

# Cadre de travail **Web/JavaScript**



*"Le plus grand arbre est né d'une graine menue." (Lao-Tseu) - 2014 © Gilles Chamillard*

# JavaScript et ses mystères

Chercher la version de JavaScript que l'on utilise s'apparente à plusieurs labyrinthes emmêlés..

## Deux informations essentielles à savoir :

- 1) Il n'existe pas de moyen fiable de vérifier la version de JavaScript !
- 2) Utile sur le principe de la compatibilité, sa connaissance est accessoire pour une application Web...

## Rapide historique/repères pour s'y retrouver :

- JavaScript a été créé par les sociétés SUN et Netscape
- Une demande de normalisation auprès de l'ECMA à aboutit à ECMA-262
- Une version **majeure** en juin 2015 s'appelle indifféremment  
**ES6**/ECMAScript6/ES2015/ECMAScript2015
- Les versions suivantes, **mineures** suivent la numérotation ES7/ES2017, etc.
- Mais en fait chaque navigateur à sa version : **Chrome** V(*Num*), **Firefox** SpiderMonkey, etc.
- Chez Microsoft JavaScript s'appelle Jscript !

**Il vaut mieux vérifier la compatibilité des fonctionnalités en fonction de la bibliothèque utilisée !**

# JavaScript et ES6

A l'heure actuelle, tous les navigateurs modernes supportent **ES6** (et les versions mineures suivantes).

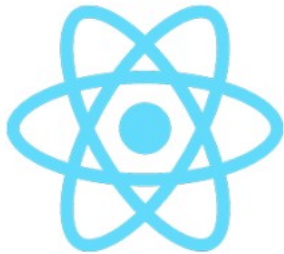
**Préférez pourtant les navigateurs à base de Chrome et son moteur V8 actuel.**

Tapez comme URL pour ce type de navigateur :

```
chrome://version/
```

Tous les frameworks **front-end** majeurs JavaScript utilisent tous la version ES6 :

