



# DÉVELOPPEMENT INFORMATIQUE LOGIQUE & PROGRAMMATION

LANGAGE JAVAScript

Y. DELVIGNE

CH. LAMBEAU

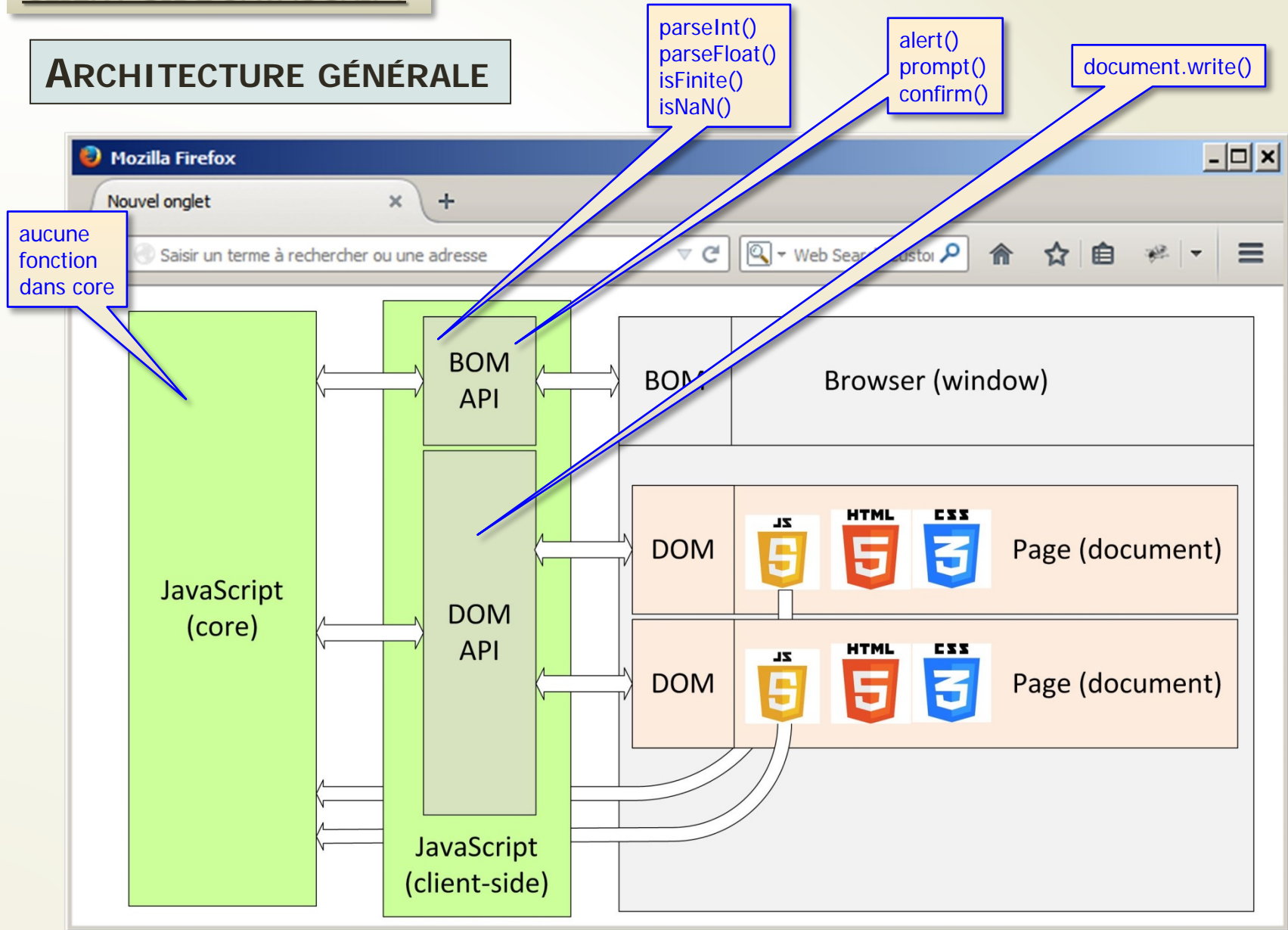


PLACER UN SCRIPT JAVAScript  
DANS UNE PAGE WEB :

