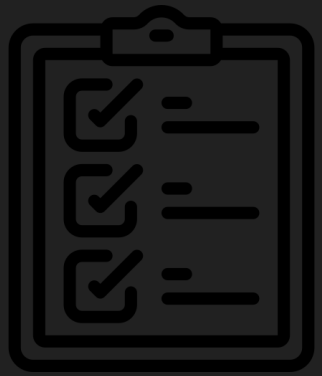
Introduction au langage JavaScript

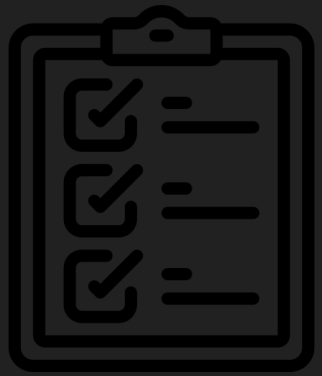
Maîtrisez le langage du web interactif

Plan du cours



- Les objectifs
- Le JavaScript
- Histoire et évolution
- Avantages
- Outils de développement
- La balise script
- Variables
- Commentaires
- Type de données
- Opérateurs arithmétiques
- Opérateurs de comparaison

Plan du cours



- Opérateurs logiques
- Structures conditionnelles
- Structures iteratives
- Les tableaux
- Les fonctions
- Les objets prédéfinis
- La POO
- Fonctions anonymes et fléchées
- JavaScript DOM
- Gestion des événements

