



XML : Introduction

- L'*Extensible Markup Language* (XML), « langage de balisage extensible » en français.
- XML est un métalangage informatique de balisage générique qui dérive du SGML.
- XML a été créé par l'organisme W3C.



XML : Exemple

```
<?xml version="1.0" encoding="UTF-8"?>
<vehicule>
  <type>4 Roues</type>
  <constructeur>Quadro</constructeur>
  <annee>2018</annee>
  <couleur>Rouge</couleur>
  <modele>Scooter</modele>
  <matricule>9999AB</matricule>
</vehicule>
```



XML : Règles

L'entête :

Exemple : `<?xml version="1.0" encoding="UTF-8"?>`

Le document doit contenir au moins une balise.

Le nom de chaque balise doit respecter les règles suivantes :

- ❑ Le nom peut contenir des lettres, des chiffres ou d'autres caractères.
- ❑ Il ne doit pas commencer par un chiffre ou un signe de ponctuation.
- ❑ Il ne doit pas non plus commencer par xml, Xml, XML...
- ❑ Aucun espace dans le nom d'une balise n'est toléré.



XML : Règles

- ❑ Chaque balise ouvrante doit avoir une balise fermante :

Exemple : `<exemple></exemple>`

- ❑ Si la balise est vide, `<exemple></exemple>` par exemple, une balise abrégée peut être utilisée en lieu et place, comme ceci : `<exemple/>`

- ❑ Les balises ne peuvent se chevaucher.

Exemple : `<data><element></data></element>`

- ❑ Les balises sont sensibles à la casse.

- ❑ Exemple : `<ELEMENT><element>`



XML : Règles

- ❑ Les valeurs des attributs des balises doivent **obligatoirement** être encadrées avec des quotes simples/doubles.
- ❑ Toutes les balises du document doivent obligatoirement être contenues dans une balise unique englobant tout le document. Cette balise s'appelle également la balise racine. Exemple :

```
<data>  
  <element attribut="valeur">élément 1</element>  
</data>
```



XML : Règles

Les caractères spéciaux interdits sont :

< (Doit être remplacé par<)

> (Doit être remplacé par>)

& (Doit être remplacé par&)

En revanche, il est possible d'utiliser une balise spéciale qui indique aux interpréteurs de ne pas tenir compte des caractères spéciaux. Exemple :

```
<description>  
    <![CDATA[Dans cette description, "l'interdit" est bravé <o_o>.]]>  
</description>
```



XML : Attributs

```
1 <personne sexe="F">  
2   <prenom>Sara</prenom>  
3   <nom>Smith</nom>  
4 </personne>
```

```
1 <personne>  
2   <sexe>F</sexe>  
3   <firstname>Sara</firstname>  
4   <lastname>Smith</lastname>  
5 </personne>
```



XML : Exercice

Créez un fichier XML nommé `iut.xml`, dont l'élément racine est `iut`. Les éléments principaux nommés `etudiant` ont comme attributs `id` (numéro d'inscription) et `nom`. Chaque élément `etudiant` peut contenir autant d'éléments `uv` que désiré. Chaque UV doit avoir un nom, une durée et une note enregistrés dans des sous-éléments. Visualisez ce fichier dans un navigateur pour vérifier qu'il est bien formé.



JSON : Introduction

- JSON: JavaScript Object Notation.
- JSON permet le stockage et d'échange de données.
- JSON est écrit avec la notation d'objet JavaScript.



JSON : Règles

La syntaxe JSON est dérivée de la syntaxe de notation d'objet JavaScript :

- Les données sont dans des paires clé/valeur.
- Les données sont séparées par des virgules.
- Les accolades contiennent des objets.
- Les crochets contiennent des tableaux



JSON : Règles

data.json

```
1 {"employees":[
2   { "firstName":"John", "lastName":"Doe" },
3   { "firstName":"Anna", "lastName":"Smith" },
4   { "firstName":"Peter", "lastName":"Jones" }
5 ]}
```

data.xml

```
1 <employees>
2   <employee>
3     <firstName>John</firstName> <lastName>Doe</lastName>
4   </employee>
5   <employee>
6     <firstName>Anna</firstName> <lastName>Smith</lastName>
7   </employee>
8   <employee>
9     <firstName>Peter</firstName> <lastName>Jones</lastName>
10  </employee>
11 </employees>
```