CLIENT-SIDE JAVAScript



### ARCHITECTURE GÉNÉRALE



### OÙ PLACER LE SCRIPT ?

#### **DANS UNE BALISE SPÉCIFIQUE : `<SCRIPT>` `</SCRIPT>`**

- A PRIORI N'IMPORTE OÙ ... MAIS ...
- LES DEUX PHASES DE LA 'VIE' D'UNE PAGE :
  - PHASE PRÉLIMINAIRE DE RENDERING  
construction par le navigateur (html, css)
  - PUIS, PHASE ÉVÉNEMENTIELLE  
réactivité suite aux événements
    - internes au navigateur
    - action de l'utilisateur (boutons, formulaire ...)

## OÙ PLACER LE SCRIPT ?

**DANS UNE BALISE SPÉCIFIQUE : `<SCRIPT>` `</SCRIPT>`**

➤ **EN PHASE DE RENDERING**

- A PRIORI N'IMPORTE OÙ ... EN `<head>` OU `<body>`
- AUTANT DE FOIS QUE L'ON VEUT ...

**CHACUN SCRIPT ARRÊTE LA PHASE DE RENDERING ...**

(➔ POSITION COURANTE DANS LA PAGE)

... **S'EXÉCUTE** (ON PEUT DONC 'ÉCRIRE' DANS LA PAGE) ...

... **PUIS LE RENDERING CONTINUE** ...

... **À LA FIN** (APRÈS `</body>`), **PERTE DE POSITION COURANTE**

➔ `document.write()` DEVIENT DESTRUCTEUR POUR LE CONTENU

## NOS INTENTIONS ?

REEMPLACER LES ~~alert()~~ MODAUX (BLOQUANTS)

(MÉTHODE DU BOM, DU NAVIGATEUR)

## PAR QUOI ?

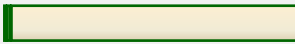
`document.write()`

(MÉTHODE DU DOM, DE LA PAGE)

