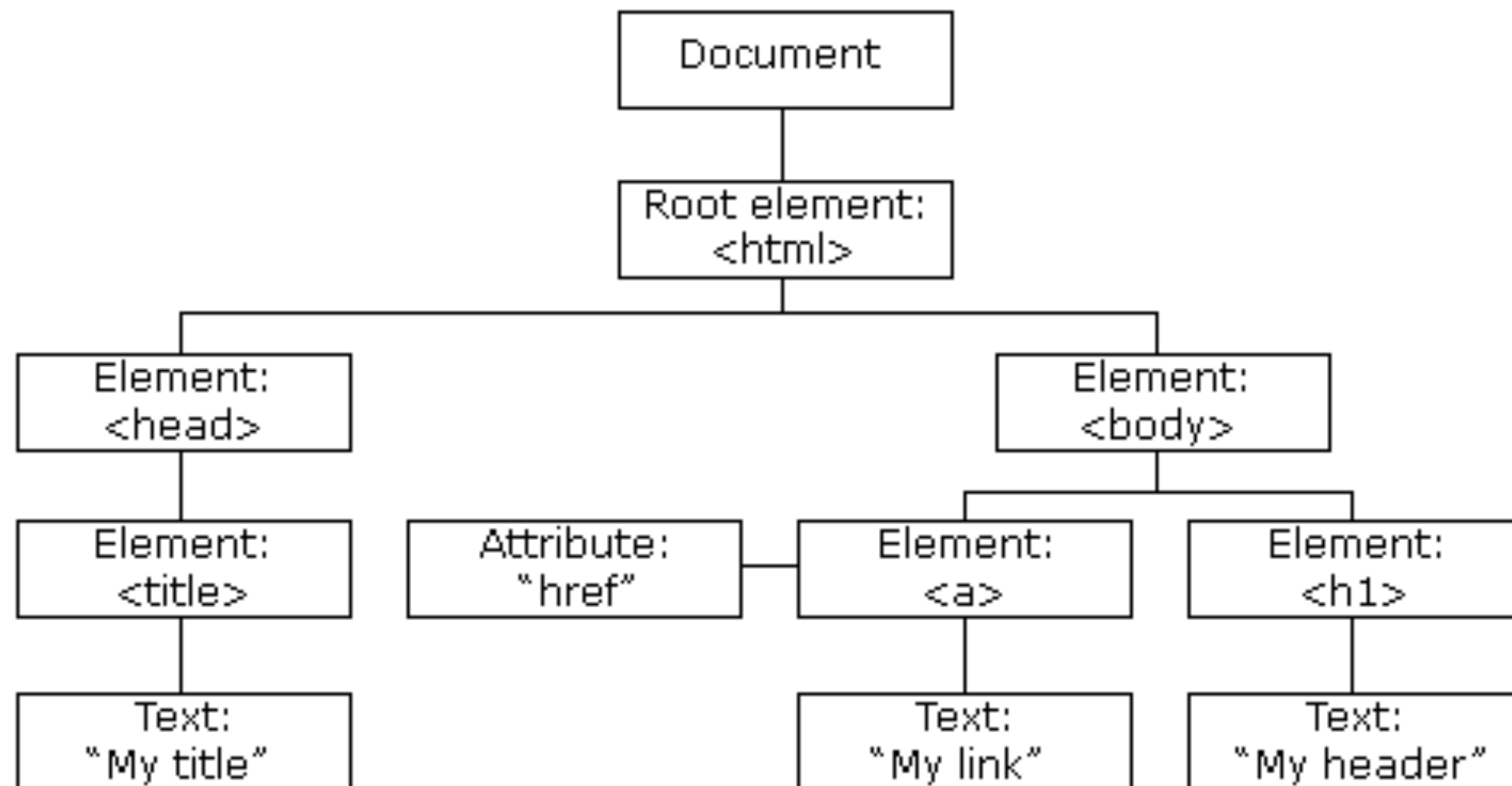


Javascript

Le Document Object Model (DOM)

Définition

Représentation de la structure et du contenu d'une page HTML sous la forme d'un arbre d'objets



