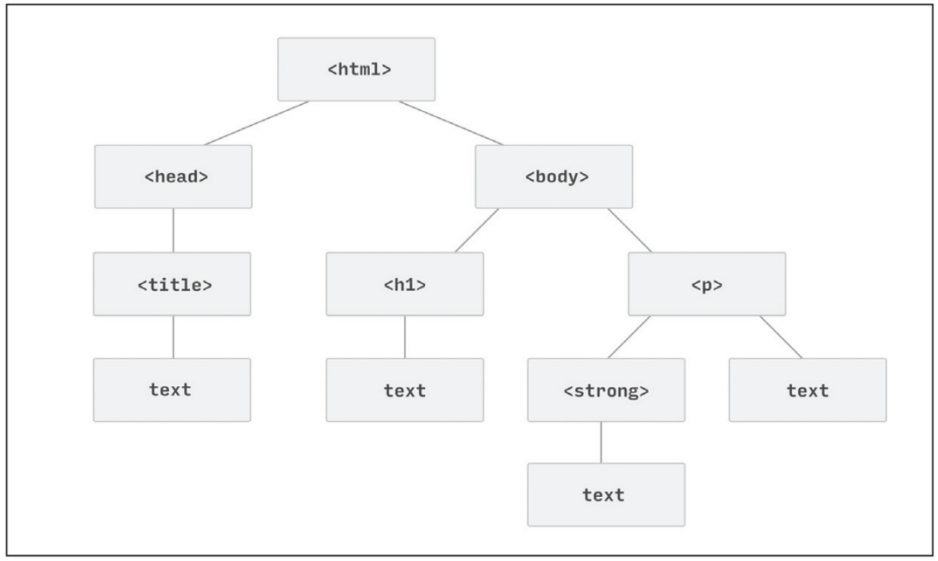
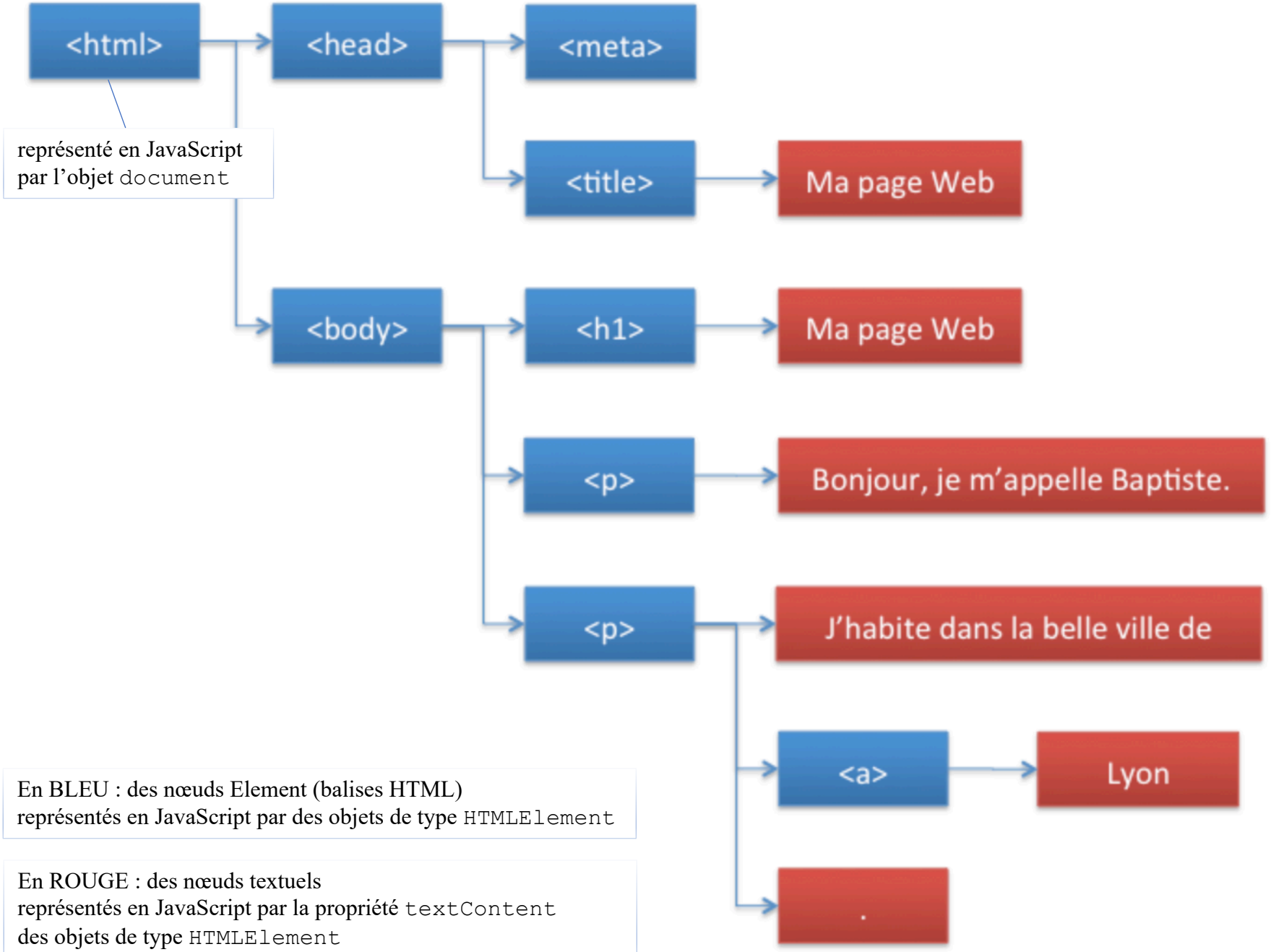
JAVASCRIPT POUR LE WEB : Le DOM (*Document Object Model*) et la POO *(Programmation Orientée Objet)*