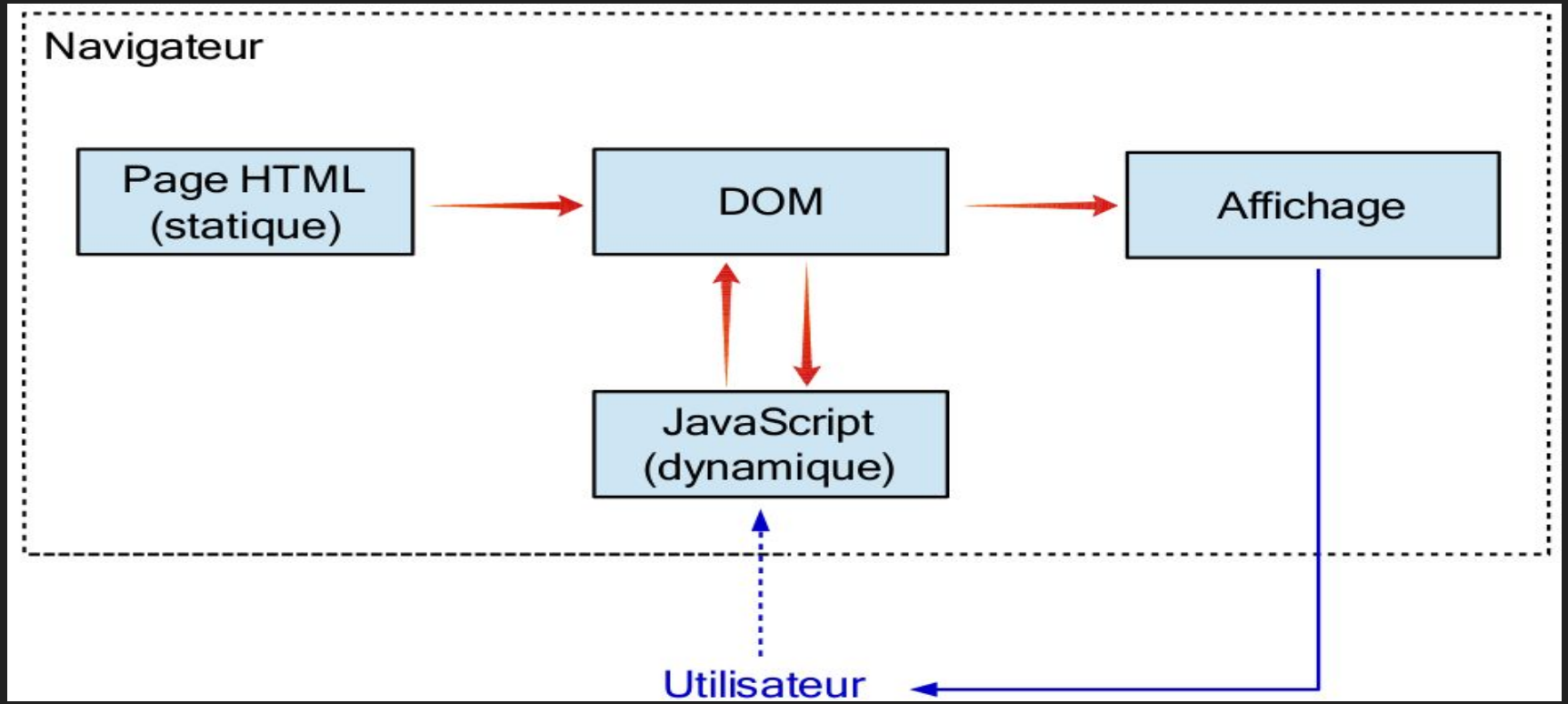
DOM

DOM

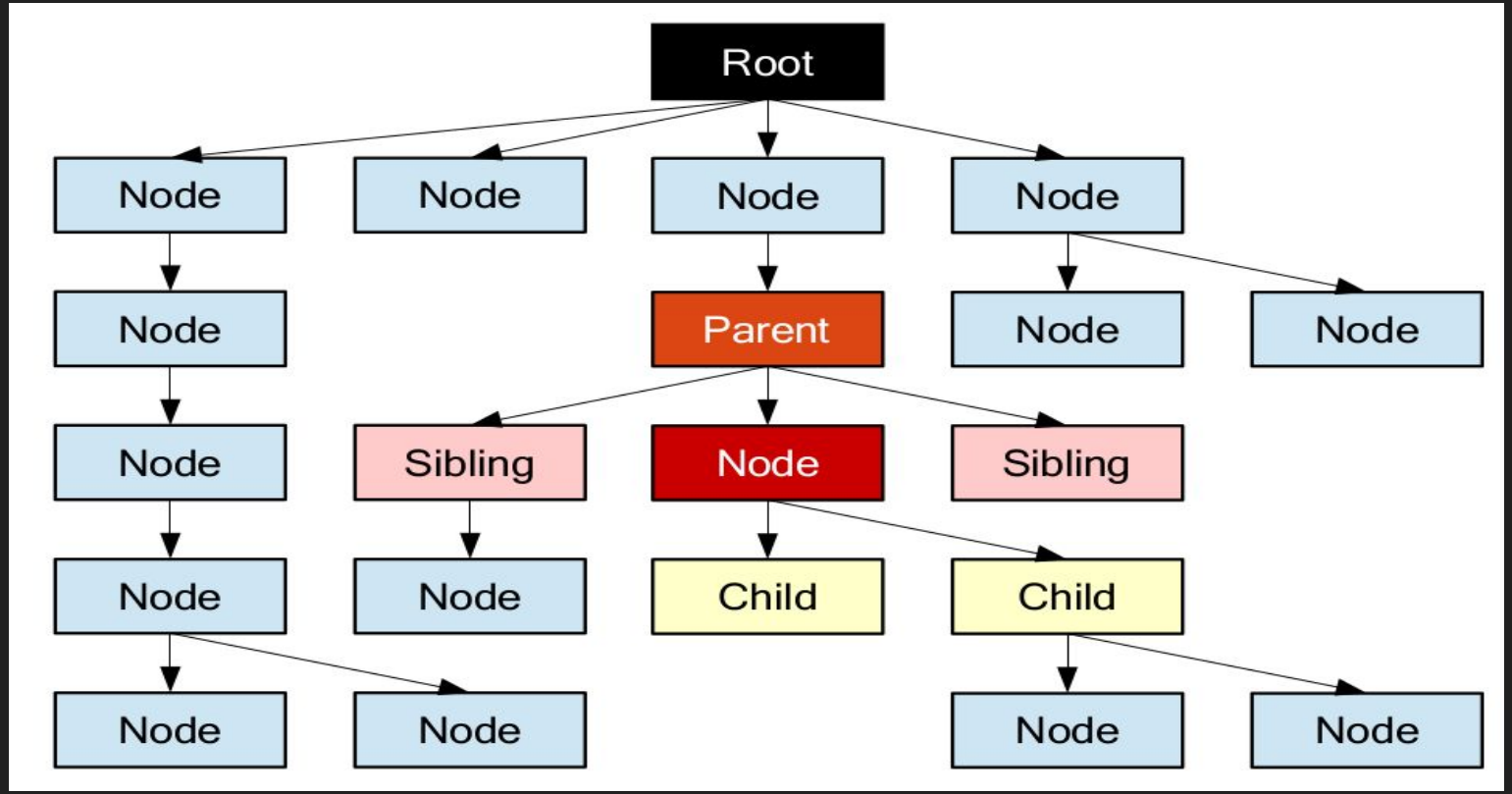
Le DOM, ou Document Object Model, est une interface de programmation pour les documents HTML. Il représente la structure logique d'un document et permet aux programmes informatiques de manipuler le contenu, la structure et le style du document.

Le DOM représente un document comme une hiérarchie d'objets, où chaque élément du HTML (attribut, texte et commentaire) du document est représenté par un nœud. Les nœuds du DOM peuvent être modifiés à l'aide de langages de programmation tels que JavaScript.

