

XPath

- Qu'est-ce que XPath ?
- Analogies entre XPath et file systems
- Fonctionnalités de base :
 - Sélection d'un ensemble de nœuds
 - Sélection de nœud d'attribut
- Modèle de données XPath
- Chemins et étapes de localisation, tests de nœud, jokers
- Axes XPath
- Formes abrégées et verbeuses
- Predicats, collections, prédicats composés
- Fonctions XPath :
 - fonctions d'ensembles de nœuds,
 - fonctions de chaînes,
 - fonctions booléennes,
 - fonctions numériques
- Divers

Standard du w3c

<http://www.w3c.org/TR/xpath>

(syntaxe non XML)

XPath est un langage non XML utilisé pour **adresser** des items dans un document XML
(modèle logique)

Il est intensivement utilisé dans XSLT

- Sa syntaxe élémentaire ressemble à l'expression d'un chemin dans un file system
- Sa syntaxe formelle plus du tout

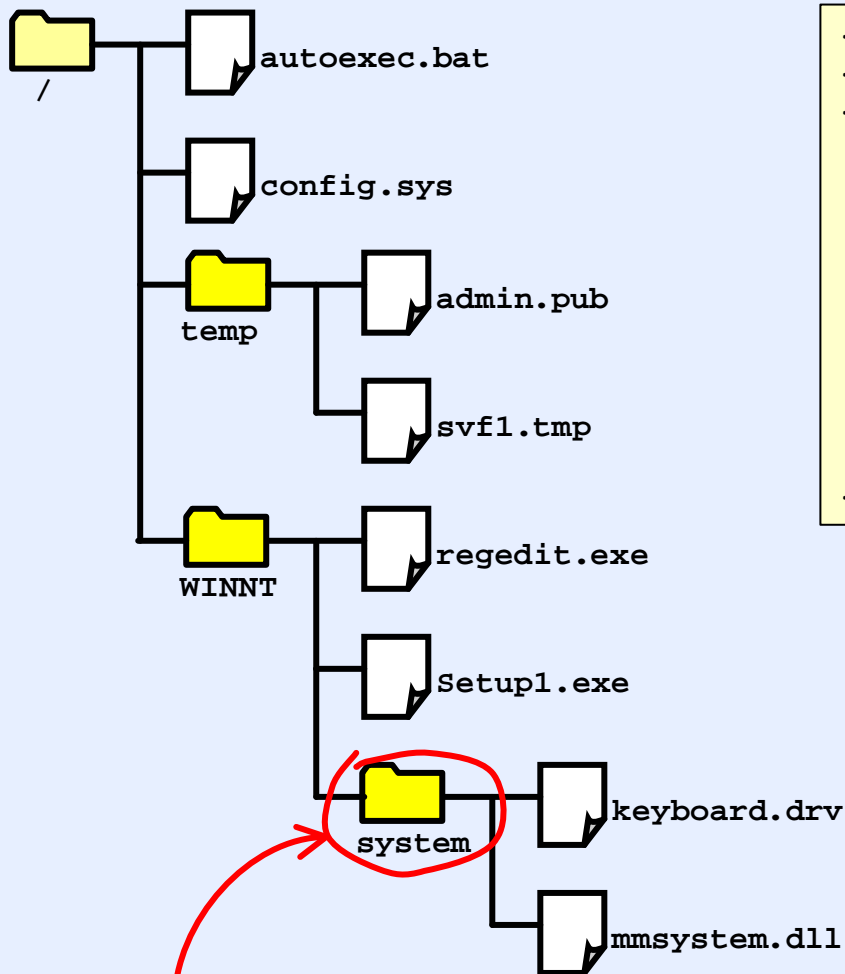
Une **expression XPath** peut s'exprimer :

- relativement, vis à vis d'un **nœud contextuel**
- d'une manière absolue

Le résultat d'une expression XPath peut être :

- **un ensemble de nœuds** (ou un nœud seul)
- une chaîne de caractères
- un nombre
- un booléen

XPath est un langage
de requêtes pour
extraire des données
d'un document XML



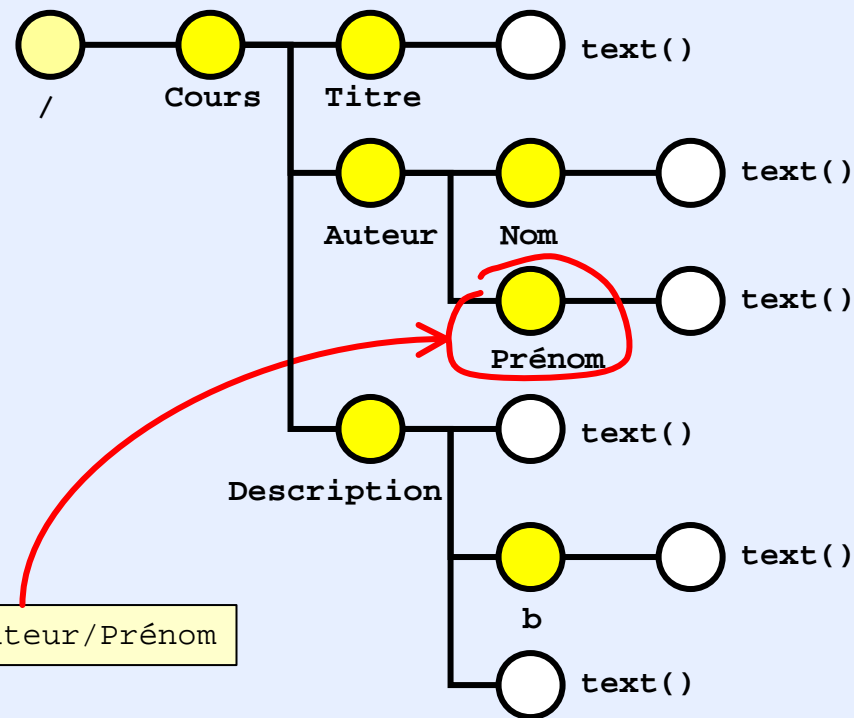
/WINNT/system

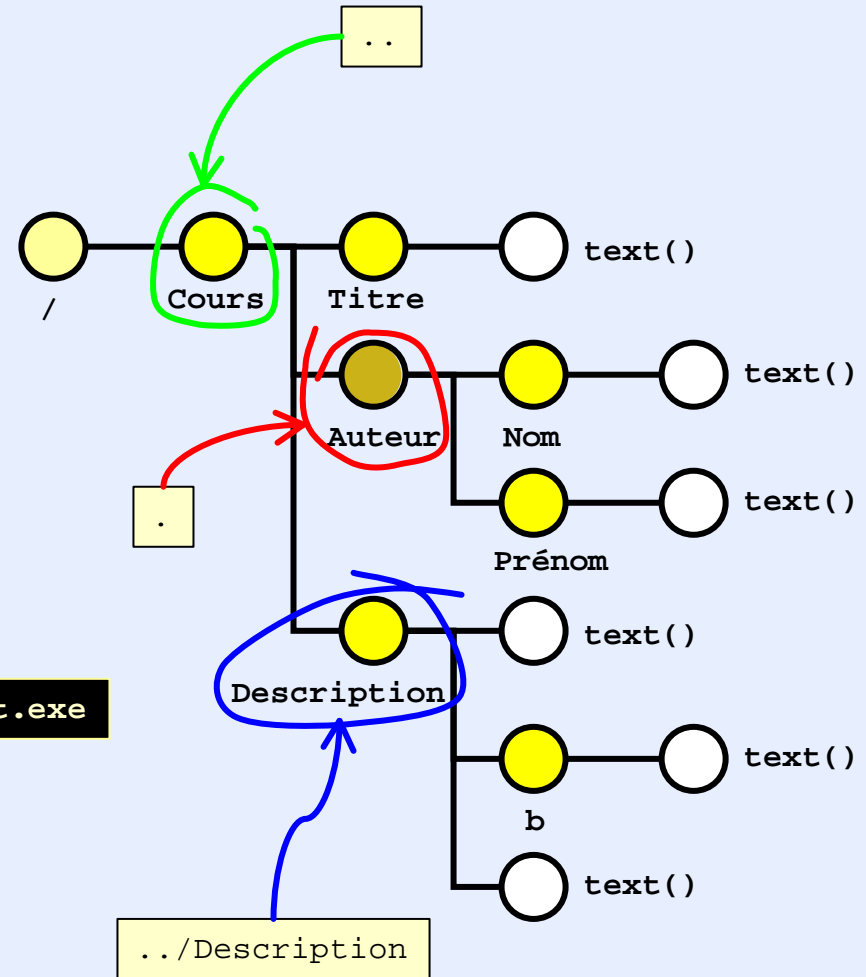
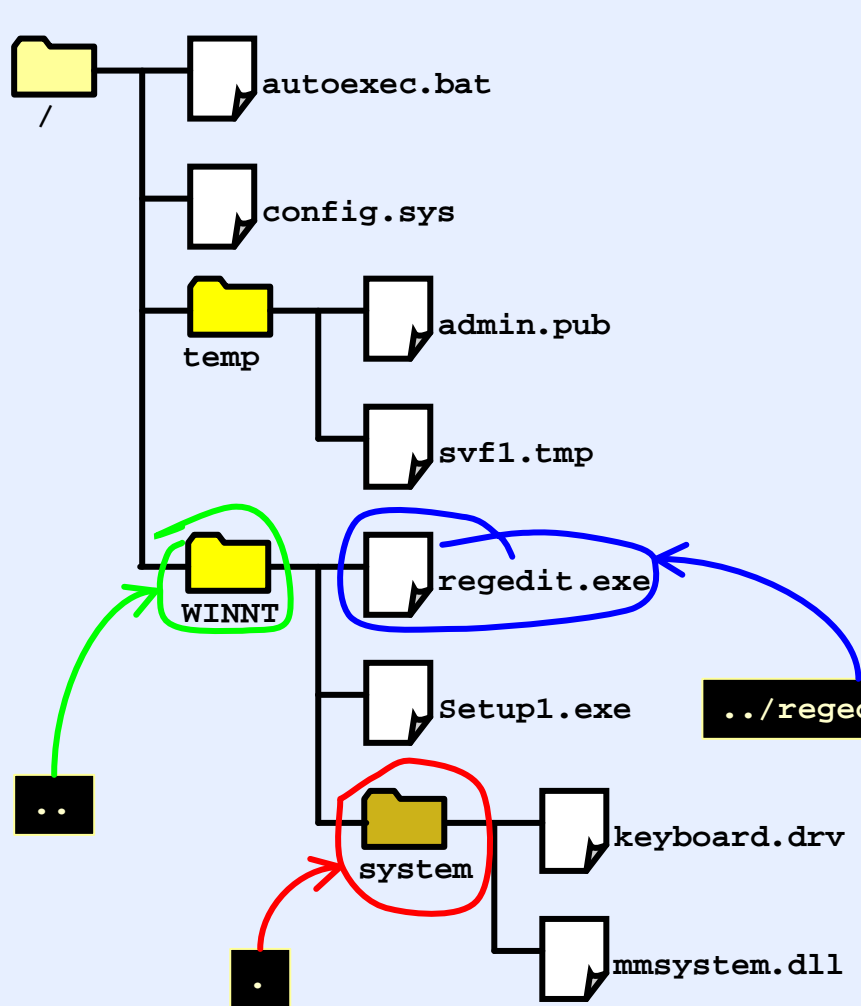
Expression XPath qui retourne 1 nœud

