

Conception d'Interfaces Web

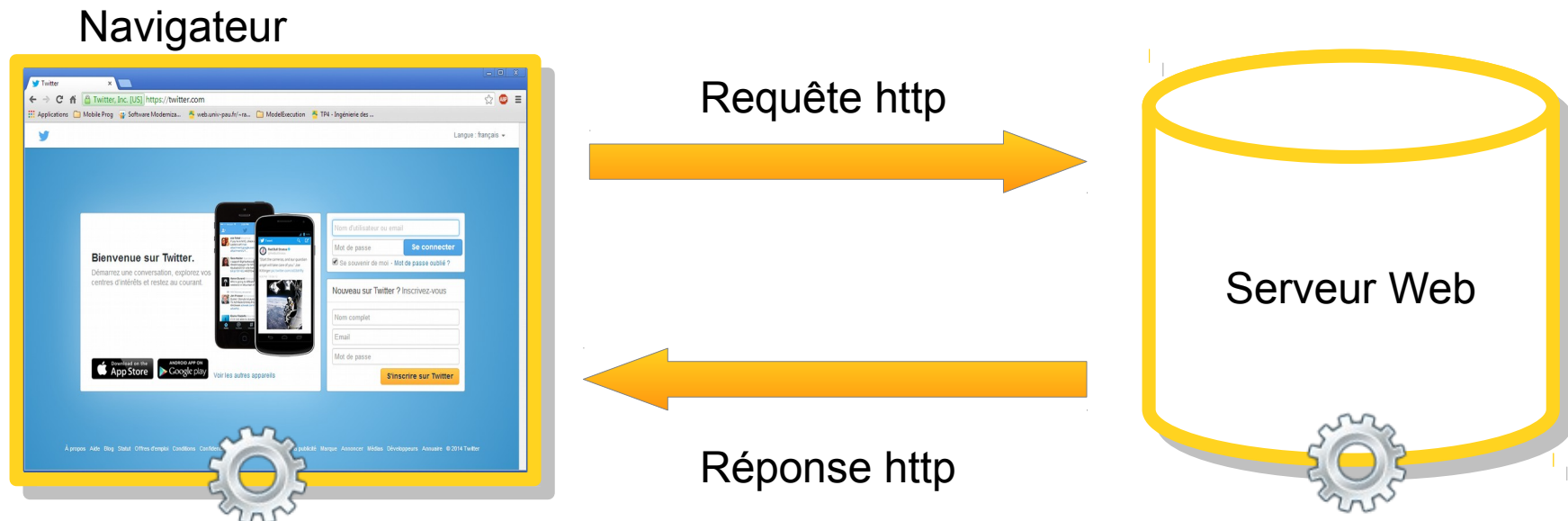
Licence 1

5. Pages Web interactives avec JavaScript

Bruno Jobard

Pages Web statiques

- Une page HTML/CSS est statique : une fois chargée dans le navigateur elle ne change pas.
- Seul un clic sur un lien peut modifier le contenu du navigateur. Cette modification requiert un aller-retour avec un serveur Web.



Pages Web interactives

- On rend une page web interactive (réagissant aux interactions de l'utilisateur, sans rechargement) en la modifiant à l'aide d'un langage de script : le **Javascript**
- Cette interaction permet par exemple de faire des vérifications de formulaires ou faire tourner des applications Web.