JSON : Règles

Type	Exemple
String	<code>{"name" : "John"}</code>
Number	<code>{"age" : 30}</code>
Object	<code>{"employee" : {"name" : "John", "age" : 30, "city" : "New York"}}</code>
Array	<code>{"employees" : ["John", "Anna", "Peter"]}</code>
Boolean	<code>{"sale" : true}</code>
Null	<code>{"middlename" : null}</code>

JSON : Exercice

Transformer la ressource ci-dessous en JSON :

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <person>
3   <firstName>John</firstName>
4   <lastName>Smith</lastName>
5   <age>22</age>
6   <address>
7     <streetAddress>21 2nd Street</streetAddress>
8     <city>New York</city>
9     <state postalCode="10021" name="NY"/>
10  </address>
11  <phoneNumber type="home">
12    212 555-1234
13  </phoneNumber>
14  <phoneNumber type="fax">
15    646 555-4567
16  </phoneNumber>
17 </person>
```

L'Ajax : INTRO

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <p id="hello"></p>
5 <script>
6
7
8 class MyClass {
9     constructor(){
10         this.onload = null;
11         this.responseText = "Hello tout le monde!";
12     }
13 }
14
15 m = new MyClass();
16 m.onload = function(){
17     document.getElementById("hello").innerHTML = this.responseText;
18 }
19 m.onload() ; // au chargement de la page le script sera exécuté (onLoad)
20 </script>
21 </body>
22 </html>
```



L'Ajax : XMLHttpRequest

```
1 <!DOCTYPE html>
2 <html>
3 <meta charset="utf-8"/>
4 <body>
5 <script>
6
7
8 // Create an XMLHttpRequest object
9 var xmlhttp = new XMLHttpRequest();
10
11 // Define a callback function
12 xmlhttp.onload = function() {
13     let data = this.responseText;
14     console.log(data);
15 }
16
17 // Send a request
18 xmlhttp.open("GET", "https://geo.api.gouv.fr/regions");
19 xmlhttp.send();
20
21 </script>
22 </body>
23 </html>
```



L'Ajaj : Parsing JSON

```
1 <!DOCTYPE html>
2 <html>
3 <meta charset="utf-8"/>
4 <body>
5 <script>
6
7
8 // Create an XMLHttpRequest object
9 var xmlhttp = new XMLHttpRequest()
10
11 // Define a callback function
12 xmlhttp.onload = function() {
13     let data = JSON.parse(this.responseText)
14     console.log(data[0].nom)
15 }
16
17 // Send a request
18 xmlhttp.open("GET", "https://geo.api.gouv.fr/regions")
19 xmlhttp.send()
20
21 </script>
22 </body>
23 </html>
```




L'Ajax : Parsing XML

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <person>
3      <firstName>John</firstName>
4      <lastName>Smith</lastName>
5      <age>22</age>
6      <address>
7          <streetAddress>21 2nd Street</streetAddress>
8          <city>New York</city>
9          <state postalCode="10021" name="NY"/>
10     </address>
11     <phoneNumber type="home">
12         212 555-1234
13     </phoneNumber>
14     <phoneNumber type="fax">
15         646 555-4567
16     </phoneNumber>
17 </person>
```

L'Ajax : Parsing XML

```
1 <!DOCTYPE html>
2 <html>
3 <meta charset="utf-8"/>
4 <body>
5
6
7 <span id="demo"></span>
8
9 <script>
10
11 var xmlhttp = new XMLHttpRequest()
12
13 xmlhttp.onload = function() {
14     let data = this.responseXML
15     console.log(data)
16     document.getElementById("demo").innerHTML = data.getElementsByTagName("firstName")[0].innerHTML
17 }
18
19 xmlhttp.open("GET", "http://localhost/api/data.xml")
20 xmlhttp.send()
21
22 </script>
23 </body>
24 </html>
```