3ème partie

JavaScript et HTML

Jean-Christophe DUBOIS
jean-christophe.dubois@univ-rennes.fr

Objet Window

Fenêtre du navigateur

Window

- L'objet window est l'objet principal dans JavaScript, il est le parent de chaque objet qui compose la page web
- L'objet window est l'élément implicite
- Il contient :
  - l'objet Document : la page HTML
  - l'objet Location : informations sur l'adresse de la page
  - l'objet History : historique des pages visitées
  - l'objet Navigator : informations sur le navigateur
  - l'objet Screen : informations sur l'écran de l'utilisateur
  - l'objet Cookies : mémorisation des cookies

Window

- Exemple : Objet Navigator

Afficher les informations du navigateur suivantes :

- Le nom

```
console.log(window.navigator.appCodeName);     // Mozilla
```

- La version

```
console.log(window.navigator.appName);    // Netscape
```

- Le système d'exploitation (plateforme)

```
console.log(window.navigator.platform);     // MacIntel
```

Propriétés de Window

- Propriétés de window :
  - closed : indicateur de fermeture de fenêtre
  - name : nom de la fenêtre
  - screenLeft : coordonnée x de la fenêtre
  - screenTop : coordonnée y de la fenêtre
  - innerHeight : hauteur intérieure de la fenêtre
  - innerWidth : largeur intérieure de la fenêtre
  - outerHeight : hauteur totale de la fenêtre
  - outerWidth : largeur totale de la fenêtre
  - status : texte de la barre de statut du navigateur
  - parent : fenêtre parent de la fenêtre active
  - self : fenêtre active
  - top : fenêtre de plus au niveau
  - …

Méthodes de Window

- Méthodes de window:
  - *alert(s)*, *confirm(s)* et *prompt(s)* : affichage de fenêtres
  - *open("URL","nom","p1=v1, p2=v2") :* ouverture d'une fenêtre
  - *close()* : fermeture d'une fenêtre
  - *moveBy(x,y)* : déplacement d'une fenêtre de x et y (px)
  - *moveTo(x,y)* : positionnement d'une fenêtre en x et y (px)
  - *scrollBy(x,y)* : déplacement de l'ascenseur de x et y (px)
  - *scrollTo(x,y)* : positionnement de l'ascenseur en x et y (px)
  - *resizeBy(x,y)* : agrandissement de x et y (px)
  - *resizeTo(w,h)* : redimensionnement à w et h (px)
  - *focus()*, *blur()* : prise (1er plan) ou perte de focus (arr. plan)
  - *print()* : impression du contenu de la fenêtre

D.O.M.