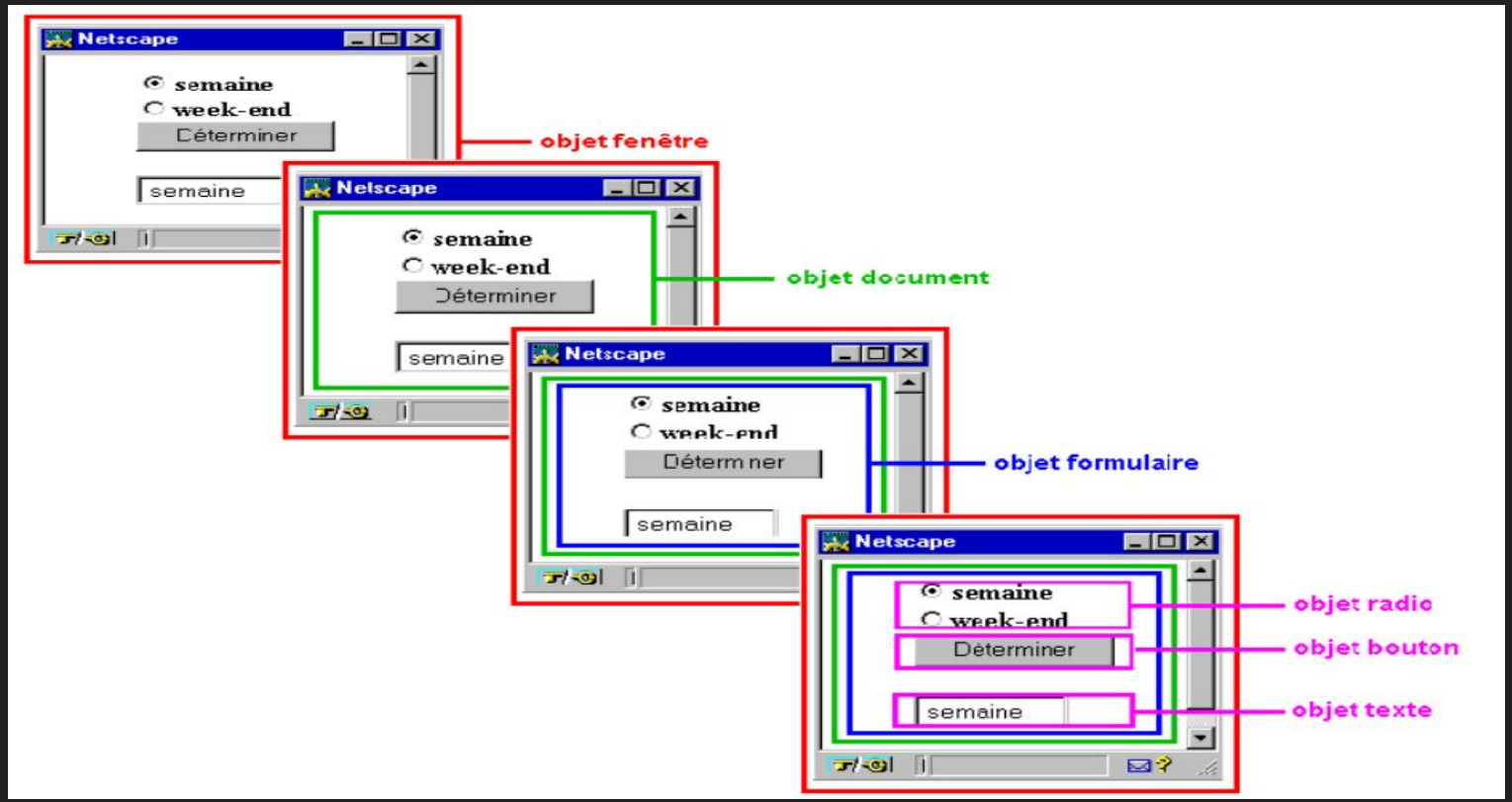
DOM


Navigateur
Page HTML (statique)
DOM
Affichage
JavaScript (dynamique)
Utilisateur