- Une page Web est un document contenant du texte structuré par des balises HTML (`<h…>` pour les titres, `<p>` pour les paragraphes, etc.).
- Le **DOM**, ou ***Document Object Model (*Modèle Objet du Document*)*** représente la page Web sous forme d'un arbre et définit la manière dont JavaScript peut la manipuler. Le DOM est une interface de programmation, ou **API** (***Application Programming Interface***).



Le DOM représente le document par une hiérarchie d'objets
Chaque nœud de l'arbre logique du document est un objet en Javascript

- Chaque nœud dans l'arbre logique de la page Web correspond à un objet du DOM :
  - la fenêtre du navigateur est représentée par l'objet `window` ;
  - la racine de l'arbre logique (l'élément `<html>` de la page) est représentée par l'objet `document` ;
  - chacune des balises HTML est représentée par un objet de type `HTMLElement ;`
- Chaque objet du DOM possède des propriétés et des méthodes utilisables avec JavaScript : par exemple,
  - chacun des attributs d'une balise HTML est représenté par une propriété de l'objet `HTMLElement ;`
  - `nodeType` est une propriété qui représente le type de nœud, `childNodes` contient une collection de nœuds enfants ;
  - `getElementById` est une méthode de l'objet `document` qui renvoie un objet de type `HTMLElement` représentant la balise qui a cet `id` ;
- Les nœuds peuvent aussi avoir des gestionnaires d'événements qui se déclenchent lorsqu'un événement se produit sur un nœud.



représenté en JavaScript
par l'objet `document`

En BLEU : des nœuds Element (balises HTML)
représentés en JavaScript par des objets de type `HTMLElement`

En ROUGE : des nœuds textuels
représentés en JavaScript par la propriété `textContent`
des objets de type `HTMLElement`

LES OBJETS EN JAVASCRIPT : LES BASES DE LA POO (Programmation Orientée Objet)

OBJET, PROPRIÉTÉS, MÉTHODES

Définitions :

- Propriété = une variable associée à un objet
- Méthode = une fonction associée à un objet
- Objet = liste de propriétés et de méthodes qui vont ensemble
- Classe = ensemble d'objets : les objets sont regroupés en classes.
  Une classe est une représentation abstraite des objets.
  Un objet est une *instance* (c'est-à-dire un exemplaire particulier) de sa classe.
  Quand un objet appartient à une classe, on dit qu'il est de type … (*le nom de la classe*).
  Exemple : une balise `p` (dans un document HTML) est un objet de type `HTMLElement`.