Username

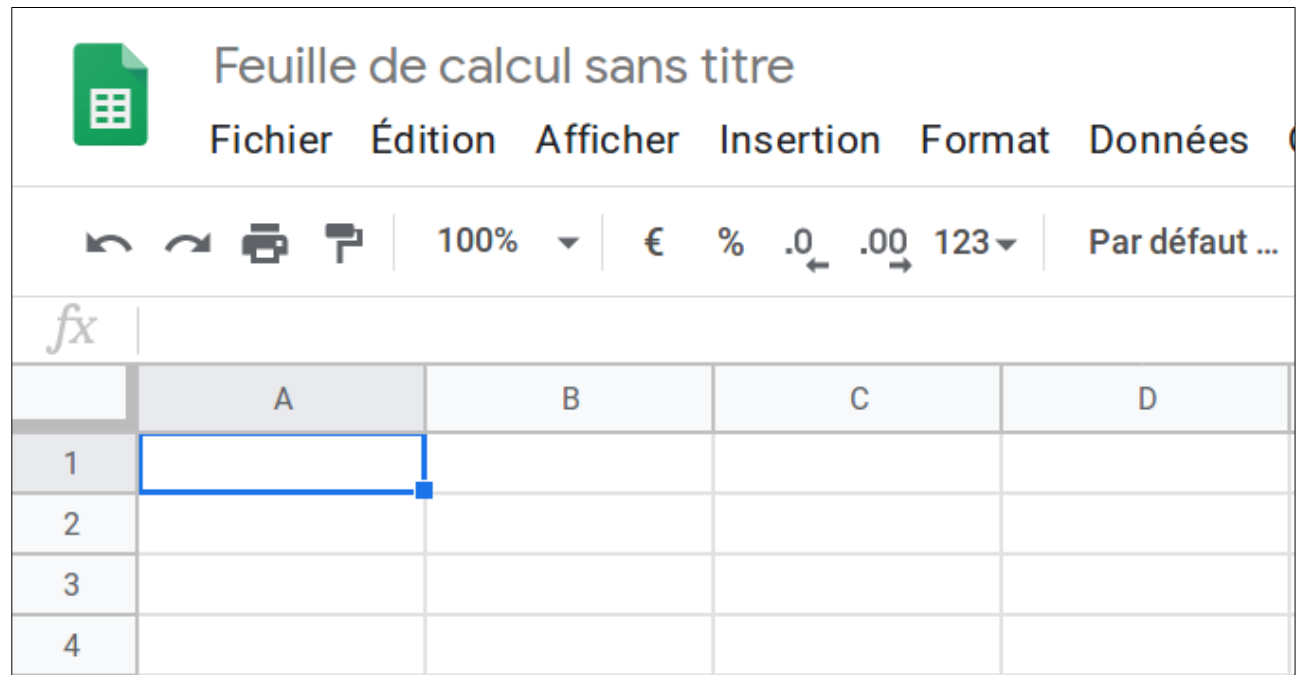
Username is required

newuser@gmail.com

.....

.....

Register



Feuille de calcul sans titre

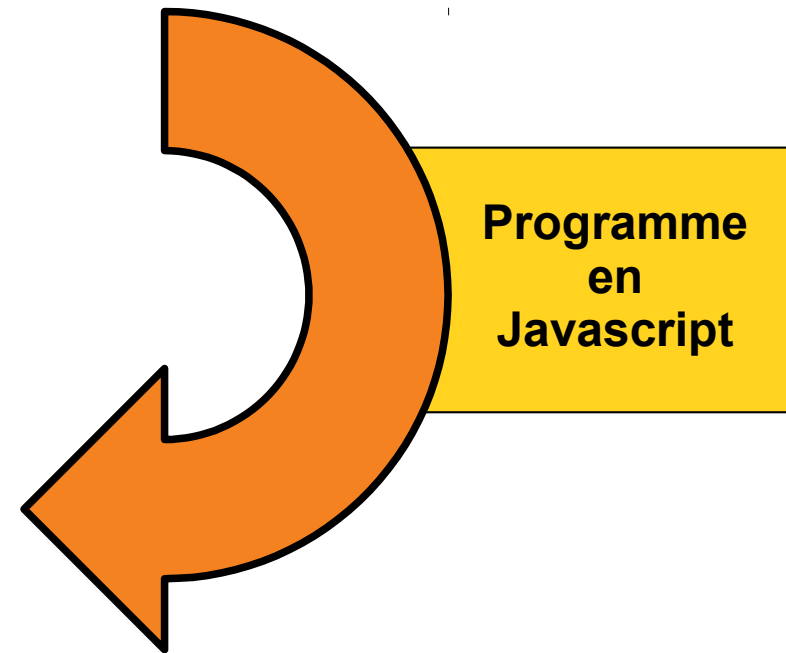
Fichier Édition Afficher Insertion Format Données

100% € % .0 .00 123 Par défaut ...