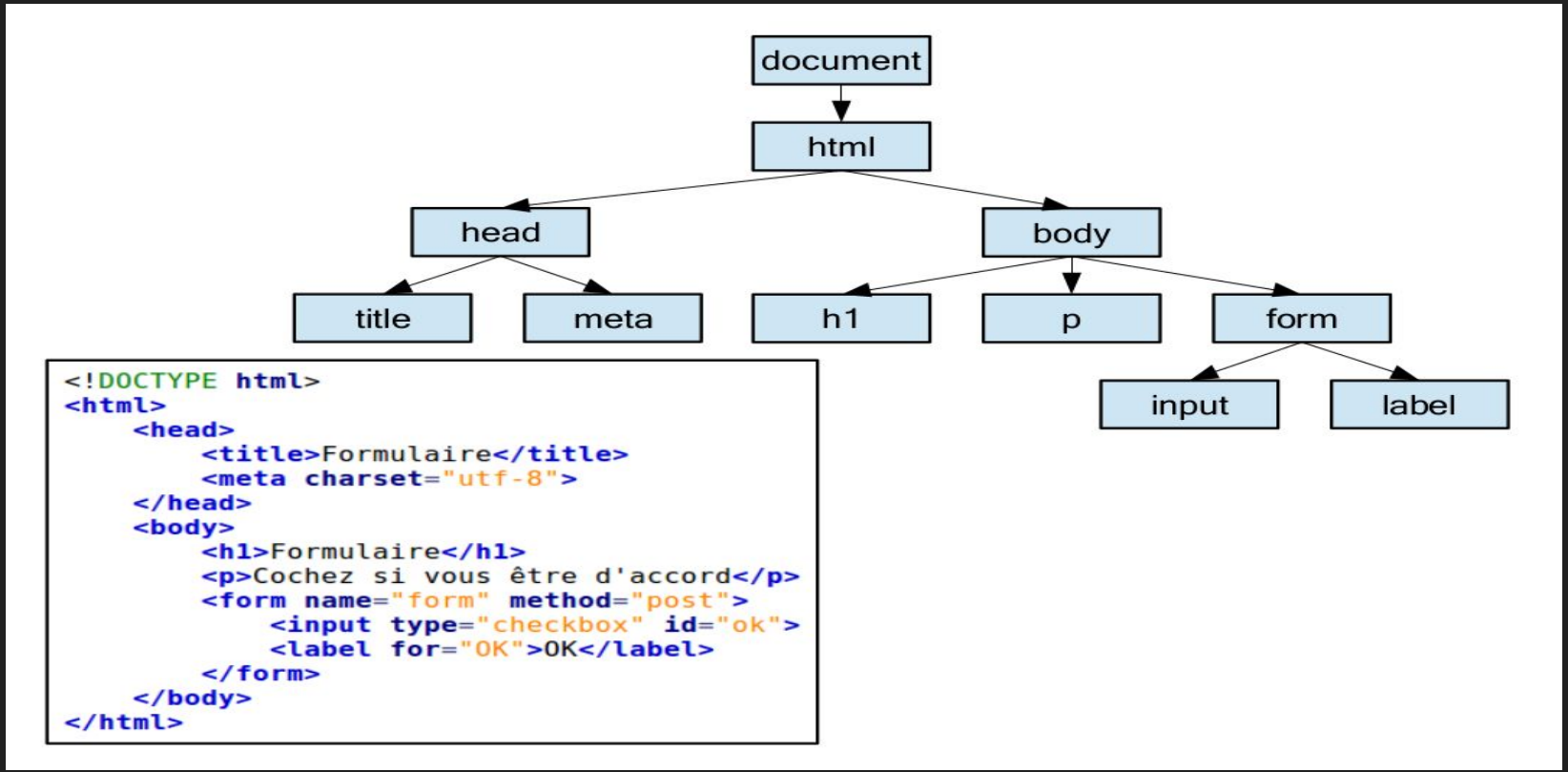
DOM


Root
Node
Node
Node
Node
Node
Parent
Node
Node
Node
Sibling
Node
Sibling
Node
Node
Child
Child
Node
Node
Node
Node