On écrit toujours

- **`OBJET.propriété = valeur;`**
- **`OBJET.méthode(`*`paramètre`*`);`**

Exemple :

Il faut avoir ici une expression qui s'évalue à un objet de type `HTMLElement`

- `element.className = "pair";`
- `document.getElementById("nav");`

Contraintes :

- On ne peut pas écrire la propriété ou la méthode toute seule [par exemple, on ne peut pas écrire `getElementById("nav");`]
  il faut toujours écrire, devant la propriété ou devant la méthode, le nom de l'objet auquel elle appartient
  [par exemple, il faut écrire `document.getElementById("nav");`].
  Exception : l'objet `window` peut être omis [par exemple, on devrait écrire `window.alert('hello');` , mais on peut écrire `alert('hello');`].
- Il faut obligatoirement que la propriété ou la méthode qu'on place à droite de l'objet
  soit bien dans la liste des méthodes et des propriétés de cet objet (ou dans celle de son parent, dont il hérite, voir ci-après).
  On ne peut par exemple par écrire `document.value` (`value` est une propriété de l'objet `HTMLInputElement`, pas de l'objet `document`)

LES OBJETS EN JAVASCRIPT : LES BASES DE LA POO (Programmation Orientée Objet)

LE CHAÎNAGE

Le principe :

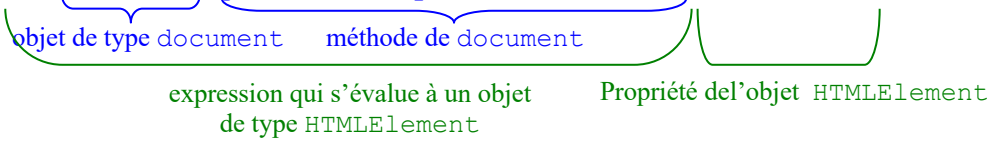
On écrit toujours

- **`OBJET.propriété`**
- **`OBJET.méthode`**

Cet objet peut être soit un nom très simple, soit une expression plus complexe qui s'évalue à un objet du bon type

Exemple :

- Je peux écrire
  `document.getElementById("monBouton");` (exemple simple)
- Imaginons qu'on a un code HTML `<input id="monBouton" type="button" value="Affiche bonjour" />`
  `document.getElementById("monBouton");` est une expression qui s'évalue à un objet de type `HTMLElement`
  (parce que la méthode `getElementById("monBouton")`, qui est donc une fonction, retourne un nœud `HTMLElement`, voir documentation)
- Je peux donc écrire
  `var monBouton = document.getElementById("monBouton");`
  Cette variable contient donc un objet de type `HTMLElement`.
- Donc maintenant, je peux écrire-
  `monBouton.className = "boutonJS";`
  (parce que la propriété `className` est une propriété de la classe `HTMLElement` et que `monBouton` est bien un objet qui appartient à la classe `HTMLElement`)
- => je peux aussi écrire
  `document.getElementById("monBouton").className = "boutonJS";`



- On peut enchaîner `OBJET.méthode().méthode().propriété = valeur;` (c'est le chaînage)
  Condition : il faut que le `OBJET.méthode()` qui est devant retourne un objet qui possède la méthode ou la propriété qu'on écrit après.