	A	B	C	D
1				
2				
3				
4				

Pages Web interactives

- Pour qu'une page se modifie en réaction aux interactions de l'utilisateur, il faut que le navigateur puisse :
 - **capturer des « événements »**
 - clic de souris,
 - changement d'un champ texte,
 - déclenchement d'une alarme,
 - ...
 - **modifier la page Web**
 - changement de sa présentation
 - ajout/suppression d'éléments
 - modification de sa structure
 - ...



Réagir à des événements

- On peut spécifier à quels événements un élément HTML peut être sensible.
- En cas de déclenchement d'un tel événement, un script Javascript est appelé.
- On précise cette association événement → action comme attribut d'une balise HTML :

<balise événement="script" > ... **</balise>**

Quelques événements classiques

- Quelques événements classiques applicables à tous les éléments HTML :
 - ***onclick***, ***ondblclick*** : déclenché sur un clic, sur un double clic de la souris sur l'élément;
 - ***onmousedown***, ***onmouseup*** : déclenché sur une pression, sur un relâchement, d'un bouton de la souris sur l'élément;
 - ***onmouseover***, ***onmouseout*** : déclenché sur l'arrivée, sur le départ du pointeur de la zone de l'élément;
 - ***onkeypress***, ***onkeydown*** : déclenché sur une pression, sur un relâchement d'une touche clavier sur l'élément.

Association événement → script

- Forme générale :

`<balise événement="script"> ... </balise>`

- Exemple : changement de la propriété de style « color », en rouge

` ... `

- le mot-clé **this** est la référence à l'élément courant (ici la portion de texte qui dépend de la balise `<a>`)
- l'attribut **style** permet d'accéder à tous les styles de l'élément
- **color** est la propriété de style que l'on souhaite modifier

Association événement → script

- Changement de la couleur de l'élément par une fonction

```
<a href=" " onmouseover="changeColor(this, 'red' "> ... </a>
```

Ici la fonction `changeColor()` est définie dans une section **script** :

```
<script type="text/javascript">
```

```
function changeColor(element, couleur) {  
    element.style.color = couleur;  
}
```

```
</script>
```

Passage des arguments



Accès aux éléments de la page Web via le DOM

- Le DOM (Document Object Model) est une représentation informatique d'une page Web
- Tous les éléments de la page sont accessibles via des instructions en Javascript
- Le DOM est une structure arborescente fidèle à l'imbrication des éléments HTML

Accès aux éléments de la page Web via le DOM

- Pour modifier un élément du document il faut obtenir une référence sur celui-ci.

Voici quelques moyens :

- mot-clé **this** : référence de l'élément courant.
Peut être passé en paramètre d'une fonction
- fonction **document.getElementById('nom')** :
fourni une référence sur l'élément dont
l'attribut id est « nom »
- fonction **document.querySelectorAll('li a')**
retourne un tableau contenant les références
de tous les éléments **a** imbriqués dans des **li**

Accès aux attributs et aux styles via le DOM

- Tous les attributs des balises HTML et les propriétés des styles qui régissent leur présentation sont accessibles par l'intermédiaire d'un script
- Si **elt** est une référence sur un élément alors
 - **elt.att** permet d'accéder à son attribut att
 - ex : **this.height=180;**
(si **this** est la référence d'une image)
 - **elt.style.prop** permet d'accéder à sa propriété prop
 - ex : **this.style.textAlign='center';**
(si **this** est la référence à un paragraphe, par ex.)
- Notez que les noms de propriétés composés de la forme mot1-mot2 s'écrivent mot1Mot2 dans les scripts