DOM

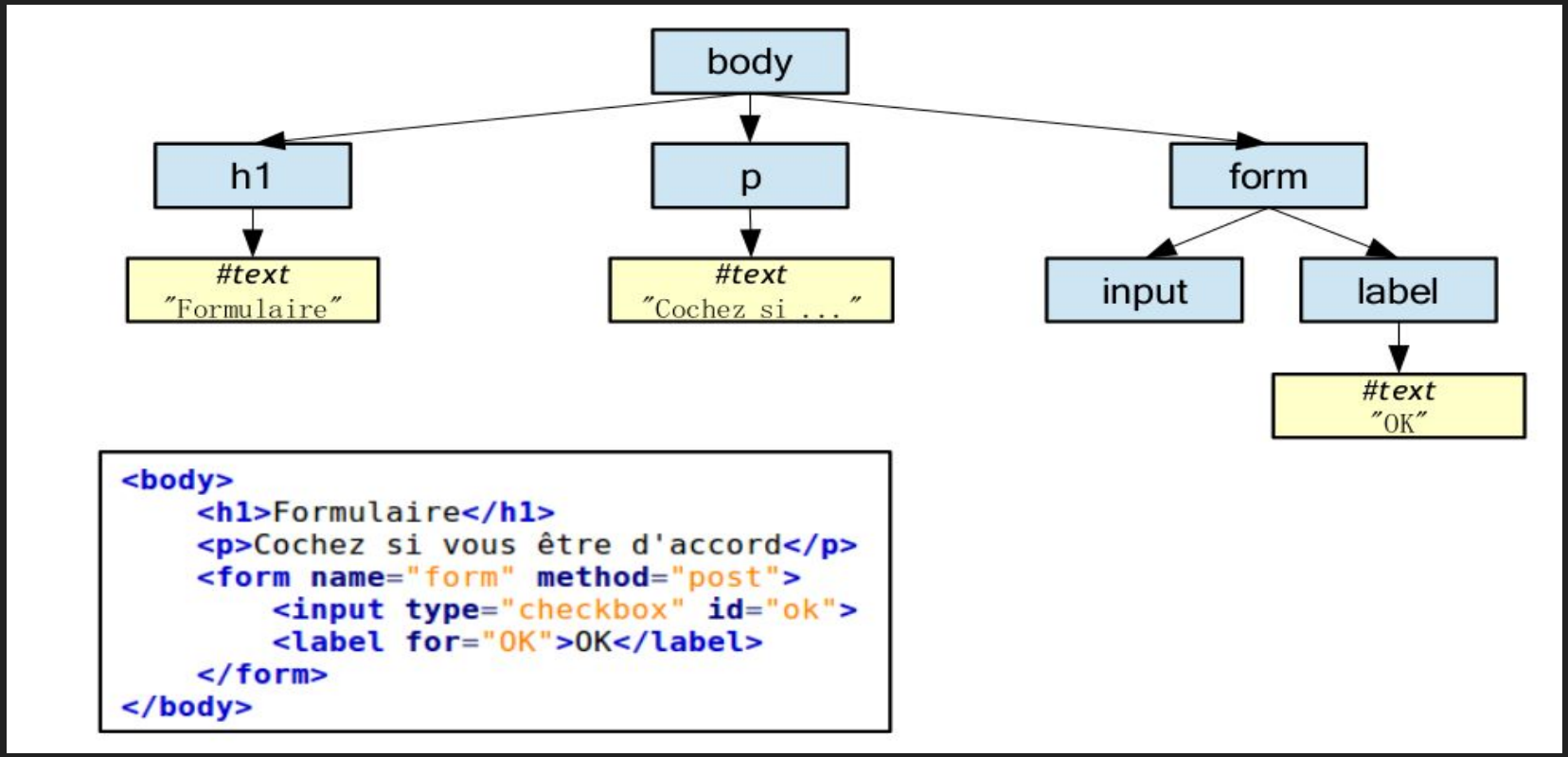

Netscape
semaine
week-end
Déterminer
semaine
objet fenêtre
objet document
objet formulaire
objet radio
objet bouton
objet texte

DOM



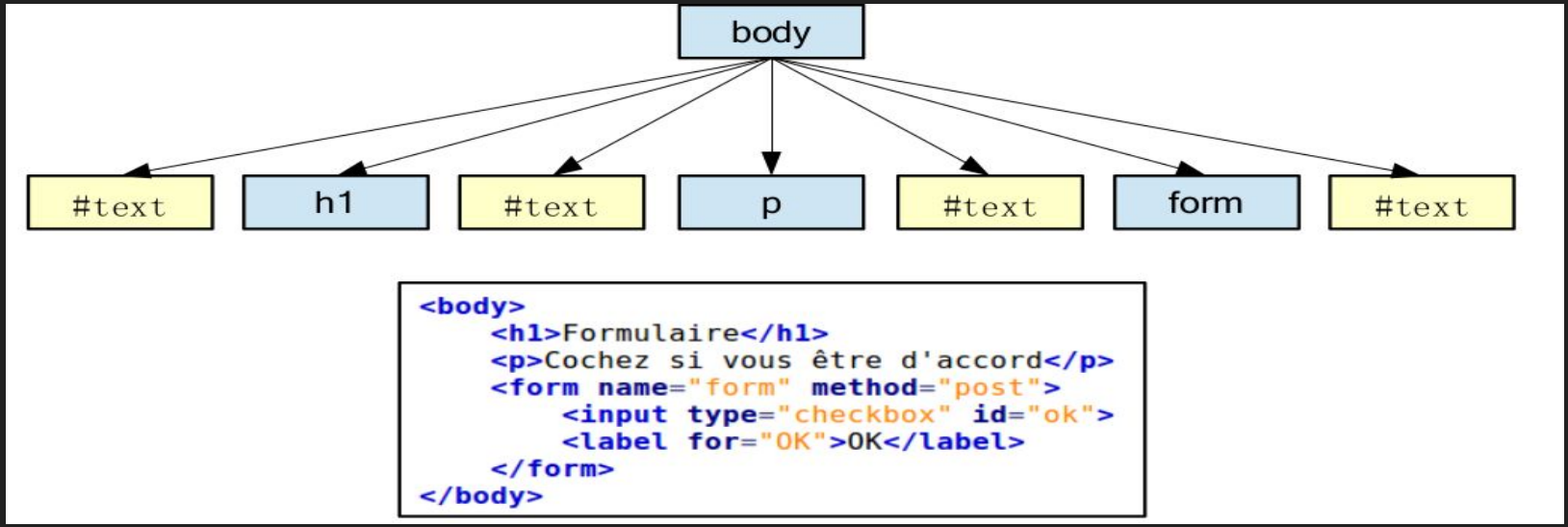

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Formulaire</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1>Formulaire</h1>
        <p>Cochez si vous être d'accord</p>
        <form name="form" method="post">
            <input type="checkbox" id="ok">
            <label for="OK">OK</label>
        </form>
    </body>
</html>
```

DOM




