Applications interactives JavaScript

Anne-Laure Ligozat

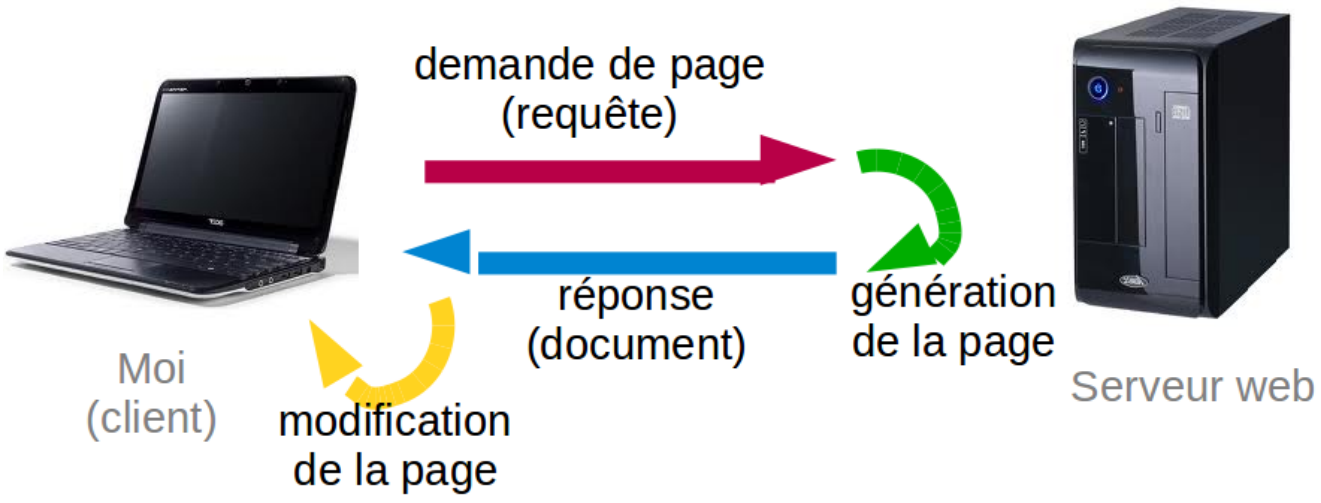
ENSIIE, 1re année

màj: février 2020

Intérêt de JavaScript

- augmenter l'interactivité de l'IHM en réduisant les échanges client-serveur
- objectifs:
  - petites applications simples (calculette, calcul de devis..)
  - amélioration de l'aspect graphique de l'interface (gestion fenêtres, modification d'image au passage de la souris...)
  - contrôle de la validité des saisies (champs obligatoires remplis, valeur saisie du bon type...)
  - aide contextuelle, menus contextuels...

Accès à une page web avec JavaScript



JavaScript

Présentation

- créé en 1995, standardisé sous le nom d'ECMAScript depuis 1997
- rien à voir avec Java
- langage de programmation objet simple, programmation événementielle
- langage interprété
- côté client : code intégré à la page HTML et exécuté par le navigateur
  - code visible, aucune sécurité
  - JavaScript désactivable par l'utilisateur
  - code contenu dans une balise <script>
- côté serveur
  - avec interpréteur (Node.js)

Généralités

Commentaires

/* plusieurs lignes */ ou // 1 ligne

Variables

nom sensible à la casse, déclaration avec `var`, typage dynamique
`boolean, number, string, object, function ou undefined`

```
var x=10; typeof x; // number
x='Test'; typeof x; // string
```

Tableaux

dynamiques, éléments pas nécessairement du même type

```
var notes = [13,4,20,15];
notes[2] // → 20
notes.length; // → 4
```

Généralités

Chaînes de caractères

délimitées par simples ou doubles guillemets
opérateur de concaténation: +

Objets (1)

objet = ensemble de propriétés ie couples (nom,valeur)

```
var book = {
  title={"Harry Potter and the Prisoner of Azkaban"},
  vol=3,
};
book.title; // -> "Harry Potter and the Prisoner of Azkaban"
book.author="J. K. Rowling";
book["lang"]="english";
```

ici, objet littéral ∼ tableau associatif de PHP
en fait en JS, tout est objet...