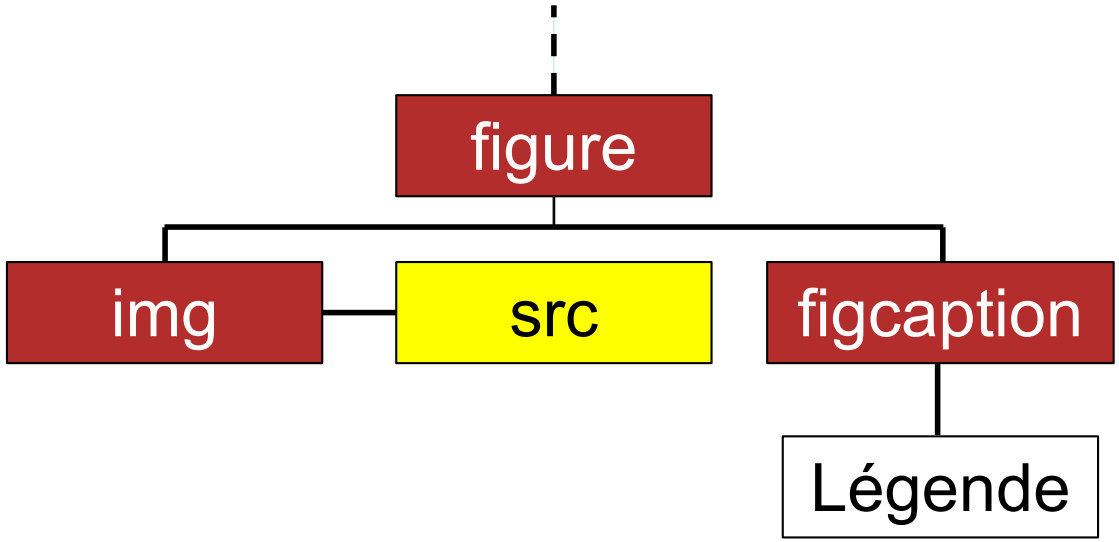
DOM – Document Object Model

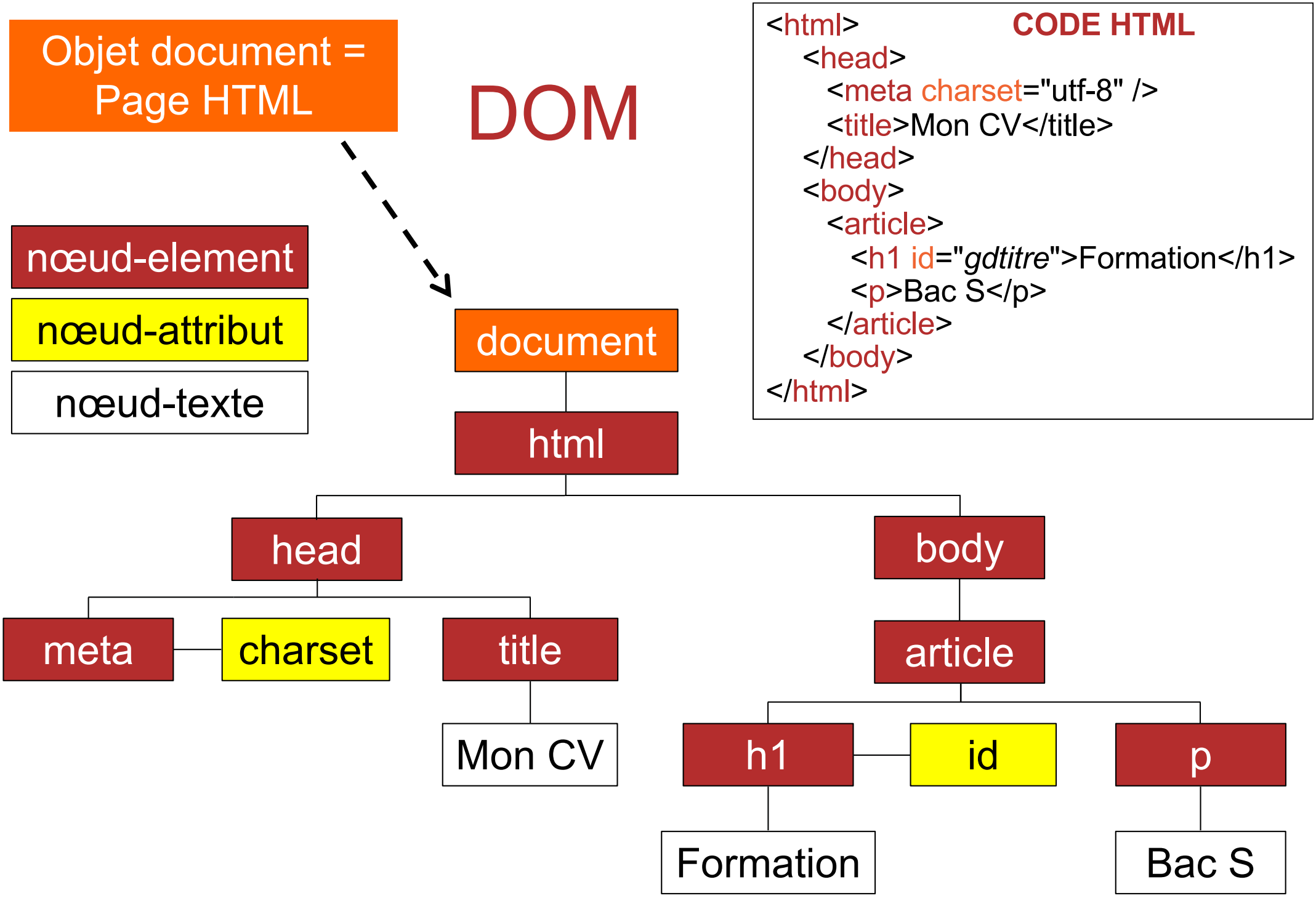
- Le Modèle d'Objet du Document (ou DOM) est une interface de programmation (Application Programming Interface) qui permet de représenter logiquement les documents HTML et XML.

  Le DOM codifie la manière dont un script peut accéder à un document pour le manipuler dynamiquement.
- Le DOM constitue le lien entre la page web et le script pour permettre :
  - la modification, l'ajout ou la suppression de code HTML,
  - la modification du code CSS,
  - le changement d'images,
  - la vérification de champs de formulaire
  - la définition d'action en réponse à des événements …

DOM – Document Object Model

- Le DOM est une structure logique orientée objet qui représente la page web sous une forme arborescente. L'arbre est constitué de nœuds (nodes en anglais).
- Le document HTML (objet document) est le nœud racine.
- Plusieurs types de nœuds peuvent être distingués :




Objet document = Page HTML
DOM
nœud-element
nœud-attribut
nœud-texte
document
html
head
body
meta
charset
title
article
Mon CV
h1
id
p
Formation
Bac S
CODE HTML
<html>
<head>
<meta charset="utf-8" />
<title>Mon CV</title>
</head>
<body>
<article>
<h1 id="gdtitre">Formation</h1>
Bac S
</article>
</body>
</html>

Objet Document

Page HTML chargée dans la fenêtre du navigateur

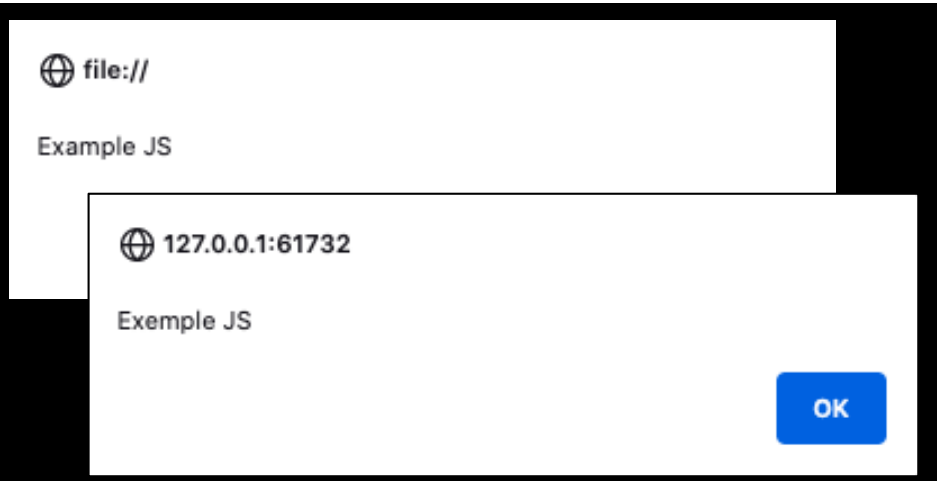
Propriétés de Document

- Principales propriétés du DOM :
  - URL : URL complète de la page HTML (ATTENTION en majuscules !)
  - title : référence au titre de la page
  - lastModified : date de dernière modification
  - head : référence à la tête de la page <head>
  - body : référence au corps de la page <body>
  - images : tableau de toutes les images insérées <img>
  - links : tableau des éléments <a> et <area> avec un attribut *href*
  - forms : tableau des formulaires insérés avec <form>
  - …