```
<body>
    <h1>Formulaire</h1>
    <p>Cochez si vous être d'accord</p>
    <form name="form" method="post">
        <input type="checkbox" id="ok">
        <label for="OK">OK</label>
    </form>
</body>
```

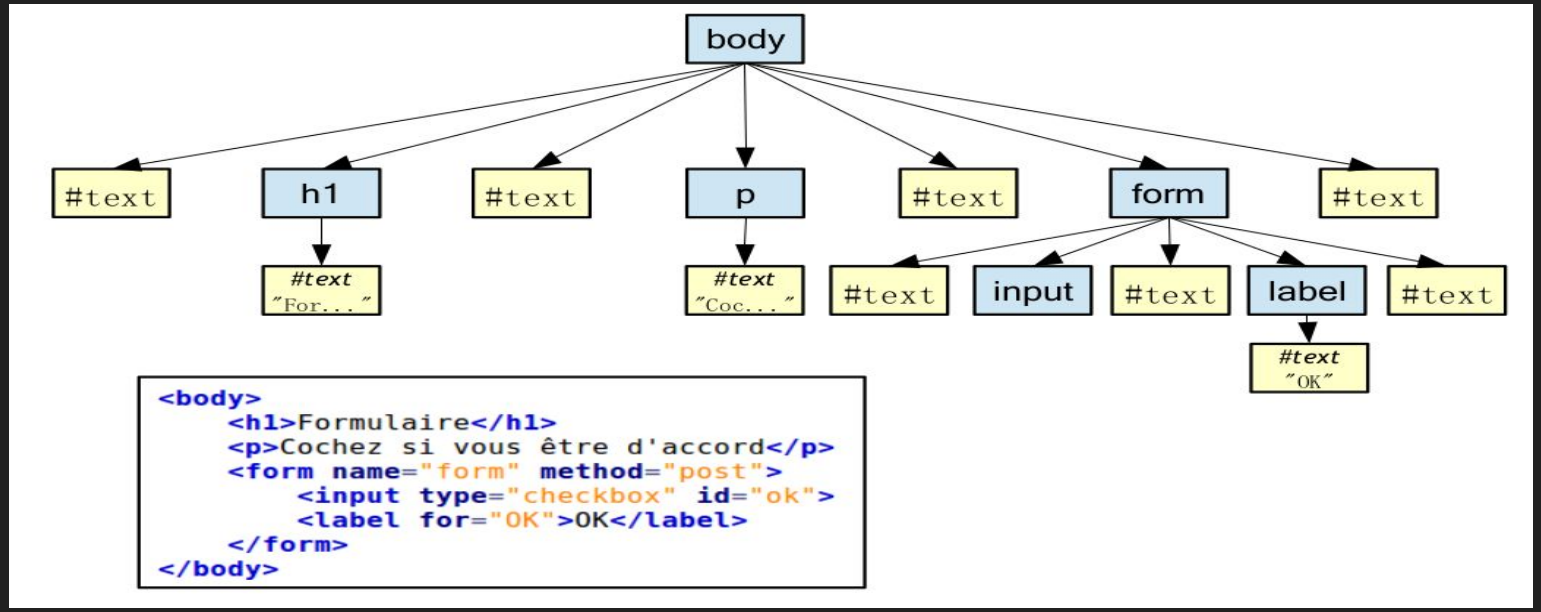
DOM



Le texte séparant deux balises, c’est-à-dire le retour à la ligne et les espaces, sont également pris en compte comme des nœuds textuels.

DOM

Les descendants complets de l'élément body sont donc




```
<body>
    <h1>Formulaire</h1>
    <p>Cochez si vous être d'accord</p>
    <form name="form" method="post">
        <input type="checkbox" id="ok">
        <label for="OK">OK</label>
    </form>
</body>
```

DOM (navigation)

L’objet document est accessible directement dans le code JavaScript à l’aide du mot clé document. L’interface de l’objet document propose des propriétés et des méthodes facilitant la navigation dans le DOM.

DOM (navigation)

- getElementById() -> Renvoie l'élément dont l'ID est celui spécifié.
- getElementsByName() -> Renvoie une liste des éléments ayant le nom donné.
- getElementsByTagName() -> Renvoie une liste des éléments ayant le nom de balise donné.
- getElementsByClassName() -> Renvoie une liste des éléments ayant le nom de classe donné.
- querySelector() -> Renvoie le premier élément qui correspond au groupe de sélecteurs passés en paramètre.
- querySelectorAll() -> Renvoie une liste des éléments correspondent au groupe de sélecteurs passés en paramètre.

DOM (navigation)

```
<body>
    <h1 class="info">Questionnaire</h1>
    <p>Quel(s) langage(s) pratiquez-vous ? </p>
    <form name="form" method="post">
        <input type="checkbox" id="js"> <label for="js">Javascript</label>
        <input type="checkbox" id="php"> <label for="php">PHP</label>
        <input type="checkbox" id="sql"> <label for="sql">SQL</label>
        <input type="button" value="clic">
    </form>
</body>
```

DOM (navigation)

```
// récupère l'élément qui a l'identifiant "js"
var checkBox = document.getElementById("js");
// récupère tous les élément labels
var labels = document.getElementsByTagName('label');
// récupère tous les élément qui un attribut class = "info"
var info = document.getElementsByClassName('info');
// récupère le input de type button
var btn = document.querySelector("input[type='button']");
// récupère tous les éléments input
var inputs = document.querySelectorAll("input");
```

DOM (navigation)

Quelques propriétés de l’objet document permettent d’accéder directement à certains éléments du DOM :

- head -> Renvoie l’élément head du document
- body -> Renvoie l’élément body du document
- links -> Renvoie tous les liens du document
- images -> Renvoie toutes les images du document
- forms -> Renvoie tous les formulaires du document

DOM (navigation)

```
// récupère le body
var body = document.body;
// récupère le head
var head = document.body;
// récupère tous les liens
var liens = document.links;
// récupère toutes les images
var imgs = document.images;
// récupère tous les formulaires
var formulaires = document.forms;
```

DOM (navigation)