Généralités

Fonctions

fonctions globales ou inline

Portée des variables

⚠ en JS, portée = fonction ≠ Java, C, PHP = portée block

Variable dans block

```
for (var i = 0; i < 2; i++) {
  document.write(i+", ");
}
document.write("final: "+i);
```

affiche: 0, 1, final: 2

Variable dans fonction

```
maFonction();
document.write("valeur de i: "+i);

function maFonction(){
  var i=2;
}
```

n'affiche rien (i pas défini)

Revenons sur les objets...

Créer des objets

JS possède la notion d'objet, mais pas de classe (jusqu'à ES6)
définition d'un constructeur via une fonction

```
function Book(titre, auteur){
  this.titre=titre;
  this.auteur=auteur;
}
var hp = new book('HP and the Prisoner of Azkaban', "J. K. Rowling");
alert(hp.titre); // nouvelle instance de l'objet
```

mais aussi: méthodes, prototype, héritage...

Insertion du code JavaScript

Insertion du code

- dans l'en-tête de préférence (notamment fonctions)
- dans le corps pour générer du HTML dynamique
- dans un événement d'objet de la page

Exemple de code dans le corps du document (w3schools)

```
<!DOCTYPE html>
<html>
<body>
...
<script>
document.write("<h1>Ceci est un titre</h1>");
document.write("<p>Ceci est un paragraphe </p>");
</script>
...
</body>
</html>
```

Code JavaScript séparé

- code JavaScript stocké dans un fichier séparé
- nom de ce fichier dans l'attribut src de la balise <script>

```
<script src="scriptControles.js">
```

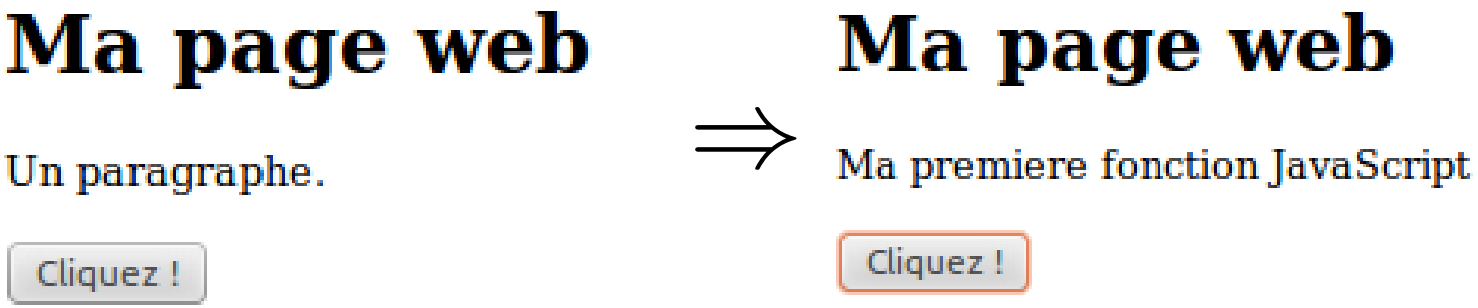
⇒ code partagé, bibliothèque de fonctions, lisibilité

Insertion du code dans l'en-tête

Exemple de code dans l'en-tête du document (w3schools)