/Cours/Auteur/Prénom

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Cours SYSTEM "Cours.dtd">
<Cours>
  <Titre>Cours XML</Titre>
  <Auteur>
    <Nom>Poulard</Nom>
    <Prénom>Philippe</Prénom>
  </Auteur>
  <Description>
    Ce cours aborde les <b>concepts</b> de
    base mis en &#339;uvre dans XML.
  </Description>
</Cours>
  
```





 Répertoire courant

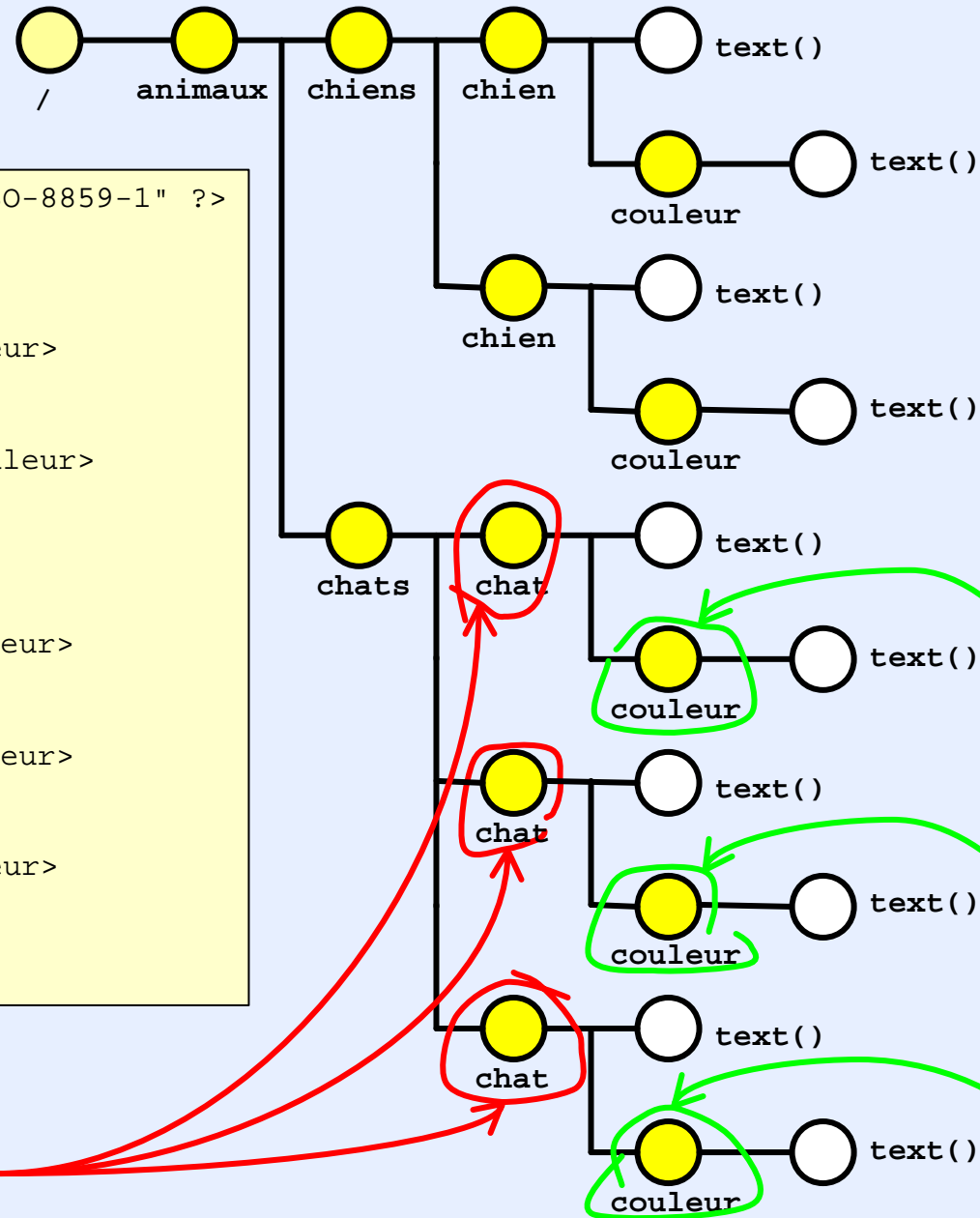
Le répertoire courant est déterminé par l'interpréteur de commandes

 Nœud contextuel

Le nœud contextuel est déterminé par l'application utilisatrice de XPath

	Expressions XPath	File system URLs
Analogies	<p>Hiérarchie constituée d'éléments et d'autres types de nœuds</p> <p>Une expression peut être évaluée à partir d'un nœud spécifique, appelé "le nœud contextuel" de la requête</p>	<p>Hiérarchie constituée de fichiers et de répertoire</p> <p>Une URL peut être évaluée à partir d'un d'un répertoire spécifique, appelé "le répertoire courant"</p>
Différences	<p>Une expression XPath identifie un ensemble d'objets qui peut être un ou plusieurs nœuds</p> <p>A chaque niveau les noms des éléments peuvent ne pas être uniques</p>	<p>Une URL identifie un fichier unique</p> <p>A chaque niveau les noms des fichiers sont uniques</p>

```
<?xml version="1.0" encoding="ISO-8859-1" ?>  
<animaux>  
  <chiens>  
    <chien>Lassie  
      <couleur>noir</couleur>  
    </chien>  
    <chien>Médor  
      <couleur>marron</couleur>  
    </chien>  
  </chiens>  
  <chats>  
    <chat>Félix  
      <couleur>crème</couleur>  
    </chat>  
    <chat>Tom  
      <couleur>crème</couleur>  
    </chat>  
    <chat>Rominet  
      <couleur>gris</couleur>  
    </chat>  
  </chats>  
</animaux>
```



/animaux/chats/chat

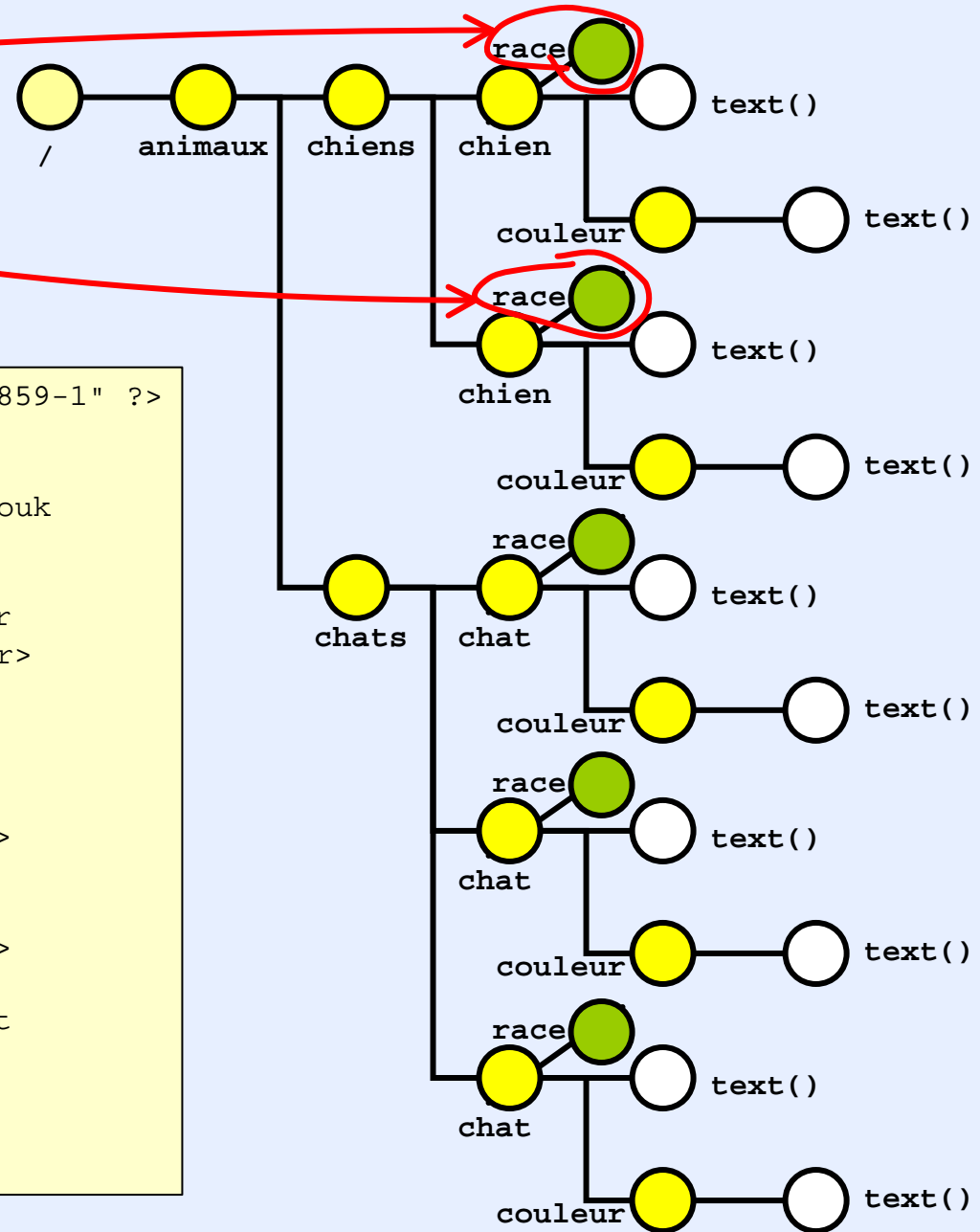
/animaux/chats/chat/couleur

@ → attribut

"at"

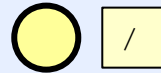
```
/animaux/chiens/chien/@race
```