**REACT**



**ANGULAR**



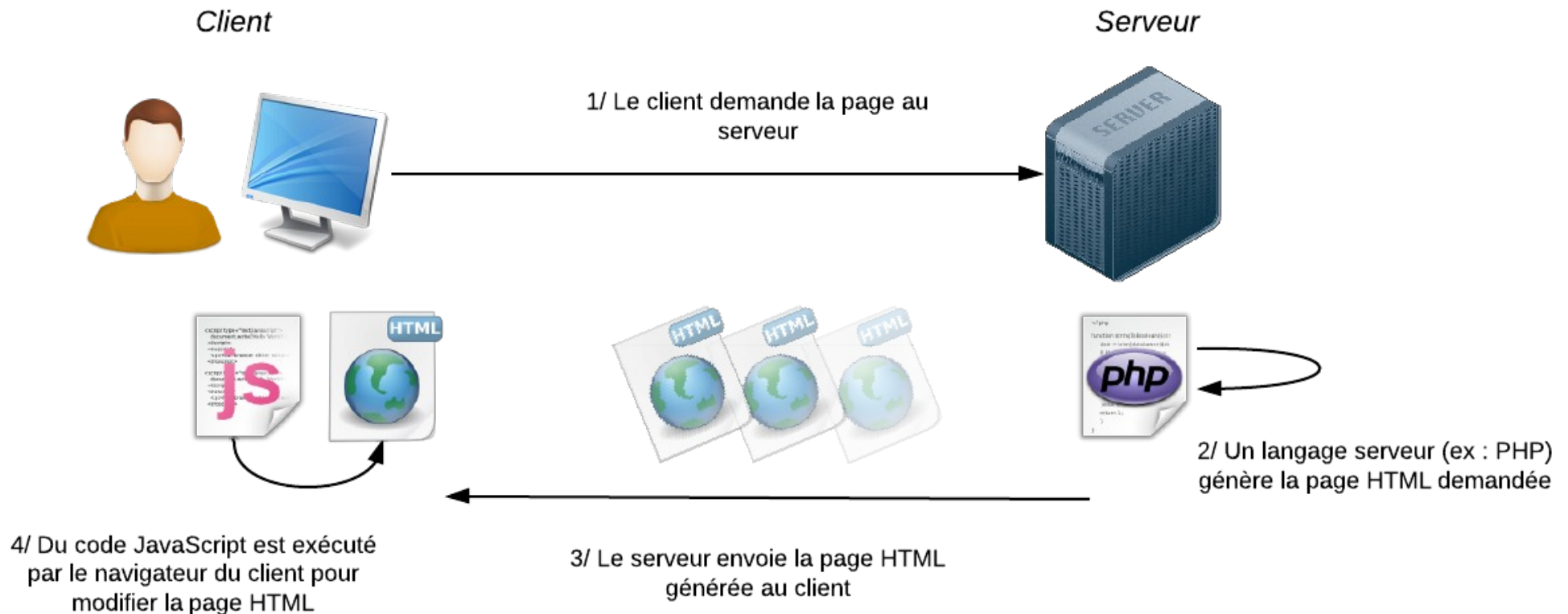
**VUE**



NodeJS, en **back-end** est construit sur « les versions modernes » de **V8**.



# JavaScript et l'architecture client/serveur

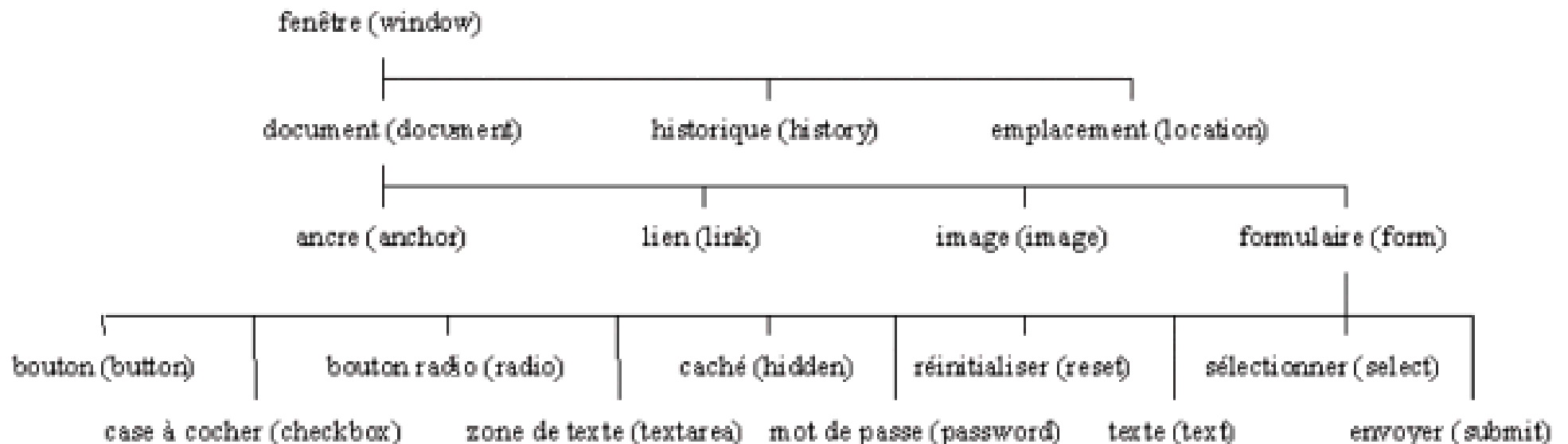


# JavaScript et le DOM

Un document HTML suit une structure de type arbre appelée **DOM** (*Document Object Model*) et interprété par le navigateur.