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
  document.getElementById("demo").innerHTML="Ma premiere fonction
JavaScript";
}
</script>
</head>
<body>
  <h1>Ma page web</h1>
  <p id="demo">Un paragraphe</p>
  <button type="button" onclick="myFunction()">Cliquez !</button>
</body>
</html>
```

Exemple JavaScript



Possibilités

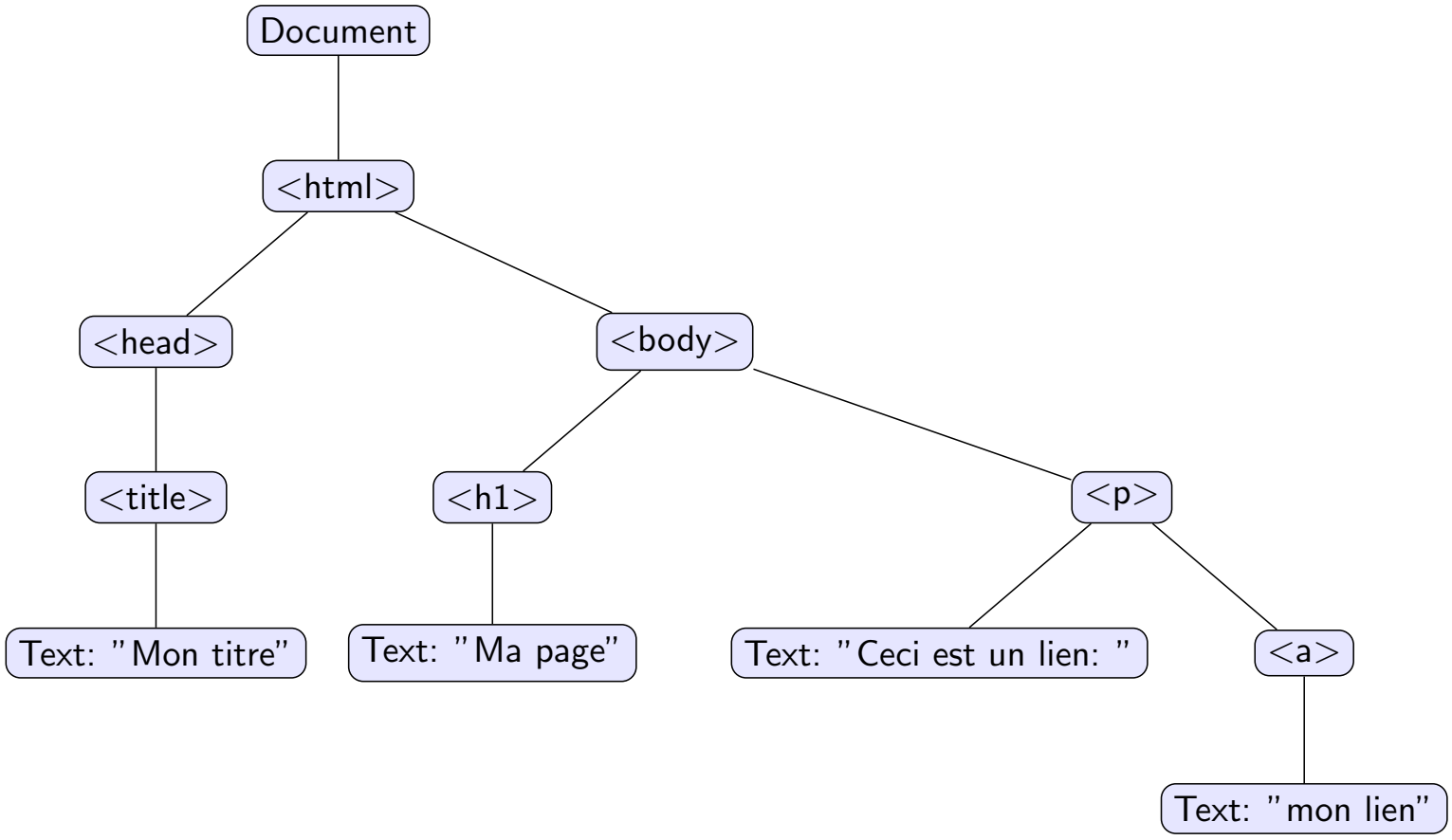
Manipulation des éléments HTML

Pour JavaScript, les différents éléments d'une page HTML sont des objets que l'on peut manipuler (interface DOM)

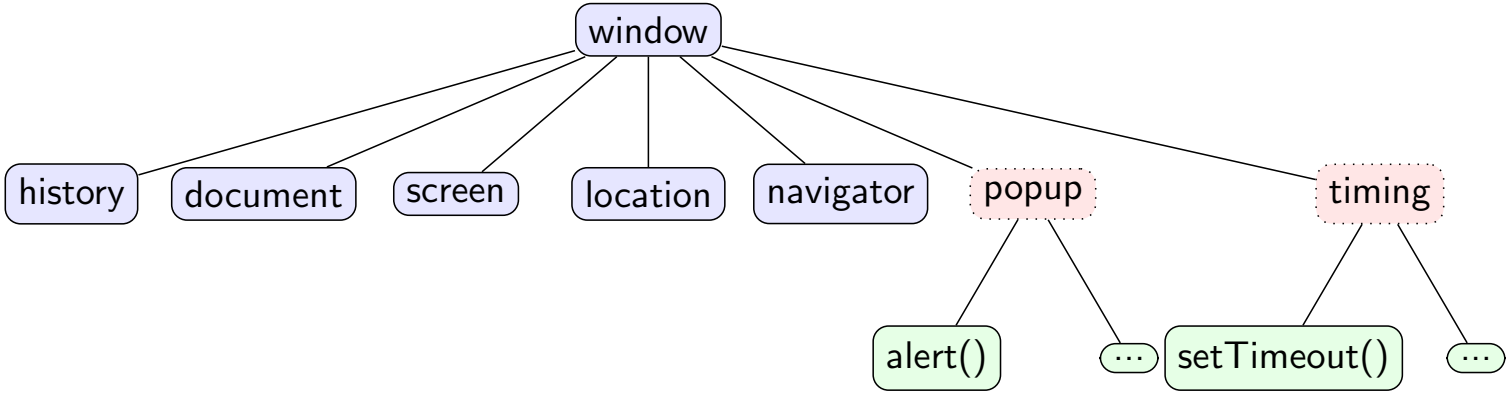
- accès aux éléments: `getElementById()`, `getElementsByTagName()` et `getElementsByName()`, `querySelector()` et `querySelectorAll()`
- modifications:
  - des propriétés: attributs `setAttribute()`, code HTML à l'intérieur de l'élément `innerHTML`
  - des méthodes de l'objet (fonctions s'exécutant avec les attributs de l'objet)
  - des événements associés à l'objet

HTML DOM (Document Object Model)

page web = arbre d'objets DOM



Hiérarchie d'objets Browser Object Model



Gestion des événements

Événements déclenchés auxquels on associe du code JS

5 catégories

- document (chargement nouvelle page, sortie ancienne page) (onLoad, onUnload)
- form (interaction avec le formulaire)
- ancre (click sur le lien)
- element (statut des images associées à une page)
- fenêtre (quelle fenêtre est active)

Exemple: les liens (ancres)

- événements associés: onMouseOver, onMouseOut, onClick
- association code JavaScript et événement:
  texte de l'ancre

Association événement-code

Peut aussi être faite par programme