```
<?xml version="1.0" encoding="ISO-8859-1" ?>  
<animaux>  
  <chiens>  
    <chien race="Labrador">Mabrouk  
      <couleur>noir</couleur>  
    </chien>  
    <chien race="Labrador">Médor  
      <couleur>marron</couleur>  
    </chien>  
  </chiens>  
  <chats>  
    <chat race="Siamois">Félix  
      <couleur>crème</couleur>  
    </chat>  
    <chat race="Birman">Tom  
      <couleur>crème</couleur>  
    </chat>  
    <chat race="Abyssin">Rominet  
      <couleur>gris</couleur>  
    </chat>  
  </chats>  
</animaux>
```

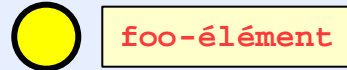


Types de nœuds XPath

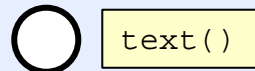
- nœud racine (nœud document)



- nœuds d'élément



- nœuds de texte



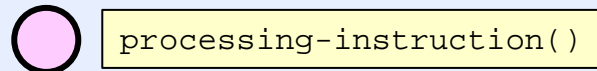
- nœuds d'attribut



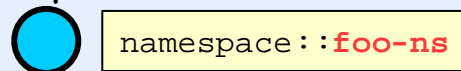
- nœuds de commentaire



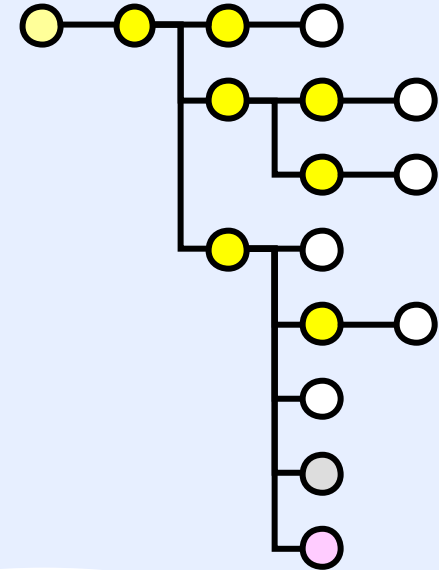
- nœuds d'instruction de traitement



- nœuds d'espace de nom



Document XML → arborescence de nœuds



Sont exclus du modèle :

- les sections CDATA
- les appels d'entités
- la clause DTD
- la déclaration XML

Nœuds XPath

Le modèle n'est pas seulement basé sur les nœuds eux-mêmes mais aussi sur leurs relations.

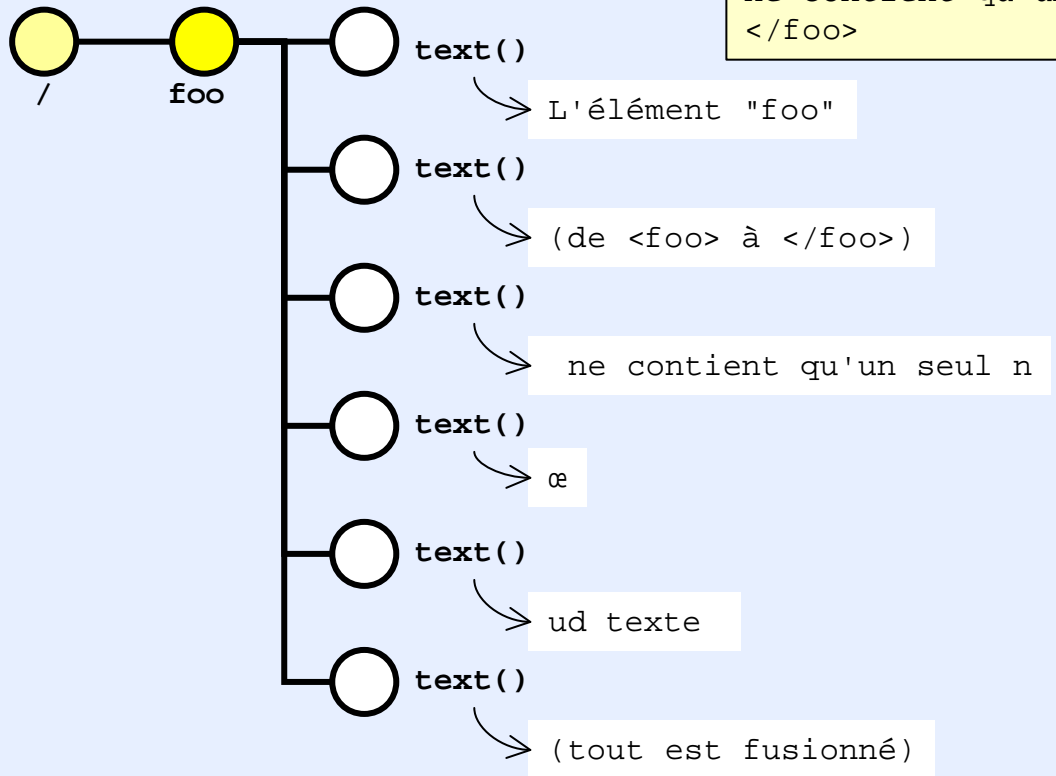
XPath ne permet pas d'accéder directement aux éléments, aux attributs et autres constructions de balisage.

Il permet seulement d'accéder aux nœuds construits à partir de ce balisage.

Les appels d'entités parsées sont résolus

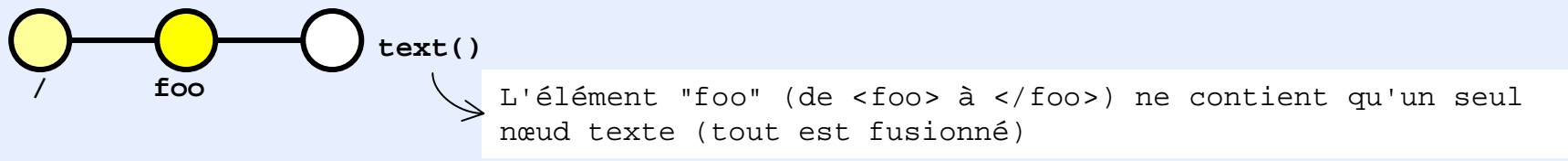
Normalisation








```
<?xml encoding="ISO-8859-1"?>
<!DOCTYPE foo [
    <!ENTITY fusion "(tout est fusionné)">
]>
<foo>L'élément "foo" <![CDATA[(de <foo> à </foo>)]]>
ne contient qu'un seul n&#339;ud texte &fusion;
</foo>
```

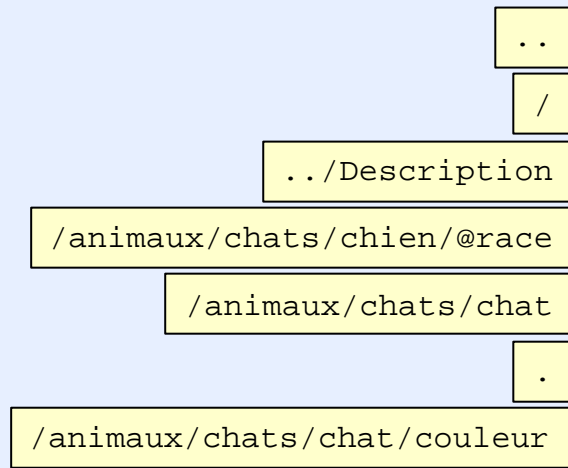


Les appels d'entité, les appels de caractère et les sections CDATA sont indiscernables des contenus textuels

normalization



Construction XML	Caractéristiques du noeud		
	Type	Nom	Valeur
-	Document		-
<élément>	Element		Nom de l'élément
attribut="valeur"	Attr		Nom de l'attribut
texte	Text		Valeur de l'attribut
<!-- commentaire -->	Comment		Texte
<?cible argument?>	ProcessingInstruction		Commentaire
xmlns:préfixe="URI"	Namespace		Nom de la cible
			Préfixe
			URI

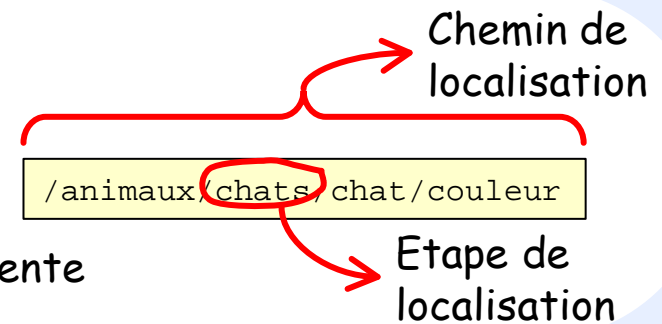


Adressage d'un ensemble hétérogène de nœuds
 → Peut être vide



Chaque étape est évaluée dans le **contexte** de l'étape précédente

Le contexte d'évaluation d'une étape est l'ensemble des nœuds obtenu par l'évaluation de l'étape précédente



Chemin de localisation de la racine : / ⚠ (il s'agit de la racine du document, pas de l'élément racine)

Unions d'étapes de localisation

Avec XSLT, on peut réaliser des unions d'expression avec |

/animaux/chats/chat | /animaux/chiens/chien

/animaux/(chiens|chats)/(chat|chien)

Tests de nœuds nommés

Nœuds d'élément S'exprime par le nom de l'élément