```
// récupère le body
var body = document.body;
// récupère le head
var head = document.body;
// récupère tous les liens
var liens = document.links;
// récupère toutes les images
var imgs = document.images;
// récupère tous les formulaires
var formulaires = document.forms;
```

DOM (manipulation)

Une fois qu’un élément a été récupéré, plusieurs propriétés et de méthodes permettent de le manipuler.

- hasAttributes() -> Vérifie si l'élément possède au moins un attribut.
- hasAttribute(name) -> Vérifie si l'élément possède ou non l'attribut spécifié.
- getAttribute(name) -> renvoie la valeur d'un attribut.
- setAttribute(name, value) -> Ajoute un nouvel attribut ou change la valeur d'un attribut existant.
- removeAttribute(name) -> Supprime l’attribut spécifié.

DOM (manipulation)

Une fois qu’un élément a été récupéré, plusieurs propriétés et de méthodes permettent de le manipuler.

- element.className -> retourne une liste de toutes les classes de element mais aussi de modifier ou d’ajouter une classe
- element.classList -> retourne une liste de toutes les classe de element

pour ajouter, supprimer ou remplacer des classes sur l'élément il est recommandé d'utiliser classList qui offre des méthodes plus précises.

DOM (manipulation)

```
<body id="test">
    <div id="myElement" class="btn primary"></div>

    <script>
        var element = document.getElementById("myElement");

        // Accéder à la valeur de la classe
        console.log(element.className); // Affiche "btn primary"

        // Modifier la valeur de la classe
        element.className = "secondary";
        console.log(element.className); // Affiche "secondary"
    </script>
</body>
```

DOM (manipulation)

```
<body id="test">
    <div id="myElement" class="btn primary">Contenu de l'élément</div>

    <script>
        var element = document.getElementById("myElement");

        // Ajouter une classe à l'élément
        element.classList.add("highlight");

        // Vérifier les classes de l'élément après l'ajout
        console.log(element.className); // Affiche "btn primary highlight"
    </script>
</body>
```

DOM (manipulation)

```
<body id="test">
    <div id="myElement"  class="btn primary">Contenu de l'élément </div>

    <script>
    var element = document.getElementById("myElement");

    // Supprimer une classe de l'élément
    element.classList.remove("primary");

    // Vérifier les classes de l'élément après la suppression
    console.log(element.className); // Affiche "btn"
    </script>
</body>
```

DOM (manipulation)

```
<body id="test">
    <div id="elmnt">Contenu de l'élément</div>

    <script>
        var element = document.getElementById("elmnt");

        // Accéder à l'ID de l'élément
        console.log(element.id); // Affiche "elmnt"

        // Modifier l'ID de l'élément
        element.id = "newElmnt";
        console.log(element.id); // Affiche "newElmnt"
    </script>
</body>
```

DOM (manipulation)

Il est possible de lire ou d’écrire le style d’un élément grâce à son attribut style.
Par exemple :

element.style.color => pour modifier la couleur de l'élément.

element.style.backgroundColor => pour modifier le background de l'élément.

element.style.fontSize => pour modifier la taille de police de l'élément.

DOM (manipulation)

```
<body>
 <p id="monElement"  style="background-color: yellow; font-size: 24px;" >Contenu de mon
élément</p>

 <script>
   // Sélection de l'élément par son identifiant
   var element = document.getElementById("monElement");

   // Modification du style de l'élément en utilisant l'attribut style
   element.style.backgroundColor = "blue";
   element.style.fontSize = "18px";
 </script>
</body>
```

DOM (manipulation)

`innerHTML:`

L'attribut innerHTML est utilisé pour obtenir ou définir le contenu HTML d'un élément du DOM

DOM (manipulation)

```
<body>
   <div id="monElement">Contenu initial</div>
    <script>
       // Sélection de l'élément par son identifiant
       var element = document.getElementById("monElement");
       // Obtenir le contenu HTML de l'élément
       var contenu = element.innerHTML;
       console.log(contenu); // Affiche "Contenu initial"
       // Définir un nouveau contenu HTML pour l'élément
       element.innerHTML = "<strong>Nouveau contenu</strong>";
   </script>
</body>
```

DOM (manipulation)

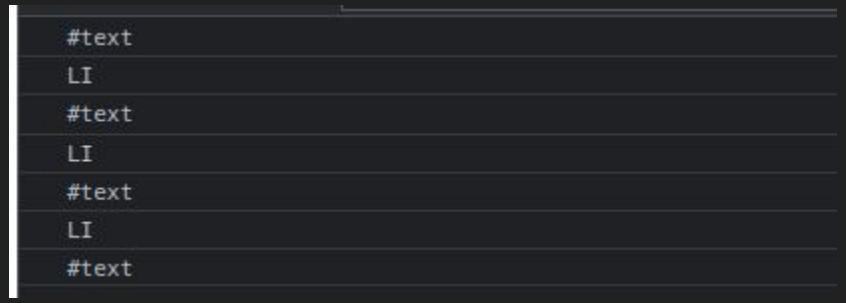
childNodes:

La propriété childNodes est utilisée pour accéder à la liste des nœuds enfants d'un élément dans le DOM. Elle retourne une liste (ou une collection) de tous les nœuds enfants, y compris les nœuds texte, les nœuds éléments et les nœuds de commentaire.

DOM (manipulation)