```
<body onFocus="maFocusFonction();" onBlur = "maBlurFonction();">
<script>
window.document.onfocus = maFocusFonction;
window.document.onblur = maBlurFonction;
</script>
```

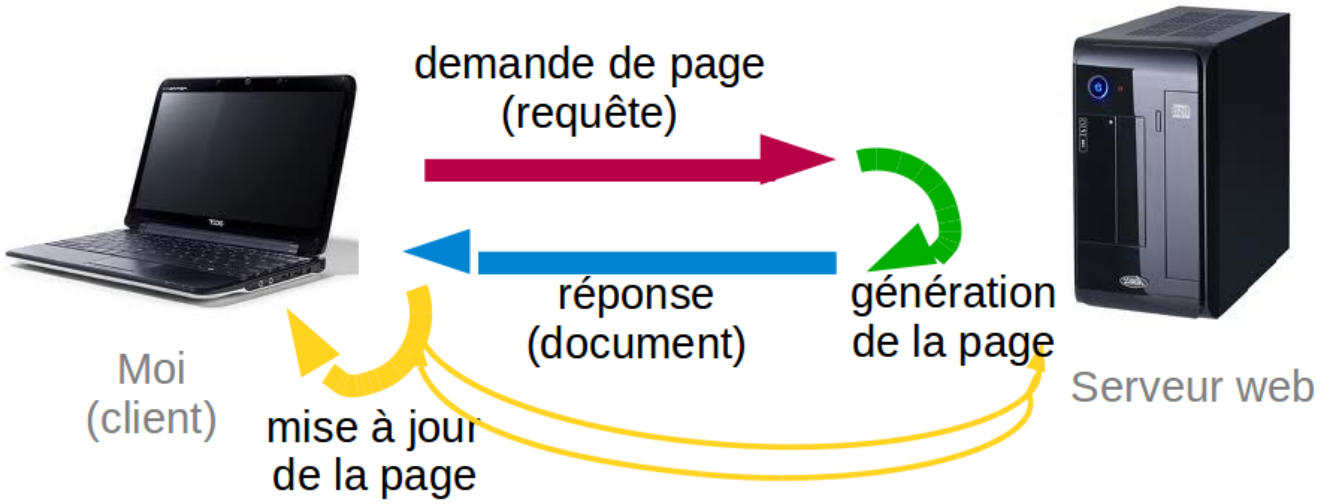
Exemple de page