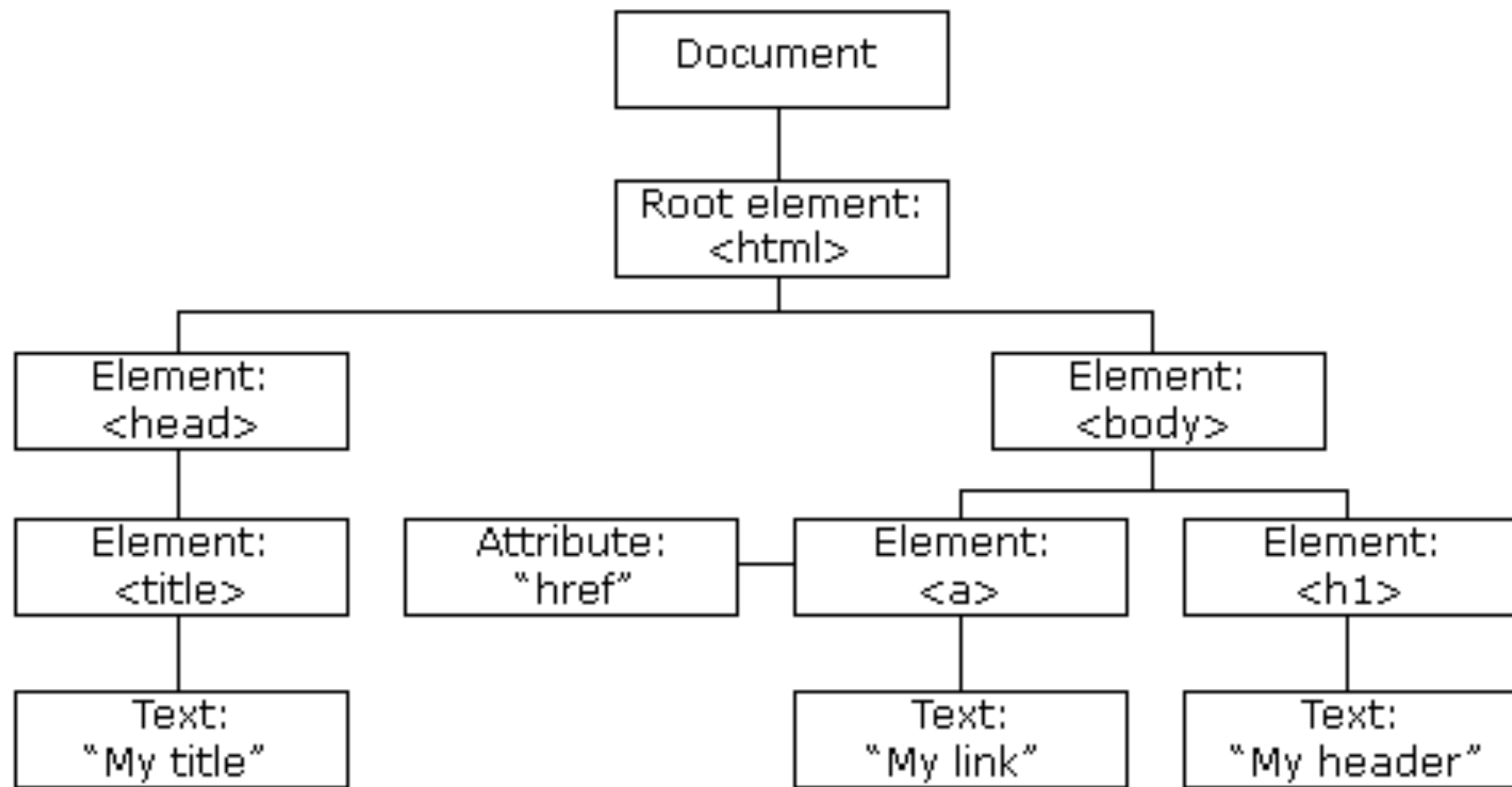
Utilité

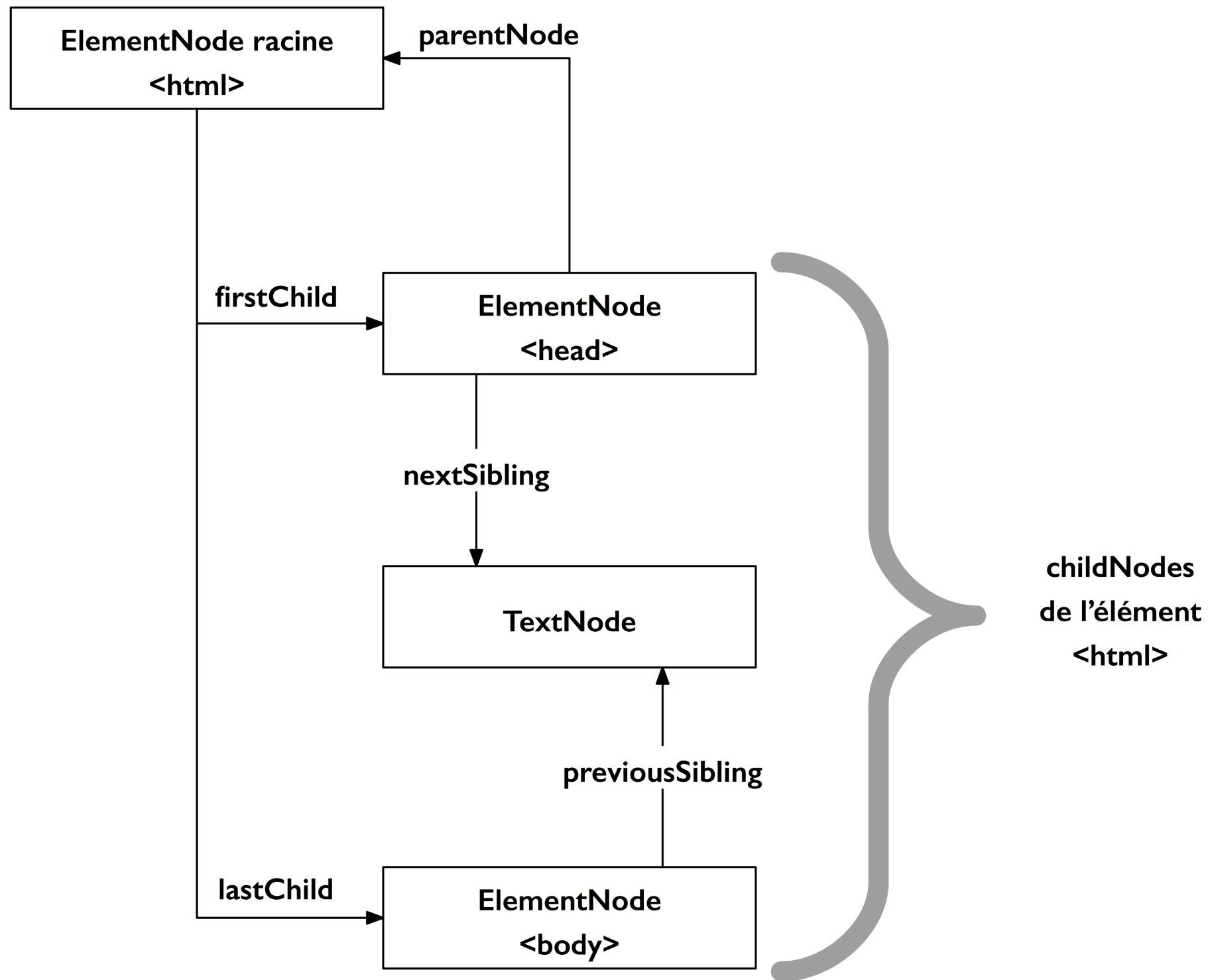
- Modifier dynamiquement des éléments (balises) ou des attributs (class, style ...)
- Supprimer dynamiquement des éléments ou attributs
- Ajouter dynamiquement des éléments ou attributs
- Capturer des événements

Le parcours

Quelques attributs

document.body	la balise <body> et son contenu
document.head	la balise <head> et son contenu
document.images	tous les éléments de type
document.forms	tous les éléments de type <form>
document.links	tous les éléments de type <a> qui ont un attribut href
document.title	le contenu de la balise <title>





Les relations

propriété	description
parentNode	Le noeud parent
firstChild	Le premier noeud enfant
lastChild	Le dernier noeud enfant
nextSibling	Le noeud frère suivant
previousSibling	Le noeud frère précédent
childNodes[]	Un tableau de l'ensemble des noeuds enfants (NodeList)
children[]	un tableau des noeuds enfant de type Element (HTMLCollection)