- **L'accès à un objet**

En JavaScript, l'accès à un objet dépend essentiellement de la hiérarchie d'objets du navigateur Web :



Toutes les pages WEB possèdent les objets `window`, `document`, `history` et `location`. Pour accéder à la valeur d'un champ texte d'un formulaire on utilisera la syntaxe suivante :

```
document.form.text.value
```

# Insertion de code JS dans la page Web


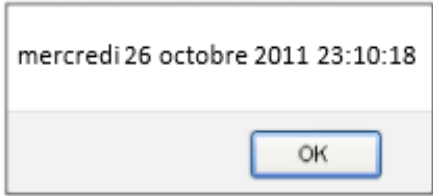
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>test JS</title>
  <script type="text/javascript" src="script.js"></script>
</head>
<body>
  <script type="text/javascript">
    <!--
      function affiche_date(){
        var aujourd'hui = new Date();
        alert(aujourd'hui.toLocaleString());
      }
    // -->
  </script>
  <noscript><p>activez JS !</p></noscript>
  <div>
    <h1>Test JS</h1>
    <a href="javascript:affiche_date()">date</a>
    <a href="javascript:alert('hello')">hello</a>
    <form action="test.php">
      <div><input type="button" value="clic" onclick="pair()"></div>
    </form>
  </div>
</body>
</html>
```

1

2

3

4