```
<html>
<head>
<title> Test balise input</title>
<script> Function verifNum(valeur){
//vérification que la chaîne de caractères représentée par le
//paramètre "valeur" ne contient que des caractères numériques
for (var i=0; i<valeur.length; i++){
   var caractere=valeur.substring(i , i+1);
   if (caractere < "0" || caractere > "9")
      return false;
   }
   return true;
}
</script></head>
<body>
<form>
<input name="montant" onChange="if (!verifNum(this.value)){alert ("ne saisissez
que des chiffres!"); return false}">
</form>
</body>
</html>
```

AJAX

- AJAX = Asynchronous JavaScript and XML
  - client “riche”
  - échange de données avec un serveur sans mise à jour de la page complète

Accès à une page web avec AJAX



Exemple d'utilisation d'AJAX

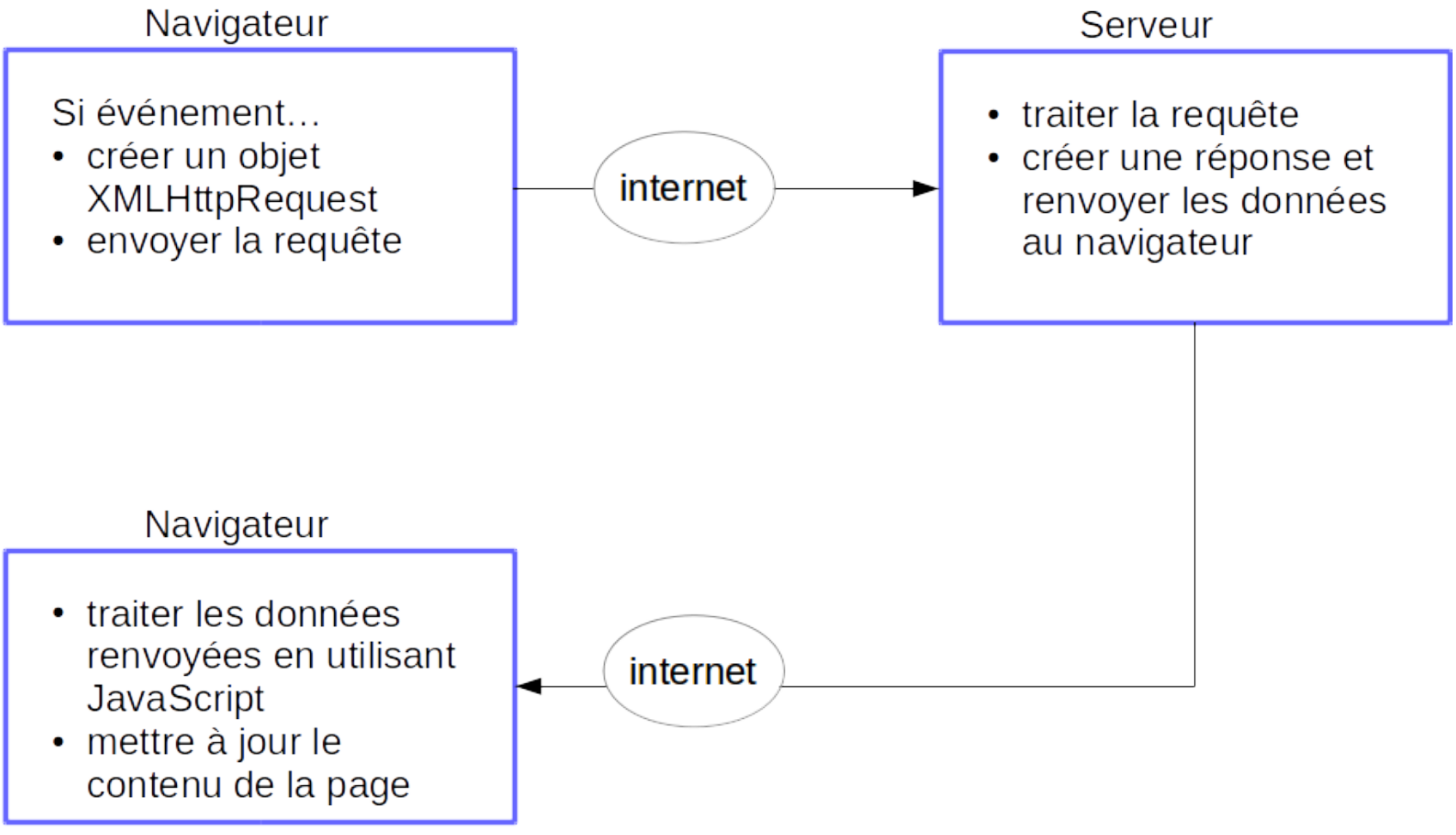
Formulaire web

- sans:
  - demande à l'utilisateur de saisir des informations
  - vérification de format avec JavaScript
  - envoyer la page au serveur pour validation
- avec:
  - validation des données entrées
  - chargement à la volée d'informations en fonction des entrées
  - exemple: Google Suggest

Exemple

formulaire avec suggestions

Fonctionnement d'Ajax



2. JSON
   - Présentation de JSON
   - Petit détour par XML

Échange structuré d'information

Besoin

échange d'information structurée entre applications

- lecture et écriture simplifiée (parser et pretty-printer génériques)
- validation stricte possible

Solutions

- format texte ad-hoc
- format texte structuré (csv)
- format binaire ad-hoc
- XML
- JSON : JavaScript Object Notation

JSON

Syntaxe

format de données textuelles dérivé de la notation des objets du langage JavaScript (Wikipédia)
1re norme : 2003
types de base :

- constante `null`
- booléens : constantes `true` et `false`
- nombres
- chaînes de caractères, délimitées par doubles guillemets (") , avec séquences d'échappement habutuelles (`\n`...)
- tableaux : suite de valeurs séparées par des virgules et entre crochets ([et])
- objets : collection de couples nom-valeur, les noms étant des chaînes de caractères

JSON

Objets : exemple

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "defeats" : null,
  "members": [
    { "name": "Molecule Man",
      "age": 29,
      "secretIdentity": ["firstname" : "Dan", "lastname" : "Jukes"],
      "powers": [ "Radiation resistance", "Turning tiny"]
    },
    { "name": "Madame Uppercut",
      "age": 39,
      "secretIdentity": ["firstname" : "Jane", "lastname" : "Wilson"],
      "powers": [ "Million tonne punch", "Damage resistance" ]
    }
  ]
}
```