Propriétés de Document

- Exemple : modification de la propriété title

```
<!DOCTYPE html>
<html>
    <head>
        <title>Example JS</title>
    </head>
    <body>
        <script>
            alert(document.title);              // Affiche "Example JS"
            document.title = "Exemple JS";
            alert(document.title);              // Affiche "Exemple JS"
        </script>
    </body>
</html>
```



Méthodes d'accès de Document

- Méthodes d'accès aux éléments de document :
  - getElementById("nomId") retourne l'élément dont l'Id est passé en paramètre, null sinon

```
let element = document.getElementById("nomId");
```

  - getElementsByClassName("nomClass") retourne tous les éléments dont la classe est passée en paramètre

```
let elems = document.getElementsByClassName("nomClass");
```

  - getElementsByTagName("balise") retourne tous les éléments dont la balise est passée en paramètre

```
let elems = document.getElementsByTagName("article");
```

Remarque : * remplace toutes les balises

Méthodes d'accès de Document

- Méthodes d'accès aux éléments de document (suite) :
  - querySelector(nomId et/ou nomClass) retourne le premier élément porteur du sélecteur ou groupe de sélecteurs passé en paramètre.

```
let element = document.querySelector("#menu p.intro");
```

  - querySelectorAll(nomId et/ou nomClass) retourne tous les éléments porteurs du sélecteur ou groupe de sélecteurs passé en paramètre.

```
let elems = document.querySelectorAll("main > p");
let elems = document.querySelectorAll("p.debut, p.fin");
```

Méthodes d'accès de Document

- Exercice : Méthodes d'accès aux éléments du DOM
  - Contenu du 1er *.lang*
  - Contenu du 2ème *p*
  - Eléments contenus dans le 2ème *.bloc*
  - Contenu de *#titre1*
  - Contenu du 2ème *#deux .lang*

```
<body>
    <h1>Diplôme</h1>
    <article id="un" class="bloc">
        <h2 id="titre1">Diplôme</h2>
        <p class="diplo">Bac S</p>
        <p class="diplo">BUT</p>
    </article>
    <article id="deux" class="bloc">
        <h2 id="titre2">Langages</h2>
        <p class="lang">HTML</p>
        <p class="lang">CSS</p>
    </article>
</body>
```

Chaque méthode d'accès ne doit être utilisée qu'une seule fois !
La propriété innerHTML permet l'accès au contenu d'un Elément

Méthodes de création de Document

- Méthodes de création de nœuds du DOM :
  - createElement() : création d'un nœud élément
  - createAttribute() : création d'un nœud attribut
  - createTextNode() : création d'un nœud texte

```
let paragraphe = document.createElement("p"); // nœud Element
let attribut = document.createAttribute("class");  // nœud Attribut
attribut.value = "maclasse";
paragraphe.setAttributeNode(attribut);                // nœud Texte
let texte = document.createTextNode("Texte paragraphe");
paragraphe.appendChild(texte);
document.body.appendChild(paragraphe);
```

  - createComment() : création d'un nœud commentaire

Méthodes de Document

- Méthode d'écriture dans Document :
  - write() : insertion de texte/balises HTML dans le document

Attention : write() s'utilise pour la création d'une fenêtre.
Si la page est déjà chargée, **write()** supprime le code existant !
Il est préférable d'utiliser les méthodes de création: createElement, createAttribute, createTextNode

```
maFenetre=window.open("", "pop", "width=300, height=100");
maFenetre.document.write("<h1>Titre inséré</h1>");
maFenetre.document.write("<p>Paragraphe inséré</p>");
```

Objet Element

Sous-classe de Node

Balises de la page HTML

Propriétés d'Element

- Propriétés permettant d'accéder à des **attributs** :
  - className : accès au contenu (string) de l'attribut ***class***

ATTENTION class peut contenir une liste de classes

  - classList : accès au contenu (tableau) de l'attribut ***class***

Méthodes
- add() : ajoute une classe
- remove() : supprime une classe
- toggle() : ajoute/supprime une classe absente/présente
- contains() : vérifie la présence de la classe
- replace() : remplace une classe par une autre

  - id : contenu de l'attribut ***identifiant***

Propriétés d'Element

- Propriété permettant d'accéder à la balise HTML
  - tagName : balise HTML de l'élément

- Propriétés permettant d'accéder au contenu :
  - innerHTML : contenu avec **code HTML et texte**
  - textContent : contenu **textuel** (sans les balises HTML)

- Propriétés permettant d'accéder à des **attributs** :
  - value : contenu de l'attribut **valeur** d'un **champ** (formulaire)
  - name : contenu de l'attribut **name** (formulaire)
  - style : contenu de l'attribut **style**

Propriétés d'Element

- Exemple : Afficher les classes appliquées à #monId

```
<head>
    <style>
            .monStyle1 {color: blue; }
            .monStyle2 {background-color: yellow;}
    </style>
</head>
<body>
    <header id="monId" class="monStyle1 monStyle2">
    </header>
    <script>



    </script>
</body>
```