```
function pair(){
  nb = prompt('entrez un nombre', '');
  (nb%2 == 0) ? alert('pair') : alert('impair');
}
```

script.js

1

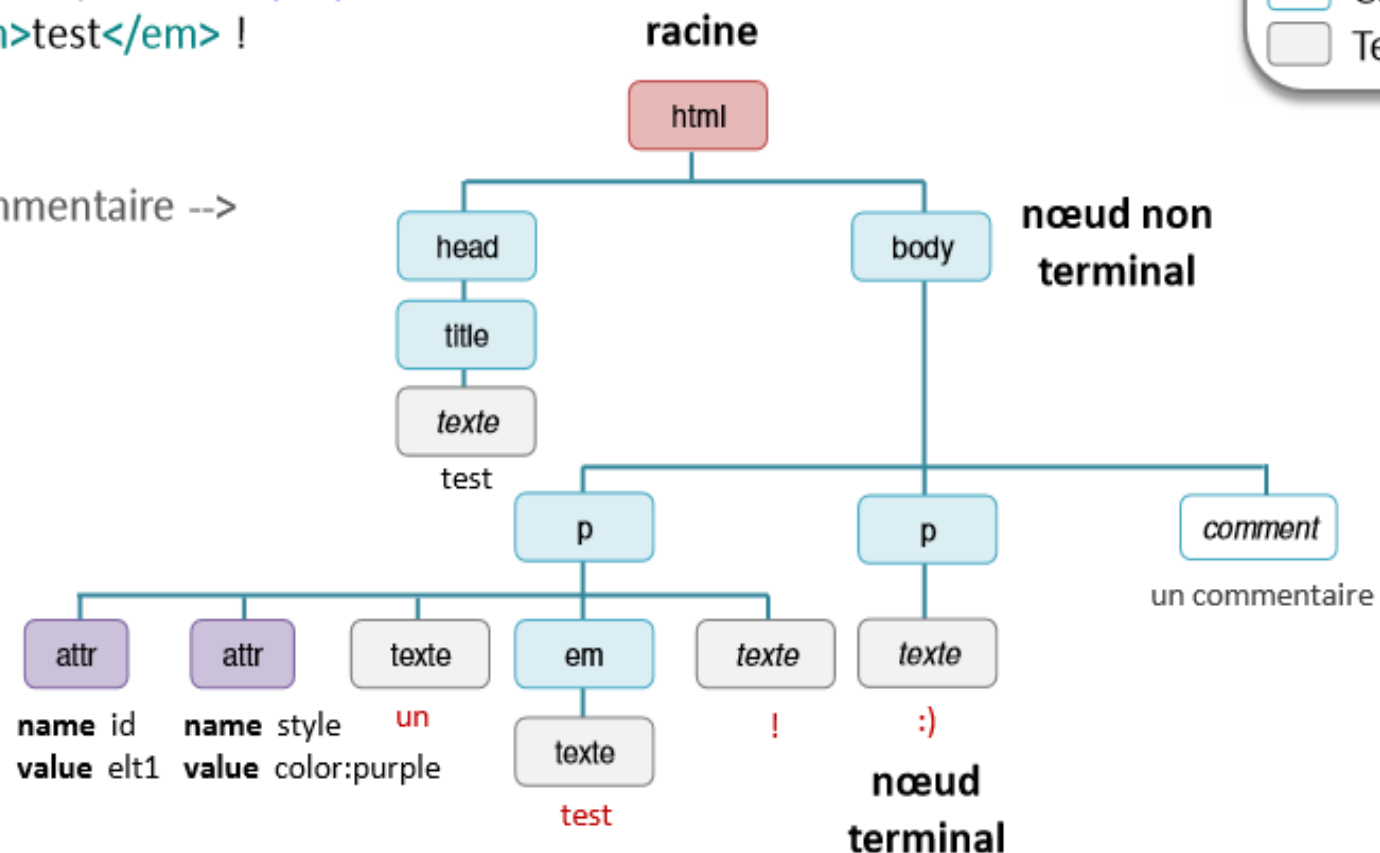


Le **code** peut être placé dans :

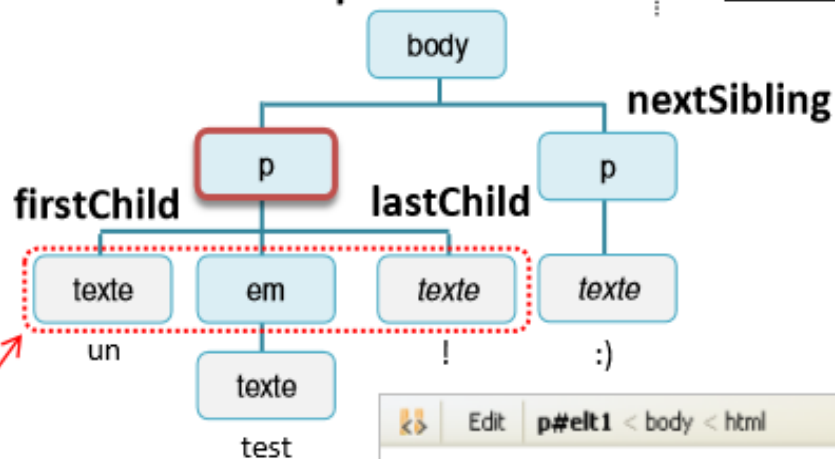
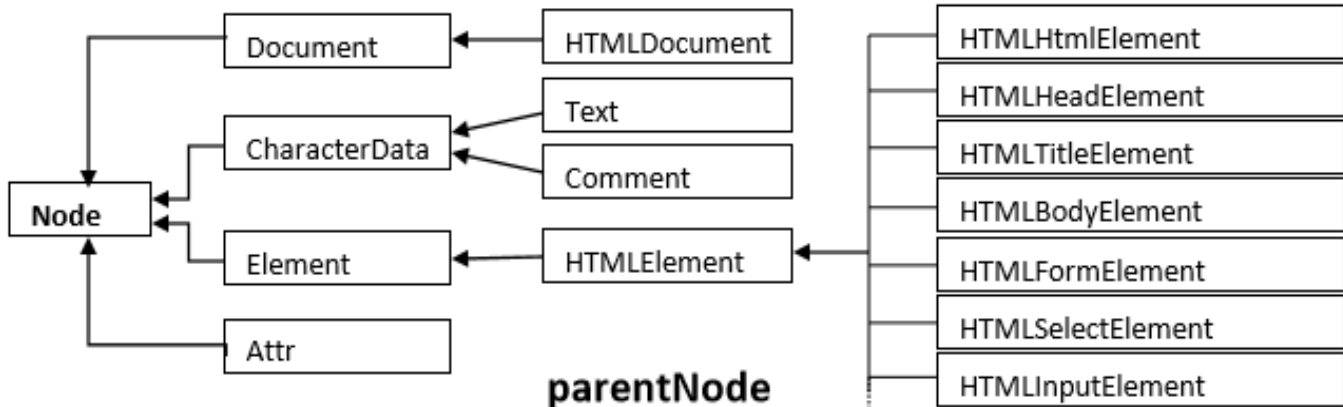
- (1) fichier externe .js
- (2) élément script
- (3) attribut contenant URL
- (4) attribut événementiel  
onclick, onchange, onselect,...

# DOM – Arborescence du document

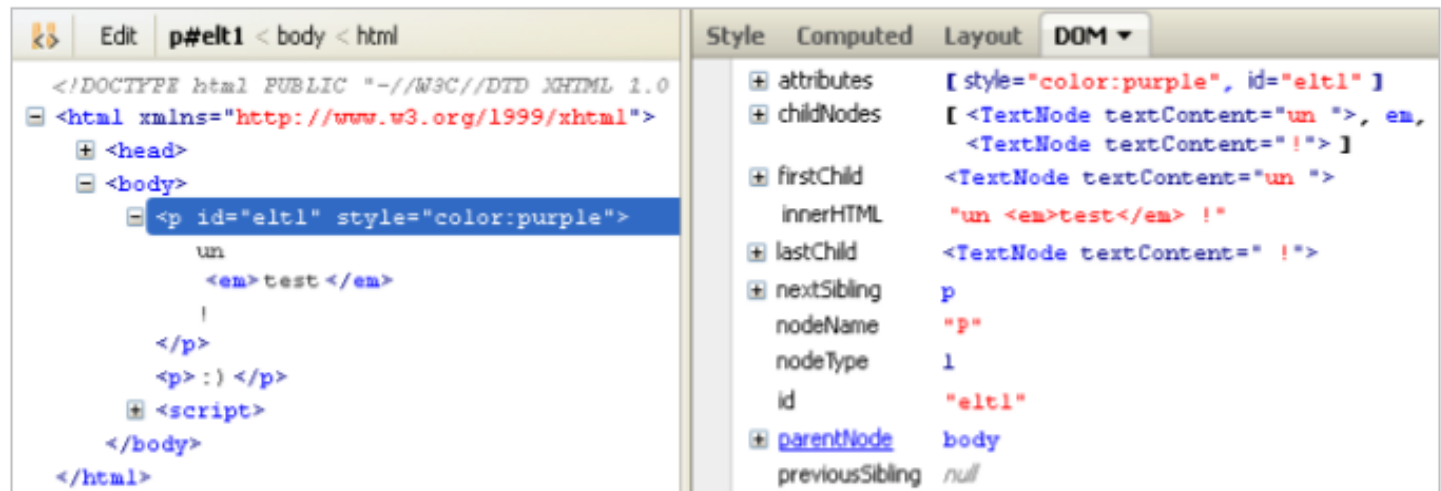
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>test</title>
</head>
<body>
  <p id="elt1" style="color:purple">
    un <em>test</em> !
  </p>
  <p>:</p>
  <!-- un commentaire -->
</body>
</html>
```