ÉCRITURE DE L'ARGUMENT À LA POSITION COURANTE

argument : de préférence  
du HTML : texte balisé  
pour pouvoir utiliser CSS

## OÙ PLACER LE SCRIPT ?

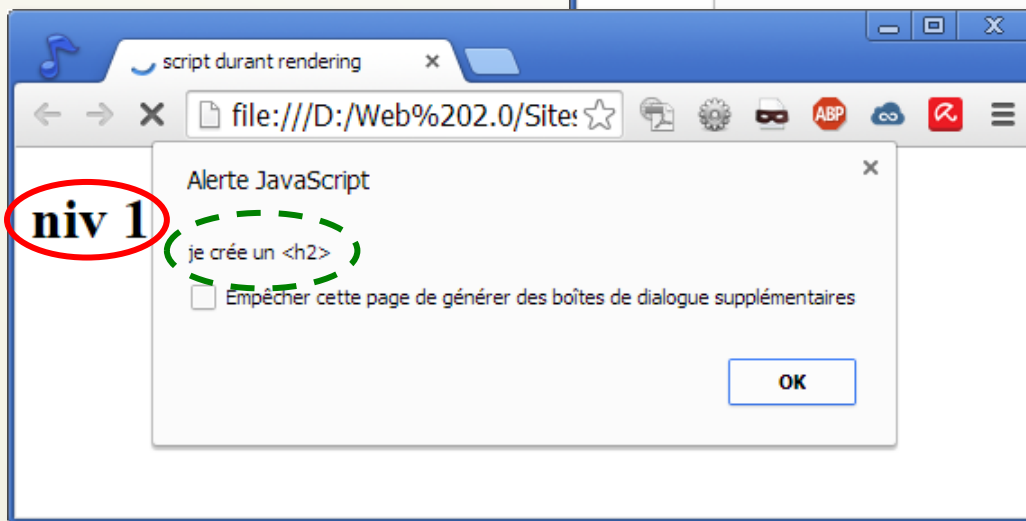
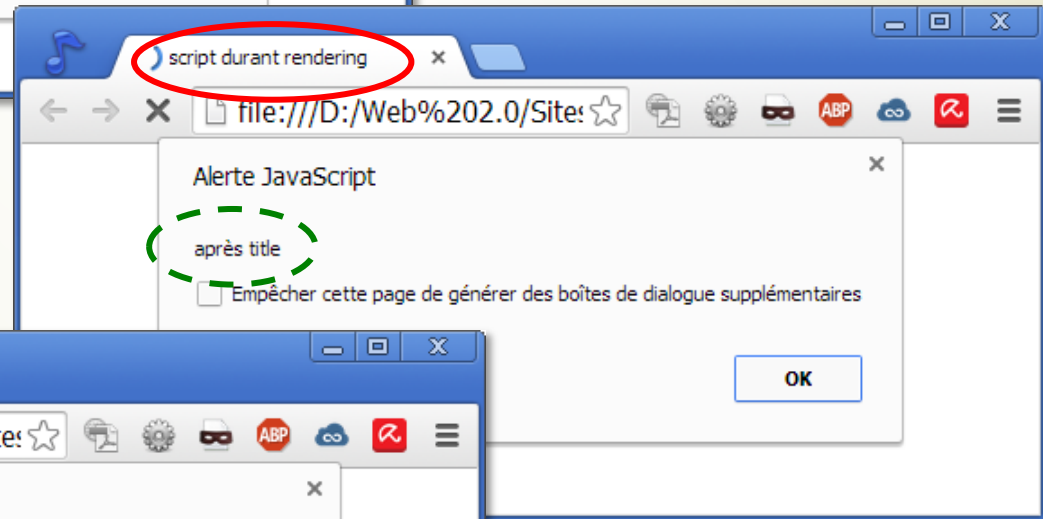
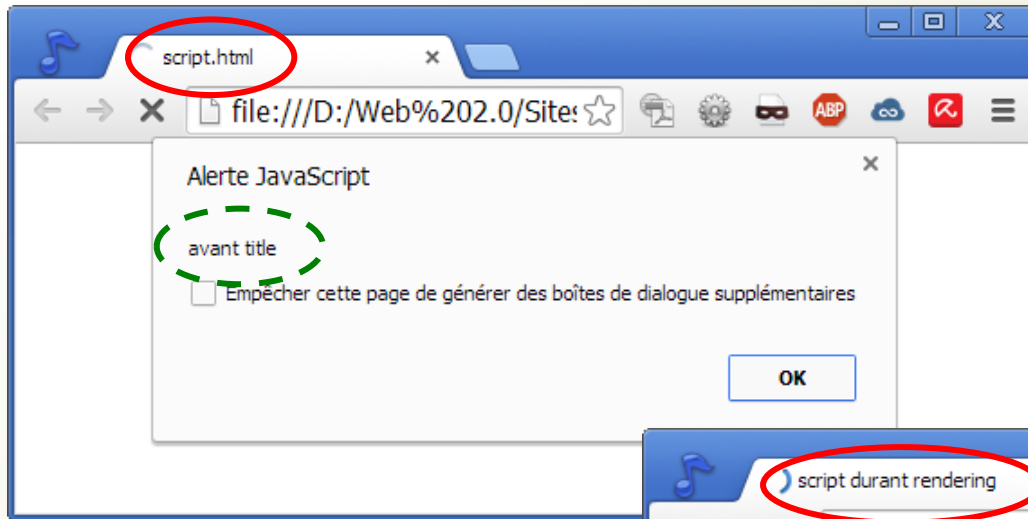
EN ATTENDANT MIEUX   
ENTRE `<script>` `</script>`  
... ET JUSTE AVANT `</body>`