```
chats
```

Nœuds d'attribut S'exprime par le nom de l'attribut précédé de @

```
@race
```

Nœuds d'instruction de traitement

```
processing-instruction("xmlstylesheet")
```

Tests de type de nœuds

Nœuds de commentaire

```
comment()
```

Nœuds de texte

```
text()
```

Nœuds d'instruction de traitement

```
processing-instruction()
```



Les jokers permettent de sélectionner différents items et types de nœuds différents

Joker pour les éléments : *

Exemple : `/animaux/(chiens|chats)/(chien|chat)`

`/animaux/**`

S'il n'y a pas d'autres animaux, ces expressions sont équivalentes

Préfixage par un espace de noms : `svg:*`

[c'est l'URI associée qui est prise en compte, et non pas le préfixe lui-même]
l'espace de nom doit être spécifié au niveau du processeur XPath



dépend du processeur et de l'application utilisatrice

Joker pour les attributs : @*

Exemple : `/animaux/chats/chat/@*`

Sélectionne tous les attributs de l'élément chat, s'il y en a plusieurs

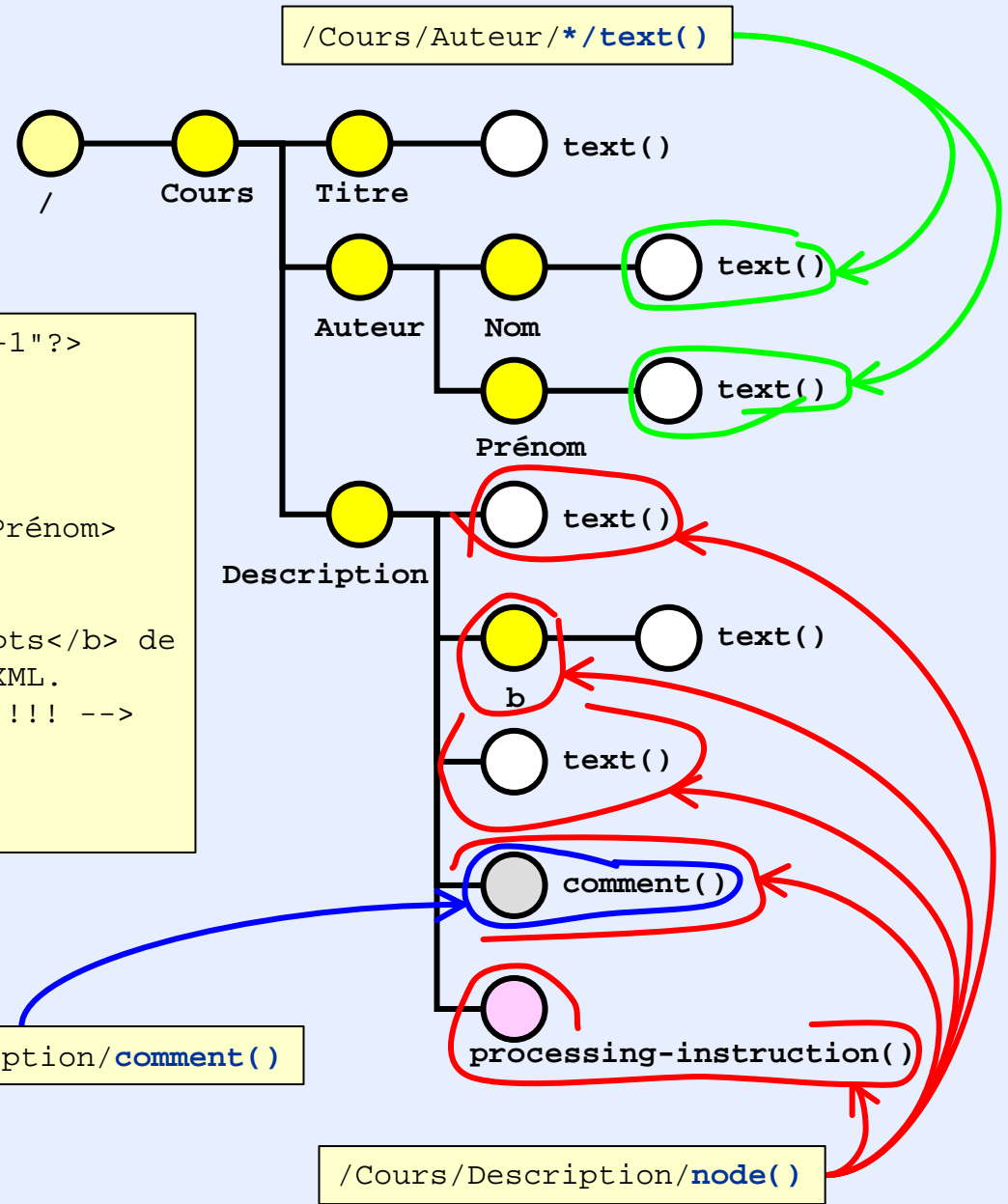
Joker universel : node()

Sélectionne tous les nœuds, quel que soit leur type

Exemple : `/animaux/chats/chat/node()`

Sélectionne tous les nœuds de l'élément chat, s'il y en a plusieurs

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Cours>
  <Titre>Cours XML</Titre>
  <Auteur>
    <Nom>Poulard</Nom>
    <Prénom>Philippe</Prénom>
  </Auteur>
  <Description>
    Ce cours aborde les <b>concepts</b> de
    base mis en &#339;uvre dans XML.
    <!-- c'est le meilleur cours !!! -->
    <?diaporama file="xml.ppt"?>
  </Description>
</Cours>
```



XPath permet de spécifier la "direction" dans laquelle un test de nœud doit être réalisé. Il y a 13 axes différents.

Syntaxe : `axe::test-de-noeud`

XPath propose une syntaxe abrégée pour les axes les plus utilisés.