## DOM – Propriétés des nœuds

**childNodes**

node			
nodeName		nodeType	
parentNode		childNodes	
firstChild		lastChild	
nextSibling		previousSibling	
attributes			
élément		élément HTML	
tagName		className	title
			id
attribut		texte	
name	value	data	length



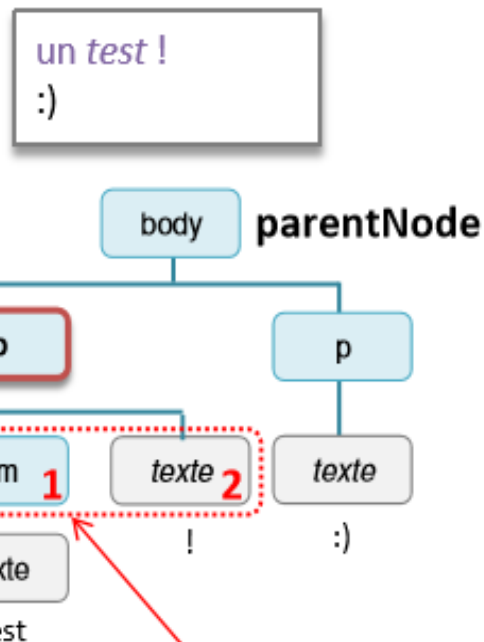


# DOM – Accès aux nœuds

## Accès par

ID	document. <b>getElementById</b> ('id_element')	-> nœud
Nom d'élément	document. <b>getElementsByTagName</b> ('nom_element')	-> collection de nœuds
Valeur de l'attribut name	document. <b>getElementsByName</b> ('val_attribut')	-> collection de nœuds

```
<body>
<p id="elt1" style="color:purple">
un <em>test</em> !</p>
<p>:</p>
</body>
```