Les relations

propriété	description
childNodes	Le nombre de noeuds de type Element (balise)
firstElementChild	Le premier noeud enfant de type Element
lastElementChild	Le dernier noeud enfant de type Element
nextElementSibling	Le noeud frère suivant de type Element
previousElementSibling	Le noeud frère précédent de type Element

Les propriétés de noeuds

propriété	description
nodeType	Le type du noeud sous la forme d'un entier (voir diapo suivante)
nodeName	Le nom du noeud (nom de la balise ou de l'attribut)
nodeValue	La valeur du noeud
textContent	Le contenu textuel (sans les balises enfant) d'un noeud de type élément plus rapide que innerHTML
innerHTML	Contenu html d'un noeud de type élément

Les types de noeuds

constante	description	valeur
Node.ELEMENT_NODE	un élément (balise)	1
Node.ATTRIBUTE_NODE	un attribut d'une balise	2
Node.TEXT_NODE	une valeur (contenu d'une balise ou d'un attribut, espace entre deux balises)	3
Node.COMMENT_NODE	un commentaire	8

Elements Console Sources Network Timeline

top ▾ ☐ Preserve log

H1

▼ [text, li, text, li, text] ⓘ

- ▶ 0: text
- ▶ 1: li
- ▶ 2: text
- ▶ 3: li
- ▶ 4: text
- length: 5

▼ __proto__: NodeList

- ▶ constructor: function NodeList()
- ▶ entries: function entries()
- ▶ forEach: function forEach()
- ▶ item: function item()
- ▶ keys: function keys()
- length: (...)
- ▶ get length: function ()
- ▶ values: function values()
- ▶ Symbol(Symbol.iterator): function ()
- Symbol(Symbol.toStringTag): "NodeList"
- ▶ __proto__: Object

▶ ["pommes", "poires"]

▶ ...

⋮ Console

Un objet NodeList
ressemble à un tableau,
mais n'en a pas tous les
attributs

Exemple

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Titre de la page</title>
  </head>
  <body>
    <h1 id="intro">Ma page</h1>
    <p id="demo">Hello!</p>
    <ul class="list">
      <li>pommes</li>
      <li>poires</li>
    </ul>

    </body>
  </html>
```


Changer le texte de la balise <h1>

```
var html = document.childNodes[1];  
var body = html.lastChild;
```

```
var h1 = body.firstElementChild;
```

```
var content = "Mon titre";
```

```
h1.textContent = content;
```

```
//ou
```

```
h1.firstChild.nodeValue = content;
```

```
//ou
```

```
h1.innerHTML = content;
```


Récupérer tous les éléments de la liste

```
var html = document.childNodes[1];

var ul = html.lastChild
        .firstElementChild
        .nextElementSibling
        .nextElementSibling
        ;
var list = [];
var childCount = ul.childNodes.length;
var currentItem;

for(var i= 0 ; i< childCount; i++){
    currentItem = ul.childNodes[i];
    if(currentItem.nodeType == Node.ELEMENT_NODE){
        list.push(currentItem.innerHTML);
    }
}
```


Autre syntaxe

```
var html = document.childNodes[1];

var ul = html.lastChild
        .firstElementChild
        .nextElementSibling
        .nextElementSibling
        ;
var list = [];

ul.childNodes.forEach(function(item){
    if(item.nodeType == Node.ELEMENT_NODE){
        list.push(item.innerHTML);
    }
});
```


Problèmes

- Le parcours est verbeux et fastidieux
- Si la structure du document change, il faut modifier le code

Les sélecteurs

méthode	description
<code>getElementById(id)</code>	Retourne le noeud identifié par l'Id passé en argument
<code>getElementsByTagName(tag)</code>	Retourne la liste des noeuds (nodeList) identifiés par la balise passée en argument
<code>getElementByClassName(class)</code>	Retourne la liste des noeuds (nodeList) identifiés par la classe passée en argument
<code>getElementsByName(name)</code>	Retourne la liste des noeuds (nodeList) identifiés par l'attribut name passé en argument
<code>querySelector(selector)</code>	Retourne le premier noeud identifié par le sélecteur CSS passé en argument
<code>querySelectorAll(selector)</code>	Retourne la liste des noeuds (nodeList) identifiés par le sélecteur CSS passé en argument

Exemples