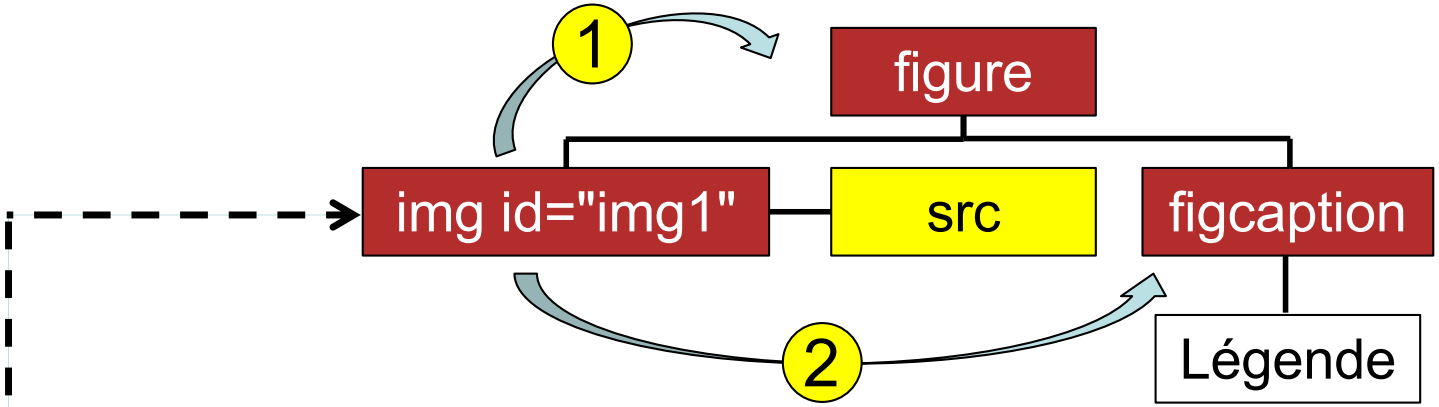
Propriétés d'Element

- Solution : Afficher les classes appliquées à #monId

```
<head>
    <style>
            .monStyle1 {color: blue; }
            .monStyle2 {background-color: yellow;}
    </style>
</head>
<body>
    <header id="monId" class="monStyle1 monStyle2">
    </header>
    <script>
        let liste = document.getElementById("monId").className;
        alert(liste);          // Affichage de monStyle1et monStyle2
    </script>
</body>
```

Propriétés d'Element

- Propriétés pour se positionner par rapport à un nœud :



  - parentNode : Nœud parent

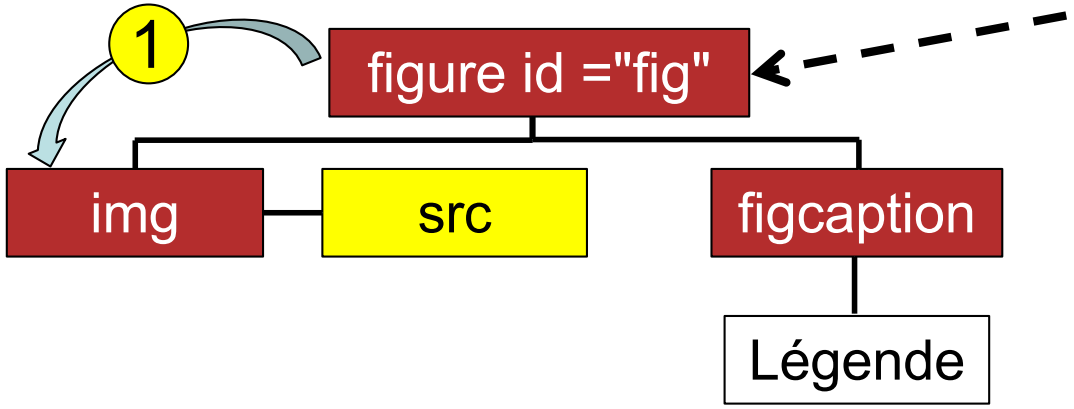
```
    let monImage = document.getElementById("img1");
1   let maFigure = monImage.parentNode;
```

  - previousSibling : Nœud frère précédent (tous types)
  - nextSibling : Nœud frère suivant (tous types)

```
2   let maFigCaption = monImage.nextSibling;
```

Propriétés d'Element

- Propriétés pour se positionner par rapport à un nœud :



  - childNodes : Tableau de tous les nœuds enfants (tous types)

  (childNodes.length : Nombre de nœuds enfants)

  - firstChild : Premier nœud enfant (tous types)

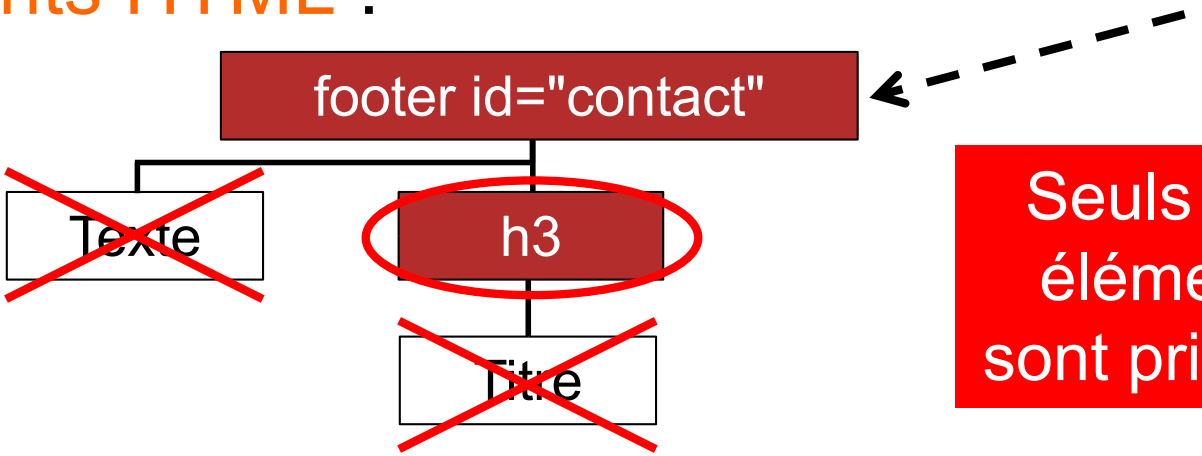
```
let maFigure = document.getElementById("fig");
let monImage = maFigure.firstChild;
```