attributes

childNodes

## Collection = tableau de nœuds

prédéfinies : **forms**, **images**, **links**  
fils d'un nœud : **childNodes**  
attributs d'un nœud : **attributes**  
nom : **getElementsByName**  
attr. name : **getElementsByTagName**  
accès à un nœud : **[i]** ou **item(i)**  
avec  $i \geq 0$

```
var el = document.getElementById('elt1');
alert(el.tagName);           // P
alert(el.parentNode.nodeName); // BODY
alert(el.childNodes[1].nodeName) // EM
```

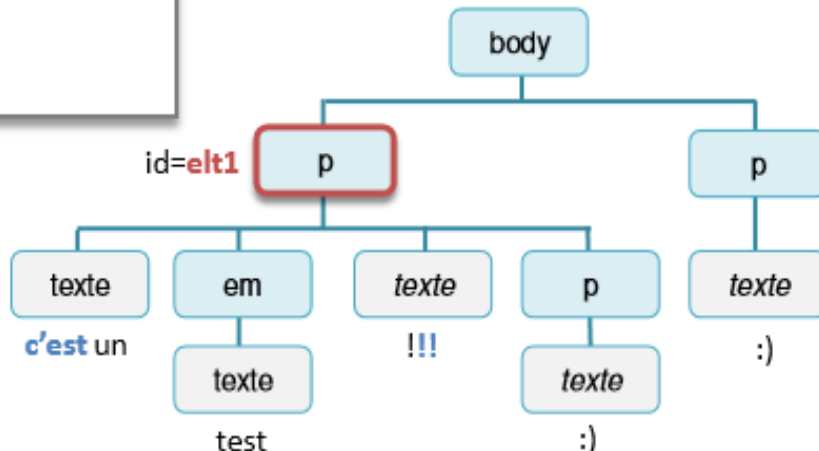
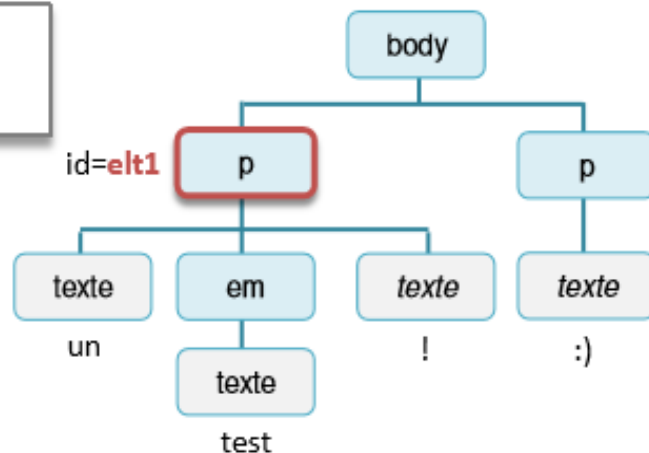
```
var tab_el = document.getElementsByTagName('p');
alert(tab_el[0].childNodes[2].nodeName); // #text
alert(tab_el[0].childNodes[2].data);      // !
```

# DOM – Modifier le document

un test !  
:)



c'est un test !!!  
:  
:)



```
var el = document.getElementById('elt1');
// DOM 0
el.innerHTML = "c'est un <em>test</em> !!!<p>:</p>";
```

## Contenu

DOM 0 : **innerHTML**, **innerText**

Lecture : **data**

Ajout : **appendData**(txt), **insertData**(i, txt)

Modifier : **replaceData**(i, j, txt)

Supprimer des caractères : **deleteData**(i, taille)

## Structure : manipuler les noeuds

Créer : **createElement**('nom'),  
**createTextNode**('texte')

Ajouter : **appendChild**(noeud), **insertBefore**(n1,n2)

Supprimer : **removeChild**(i)