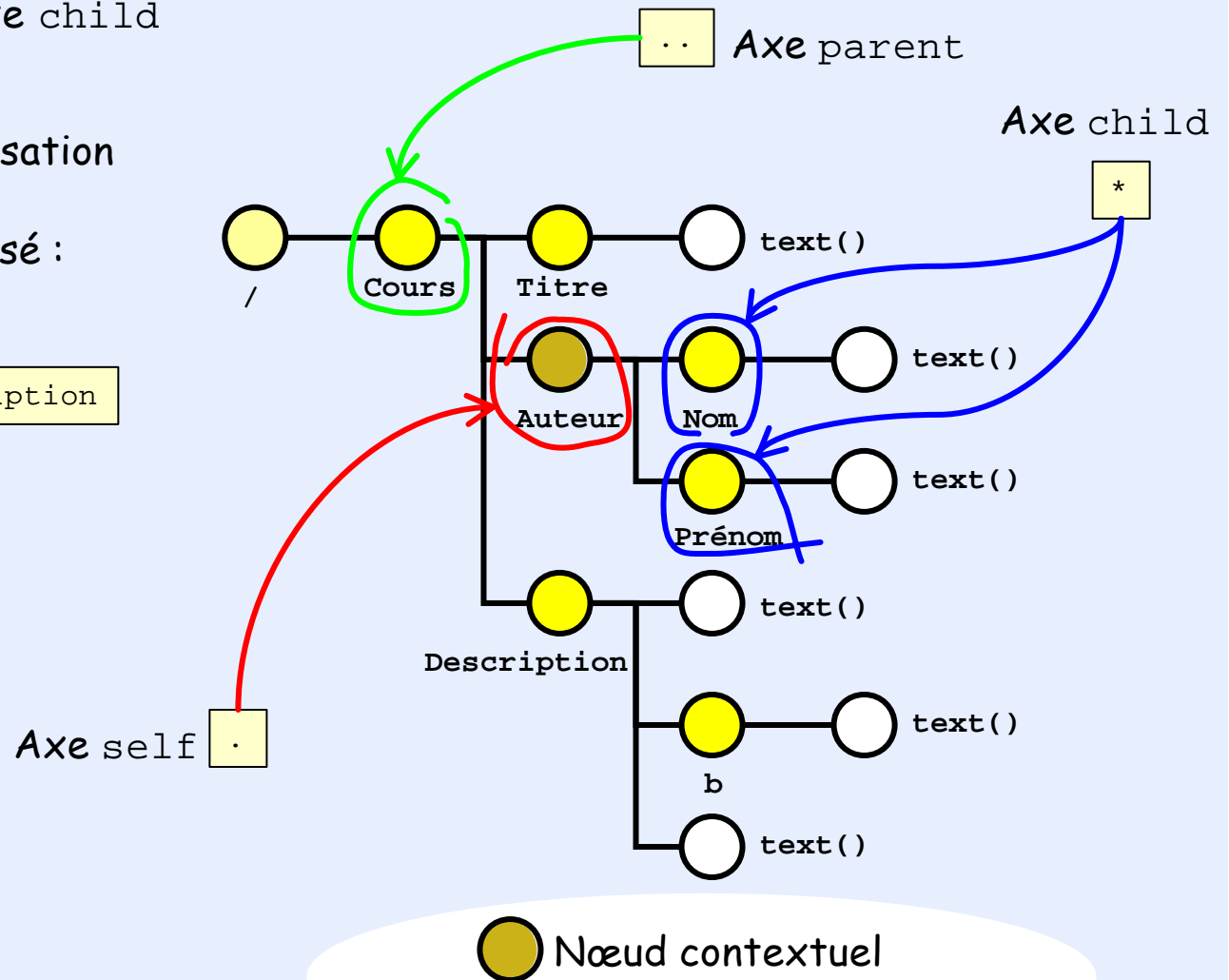
.	=	self::*
..	=	parent::*
*	=	child::*

L'axe par défaut est l'axe child

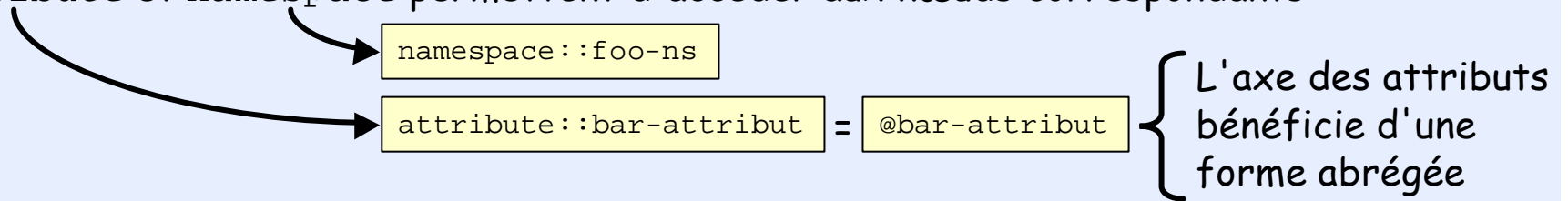
A chaque étape de localisation (séparées par /), un axe différent peut être utilisé :

`../Description`
 = `parent::* / child::Description`

- self
- child
- parent
- attribute
- namespace
- descendant
- descendant-or-self
- ancestor
- ancestor-or-self
- following
- following-sibling
- preceding
- preceding-sibling



2 axes particuliers permettent de sélectionner des types de nœud :
 attribute et namespace permettent d'accéder aux nœuds correspondants



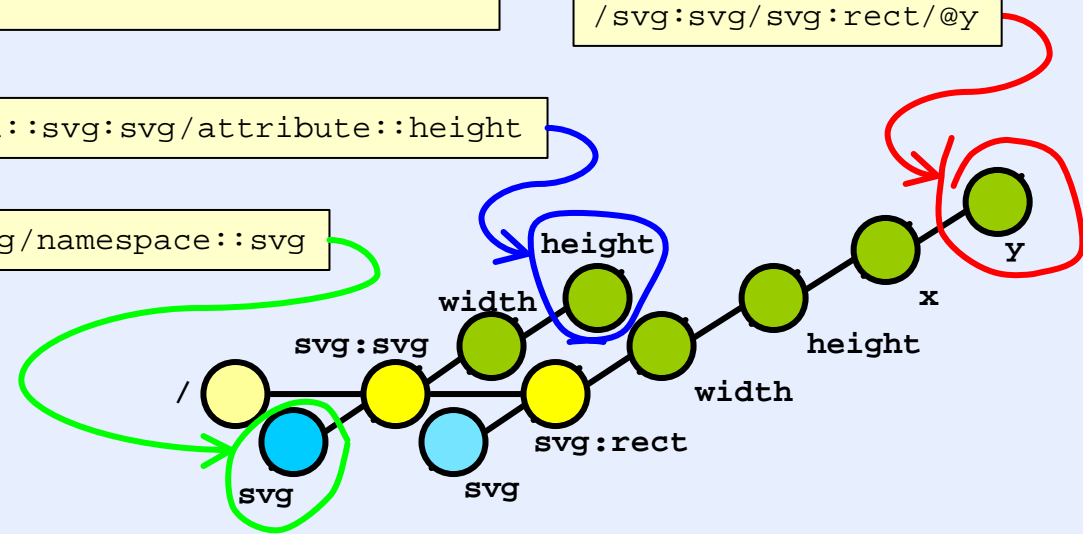
! Les 11 autres axes permettent d'accéder à tous les types de nœud, SAUF les nœuds de type attribut et les nœuds d'espace de nommage

```
<?xml version="1.0"?>
<svg:svg width="18cm" height="6cm"
  xmlns:svg="http://www.w3.org/2000/svg">
  <svg:rect x="1" y="1" width="198"
    height="118"/>
</svg:svg>
```

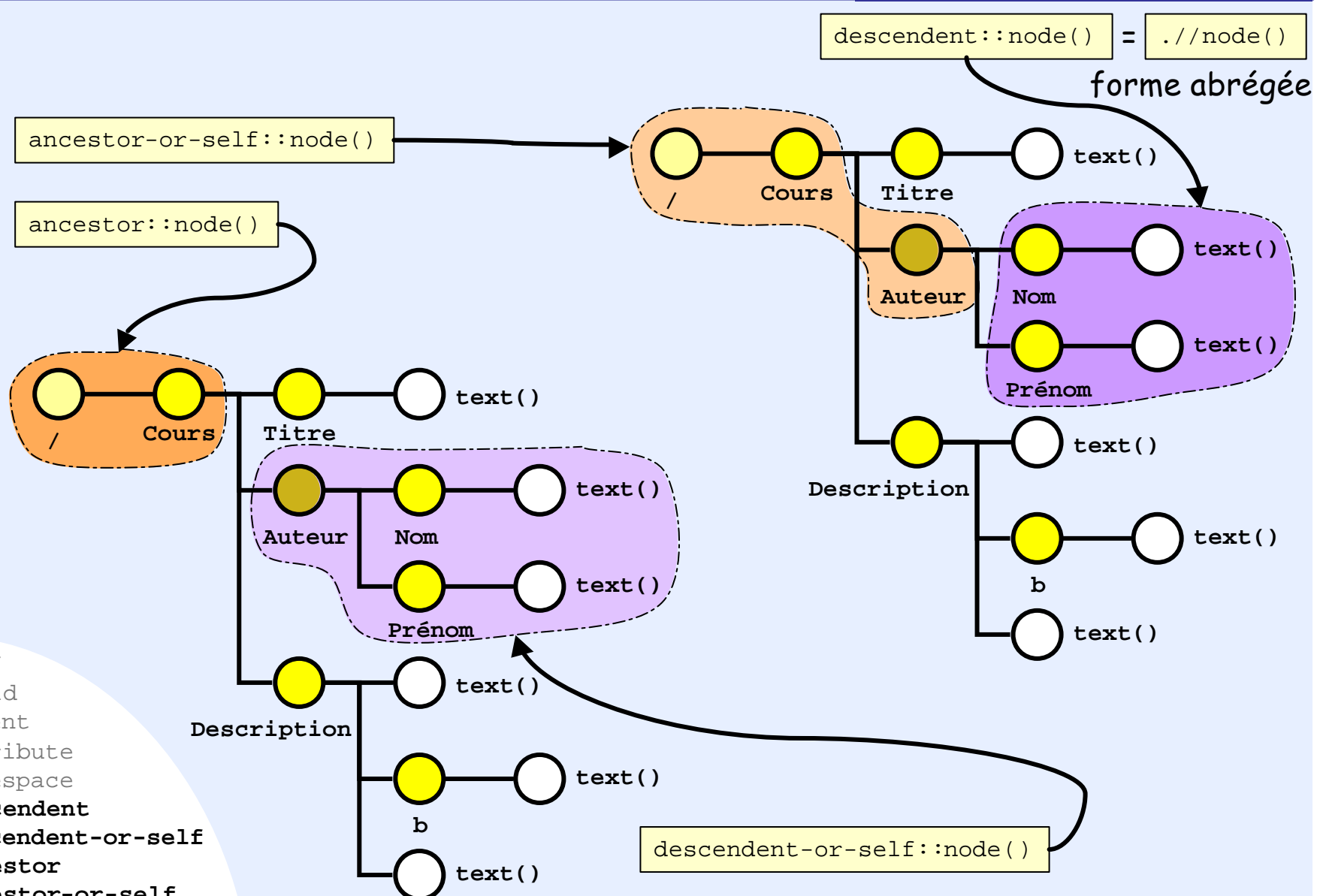
/svg:svg/svg:rect/@y

/child::svg:svg/attribute::height

/child::svg:svg/namespace::svg



- self
- child
- parent
- attribute
- namespace
- descendant
- descendant-or-self
- ancestor
- ancestor-or-self
- following
- following-sibling
- preceding
- preceding-sibling



- self
- child
- parent
- attribute
- namespace
- descendant**
- descendant-or-self**
- ancestor**
- ancestor-or-self**
- following
- following-sibling
- preceding
- preceding-sibling

