qui héritent de plusieurs objets par héritage prototypal.

Autrement dit chaque noeud peut utiliser des propriétés se trouvant sur d'autres objets

Relevez que comme il n'y a pas d'héritage multiple en JavaScript, chaque classe hérite d'une seule classe.

Nous verrons plus tard toutes ces notions en détails plus tard dans la formation.

Accéder au nom du noeud ou de l'élément

Pour accéder au nom d'un noeud il suffit d'utiliser la propriété `nodeName`.

Pour accéder au nom d'un élément, il faut utiliser `tagName`.

Exemples :

```
document.body.tagName; // BODY
document.nodeName; // #document
document.tagName; // undefined car document n’est pas un élément
```

La propriété `innerHTML` des éléments

Les éléments ont une propriété `innerHTML` qui permet de récupérer ou de définir le `HTML` au format chaîne de caractères de l'ensemble des descendants de l'élément.

```
const contenuHTML = element.innerHTML;
```

Ou on peut aussi modifier le contenu :

```
document.body.innerHTML = "<p>Salut ! </p>";
```

La propriété `outerHTML` des éléments

La propriété `outerHTML` permet de récupérer ou de définir le `HTML` au format chaîne de caractères de l'élément et de l'ensemble des descendants de l'élément.

Il correspond en fait à `innerHTML` plus le `HTML` au format chaîne de caractères de l'élément.

Si nous avons le `HTML` suivant :

```
<div id="ma-div">
  <p>Un paragraphe</p>
  <p>Deux paragraphe</p>
</div>
```

Et que nous faisons :

```
const maDiv = document.getElementById("ma-div");
console.log(maDiv.outerHTML);
```

Nous aurons une chaîne de caractères contenant la `div` et les paragraphes. Elle contiendra aussi les espaces et les sauts de ligne.

Les propriétés `data` et `nodeValue` des noeuds

Les deux propriétés précédentes sont utilisables uniquement sur les éléments `DOM`.

Pour les noeuds qui ne sont pas des éléments comme les `Text` et les `Comment` vous pouvez utiliser les propriétés `data` ou `nodeValue`.

Ces propriétés ont le même effet, **elles retournent une chaîne de caractères contenant la valeur du noeud**.

Les propriétés `textContent` et `innerText`

`textContent` et `innerText` permettent de récupérer le contenu textuel d'un noeud et de ses descendants.

Ces deux propriétés ont d'importantes différences qu'il est nécessaire de connaître.

A l'inverse de `innerText, textContent` récupère le contenu de tous les éléments (comme par exemple <script> ou <style>).

`innerText` retourne le texte de l'élément mais en appliquant le style. Il ne retournera rien pour les éléments ayant la propriété `hidden`.

Par exemple si nous avons en `HTML` :

```
<body>
  <div id="ma-div">
    Ceci est un <span>test</span> de text
    <span hidden>avec du texte caché</span>
  </div>
</body>
```

Nous aurions :

```
document.getElementById("ma-div").innerText; // Ceci est un test de text


document.getElementById("ma-div").textContent;
//
//    Ceci est un test de text
//    avec du texte caché
//
```