JSON

Syntaxe : détails

- pas de syntaxe pour les commentaires
- une propriété est n'importe quelle chaîne syntaxiquement valide
- les tableaux peuvent contenir des types différents
- les espaces en dehors des chaînes ne sont pas significatifs
- les chaînes ne peuvent pas être sur plusieurs lignes

API JavaScript pour JSON

deux méthodes :

- `JSON.stringify(v)` : convertit la valeur `v` en une chaîne de caractères représentant son encodage JSON
- `JSON.parse(s)` : décode la chaîne `s` (représentant du JSON) en un objet JavaScript

XML

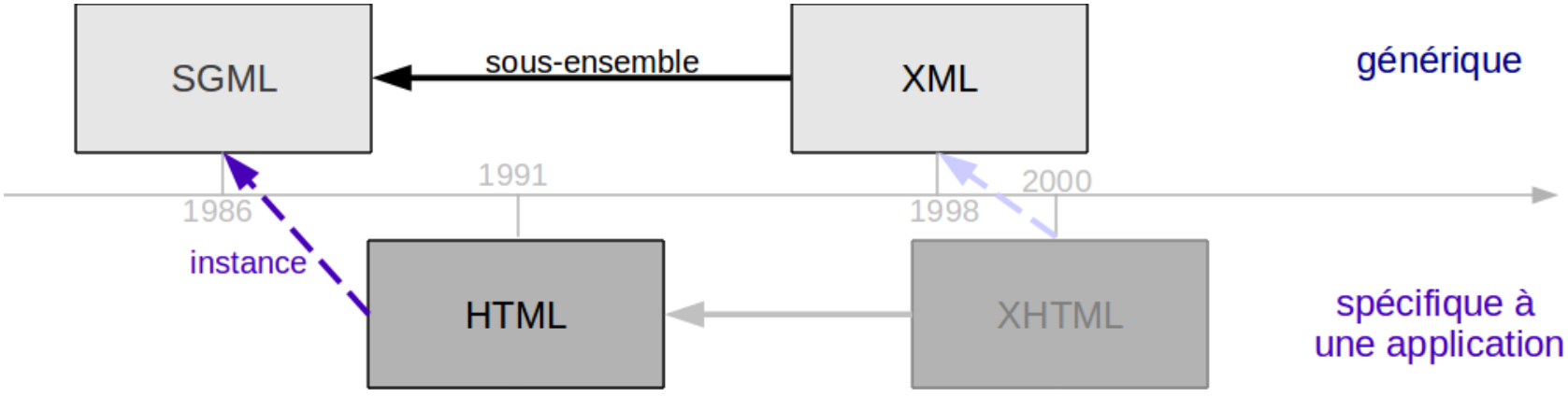
Objectifs de XML

- XML = eXtensible Markup Language
- XML = méta-langage universel pour les données sur le Web, qui permet au développeur de délivrer du contenu depuis les applications à d'autres applications ou aux navigateurs

Apports décisifs

- extensibilité et structure
- modularité et réutilisation de structures types
- contrôle de validité
- dissociation forme (présentation) et fond (données)
- format d'échange général indépendant du matériel et logiciel (texte)
- richesse des standards dérivés

Historique



XML

- langage d'échange de documents fondé sur le balisage
  - plus simple que SGML
  - pas limité "présentation" comme HTML
- développé par le XML Working Group dirigé par le W3C (depuis 1996)
- XML 1.0 = recommandation officielle du W3C depuis le 10 février 1998, 5e version en 2008

Applications

- publication multisupports
- échange de données (B2B, ETL, XML EDI...)
- gestion de documents semi-structurés
- intégration de données hétérogènes
- ...

Commençons par un exemple !