LES CLASSES D'OBJETS DE JAVASCRIPT

RELATIONS D'EMBOÎTEMENT

Le principe :

- ***Certains objets sont eux-mêmes des propriétés d'autres objets***

Ce qui implique qu'on pourra écrire

- **`OBJET.OBJET.propriété`**
- **`OBJET.OBJET.méthode`**

À nouveau, chaque objet peut être soit un nom très simple, soit une expression plus complexe qui s'évalue à un objet du bon type

Exemple :

- **`style`** est une propriété de l'objet **`HTMLElement`** (voir documentation)
- Donc, je peux écrire une expression du genre « un **objet** de type `HTMLElement` »**`.style`**
  Par exemple, je peux écrire

```
document.getElementById("monBouton").style
```

objet de type `document` — méthode de `document`

expression qui s'évalue à un objet de type `HTMLElement` — propriété de l'objet `HTMLElement`

- Mais style est en même temps un objet de la classe **`Style`** (voir documentation).
  Cela veut dire que, à l'intérieur de cette classe **`Style`**, il y a encore des propriétés et des méthodes que je peux utiliser.
- Donc, ça s'emboîte, ça s'enchaîne : je peux continuer l'expression précédente avec `style.propriété` ou `style.méthode`
- Je pourrais donc par exemple écrire

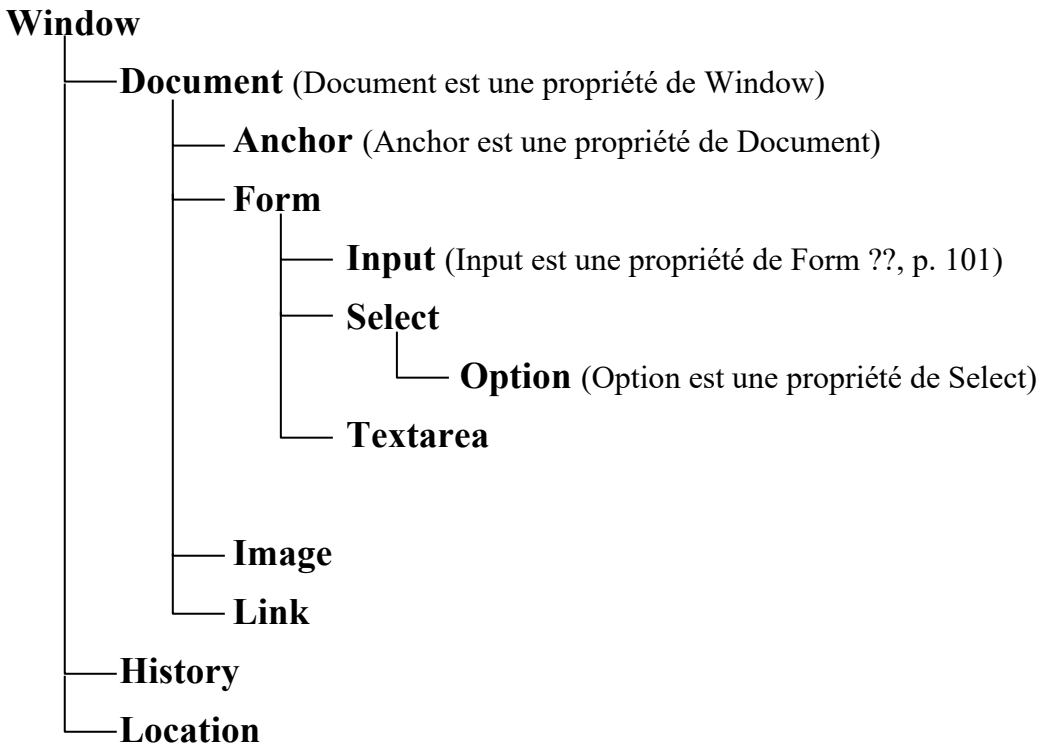
```
document.getElementById("monBouton").style.backgroundColor="red" ;
```

objet de type `Style` — propriété de `Style`

LES CLASSES D'OBJETS DE JAVASCRIPT

RELATIONS D'EMBOÎTEMENT

Dans le DOM et dans JS côté client



Element

Style (Style est une propriété de Element)

LES CLASSES D'OBJETS DE JAVASCRIPT

RELATIONS D'HÉRITAGE

Le principe :

- ***Certaines classes sont des sous-classes d'autres classes : elles héritent alors les propriétés et les méthodes de leur classe parent***
- Une classe possède ses propriétés et ses méthodes (un objet est un ensemble de propriétés et de méthodes)
  Exemple :