1

  - lastChild : Dernier nœud enfant (tous types)

Propriétés d'Element

- Propriétés spécifiques aux nœuds constitués d'éléments HTML :



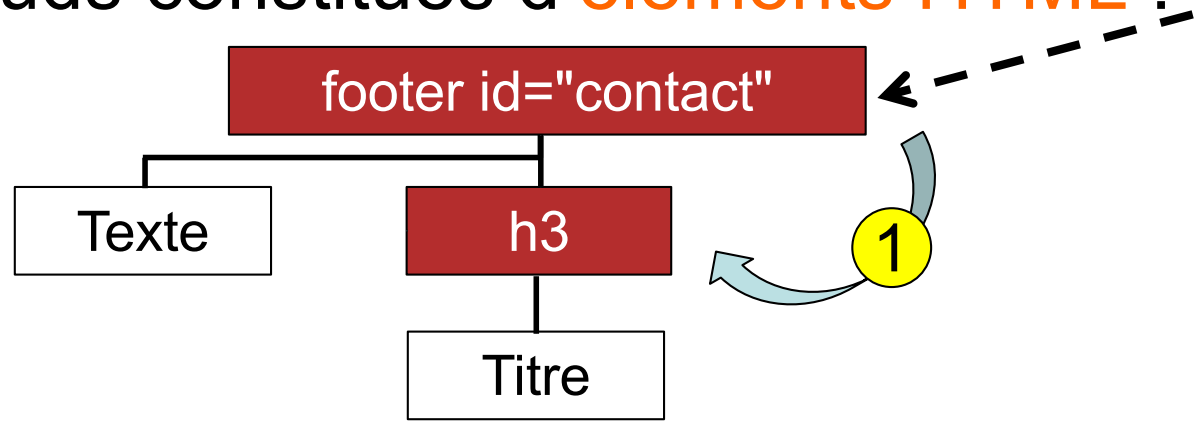
Seuls les nœuds éléments HTML sont pris en compte

  - children : Tableau de tous les éléments HTML enfants
  - childElementCount : Nombre de nœuds HTML enfants

```
let monContact = document.getElementById("contact");
alert(monContact.childElementCount);        // 1 seul enfant
alert(monContact.children[0].innerHTML;     // Titre
```

Propriétés d'Element

- Propriétés pour se positionner uniquement par rapport aux nœuds constitués d'éléments HTML :



  - firstElementChild/lastElementChild : Premier/Dernier nœud enfant constitué d'un élément HTML

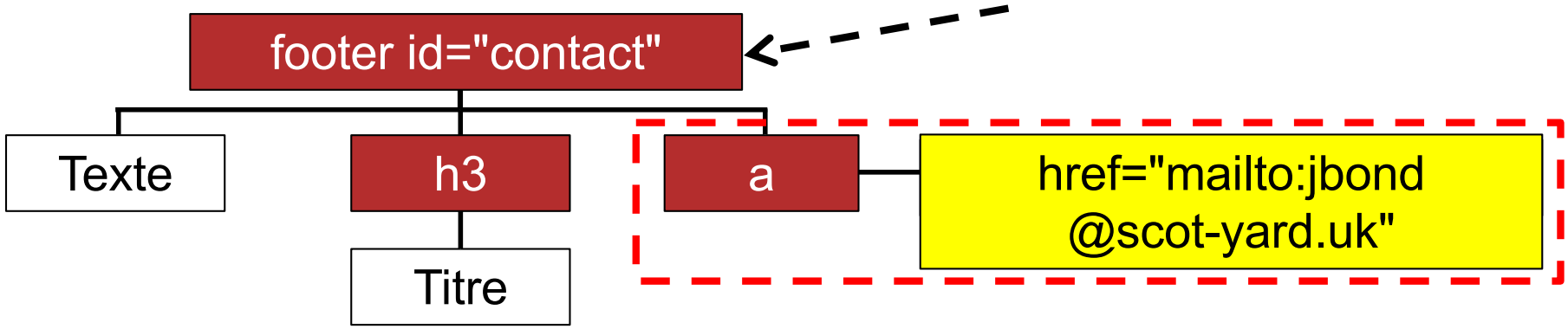
```
let monContact = document.getElementById("contact");
let monTitre= monContact.firstElementChild;
```
1

  - previousElementSibling/nextElementSibling : Nœud frère précédent/suivant constitué d'un élément HTML

Méthodes d'Element

- Méthodes de modification dynamique du DOM :
  - hasChildNodes() : teste si le nœud a des enfants (true/false)
  - appendChild() : insertion d'un nœud enfant dans un parent (avec body, l'ajout est en fin de page)

```
let monAdr = document.createElement("a");
let attribut = document.createAttribute("href");
attribut.value =  "mailto:jbond@scot-yard.uk";
monAdr.setAttributeNode(attribut);
monContact.appendChild(monAdr);
```