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Catalogue d'une librairie -->
<librairie>
  <livre categorie="enfants">
    <titre langue="en">
      Harry Potter
    </titre>
    <auteur>J K. Rowling</auteur>
    <annee>2005</annee>
    <prix>29,99</prix>
  </livre>
  <livre categorie="programmation">
    <titre langue="fr">
      Apprentissage de la programmation
          avec Ocaml
    </titre>
    <auteur>Catherine Dubois</auteur>
    <annee>2004</annee>
    <prix>39,95</prix>
  </livre>
</librairie>
```

Document XML composé de :

- un prologue
- des éléments organisés hiérarchiquement
  - dont un élément racine
- des commentaires

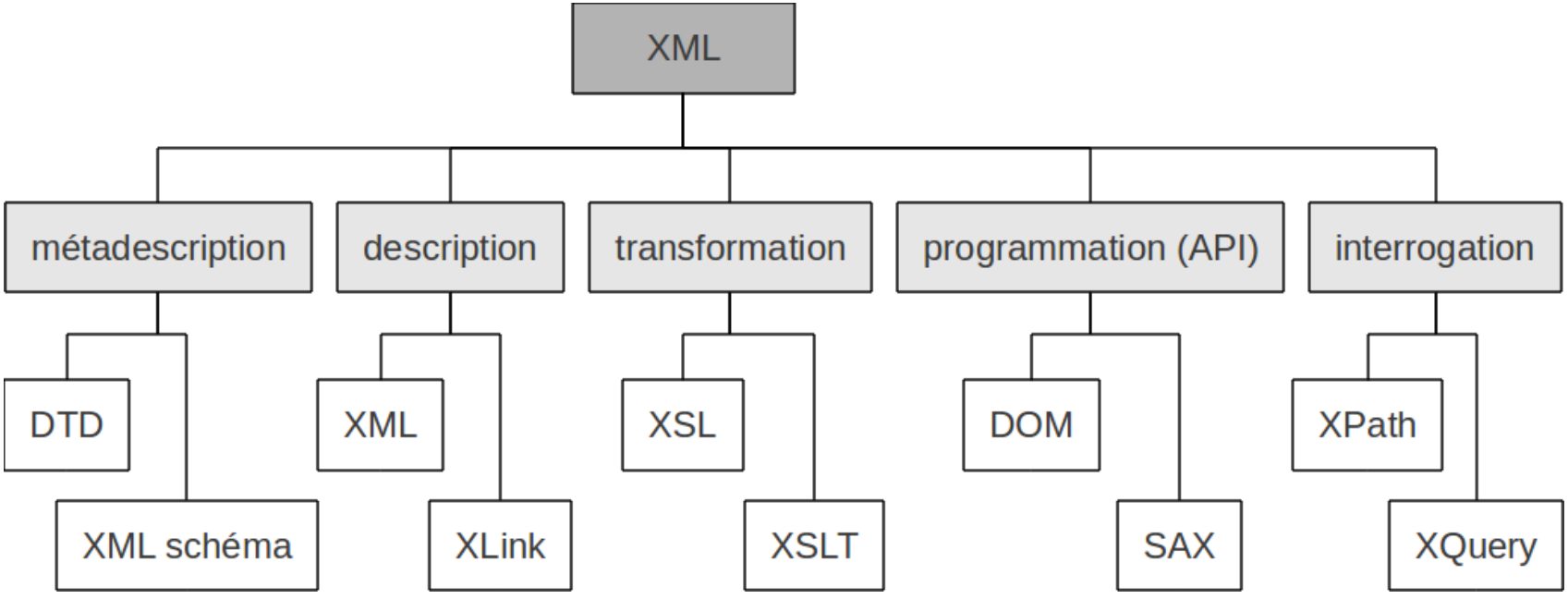
Deux niveaux de validation

1. document bien formé ?
   - le document correspond-il à une structure d'arbre ?

   $\Rightarrow$ respect de la syntaxe XML
2. document valide ?
   - le document correspond-il à un dialecte XML spécifique ?

   $\Rightarrow$ conformité à une structure et un typage prévus

Différents standards



XML vs JSON

JSON: JavaScript Object Notation

XML

```
<employees>
    <employee>
        <firstName>John</firstName>
        <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName>
        <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName>
        <lastName>Jones</lastName>
    </employee>
</employees>
```

JSON

```
{"employees":[
    {"firstName":"John",
     "lastName":"Doe"},
    {"firstName":"Anna",
     "lastName":"Smith"},
    {"firstName":"Peter",
      "lastName":"Jones"}
]}
```

XML vs JSON

XML

+	• lisibilité • annotations • validation • typage • outils/standards • durée dans le temps, standard partagé
-	• taille des données, verbeux

JSON

+	• lisibilité • syntaxe simple • facilité d'intégration en programmation web • validation (JSON schéma)
-	• annotation