Modification des valeurs dans le DOM

- Après avoir navigué dans le DOM et atteint le nœud de son choix, il faut agir dessus en modifiant ses propriétés

```

```

```
<p>Ceci est <b>une poire</b> !</p>
```

```
<script>
```

```
let imgTag = document.getElementById('fruit'); // navigation
```

```
imgTag.src = 'ananas.jpg'; // modification !
```

```
let bTag = document.querySelectorAll('p b'); // navigation
```

```
bTag[0].textContent = 'un ananas'; // modification !
```

```
</script>
```

Exemple – côté HTML

```
<ul>
<li>
  <a id="lien1" href=""
    onmouseover="this.style.color='red'"
    onmouseout ="this.style.color='black'"> lien 1 </a>
</li><li>
  <a id="lien2" href=""
    onmouseover="changeColor1()"
    onmouseout ="changeColor2()"> lien 2 </a>
</li><li>
  <a id="lien3" href=""
    onmouseover="changeColor3('cyan')"
    onmouseout ="changeColor3('black')"> lien 3 </a>
</li><li>
  <a href=""
    onmouseover="changeColor4(this,'brown')"
    onmouseout ="changeColor4(this,'black')"> lien 4 </a>
</li><li>
  <a href=""
    onmouseover="changeWeight(this,'bold')"
    onmouseout ="changeWeight(this,'normal')"> lien 5 </
</li>
</ul>
```

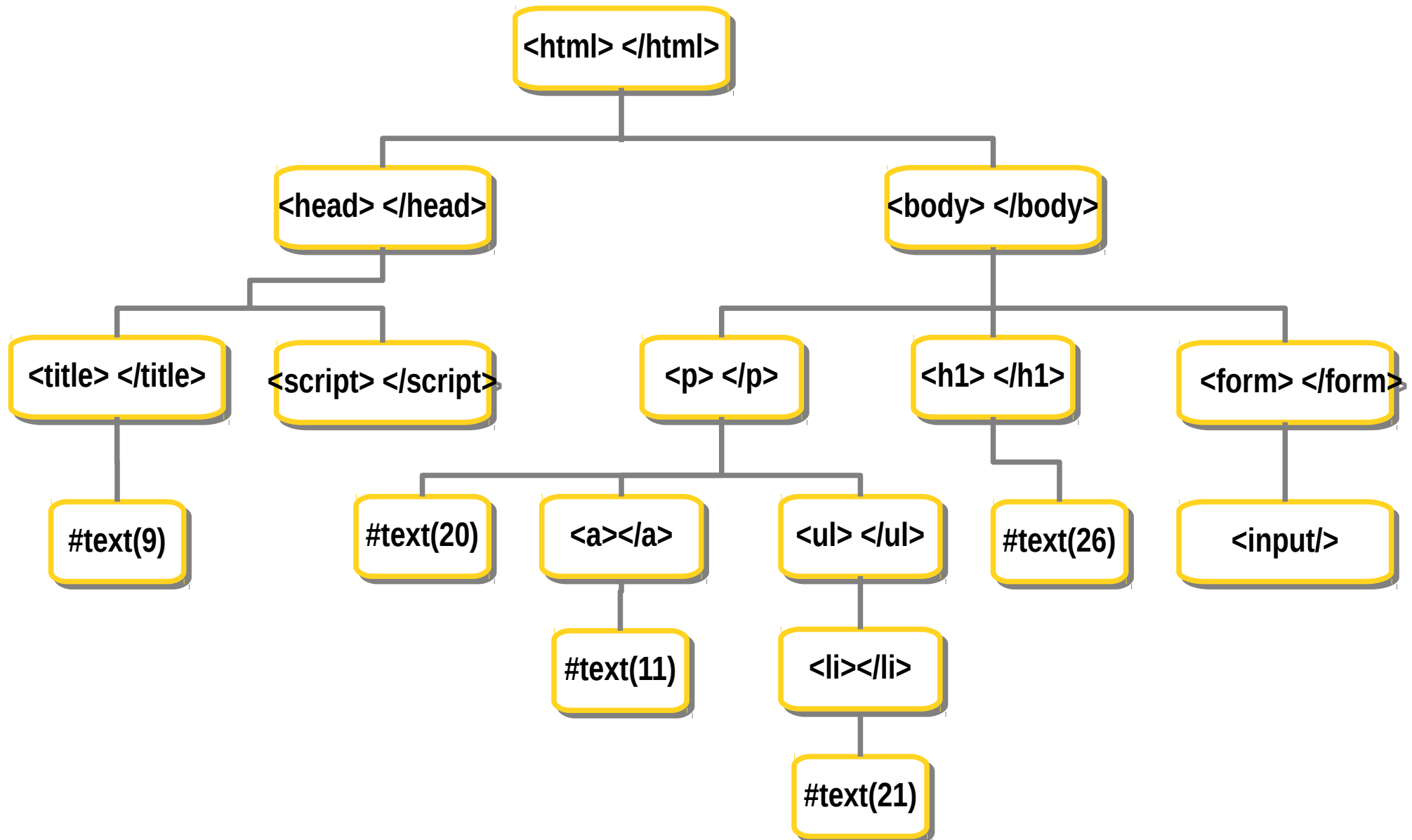
- lien 1
- lien 2
- lien 3
- lien 4
- lien 5