FORMULAIRES,  
API DU DOM,  
PROGRAMMATION ÉVÉNEMENTIELLE,  
SCRIPT EXTERNE DANS `<head>`

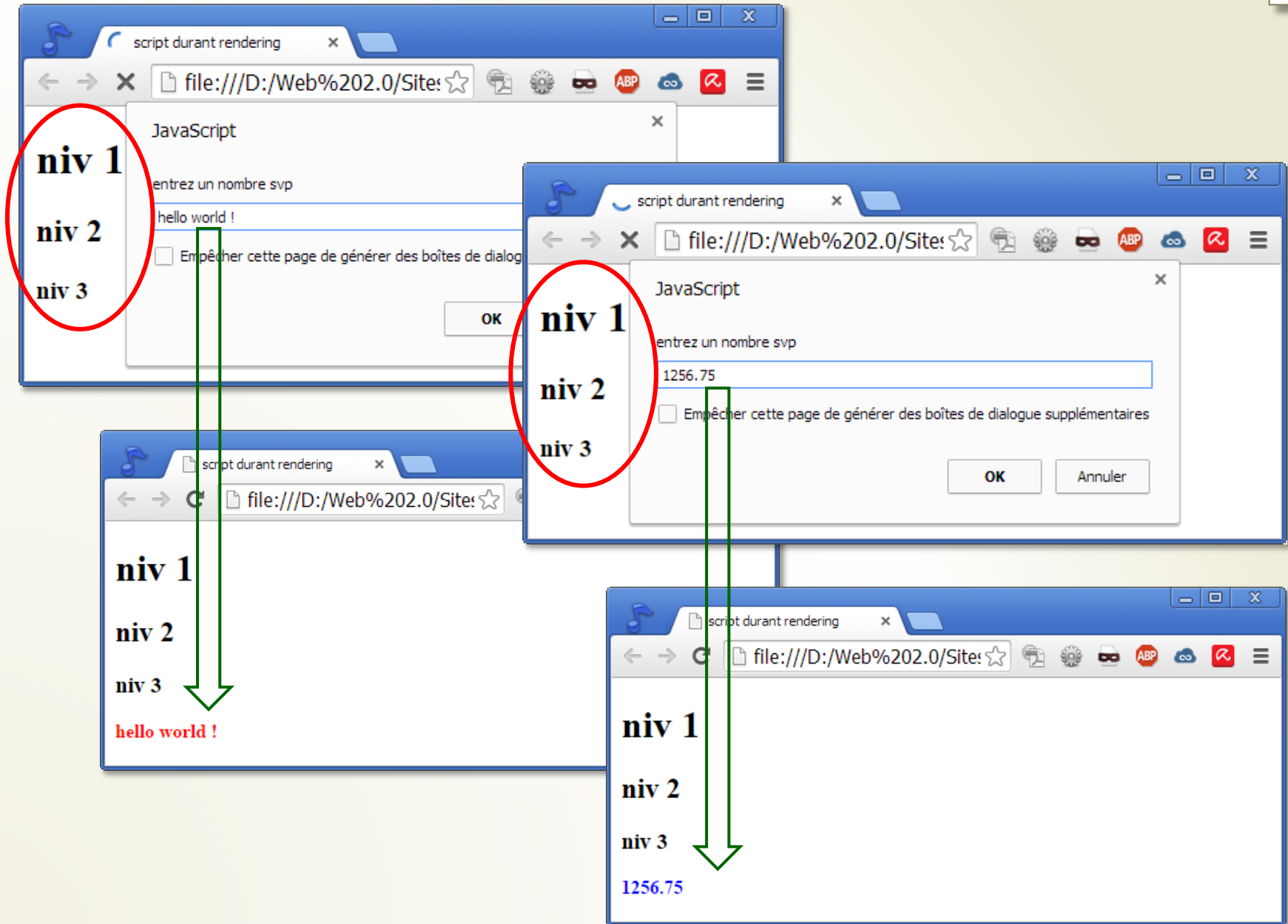


## UN EXEMPLE ...

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=utf-8>
    <script>alert('avant title');</script>
    <title>script durant rendering</title>
    <script>alert('après title');</script>
    <style>
      p {font-weight: bold;}
      .ok {color: blue;}
      .err {color: red;}
    </style>
  </head>
  <body>
    <h1>titre niveau 1</h1>
    <script>alert('je crée un <h2>');</script>
    <script>document.write('<h2>titre niveau </h2>');</script>
    <h3>titre niveau 3</h3>
    <script>
      var reponse, nombre;
      reponse = prompt('entrez un nombre svp');
      nombre = parseFloat(reponse);
      if (isNaN(nombre)) {
        document.write('<p class=err>' + reponse + '</p>'); }