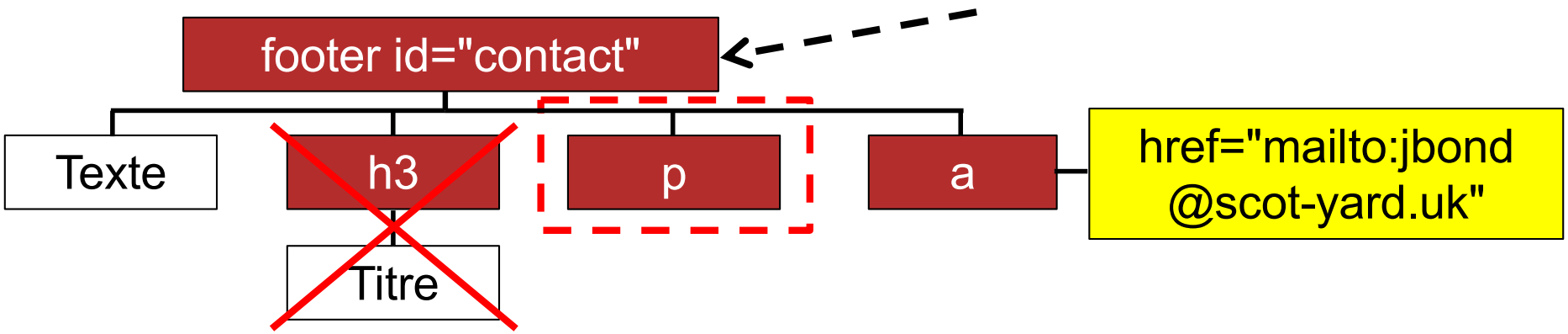
Méthodes d'Element

- Méthodes de modification dynamique du DOM :
  - insertBefore/insertAfter() : insertion d'un nœud avant/après un autre

```
let monTel = document.createElement("p");
monContact.insertBefore(monTel, monAdr);
```

  - removeChild(Elt) : suppression d'un nœud enfant

```
monContact.removeChild(monContact.children[0]);
```



Méthodes d'Element

- Méthodes de gestion des attributs des éléments :
  - hasAttribute("attribut") : teste l'**existence** d'un **attribut**
  - hasAttributes() : teste l'**existence** d'**attributs**
  - setAttribute("attribut","valeur") : **définition** d'un **attribut et affectation** d'une **valeur**

```
monAdr.setAttribute("href",  "mailto:jbond@scot-yard.uk");
monAdr.setAttribute("style", "background-color: red;");
```

ATTENTION les autres styles seront écrasés !!!

  - getAttribute("attribut") : **accès** à la **valeur** d'un attribut

```
alert(monAdr.getAttribute("href"));
// affichage de mailto:jbond@scot-yard.uk
```

Méthodes d'Element

- Méthodes de gestion des attributs des éléments :
  - setAttributeNode(noeudAttribut) **ajoute** un nœud **attribut** (**sans** lui affecter de valeur contrairement à setAttribute() !)

```
let monTitre = document.createElement("h1");
let monAttribut = document.createAttribute("class");
monAttribut.value = "maclasse";
monTitre.setAttributeNode(monAttribut);
document.body.appendChild(monTitre);
```

  - removeAttribute("attribut") **supprime** l'**attribut**

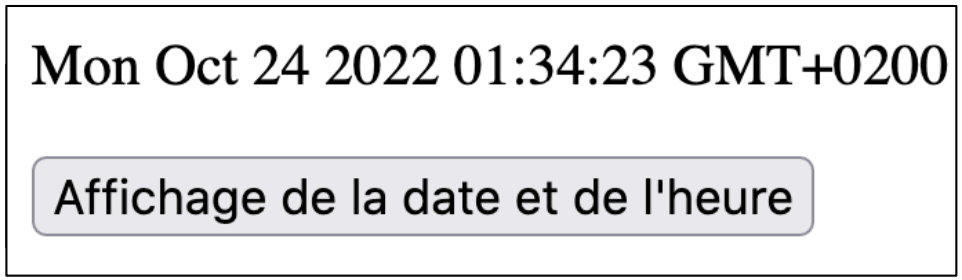
```
monTitre.removeAttribute("class");
```

Propriétés de Node

- Propriétés d'un nœud du DOM :
  - nodeName: retourne le **nom du nœud**
    - 1 - nom de la balise (en majuscule !)
    - 2 - nom de l'attribut
    - 3 - #text
    - 8 - #comment
  - nodeType : retourne le **type du nœud**
    - 1 - nœud élément
    - 2 - nœud attribut
    - 3 - nœud texte
    - 8 - nœud commentaire
  - nodeValue : retourne la **valeur d'un nœud**

Evénements

Evénements



- Un gestionnaire d'événement est un code JavaScript associé à un événement déclenché par l'utilisateur (via la souris ou le clavier) ou par le navigateur :
  - Code associé à l'événement à l'aide d'un attribut HTML

```
<p id="laDate"></p>
<button onclick='getElementById("laDate").innerHTML=Date()'>
Affichage de la date et de l'heure</button>
```

  - Appel de fonction associé à l'événement

```
<p id="laDate"></p>
<button id="bt1" onclick="maFonction()">Test</button>
<script>
function maFonction() {
    document.getElementById("laDate").innerHTML=Date(); }
</script>
```

Evénements du DOM

(on)abort : Interruption du chargement de l'image
(on)blur : Perte du focus sur l'élément
(on)change : Modification du contenu d'un champ de données
(on)click : Clic souris sur l'élément associé à l'événement
(on)dblclick : Double-clic sur l'élément associé à l'événement
(on)dragdrop : G*lisser-déposer* sur la fenêtre du navigateur
(on)error : Erreur lors du chargement de la page
(on)focus : Focus sur l'élément
(on)keydown : Pression exercée sur une touche du clavier
(on)keypress :Maintient d'une touche du clavier enfoncée
(on)keyup : Relâchement d'une touche du clavier
(on)load : Chargement de la page en cours
(on)mouseover : Survol de l'élément avec la souris
(on)mouseout : Fin du survol d'un élément avec la souris
(on)reset : Réinitialisation des données d'un formulaire
(on)resize : Redimensionnement de la fenêtre du navigateur
(on)select : Sélection d'un texte (ou d'une partie) dans un champ text/textarea
(on)submit : Soumission d'un formulaire
(on)unload : Départ de la page en cours

Evénements du DOM

- Méthode de création d'un lien entre une action et un élément du DOM :
  - addEventListener("event", fonction);

```
let elem = document.getElementById("monId1");

elem.addEventListener("click", fonction1);
elem.addEventListener("mousemove", fonction2);
elem.addEventListener("mousemove", fonction3);
```