Element
- **Propriétés**
  - `className`
  - `style`
- **Méthodes**
  - `getAttribute(`*`nom`*`)`
  - `setAttribute(`*`nom`*`)`

- Si cette classe est une sous- classe d'une autre classe, elle possède ses propriétés et ses méthodes **PLUS** les propriétés et les méthodes de sa classe parent.
  Exemple :
  `Element` est une sous-classe de `Node` (on dit aussi que `Element` hérite de `Node`)

Node
- **Propriétés**
  - `firstChild`
  - `nextSibling`
- **Méthodes**
  - `addEventListener(`*`type, observateur, utiliseCapture`*`)`
  - `appendChild(`*`nouvelEnfant`*`)`

Element
- **Propriétés**
  - `className`
  - `style`
- **Méthodes**
  - `getAttribute(`*`nom`*`)`
  - `setAttribute(`*`nom`*`)`

⟹ Un objet de type `HTMLElement`
possède ses méthodes et ses propriétés à lui
+ les méthodes et les propriétés de `Element`
+ les méthodes et les propriétés de `Node`
+ les méthodes et les propriétés de `EventTarget`
+ les méthodes et les propriétés de `Object`

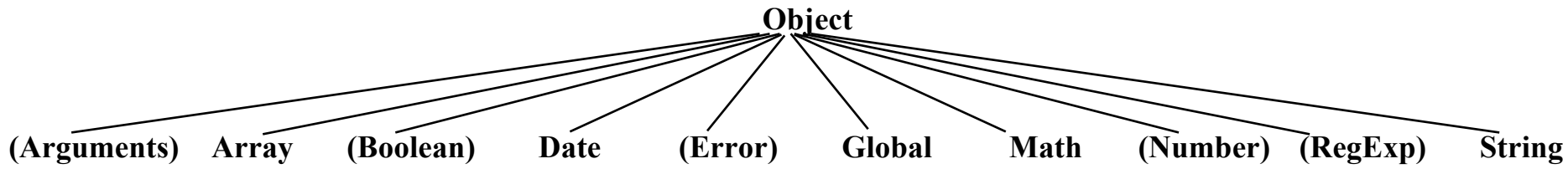
- Ceci implique qu'on pourra écrire :

```
document.getElementById("monBouton").addEventListener("click", direBonjour, false);
```