      else {
        document.write('<p class=ok>' + nombre + '</p>'); }
    </script>
  </body>
</html>
```









MANIPULATION DE LA PAGE

INTRODUCTION AU  
DOCUMENT OBJECT MODEL

??? DOM ???

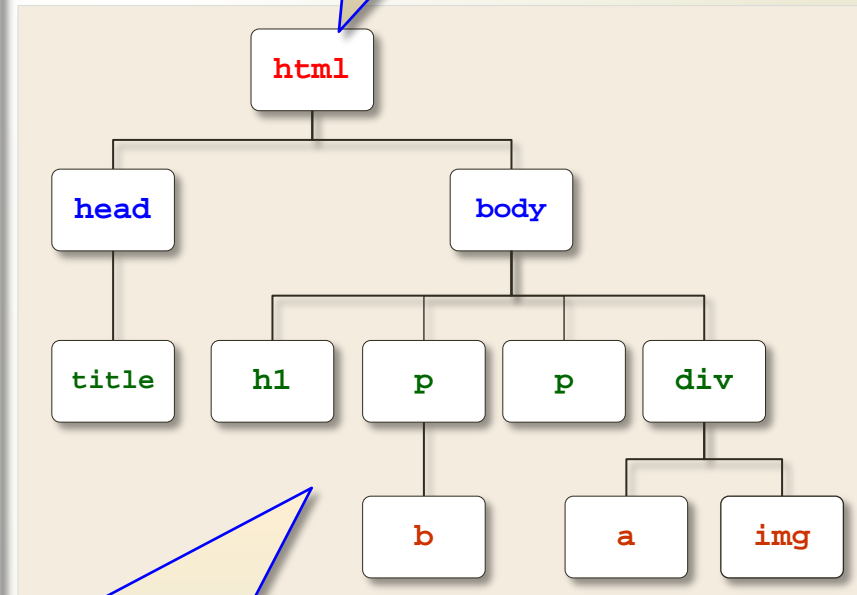
INTERFACE PROGRAMMATIQUE  
VISION STRUCTURELLE BASIQUE

représentation interne de la hiérarchie

- 1 balise peut contenir N balises 'enfants'
- 1 balise est contenue dans 1 seule balise 'parent'

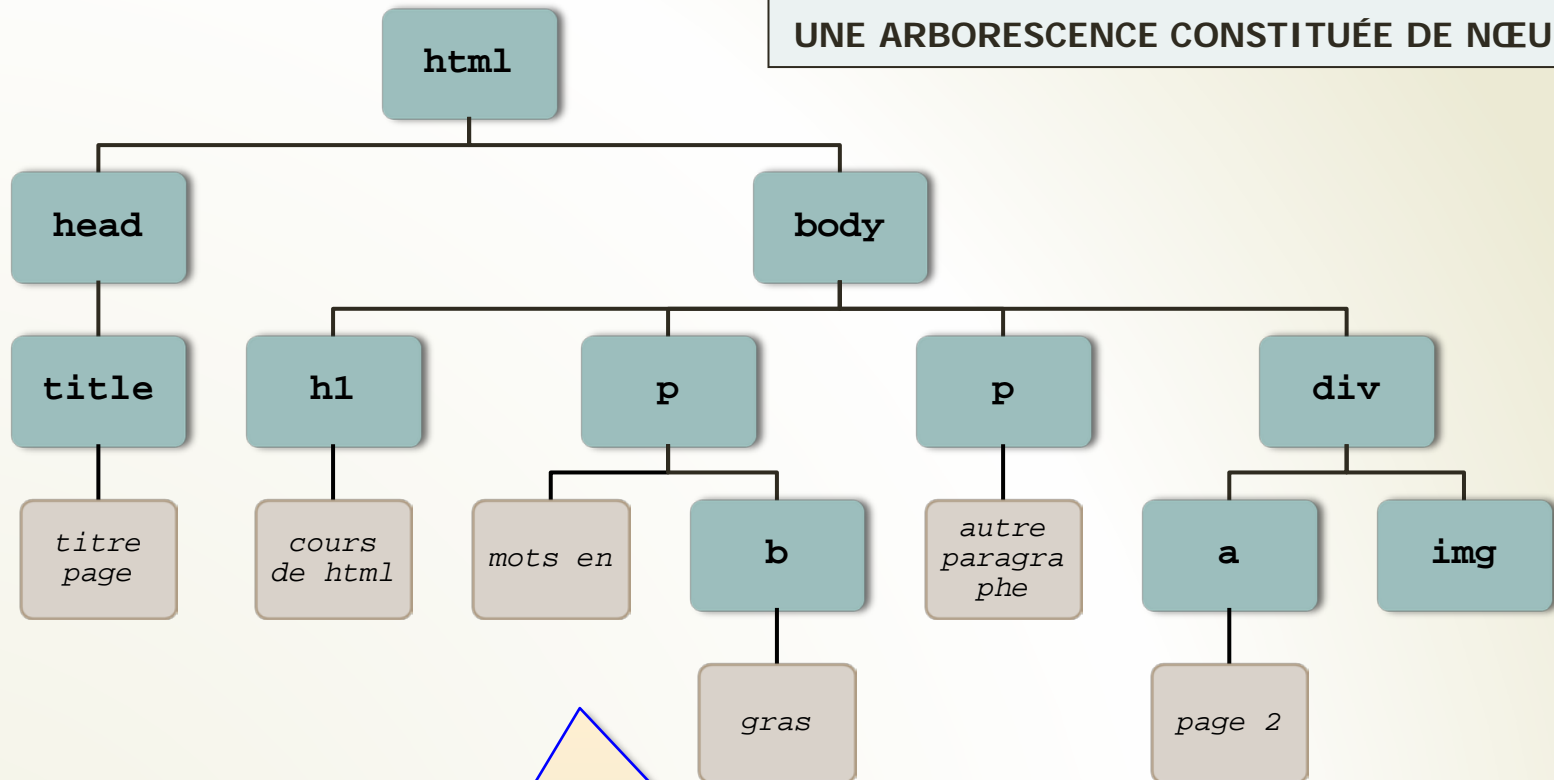
```
<!DOCTYPE html>
<html>
  <head>
    <title>titre page</title>
  </head>
  <body>
    <h1>cours de html</h1>
    <p>mots en <b>gras</b></p>
    <p>autre paragraphe</p>
    <div>
      <a href=page2.html>page 2</a>
      <img src=maPhoto.jpg alt=moi>
    </div>
  </body>
</html>
```

représentation basique : uniquement les éléments



### INTERFACE PROGRAMMATIQUE *VISION PLUS RÉELLE*

UNE ARBORESCENCE CONSTITUÉE DE NŒUDS



représentation réelle :

- les balises sont des nœuds de type "element"
- leur contenu est un nœud de type "text"
- les attributs ne sont pas représentés et ne sont pas des nœuds



### ÉCRIRE DANS LA PAGE HTML (1)

#### **BUT(s) :**

- SE DÉBARRASSER DE **DOCUMENT.WRITE()** (QUI EST DESTRUCTEUR POUR LE CONTENU DE LA PAGE)
- AFFECTER UN CONTENU À UN ÉLÉMENT HTML DE LA PAGE

**PRÉREQUIS :** L'ÉLÉMENT HTML DOIT ÊTRE IDENTIFIABLE SANS ÉQUIVOQUE

- UTILISER **L'ATTRIBUT ID** DANS LA BALISE DE L'ÉLÉMENT
- UTILISER **DOCUMENT.GETELEMENTBYID()**



## ÉCRIRE DANS LA PAGE HTML (2) : LA PAGE HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>exemple DOM</title>
  </head>
  <body>
    <h1>exemple DOM</h1>
    <p id=msg></p>
  </body>
</html>
```