- Méthode de suppression d'un lien
  - removeEventListener("event", fonction);

```
elem.removeEventListener("mousemove", fonction2);
```

Evénements du DOM

- Méthode addEventListener et fonction anonyme

```
<p id="laDate"></p>
<button id="bt1">Test</button>          // Définition du bouton #bt1

<script>
    let elem = document.getElementById("bt1");
    elem.addEventListener("click", function() {
        document.getElementById("laDate").innerHTML = Date();
    });
</script>
```

Objet Event

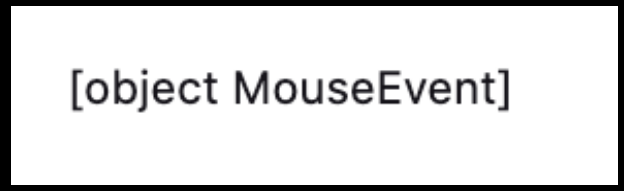
Evènements qui se produisent
sur la page HTML

Objet Event

- Récupération de l'événement déclencheur : event

```
<p id="laDate"></p>
<button id="bt1">Test</button>          // Définition du bouton #bt1

<script>
    let elem = document.getElementById("bt1");
    elem.addEventListener("click", maFonction);
    function maFonction (event) {
        document.getElementById("laDate").innerHTML = Date();
        alert(event);
    }
</script>
```



Objet Event

- Propriétés de l'objet événement :
  - type: chaine contenant le type d'événement ("click", "mouseover", "keydown"…)
  - key : touche pressée (événement clavier)
  - clientX / clientY: coordonnées horizontales / verticales du pointeur de la souris au sein de la fenêtre du navigateur (événement souris)

```
<script>
    document.addEventListener('mousemove', function(event) {
            console.log('Coordonnées X : ' + event.clientX + ‘,
                                Coordonnées Y : ' + event.clientY); });
</script>
```

Objet Event

- currentTarget : propriété contenant l'élément auquel l'événement est rattaché

```
…
<body onclick="maFonction(event)">
    <p>Quel est l'élément cliqué ?</p>
    <script>
        function maFonction(event) {
            alert(event.currentTarget.nodeName);
            alert(event.target.nodeName);
        }
    </script>
</body>
```

- target : propriété contenant l'élément sur lequel l'événement a été déclenché

Objet Event

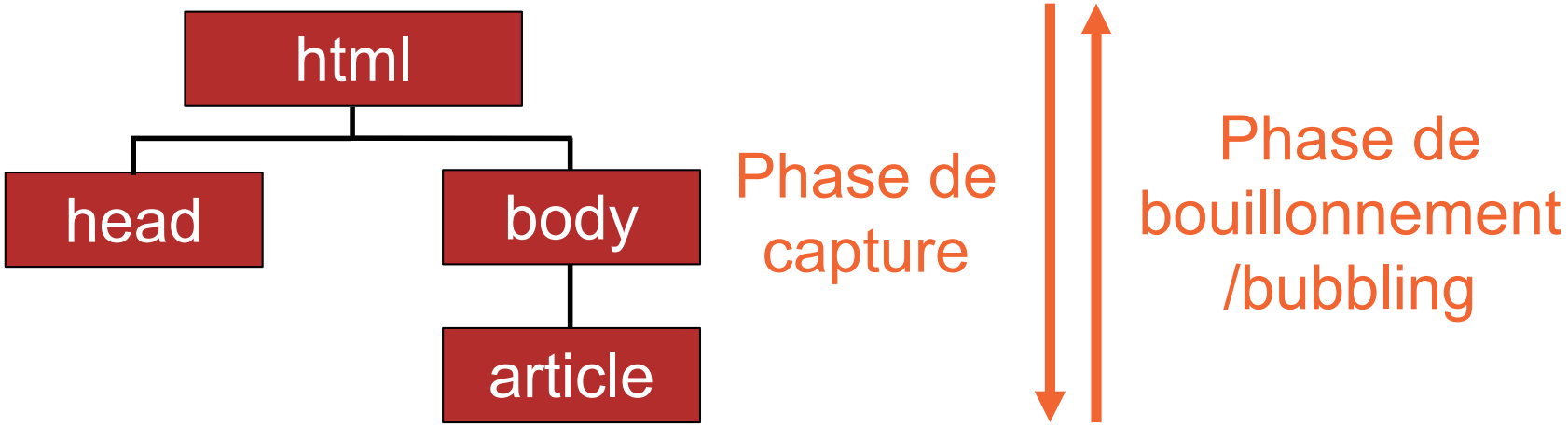
- currentTarget : propriété contenant l'élément auquel l'événement est rattaché

```
…
<body onclick="maFonction(event)">
    <p>Quel est l'élément cliqué ?</p>
    <script>
        function maFonction(event) {
            alert(event.currentTarget.nodeName);
            alert(event.target.nodeName);
        }
    </script>
</body>
```

- target : propriété contenant l'élément sur lequel l'événement a été déclenché

Propagation des événements

- Un événement descend de la racine `html` vers l'élément HTML `article` porteur de l'écouteur