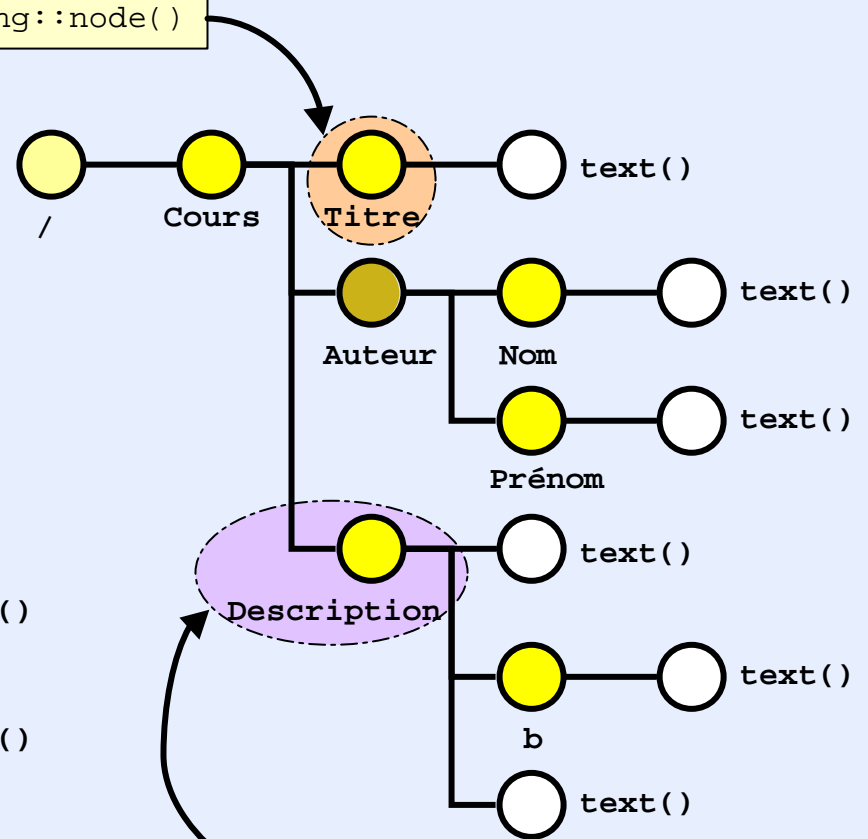
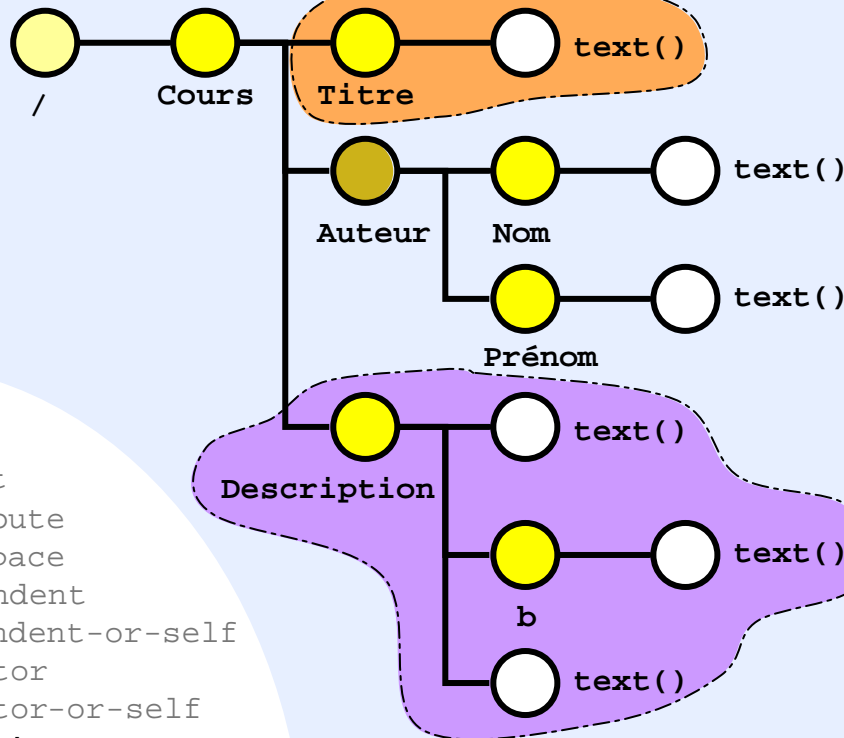
● Nœud contextuel

Même parent que le nœud de référence

`preceding-sibling::node()`

Nœuds terminés avant le début du nœud de référence

`preceding::node()`



`following-sibling::node()`

Même parent que le nœud de référence

`following::node()`

Nœuds commençant après la fin du nœud de référence

● Nœud contextuel

- self
- child
- parent
- attribute
- namespace
- descendant
- descendant-or-self
- ancestor
- ancestor-or-self
- following**
- following-sibling**
- preceding**
- preceding-sibling**

Joker universel : `node()`

Sélectionne tous les nœuds, quel que soit leur type

Exemple : `/animaux/chats/chat/node()` Sélectionne tous les nœuds de l'élément `chat`, s'il y en a plusieurs



Sont exclus : les **attributs** et les **nœuds d'espace de noms**, puisqu'ils ne sont pas dans l'axe par défaut `child`

L'axe `child` contient seulement :

- les nœuds d'élément
- les nœuds de texte
- les nœuds de commentaire
- les nœuds d'instruction de traitement

```
/animaux/chats/chat/child::node()
```

```
/animaux/chats/chat/attribute::node()
```

```
/animaux/chats/chat/attribute::*
```

Ces 2 expressions sont équivalentes puisqu'il n'y a qu'un seul type de nœud dans l'axe `attribut`

Forme verbeuse

```

child::foo-élément
child::*
attribute::foo-élément
attribute::*
/descendant-or-self::node()
self::node()
parent::node()
[position()=n]

```

Exemples

```

/child::animaux
/child::*
attribute::race
attribute::*
/descendant-or-self()::node()/chat
self::node()/text()
parent::node()/chiens
//chat[position()=3]

```

```

/animaux
/*
@race
@*
//chat
text()
../chiens
//chat[3]

```

Forme abrégée

```

foo-élément
*
@foo-élément
@*
//
.
..
[n]

```

Limitation des axes XPath : il n'existe pas d'axe de parcours "naturel" d'un arbre

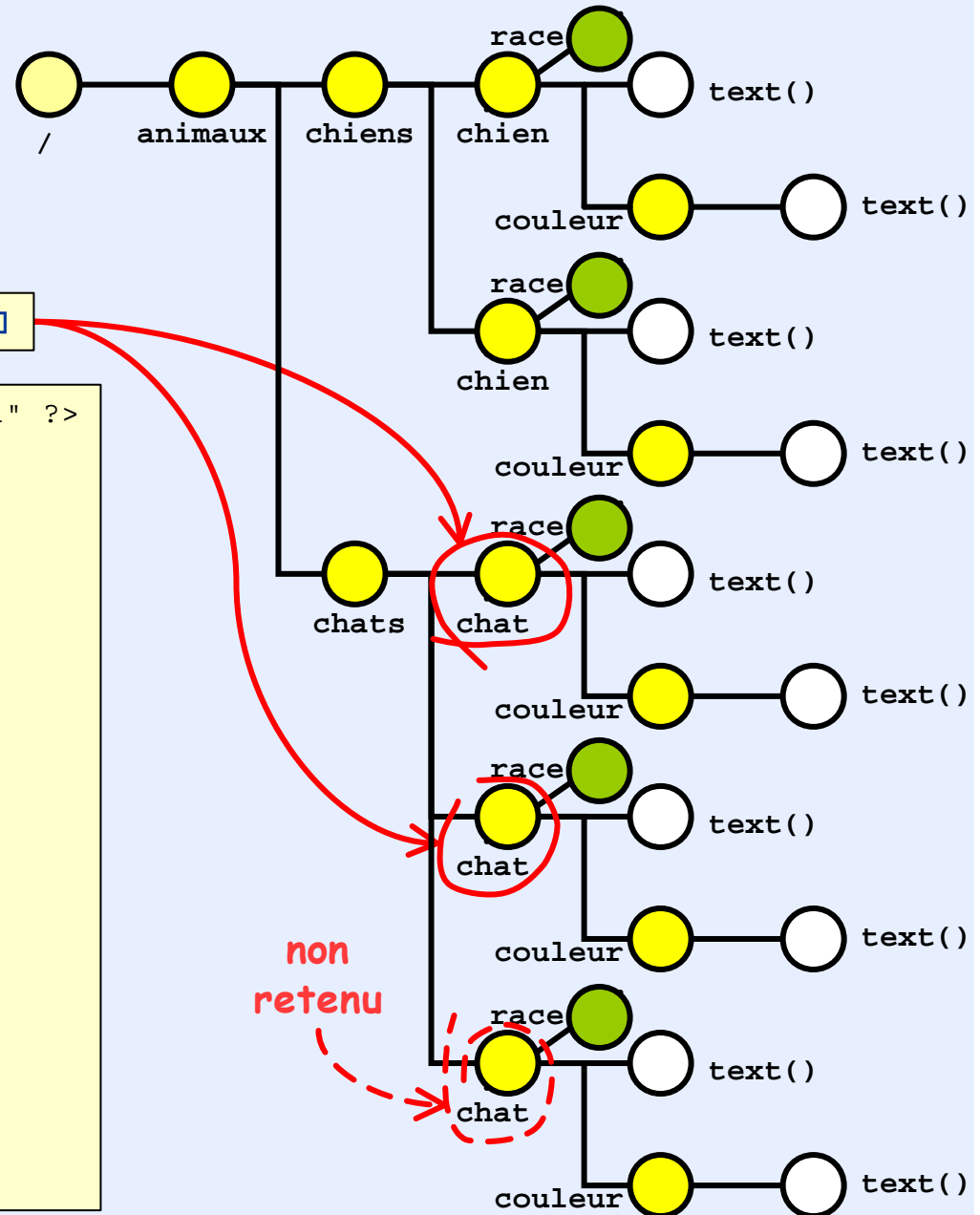
Un prédicat permet de spécifier un filtre dans une étape de localisation

Un prédicat est une expression booléenne qui sera testée pour chaque nœud de l'ensemble des nœuds de l'étape de localisation