Remplacer : **replaceChild**(noeud\_src, noeud\_cible)

Cloner avec arborescence : **cloneNode**(true)

```
var el = document.getElementById('elt1');
el.firstChild.insertData(0, "c'est ");
el.lastChild.append("!!!") ;
node = el.nextSibling.cloneNode(true);
el.appendChild(node);
```

## DOM – Modifier le style

### Modifier le style

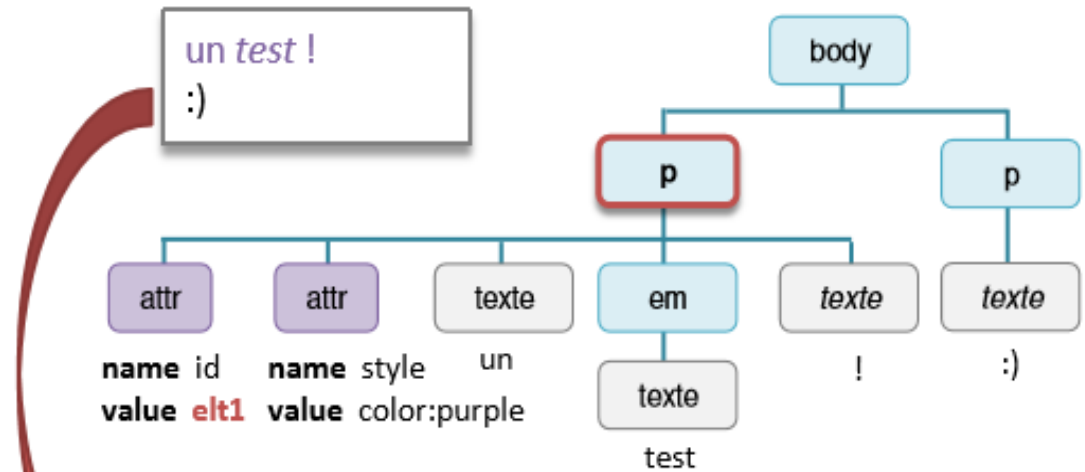
### Propriété CSS :

```
node.style.propriete = 'valeur';
```

Classe de style :