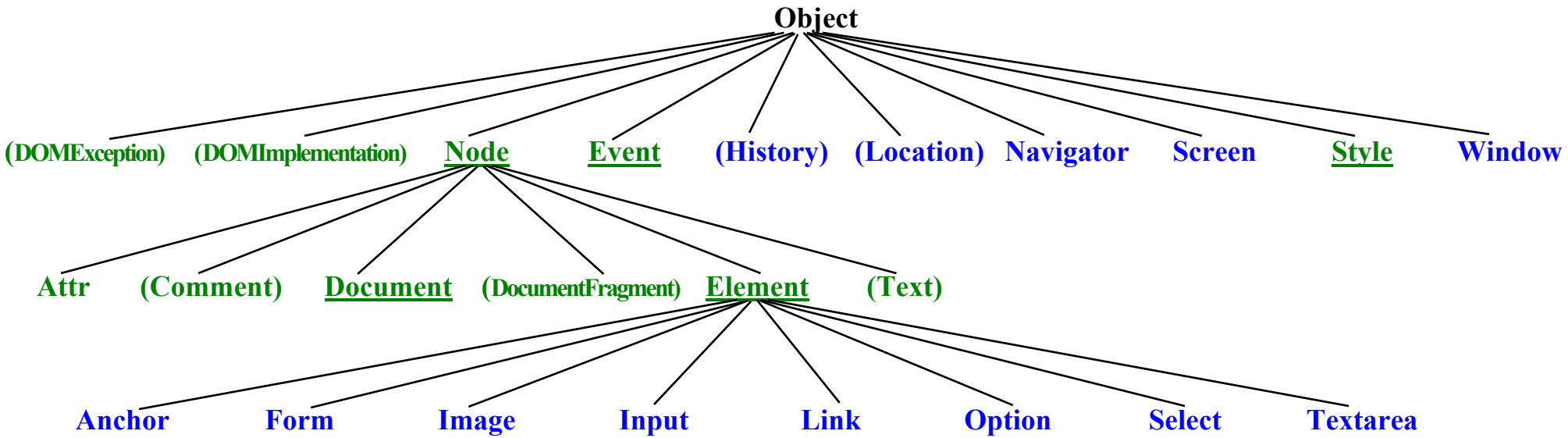
LES CLASSES D'OBJETS DE JAVASCRIPT

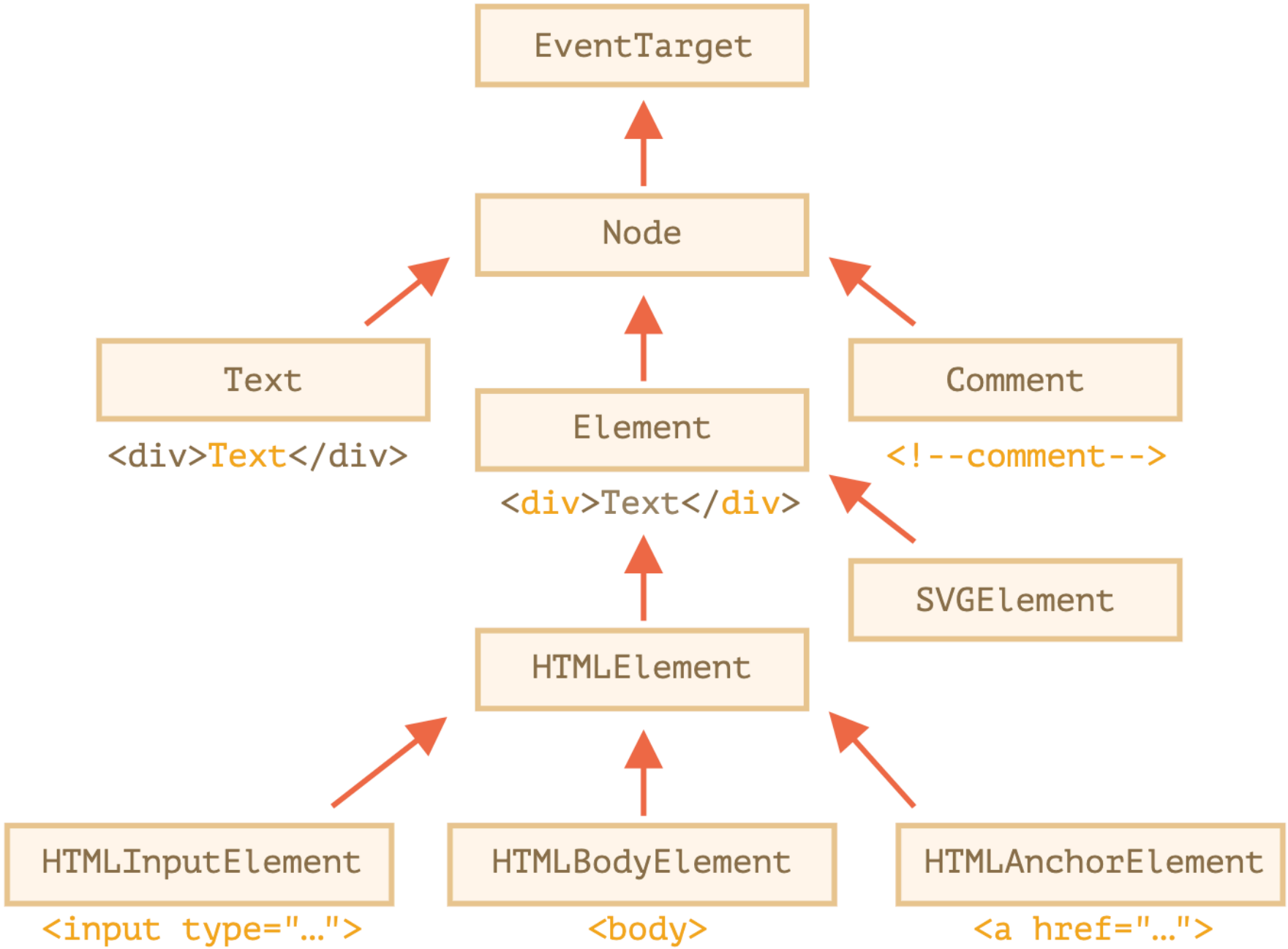
RELATIONS D'HÉRITAGE

1. Le noyau de JS (1.0, 1.1, 1.2, 1.5)



2. Le DOM et JS côté client




EventTarget
Node
Text
Text
Element
Text
Comment
<!--comment-->
SVGElement
HTMLElement
HTMLInputElement
<input type="…">
HTMLBodyElement
<body>
HTMLAnchorElement
<a href="…">

API = « Application Programming Interface »
DOM = « Document Object Model »

Anchor	JS1.2cc	hérite de Element
Applet	JS1.2cc	
Arguments	JS1.1Noyau	
Array	JS1.1Noyau	
Attr	DOM1	hérite de Node
Boolean	JS1.1Noyau	
Comment	DOM1	hérite de Node
DOMException	DOM1	
DOMImplementation	DOM1	
Date	JS1.0Noyau	
Document	JS1.0cc, DOM1	hérite de Node
DocumentFragment	DOM1	hérite de Node
Element	DOM1	hérite de Node
Error	JS1.5Noyau	hérite de Object
Event	DOM2	
Form	JS1.0cc	hérite de Element
Function	JS1.0cc	
Global	JS1.0Noyau	
History	JS1.0cc	
Image	JS1.1cc	hérite de Element
Input	JS1.0cc	hérite de Element
Layer	N4cc	
Link	JS1.0cc	hérite de Element
Location	JS1.0cc	
Math	JS1.0Noyau	
Navigator	JS1.0cc	
Node	DOM1	
Number	JS1.1Noyau	
Object	JS1.0Noyau : Superclasse de TOUS les objets JS	