`//chat [couleur="crème"]`

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<animaux>
  <chiens>
    <chien race="Labrador">Mabrouk
      <couleur>noir</couleur>
    </chien>
    <chien race="Labrador">Médor
      <couleur>marron</couleur>
    </chien>
  </chiens>
  <chats>
    <chat race="Siamois">Félix
      <couleur>crème</couleur>
    </chat>
    <chat race="Birman">Tom
      <couleur>crème</couleur>
    </chat>
    <chat race="Abyssin">Romet
      <couleur>gris</couleur>
    </chat>
  </chats>
</animaux>
```

Syntaxe : `axe::test-de-noeud[prédicat]`



and
or
not()
=
!=
<
<=
>
>=
()

Test sur un attribut : `//*[*/@race="Birman"]`

`//*[(./@race!="Birman") and (./@race!="Siamois")]`

Expressions non booléennes

Les prédicats qui retournent des expressions non booléennes sont converties :

Type	Résultat	Valeur
Chaîne	Vide	Faux
	Non vide	Vrai
Ensemble de nœuds	Vide	Faux
	Non vide	Vrai
Numérique	Numéro du nœud identique	Vrai
	Numéro du nœud différent	Faux

`//chat [string(couleur)]`

`//chat [couleur]`

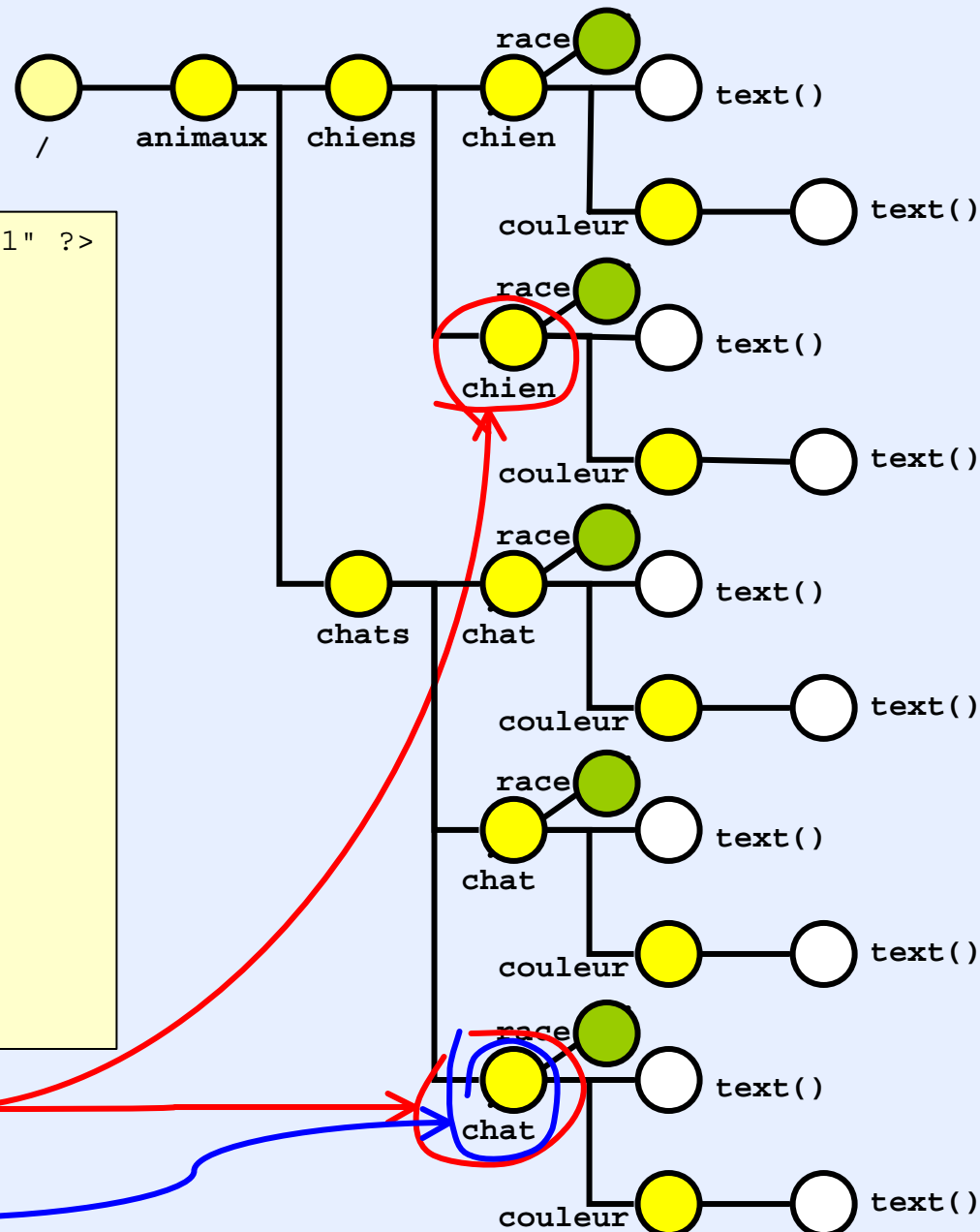
`//chat [2]`

Tous les nœuds peuvent être convertis en chaîne de caractère

Les expressions booléennes peuvent être exprimées avec un jeu de fonctions XPath, comme `string()`, `position()`, `last()`...

Les crochets [et] ont une priorité plus grande que le slash /, mais on peut aussi utiliser des parenthèses (et).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<animaux>
  <chiens>
    <chien race="Labrador">Mabrouk
      <couleur>noir</couleur>
    </chien>
    <chien race="Labrador">Médor
      <couleur>marron</couleur>
    </chien>
  </chiens>
  <chats>
    <chat race="Siamois">Félix
      <couleur>crème</couleur>
    </chat>
    <chat race="Birman">Tom
      <couleur>crème</couleur>
    </chat>
    <chat race="Abyssin">Rominet
      <couleur>gris</couleur>
    </chat>
  </chats>
</animaux>
```