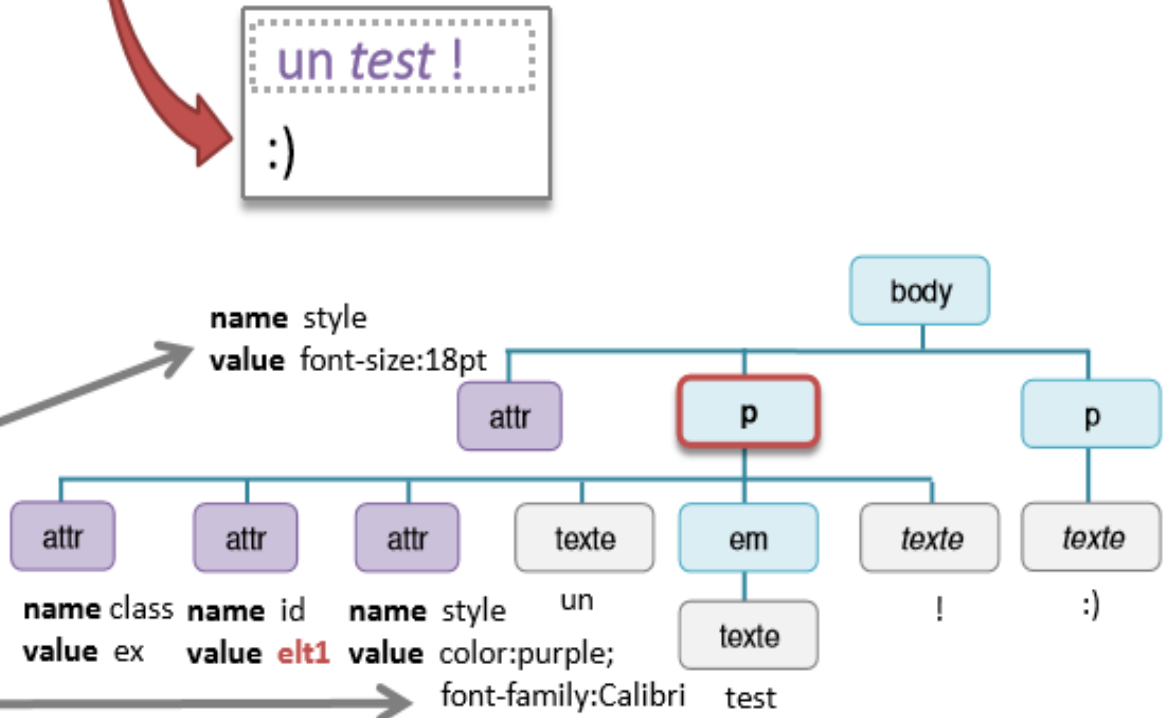
```
node.className = 'nom_classe';
```

Possibilité de modifier/ajouter/supprimer des règles (accès à la feuille de style CSS)



```
<style type="text/css">
.ex {
    border:2px dotted silver;
}
</style>
```

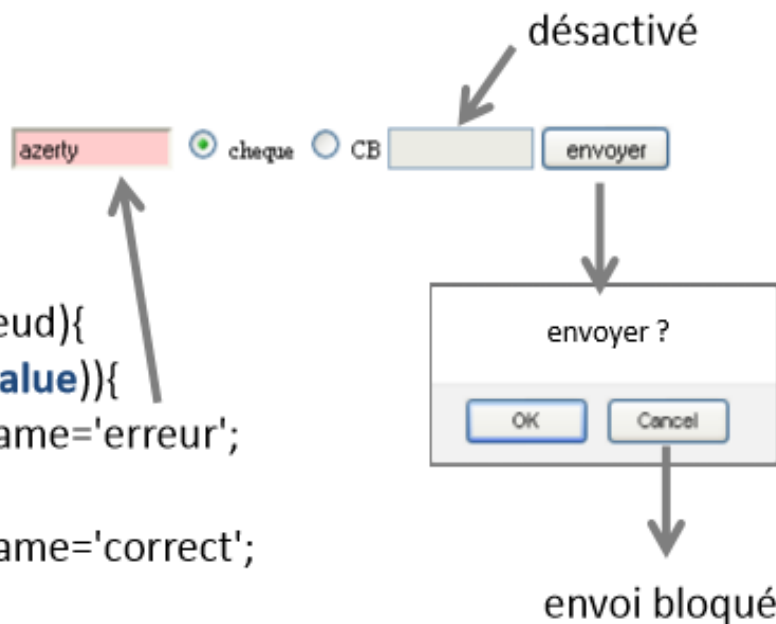
```
var el = document.getElementById('elt1');  
el.parentNode.style.fontSize = '18pt';  
el.className = 'ex';  
el.style.fontFamily = 'Calibri';
```



# DOM – Manipuler un formulaire

```
<form action='test.php' onsubmit="return confirm('envoyer?')">
<div>
<input type="text" name="nb" onchange="verifNb(this)">
<input type="radio" name="choix" value="1"
  onclick="cb.disabled=true"> cheque
<input type="radio" name="choix" value="2"
  onclick="cb.disabled=false" checked> CB
<input type="text" name="cb">
<input type="submit" value="envoyer">
</div>
</form>
```

```
function verifNb(noeud){
  if (isNaN(noeud.value)){
    noeud.className='erreur';
  } else {
    noeud.className='correct';
  }
}
```



```
.erreur { background-color:#FFCCCC }
.correct { background-color:white }
```

## propriétés - champs textes

text, hidden, password, textarea,  
boutons submit, reset, button

**value**

**defaultValue**

## propriétés - radio / cases à cocher

**value**

**checked**

## propriétés - listes de choix

**selectedIndex** **options** (collection)  
prop. options : **selected**, **value**, **text**

## Événements

onsubmit, onreset, onclick,  
onchange, onblur, onfocus, onselect

## Bloquer un événement

**return false;**

## Activer/désactiver des champs

propriétés **disabled** et **readOnly**