```
//Le premier h1 du document
h1 = document.getElementsByTagName("h1")[0];

//l'élément dont l'attribut id est intro
h1 = document.getElementById("intro");

//Le premier élément de class list
list = document.getElementsByClassName("list")[0];

//Le premier ul de classe list
list = document.querySelector("ul.list");

//La liste de tous les li
items = document.querySelectorAll("ul.list li");
```


Les événements

Gestionnaire d'événements

fonction de callback réagissant à un événement sur un élément

```
element.onEventName = eventHandler
```

```
var bt = document.getElementById("bt1");  
bt.onClick = clickHandler;
```


Avec une fonction anonyme

```
var bt = document.getElementById("bt1");  
bt.onClick = function(){  
    console.log("click");  
};
```


Comportements par défaut

- Chargement des liens (``)
- Soumission des formulaires
(`type="submit"` ou touche ENTER)
- Focus sur les éléments de formulaire
avec TAB

Si le gestionnaire retourne false, le comportement par défaut ne s'applique pas

Quelques événements

onclick	clic de la souris
ondblclick	double clic
onmouseup	bouton de la souris relâché
onmousedown	bouton de la souris enfoncé
onmouseover	le pointeur de la souris est sur l'élément
onmouseout	le pointeur de la souris quitte l'élément
onmousemove	le pointeur de la souris bouge
oncontextmenu	clic avec le bouton droit

Quelques événements

onfocus	focus sur un champ de saisie
onblur	perte du focus
onchange	modification d'un contrôle de formulaire
onsubmit	soumission d'un formulaire
onreset	annulation de la saisie d'un formulaire
onscroll	déplacement de la barre de défilement d'un élément
onload	chargement d'une ressource (window, image, link...)

L'objet Event

Définition

- Un objet passé en argument des gestionnaires d'évènements
- Indique le contexte de l'évènement

Example

```
var bt1 = document.getElementById("bt1");  
  
bt1.onclick = function(evt){  
    console.log(evt);  
};
```


propriétés

target	l'élément qui a déclenché l'événement
type	le nom de l'événement
timeStamp	horodatage de l'événement (en ms depuis le 01/01/1970)

propriétés de MouseEvent

clientX, clientY	Les coordonnées X et Y absolue par rapport à la fenêtre visible
pageX, pageY	Les coordonnées X et Y par rapport au document
button	le bouton de la souris qui a été enfoncé (0 : gauche, 1 : droit, 2 : roue)
which	le bouton de la souris qui a été enfoncé (0 : aucun, 1 : gauche, 2 : roue, 3 : droit)
detail	le nombre de clics successif dans un faible laps de temps
altKey, ctrlKey, shiftKey	true si la touche ALT, CTRL ou SHIFT était enfoncée lors du clic

Exemple

événement générique pour les boutons

```
var buttons = document.querySelectorAll("button");

buttons.forEach(function (item) {
    item.onclick = function(evt){
        evt.target.innerText += " cliqué";
    };
});
```


Exemple

horodateur d'événements

```
var buttons = document.querySelectorAll("button");
var btTime = {};

buttons.forEach(function (item) {
    item.onclick = function (evt) {
        var targetId = evt.target.getAttribute("id");