```
<body>
   <ul id="maListe">
     <li>Item 1</li>
     <li>Item 2</li>
     <li>Item 3</li>
   </ul>
    <script>
     // Sélection de l'élément par son identifiant
     var liste = document.getElementById("maListe");
     // Accéder à la liste des nœuds enfants de l'élément
     var enfants = liste.childNodes;
     // Parcourir la liste des nœuds enfants et afficher leurs noms
     for (var i = 0; i < enfants.length; i++) {
       console.log(enfants[i].nodeName);
     }
   </script>
</body>
```

DOM (manipulation)



DOM (manipulation)

firstChild:

La propriété firstChild est utilisée pour accéder au premier nœud enfant d'un élément dans le DOM. Elle renvoie le premier nœud enfant, qu'il s'agisse d'un nœud texte, d'un nœud élément, d'un nœud de commentaire ou d'un autre type de nœud.

DOM (manipulation)

```
<body>
 <div id="parent">
   <p>Paragraphe 1</p>
   <p>Paragraphe 2</p>
 </div>

 <script>
   // Sélection de l'élément parent par son identifiant
   var parent = document.getElementById("parent");
   // Accéder au premier nœud enfant de l'élément parent
   var premierEnfant = parent.firstChild;
   // Afficher le type et le contenu du premier nœud enfant
   console.log(premierEnfant.nodeType); // Affiche le type de nœud
   console.log(premierEnfant.nodeValue); // Affiche la valeur du nœud
 </script>
</body>
```

DOM (manipulation)

nodeValue:

La propriété nodeValue est utilisée pour accéder ou définir la valeur du nœud dans le DOM. Elle est applicable aux nœuds de type texte et aux nœuds de type commentaires.

DOM (manipulation)

```
<body>
 <div id="monElement">Contenu initial</div>

 <script>
   // Sélection de l'élément par son identifiant
   var element = document.getElementById("monElement");

   // Accéder à la valeur du nœud texte de l'élément
   var valeurInitiale = element.firstChild.nodeValue;
   console.log(valeurInitiale); // Affiche "Contenu initial"

   // Modifier la valeur du nœud texte de l'élément
   element.firstChild.nodeValue = "Nouveau contenu";
 </script>
</body>
```

DOM (manipulation)

CreateElement:

La méthode createElement() est utilisée pour créer un nouvel élément HTML dans le DOM. Elle crée un élément avec le nom de balise spécifié et renvoie une référence à cet élément nouvellement créé.

DOM (manipulation)

```
<body>
 <div id="madiv"></div>

 <script>
   // Sélection de l'élément parent par son identifiant
   var container = document.getElementById("madiv");

   // Création d'un nouvel élément <p>
   var paragraphe = document.createElement("p");

  </script>
</body>
```

DOM (manipulation)

textContent:

La propriété textContent est utilisée pour accéder au contenu textuel d'un élément du DOM, ou pour définir son contenu textuel.

DOM (manipulation)

```
<body>
 <div id="container">
   <p id="paragraphe">Contenu initial</p>
 </div>

 <script>
   // Sélection de l'élément par son identifiant
   var paragraphe = document.getElementById("paragraphe");

   // Accéder au contenu textuel de l'élément
   var contenuInitial = paragraphe.textContent;
   console.log(contenuInitial); // Affiche "Contenu initial"

   // Modifier le contenu textuel de l'élément
   paragraphe.textContent = "Nouveau contenu";
 </script>
</body>
```

DOM (manipulation)

appendChild:

La méthode appendChild() est utilisée pour ajouter un nœud en tant qu'enfant d'un autre nœud dans le DOM. Elle insère le nœud spécifié à la fin de la liste des enfants du nœud parent.

DOM (manipulation)

```
<body>
 <div id="madiv">
   <p>Contenu initial</p>
 </div>

 <script>
   // Sélection de l'élément parent par son identifiant
   var parent = document.getElementById("madiv");
   // Création d'un nouvel élément <span>
   var span = document.createElement("span");
   // Ajout de contenu texte à l'élément
   span.textContent = "Nouveau contenu";
   // Ajout du nouvel élément en tant qu'enfant du parent
   parent.appendChild(span);
 </script>
</body>
```

DOM (manipulation)

removeChild:

La méthode removeChild() est utilisée pour supprimer un nœud enfant spécifié d'un nœud parent dans le DOM. Elle supprime le nœud enfant donné de la liste des enfants du nœud parent.

DOM (manipulation)

```
<body>
 <div id="madiv">
   <p>Contenu initial</p>
   <span>Contenu supplémentaire</span>
 </div>

 <script>
   // Sélection de l'élément parent par son identifiant
   var parent = document.getElementById("madiv");

   // Sélection de l'élément enfant à supprimer
   var span = document.querySelector("span");

   // Suppression de l'élément enfant du parent
   parent.removeChild(span);
 </script>
</body>
```

Liens utils

- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- https://www.w3schools.com/js/default.asp
- https://devdocs.io/javascript/