Exemple – côté Javascript

```
<script type="text/javascript">
function changeColor1() {
    document.getElementById('lien2').style.color='red';
}
function changeColor2() {
    document.getElementById('lien2').style.color='black';
}
function changeColor3(couleur) {
    let elements = document.getElementsByTagName('a');
    // on change le troisième élément du tableau
    elements[2].style.color=couleur;
}
function changeColor4(element, couleur) {
    element.style.color=couleur;
}
function changeWeight(element, poids) {
    element.parentNode.parentNode.style.fontWeight=poids;
}
</script>
```

- lien 1
- lien 2
- lien 3
- lien 4
- lien 5

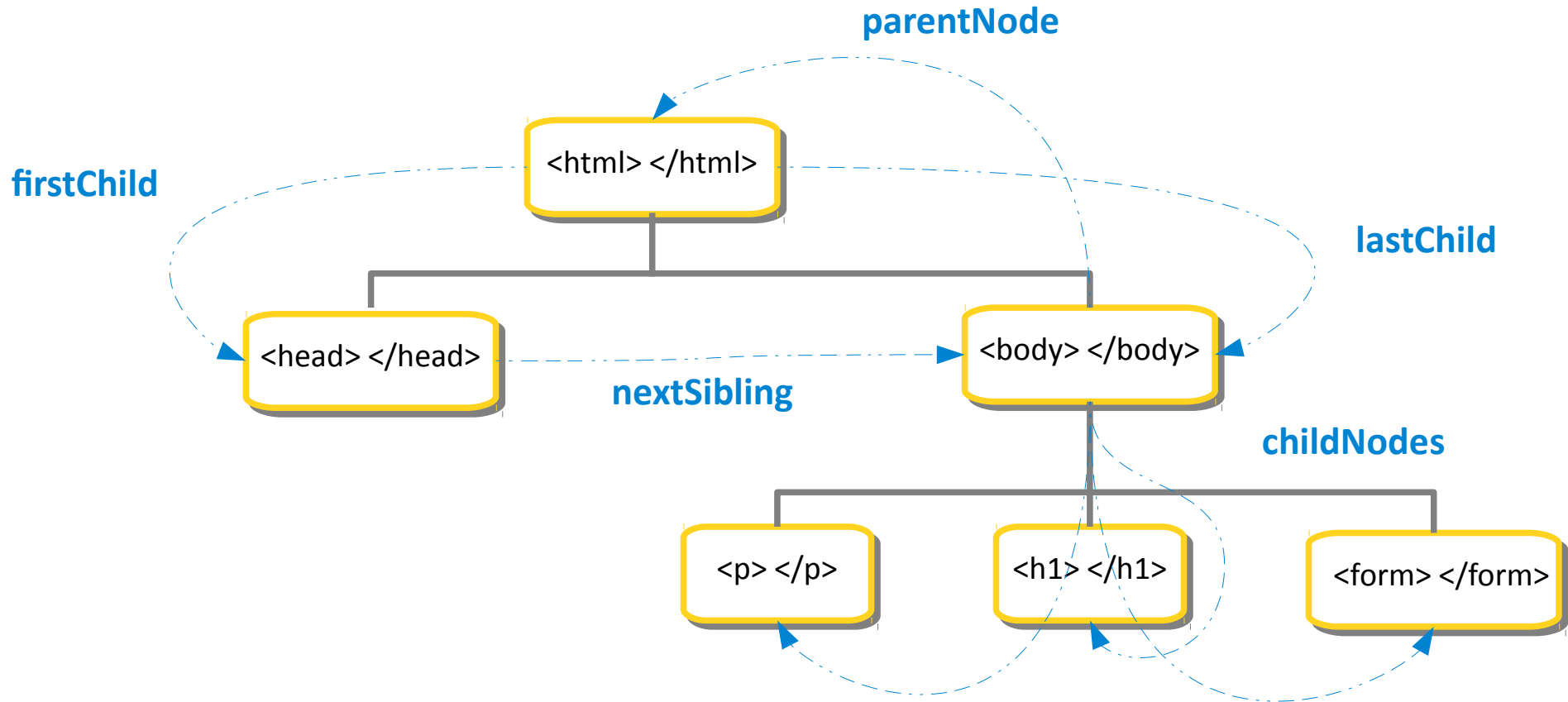
DOM : Arborescence de noeuds d'un document HTML



Propriétés d'un nœud du DOM

Propriétés	Commentaires
childNodes	nœuds enfants (Array)
firstChild	premier nœud enfant
lastChild	dernier nœud enfant
nextSibling	prochain nœud d'un type (nœud de même niveau)
parentNode	nœud parent
previousSibling	nœud précédent d'un type (nœud de même niveau)
nodeName	nom du nœud
nodeValue	valeur / contenu du nœud
nodeType	type du nœud
innerHTML	contenu littéral html du nœud

Navigation dans l'arbre DOM



Ajout/suppression de nœuds dans le DOM

Méthodes	Commentaires
<code>createElement()</code>	Méthode pour créer un nouvel élément HTML dans le document (div, p, span, a, form, input, etc...).
<code>createTextNode()</code>	Méthode pour créer un nœud texte.
<code>appendChild()</code>	Pour ajouter l'élément créé dans le document. L'élément sera ajouté comme étant le dernier nœud enfant d'un élément parent.
<code>insertBefore()</code>	Pour ajouter l'élément créé avant un autre nœud.
<code>removeChild()</code>	Pour supprimer un nœud.

Exemple d'ajout de noeuds dans le DOM

- Ajouter un item en cliquant sur une liste :

<p> Cliquez sur la liste ci-dessous pour ajouter un item :

```
<ul onclick="addItem(this)">
```

```
  <li>Un premier item</li>
```

```
</ul>
```

```
</p>
```

```
<script>
```

```
function addItem(elt) {
```

```
  //navigation vers le parent
```

```
  let ulTag = elt;
```

```
  //nouveau noeud de type <li>
```

```
  let item = document.createElement('li');
```

```
  item.innerHTML="Lorem Ipsum";
```

```
  //greffe du nouveau noeud à son parent
```