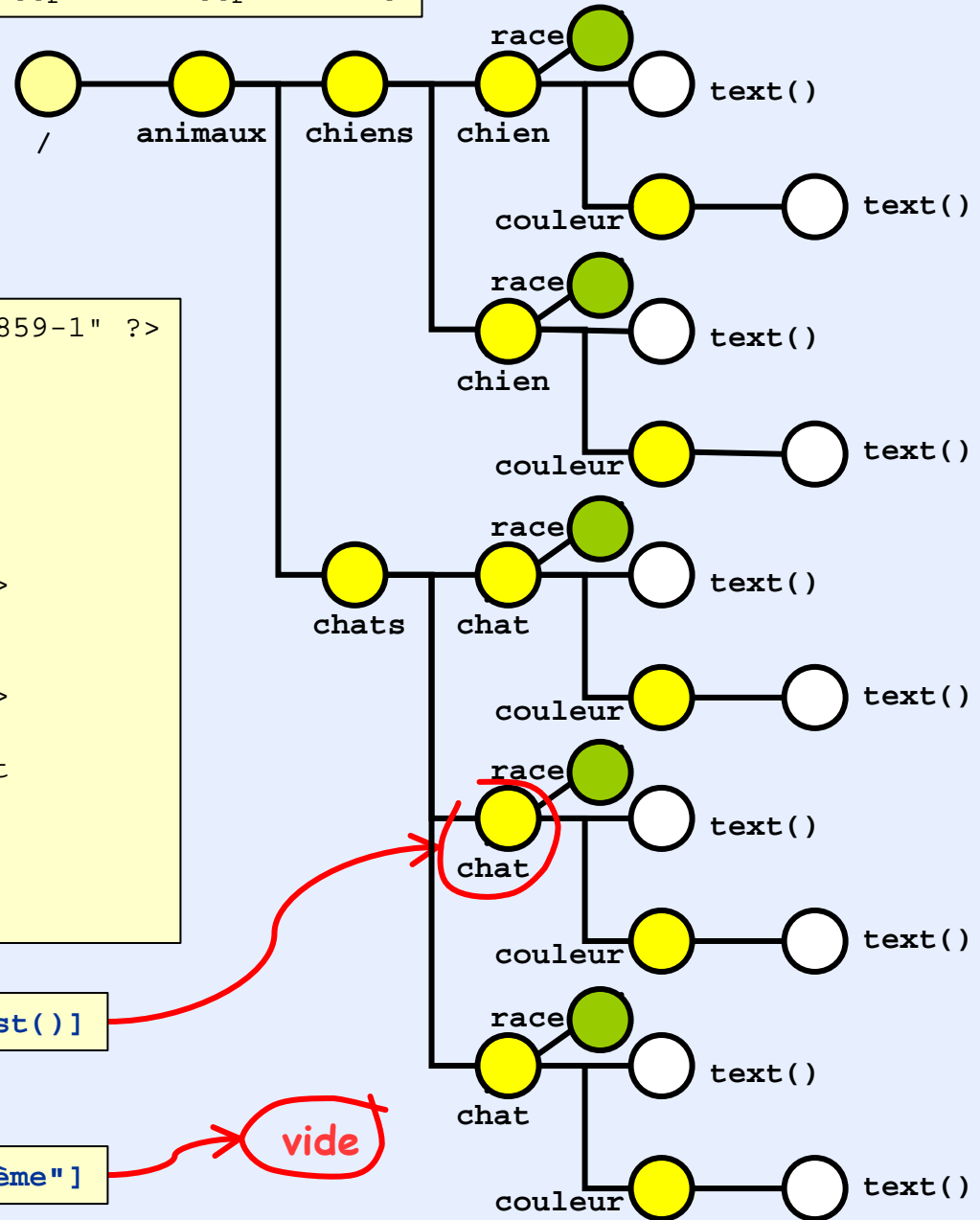
`/animaux/**[position()=last()]`

`(/animaux/**)[position()=last()]`

Syntaxe : `axe::test-de-noeud[prédicat][prédicat][prédicat]`

Les prédicats sont évalués successivement sur le résultat du précédent.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<animaux>
  <chiens>
    .../...
  </chiens>
  <chats>
    <chat race="Siamois">Félix
      <couleur>crème</couleur>
    </chat>
    <chat race="Birman">Tom
      <couleur>crème</couleur>
    </chat>
    <chat race="Abyssin">Rominet
      <couleur>gris</couleur>
    </chat>
  </chats>
</animaux>
```



`//chat [couleur="crème"] [position()=last()]`

`//chat [position()=last()] [couleur="crème"]`

vide

position()	Retourne le numéro d'index du nœud de référence dans le nœud parent <pre style="border: 1px solid black; padding: 2px;">/animaux/*/*[position()=1]</pre> Forme abrégée : <pre style="border: 1px solid black; padding: 2px;">/animaux/*/*[1]</pre>
last()	Retourne le nombre de nœuds de l'ensemble de nœuds de référence <pre style="border: 1px solid black; padding: 2px;">/animaux/*/*[position()=last()]</pre> ou <pre style="border: 1px solid black; padding: 2px;">/animaux/*/*[last()]</pre>
count(node-set)	Retourne le nombre de nœuds de l'ensemble de nœuds passé en argument <pre style="border: 1px solid black; padding: 2px;">/animaux/*[count(*)>2]</pre>
name(node-set?)	Retourne le nom qualifié du premier nœud de l'ensemble de nœuds passé en argument <pre style="border: 1px solid black; padding: 2px;">//*[name(.)!="svg:rect"]</pre>
local-name(node-set?)	Retourne la partie locale du nom du premier nœud de l'ensemble de nœuds passé en argument <pre style="border: 1px solid black; padding: 2px;">//*[local-name(.)!="rect"]</pre>
namespace-uri(node-set?)	Retourne l'URI de l'espace de nom du premier nœud de l'ensemble de nœuds passé en argument <pre style="border: 1px solid black; padding: 2px;">//*[namespace-uri(.)="http://www.w3.org/2000/svg"]</pre>
id(string)	Retourne les nœuds ayant un ID égal à l'un des mots de la chaîne passée en argument <pre style="border: 1px solid black; padding: 2px;">id("ref123 ref456 ref789")</pre>

<code>concat(string*)</code>	Retourne la concaténation des chaînes passées en argument
<code>contains(string, string)</code>	Retourne vrai si la deuxième chaîne passée en argument est contenue dans la première chaîne passée en argument
<code>starts-with(string, string)</code>	Retourne vrai si la première chaîne passée en argument commence par la deuxième chaîne passée en argument
<code>normalize-space(string)</code>	Retourne la chaîne après normalisation des espaces
<code>string(object)</code>	Retourne la chaîne correspondante
<code>string-length(string)</code>	Retourne la longueur de la chaîne passée en argument
<code>substring(string, number, number?)</code>	Retourne la partie de la chaîne passée en argument qui commence à la position indiquée par le 2e argument sur la longueur indiquée par le 3e argument
<code>substring-after(string, string)</code>	Retourne la partie de la chaîne passée en argument qui suit la première occurrence de la 2e chaîne passée en argument
<code>substring-before(string, string)</code>	Retourne la partie de la chaîne passée en argument qui précède la première occurrence de la 2e chaîne passée en argument
<code>translate(string, string, string)</code>	Retourne la chaîne passée en 1er argument dont les caractères de la 2e chaîne passée en argument ont été remplacés par ceux de même index de la 3e chaîne passée en argument

Tous les nœuds peuvent être convertis en chaîne de caractère

`string(object)`

Retourne la chaîne correspondante, en fonction du type :

Type de nœud	Résultat
Text	La valeur du nœud
Comment	La valeur du nœud
ProcessingInstruction	La valeur du nœud
Attribute	La valeur du nœud
Element	La concaténation des valeurs de nœuds texte du contenu
Document	La concaténation des valeurs de nœuds texte du contenu
Namespace	La valeur du nœud



Ensemble de nœuds

Le premier nœud seulement est converti

...donc :

`string(/animaux/*)`

retourne

Mabrouk noir Médor marron

`string(/animaux)`

retourne

Mabrouk noir Médor marron Félix crème Tom crème Rominet gris



Il n'y a pas de fonctions uppercase ni lowercase... mais il y a une fonction translate

`translate("Texte à Casse Mixte", "abcdefghijklmnopqrstuvwxyz", "ABCDEFGHIJKLMNOPQRSTUVWXYZ")`



TEXTE à CASSE MIXTE

not(boolean)

boolean(object)

Type	Résultat	Valeur
Chaîne	Vide	Faux
	Non vide	Vrai
Ensemble de nœuds	Vide	Faux
	Non vide	Vrai
Numérique	0 ou NaN	Faux
	≠0	Vrai

true()

false()

lang(string)

Retourne vrai si la langue passée en argument correspond à la langue du nœud courant.

La langue du nœud courant est déterminée par l'attribut `xml:lang` de ce nœud ou de ses ancêtres.

La casse est ignorée.

Si l'argument ne contient pas de suffixe (exemple : `fr`), la comparaison se limite à la langue non suffixée (exemple : `fr` pour `fr-ca`).

ceiling(number) Retourne l'arrondi supérieur du nombre passé en argument

round(number) Retourne le plus proche arrondi du nombre passé en argument

floor(number) Retourne l'arrondi inférieur du nombre passé en argument

number(object?)

Type	Contenu	Conversion
Chaîne	Expression numérique Autre	selon IEEE 754 NaN
Booléen	Vrai Faux	1 0
Ensemble de nœuds	Converti en chaîne, puis comme ci-dessus	

sum(node-set) Retourne la somme des nœuds de l'ensemble de nœud passé en argument, après leur conversion numérique

Opérations numériques

+

-

*

div

mod

Pour -, div et mod, il faut
laisser un espace de part et
d'autres de l'opérateur

Le sens de parcourt d'un axe est significatif pour certaines opérations.

Tous les axes "remontants" ou "arrières" commencent au plus près du nœud contextuel.

Ainsi, les opérations numériques sur les nœuds sont sensibles au sens de parcours.

Sur un même ensemble de nœud, le premier nœud dans un sens de lecture "arrière" ou "remontant" est le dernier nœud dans un sens de lecture "avant" ou "descendant"


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<data>
  <a>
    <item>1</item>
    <item>2</item>
    <item>3</item>
    <item>4</item>
    <item>5</item>
  </a>
  <b>
    <item>2</item>
    <item>3</item>
    <item>4</item>
    <item>6</item>
    <item>7</item>
  </b>
</data>
```

Tests sur les nodes

- 2 ensembles de nœuds (nodeset) sont équivalents si l'un de leur nœud est égal
- 2 ensembles de nœuds sont différents si l'un de leur nœud est différent



```
//a/item[2] = //b/item
//a/item[2] != //b/item
```

Sont 2 expressions vraies !!!

Les tests d'égalité et d'inégalité sur les nœuds doivent être compris comme des tests sur les ensembles

Un test d'égalité avec NaN retourne **faux**

NaN=NaN

est **faux** !

Un test d'inégalité avec NaN retourne **vrai**

NaN!=NaN

est **vrai** !

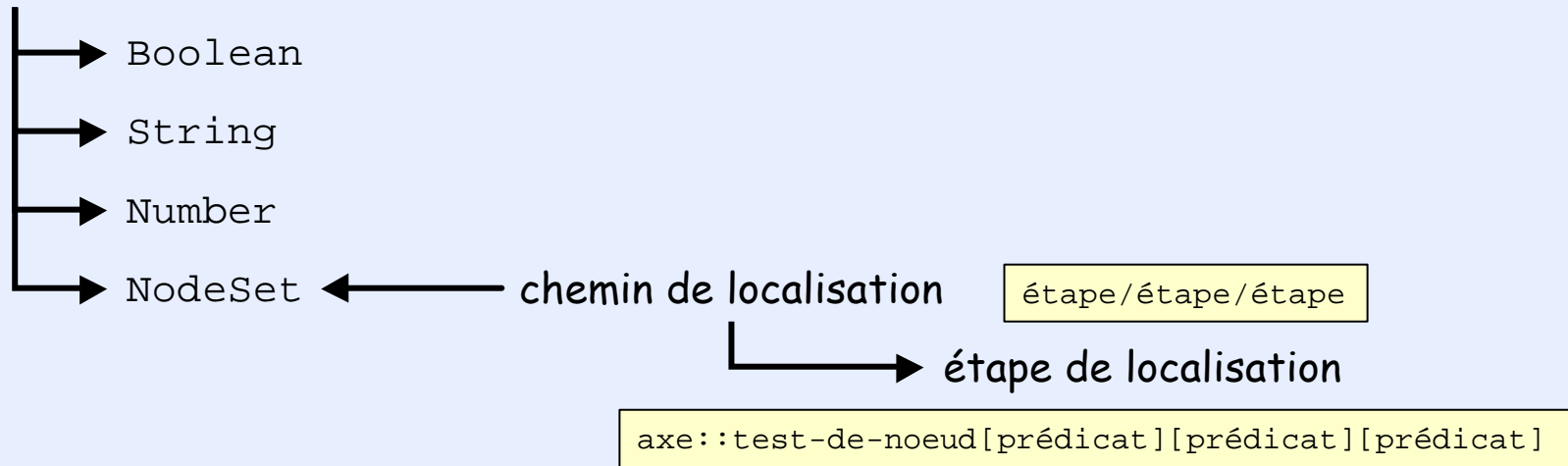
x est-il numérique ?

Si

number(x) != number(x)

{ est **vrai**, alors x n'est pas numérique
est **faux**, alors x est numérique

expression XPath



Contexte d'évaluation d'une expression XPath