        //Temps écoulé entre deux clics sur le même bouton
        if (targetId in btTime) {
            var elapsed = evt.timeStamp - btTime[targetId];
            console.log((elapsed / 1000) + " s écoulées entre deux clics");
        }
        //Horodatage du clic
        btTime[targetId] = evt.timeStamp;
    };
});
```


Exemple

position de la souris

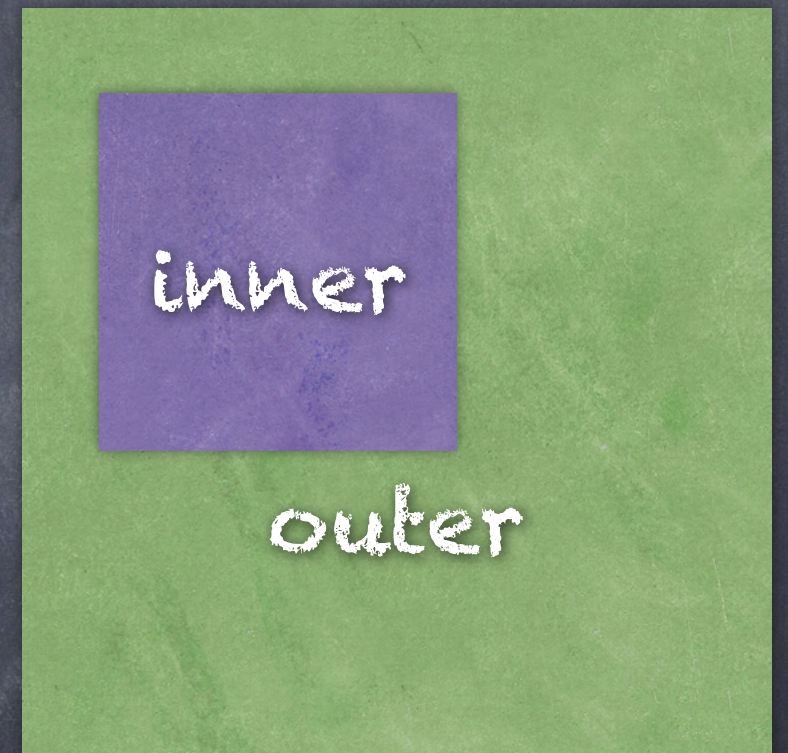
```
window.onmousemove = function(evt){  
  
    console.log("clientX = "evt.clientX);  
    console.log("pageX = "evt.pageX);  
  
    console.log("clientY = "evt.clientY);  
    console.log("pageY = "evt.pageY);  
};
```


Désactiver le comportement par défaut

```
var bt1 = document.getElementById("bt1");  
  
bt1.onclick = function(evt){  
    evt.preventDefault();  
};
```


Propagation des événements

```
inner.onclick = function(evt){  
    evt.stopPropagation();  
    console.log("inner");  
};  
  
outer.onclick = function(evt){  
    console.log("outer");  
};
```



Désactiver un gestionnaire

```
var bt1 = document.getElementById("bt1");  
var bt2 = document.getElementById("bt2");  
  
bt1.onclick = function(){  
    console.log("clic");  
};  
  
bt2.onclick = function(){  
    bt1.onclick = null;  
};
```


Écouteurs
d'événements

Utilité

Il est possible de définir plusieurs écouteurs sur un même élément et pour un même événement ce qui est impossible avec un gestionnaire d'événement

Ajouter un écouteur

```
var bt1 = document.getElementById("bt1");
```

```
function clickHandler(){  
    console.log("clic");  
}
```

```
bt1.addEventListener("click", clickHandler);
```


Supprimer un écouteur

Pour supprimer un événement il faut que sa fonction de callback soit nommée

```
var bt1 = document.getElementById("bt1");  
  
function clickHandler(){  
    console.log("clic");  
    bt1.removeEventListener("click", clickHandler);  
}  
  
bt1.addEventListener("click", clickHandler);
```


Supprimer un écouteur

Il est possible de stocker la fonction anonyme dans une variable

```
var bt1 = document.getElementById("bt1");  
  
var evtId = bt1.addEventListener("click",  
    function clickHandler(){  
        console.log("clic");  
        bt1.removeEventListener("click", evtId);  
    });
```


Les attributs

Méthodes

<code>setAttribute(name, value)</code>	change l'attribut name et lui affecte la valeur value
<code>getAttribute(name)</code>	retourne la valeur de l'attribut dont le nom est passé en argument
<code>hasAttribute(name)</code>	retourne true si l'élément possède l'attribut dont le nom est passé en argument
<code>removeAttribute(name)</code>	supprime un attribut
<code>attributes</code>	tableau indicé de tous les attributs d'un élément

Exemple

```
var bt1 = document.getElementById("bt1");
var btReset = document.getElementById("reset");

bt1.onclick = function(){
    bt1.innerText = "Cliqué";
    bt1.setAttribute("disabled","disabled");
};

btReset.onclick = function(){
    if(bt1.hasAttribute("disabled")){
        bt1.removeAttribute("disabled");
        bt1.innerText = "Cliquez moi"
    }
}
```


Les styles

```
selecteur.style.propriétéCSS
```

La propriété CSS est camélisée
font-size devient fontSize