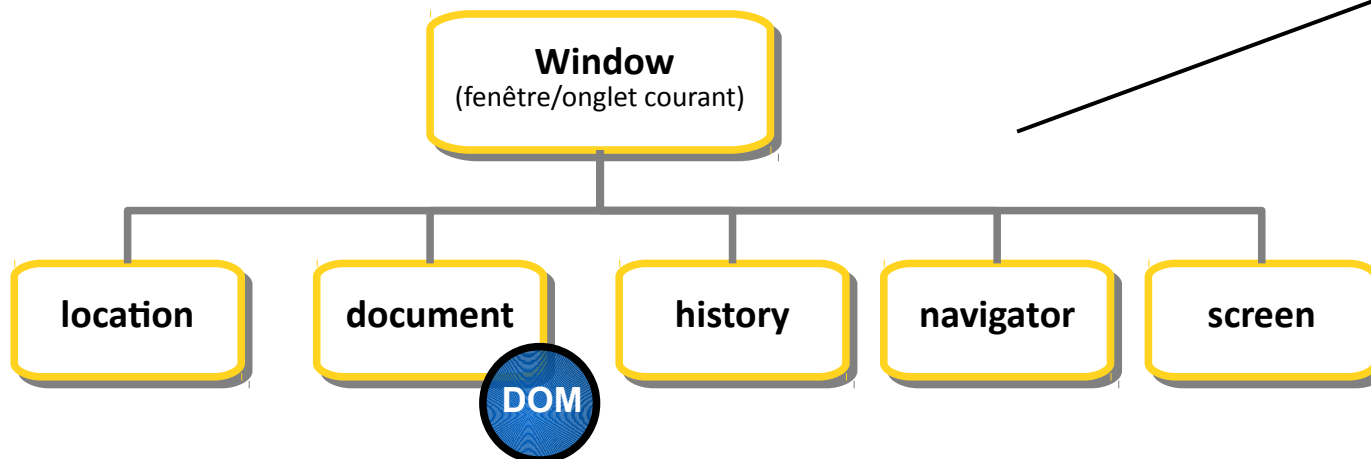
```
  ulTag.appendChild(item);
```

```
}
```

```
</script>
```

BOM: *Browser Object Model*

- Permet de manipuler le navigateur
 - Tous les navigateurs (IE, Firefox, Chrome, ...) sont des logiciels qui offrent les mêmes fonctionnalités de base
 - Ouvrir/fermer des onglets, aller à une URL, mémoriser la liste des URL précédemment consultées, etc.
- Arborescence d'objets



Chaque objet possède ses propriétés et méthodes

BOM : Entrées / sorties

- Méthodes d'interaction avec l'utilisateur par le biais de la fenêtre du navigateur
 - Utilisation de l'objet window
- 2 méthodes d'entrée

```
let age = window.prompt("Quel est votre age ?");
```

```
let doQuit = window.confirm("Voulez vous quitter cette page ?");
```

- 1 méthode de sortie

```
window.alert("Bienvenue sur ce site !");
```

BOM : exemples

//affiche dans la console le nom de code du navigateur utilisé

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

//redirige le navigateur vers une adresse quelconque

```
window.location = "http://www.univ-pau.fr";
```

//ouvre un nouvel onglet dans le navigateur

```
let onglet = window.open('http://www.youtube.com');
```

//Fais revenir une page en arrière (similaire au bouton 'Back')

```
window.history.back();
```

//Affiche dans une boîte de dialogue la résolution de l'écran utilisé

```
window.alert(window.screen.availWidth + "x" + window.screen.availHeight);
```

//Ecrit de l'html directement dans le document (et supprime l'existant)

```
window.document.write("<b>Bienvenue à l'université de Pau</b>");
```

Exercice :
Quels sont
les objets ?
Quelles sont
les propriétés ?
Quelles sont
les méthodes ?

Conclusion

- Javascript permet de réagir à des événements en modifiant instantanément la page Web
- Mais Javascript est bien plus que ce que nous venons de voir :
 - Langage de programmation à part entière (types de données, structures de contrôle, fonctions...)
 - Nombreuses bibliothèques de fonctions disponibles
 - Son étude continuera en L2 et L3...