Option	JS1.0cc	hérite de Element
RegExp	JS1.2Noyau	
Screen	JS1.2cc	
Select	JS1.0cc	hérite de Element
String	JS1.0Noyau	
Style	DOM2	= propriété de Element !
Text	DOM1	hérite de Node
Textarea	JS1.0cc	hérite de Element
Window	JS1.0cc	

Arguments	JS1.1Noyau	
Array	JS1.1Noyau	
Boolean	JS1.1Noyau	
Date	JS1.0Noyau	
Error	JS1.5Noyau	hérite de Object
Global	JS1.0Noyau	
Math	JS1.0Noyau	
Number	JS1.1Noyau	
Object	JS1.0Noyau : Superclasse de TOUS les objets JS	
RegExp	JS1.2Noyau	
String	JS1.0Noyau	
Node	DOM1	
Attr	DOM1	hérite de Node
Comment	DOM1	hérite de Node
Document	DOM1, JS1.0cc	hérite de Node
DocumentFragment	DOM1	hérite de Node
Element	DOM1	hérite de Node
Text	DOM1	hérite de Node
Style	DOM2	= propriété de Element !
Anchor	JS1.2cc	hérite de Element
Form	JS1.0cc	hérite de Element
Image	JS1.1cc	hérite de Element
Input	JS1.0cc	hérite de Element
Link	JS1.0cc	hérite de Element
Option	JS1.0cc	hérite de Element
Select	JS1.0cc	hérite de Element
Textarea	JS1.0cc	hérite de Element
Applet	JS1.2cc	
DOMException	DOM1	
DOMImplementation	DOM1	
Event	DOM2	
Function	JS1.0cc	
History	JS1.0cc	
Layer	N4cc	
Location	JS1.0cc	
Navigator	JS1.0cc	
Screen	JS1.2cc	
Window	JS1.0cc	