```
document.querySelector("body")  
    .backgroundColor = "#CCCCCC";
```


Les classes CSS

IE 10+

<code>className</code>	retourne une chaîne de caractère correspondant à la valeur de l'attribut class de l'élément
<code>classList</code>	retourne un tableau des classes affectées à l'élément (DOMTokenList)
<code>classList.contains(className)</code>	retourne true si l'élément possède la classe passée en argument
<code>classList.add(className)</code>	ajoute la classe passée en argument
<code>classList.remove(className)</code>	supprime la classe passée en argument
<code>classList.toggle(className)</code>	bascule la classe passée en argument, si elle existe elle est supprimée sinon elle est ajoutée

Ajout/suppression
de noeuds

Ajouter un noeud

- Créer un élément détaché du DOM
- Modifier les propriétés de cet élément
- Attacher cet élément au DOM

Création d'un élément

```
document.createElement("balise");
```

```
var list = document.querySelector("#list");
```

```
//création de l'élément
```

```
var item = document.createElement("li");
```

```
//Modification des attributs
```

```
item.textContent = "cerises";
```

```
item.style.backgroundColor = "red";
```


Ajout de l'élément au DOM

```
node.appendChild(element)
```

```
var list = document.querySelector("#list");
```

```
//création de l'élément
```

```
var item = document.createElement("li");
```

```
//Modification des attributs
```

```
item.textContent = "cerises";
```

```
item.style.backgroundColor = "red";
```

```
//Ajout comme dernier enfant
```

```
list.appendChild(item);
```


Insertion

```
node.insertBefore(element, cible)
```

```
var list = document.querySelector("#list");  
var firstItem = list.firstElementChild;
```

```
//création de l'élément
```

```
var newItem = document.createElement("li");
```

```
//Ajout avant le premier enfant
```

```
list.insertBefore(newItem, firstItem);
```


Remplacement

```
node.replaceChild(element, cible)
```

```
var list = document.querySelector("#list");  
var firstItem = list.firstElementChild;  
  
//création de l'élément  
var newItem = document.createElement("li");  
  
//Remplacement du premier enfant  
list.replaceChild(newItem, firstItem);
```


Suppression

```
node.removeChild(element, cible)
```

```
var list = document.querySelector("#list");  
var lastItem = list.lastElementChild;
```

```
//Suppression du dernier enfant  
list.removeChild(lastItem);
```

```
//Ou avec les derniers navigateurs  
lastItem.remove();
```


Application

- ▶ Au clic sur un bouton, changer son texte et sa couleur de fond
- ▶ Tester si un mot de passe et sa confirmation sont identiques, afficher un message d'erreur si ce n'est le cas

Application

- ▶ Rendre un paragraphe éditable
 - ▶ au clic sur la paragraphe, remplacer celui-ci par une zone de texte multi-ligne contenant le texte du paragraphe
 - ▶ à la sortie de la zone de texte (onblur) remplacer celle-ci par un paragraphe contenant le texte modifié

Application

- ▶ Créer une application qui permette d'ajouter, supprimer et modifier les éléments d'une liste html
- ▶ Permettre également le tri des éléments au clic sur un bouton

Application

- ▶ Créer un tooltip, une zone d'information qui apparaît au survol d'un élément et disparaît lorsque le pointeur quitte l'élément.
- ▶ Le texte du tooltip reprendra l'attribut title de l'élément cible.
- ▶ Si le pointeur de la souris se déplace en restant dans l'élément, le tooltip devra également se déplacer en conséquence

Application

- ▶ Créer un bouton qui affiche une fenêtre d'authentification dans un overlay.
- ▶ Au clic n'importe où sur l'overlay masquer celui-ci
- ▶ Au clic sur le bouton valider du formulaire tester la saisie et afficher un message en conséquence (login=admin, password=pass)

Application

- ▶ A partir du fichier `astronomie.js` afficher une liste des étoiles.
- ▶ Au clic sur un élément afficher les détails de cette étoile
- ▶ Créer un formulaire pour ajouter une étoile
- ▶ Mettre en place un système de recherche par constellation

Application

- ▶ A partir des fichiers `regions.js`, `regions-departements.js` et `communes-simples.js` mettre en place un menu déroulant permettant de choisir une région.
- ▶ Au choix de la région afficher une liste déroulante des départements correspondant
- ▶ Au choix d'un département afficher une liste déroulante des communes