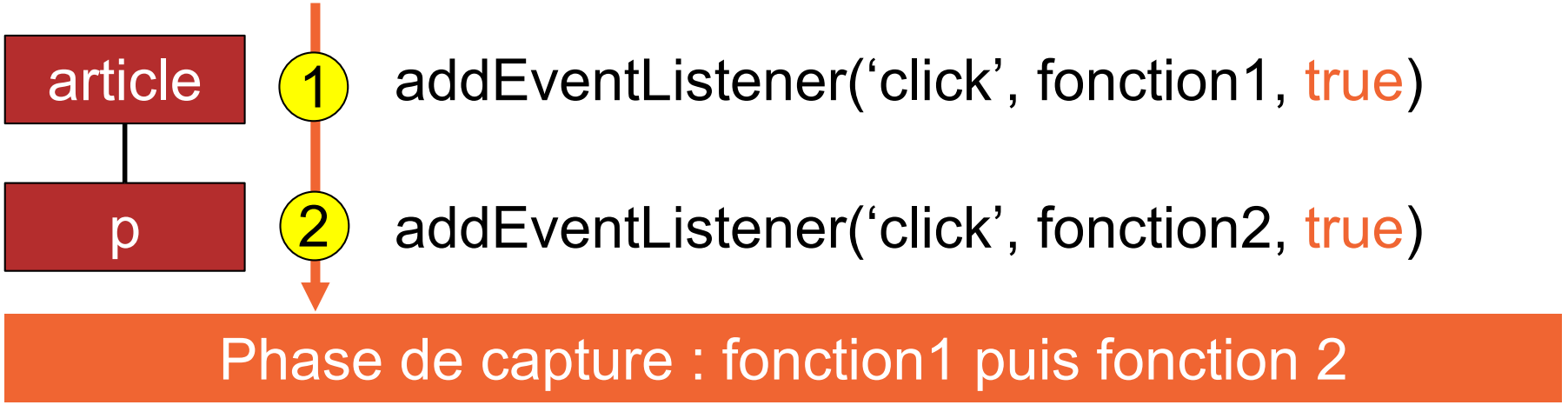
- Puis l'événement poursuit son chemin et repart de l'élément HTML `article` jusqu'à la racine `html`
- Une action n'est exécutée qu'une seule fois en phase de capture ou phase de bouillonnement

Propagation des événements

- L'écouteur addEventListener() comporte 3 paramètres
  - L'événement déclencheur
  - La fonction à exécuter
  - Le booléen de propagation (useCapture) :
    - False (valeur par défaut) : déclenchement lors de la phase de bouillonnement
    - True : déclenchement lors de la phase de capture

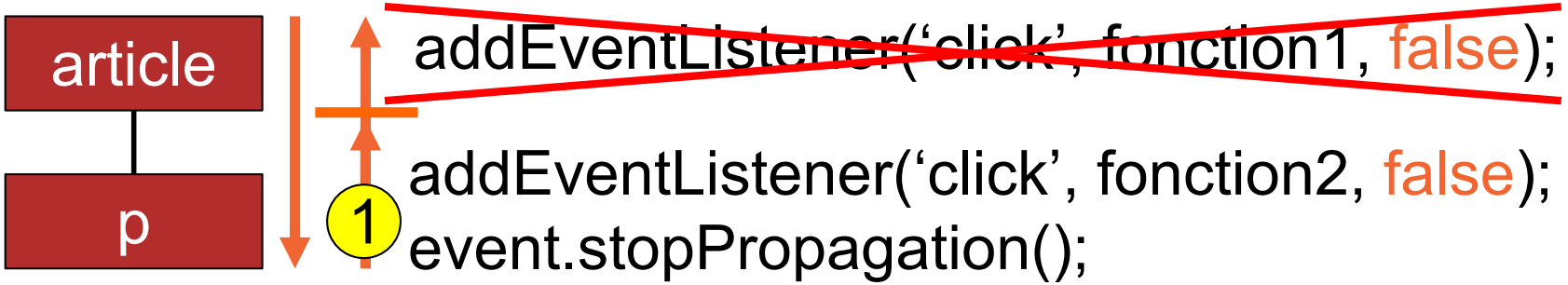
Propagation des événements

- L'écouteur addEventListener() comporte 3 paramètres
  - L'événement déclencheur
  - La fonction à exécuter
  - Le booléen de propagation :
    - False (valeur par défaut) : déclenchement lors de la phase de bouillonnement
    - True : déclenchement lors de la phase de capture

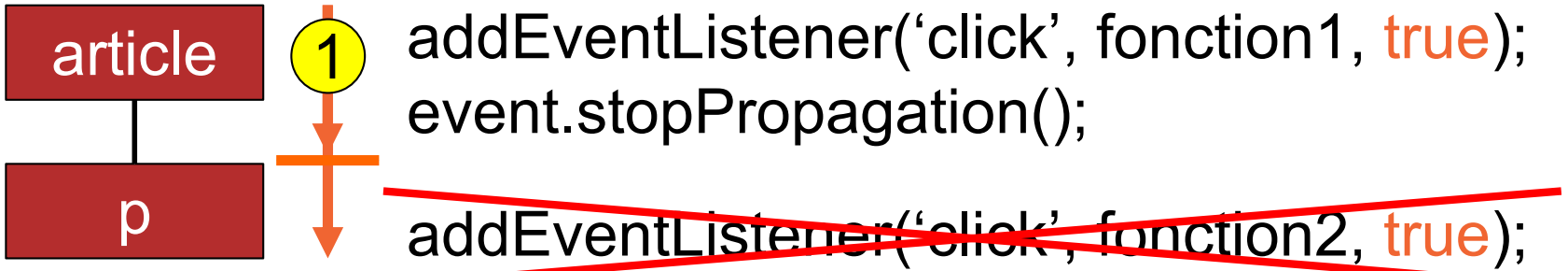


Interruption de la propagation

- Appliquée à l'objet event la méthode stopPropagation() permet d'empêcher l'événement de se propager



Phase de bouillonnement : fonction2 et Arrêt



Phase de capture : fonction1 et Arrêt

Evénements du DOM

- Evènement déclenché à la fin du chargement de la page : DOMContentLoaded

```
<script>
    document.addEventListener("DOMContentLoaded", function() {
            alert (‘Page chargée !');
    });
</script>
```

- Vérification du chargement d’une page et des éléments annexes avec la propriété readyState : loading, interactive ou complete

```
    if (document.readyState == ‘interactive') {
            alert (‘Page chargée !');
    }
```