<p> est vide !  
on veut lui donner un  
contenu via JS ...

... pour cela,  
il faut l'identifier !



## ÉCRIRE DANS LA PAGE HTML (3) : MODE CONSOLE EN JS

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>exemple DOM</title>
  </head>
  <body>
    <h1>exemple DOM</h1>
    <p id=msg></p>
  </body>
</html>
```



```
Developer Tools - file:///I:/S...
Elements >>
<top frame>
> document.getElementById("msg")
  <p id="msg"></p>
> |
```



```
Developer Tools - file:///I:/Site_01/exempleDOM1.html
Elements Network Sources Timeline Profiles >>
<top frame>
> document.getElementById("msg")
  <p id="msg"></p>
> document.getElementById("msg").innerHTML="hello world !"
  "hello world !"
> |
```



### ÉCRIRE DANS LA PAGE HTML (4) : EXPLICATION

```
<p id=msg>contenu</p>
```



camelCase !!!!

```
getElementById("<idName>")
```

renvoie tout l'élément (**objet**)



```
getElementById("<idName>").innerHTML
```

permet l'accès au contenu (**propriété**) de l'objet en lecture ou en écriture



préfixés par document.

```
getElementById("<idName>").innerHTML
```

acquisition (lecture) du contenu de l'élément

nommage !!!!

```
getElementById("<idName>").innerHTML = 'nouveau contenu'
```

affectation (écriture destructive) du nouveau contenu de l'élément





## ÉCRIRE DANS LA PAGE HTML (5) : SCRIPT JS

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>exemple DOM</title>
  </head>
  <body>
    <h1>exemple DOM</h1>

    <p id=msg></p>

    <script>
      // acquisition de l'élément via une variable
      var el = document.getElementById( 'msg' );
      // écriture du contenu
      el.innerHTML = 'hello world !';
    </script>

  </body>
</html>
```



HTML : identifiant  
sans apostrophes

via et pas dans

JS : identifiant  
entre apostrophes





## innerHTML vs innerText

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>exemple DOM</title>
  </head>
  <body>
    <h1>exemple DOM</h1>
    <p id=msg>hello <b>world</b> !</p>
    -----
    <script>
      var el = document.getElementById( 'msg' );
      alert(el.innerHTML);
      alert(el.innerText);
    </script>
    -----
  </body>
</html>
```



hello &lt;b&gt;world&lt;/b&gt; !

hello world !

`getElementById("...").innerHTML`

lecture/écriture du contenu de l'élément

`getElementById("...").innerText`lecture/écriture du contenu **texte brut** de l'élément

**innerHTML** : créer en une fois une sous-arborescence

```
<p id=msg></p>
```

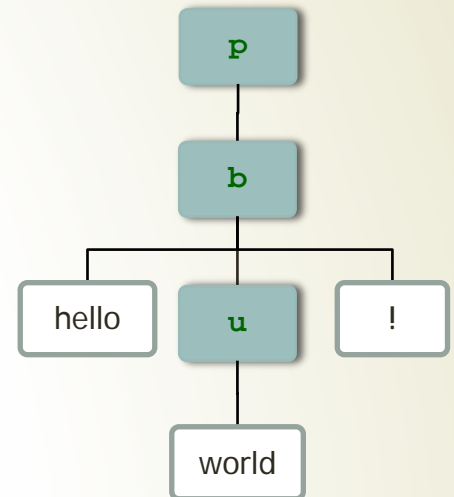
p

```
<script>
  var el = document.getElementById('msg');
  el.innerHTML = '<b>hello <u>world</u> !</b>';
</script>
```

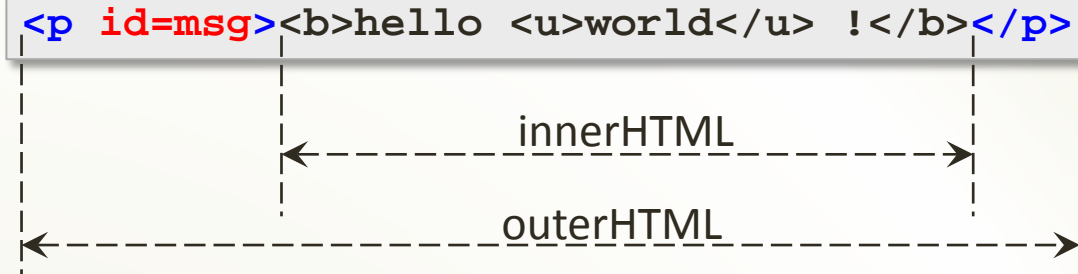
résultat

```
<p id=msg><b>hello <u>world</u> !</b></p>
```

innerHTML



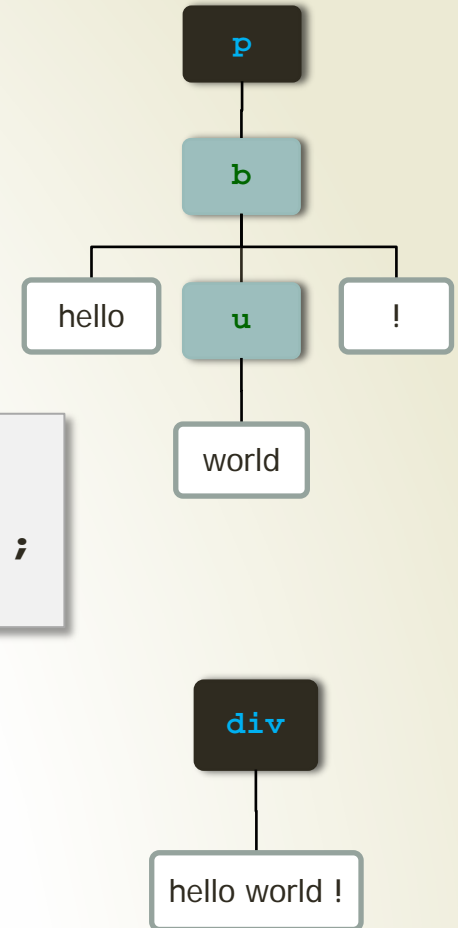
## outerHTML : accès à la balise 'conteneur'



```
<script>
  var el = document.getElementById('msg');
  el.outerHTML = '<div>' + el.innerText + '</div>';
</script>
```

résultat

```
<div>hello world !</div>
```




- 👉 innerHTML, innerText, outerHTML, outerText : non standard
- 👉 innerHTML : reconnu par tous les navigateurs

**innerHTML** : cher ! à n'utiliser que pour de petites interventions

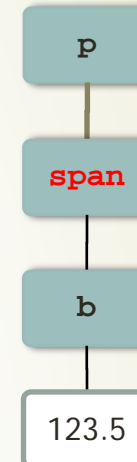
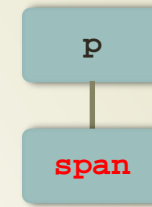
```
<p>prix à payer : <span id=prx></span>€</p>
```

```
<script>
  var prix;
  prix = // ... calculé ici disons 123.5
  var p = document.getElementById('prx');
  p.innerHTML = '<b>' + prix + '</b>';
</script>
```

résultat



```
<p>prix à payer : <span id=prx><b>123.5</b></span>€</p>
```



**innerHTML** : cher ! à n'utiliser que pour de petites interventions

```
<table id=tb>
  <tr>
    <th>N°</th>
    <th>Lib</th>
  </tr><tr>
    <td>1</td>
    <td>aaa</td>
  </tr><tr>
    <td>2</td>
    <td>bbb</td>
  </tr><tr>
    <td>3</td>
    <td>ccc</td>
  </tr><tr>
    <td>4</td>
    <td>ddd</td>
  </tr><tr>
    <td>5</td>
    <td>eee</td>
  </tr>
</table>
```

✎ ajouter une ligne à une table

*affection = remplacement  
de tout le contenu*

```
<script>
  var t = document.getElementById('tb');
  t.innerHTML += '<tr>' +
    '<td>6</td>' +
    '<td>fff</td>' +
    '</tr>';
</script>
```

N°	Lib
1	aaa
2	bbb
3	ccc
4	ddd
5	eee



N°	Lib
1	aaa
2	bbb
3	ccc
4	ddd
5	eee
6	fff

## Modification du style d'un élément (1)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>exemple DOM</title>
    <style>
      #msg { color : red; font-weight : bold; }
    </style>
  </head>
  <body>
    <h1>exemple DOM</h1>
    <p id = msg>hello world !</p>
  </body>
</html>
```

sélecteur

propriété

valeur



en CSS :

- jamais d'apostrophes,
- liste de déclarations avec ;
- tiret pour les propriétés à noms composés : **font-weight**



## Modification du style d'un élément (2)

```
<style>
  #msg {color:red; font-weight:bold;}
</style>
```



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>exemple DOM</title>
  </head>
  <body>
    <h1>exemple DOM</h1>
    <p id = msg>hello world !</p>
    <script>
      var el = document.getElementById( 'msg' );
      el.style.color = 'blue';
      el.style.fontWeight = 'normal';
    </script>
  </body>
</html>
```



propriété

valeur

en JS :

- apostrophes,
- 1 instruction par propriété,
- camelCase pour les propriétés à nom composé : **fontWeight**