

- (10) 1. Expliquez ce qu'on entend par l'*amélioration progressive* pour le développement web. Expliquez son but et les moyens d'y arriver.

- (10) 2. Soit la fonction Javascript suivante avec un nom délibérément vague ici:

```
function lambda(){  
    var res=[];  
    for (var i=0;i<10;i++)  
        res[i]=function(n){return function(x){return n*x}}(i)  
    return res;  
}
```

Expliquez ce que retourne un appel à `lambda`. Donnez un exemple d'appel qui produira un entier comme résultat.

- (10) 3. Expliquez ce que sont les notions de bonne formation et de validation en XML. Quel est le but de chacune de ces opérations et comment on les met en application.
- (10) 4. Dans le langage courant, *internet* réfère le plus souvent au *web*. Or ces deux notions sont différentes, expliquez les nuances entre ces deux termes.

- (30) 5. Vous devez construire un système simplifié de facturation pour qu'un magasin sache qui a acheté quoi. Le magasin conserve dans un fichier XML les informations sur les clients et les articles (identifiés par un attribut no unique dans tout le fichier). Les commandes font ensuite le lien entre un client et l'article qu'il a commandé ainsi que la quantité et un possible commentaire sur la commande. Voici un exemple d'un tel fichier.

```
<magasin><!-- clients et objets de Pierre Dac -->
  <clients>
    <client no="C12"><nom>Olga Laxie</nom></client>
    <client no="C35"><nom>Odile Déserte</nom></client>
    <client no="C36"><nom>Séraphin Dufin</nom></client>
  </clients>
  <articles>
    <article no="A1">
      <description>Passoire non percée pouvant servir de casserole</description>
      <prix>14.95</prix>
    </article>
    <article no="A9">
      <description>Trou pour planter un arbre</description>
      <prix>9.95</prix>
    </article>
    <article no="A10">
      <description>Mannequin géant pour haute couture</description>
      <prix>19.99</prix>
    </article>
  </articles>
  <commandes>
    <commande client="C12" article="A1">
      <quantite>2</quantite>
      <commentaire>Livraison immédiate</commentaire>
    </commande>
    <commande client="C35" article="A10">
      <quantite>5</quantite>
      <commentaire>Livrer au 8e étage</commentaire>
    </commande>
    <commande client="C12" article="A9">
      <quantite>2</quantite>
    </commande>
    <commande client="C12" article="A10">
      <quantite>2</quantite>
    </commande>
  </commandes>
</magasin>
```

- (10) (a) Donnez un schéma Relax NG forme compacte pour valider un tel fichier.
- (20) (b) Donnez les templates XSLT permettant de transformer le fichier XML ci-dessus dans le HTML correspondant à la figure suivante qui regroupe toutes les commandes des clients qui ont commandé quelque chose. Il faut également calculer le nombre total d'items commandés par le client et le prix total de sa commande.

Listes des commandes de chaque client

Olga Laxie

Article	Qté	Prix	Commentaire
Passoire non percée pouvant servir de casserole	2	14.95	Livraison immédiate
Trou pour planter un arbre	2	9.95	
Mannequin géant pour haute couture	2	19.99	
Total	6	89.78	

Odile Déserte

Article	Qté	Prix	Commentaire
Mannequin géant pour haute couture	5	19.99	Livrer au 8e étage
Total	5	99.95	

- (30) 6. Il vous faut créer une page web permettant de remplir une case d'un Sudoku 4x4. Il faut entrer dans les cases du tableau un nombre entre 1 et 4 de telle sorte que les colonnes et les lignes ne contiennent que des chiffres différents. Cette même règle s'applique pour les quatre cases de chaque coin. Cette grille est définie avec une table HTML statiquement dans la page web. Certaines valeurs déjà présentes lors de la création de la grille ne peuvent être changées au cours du jeu.

La figure suivante donne un exemple avec deux nombres déjà présents dans la grille.

Construction d'un sudoku 4 x 4

Cliquez pour sélectionner une cellule et ensuite taper un nombre entre 1 et 4 pour l'entrer dans cette cellule

	3		
		2	

Une fois la grille affichée, l'utilisateur peut cliquer sur une des cases modifiables. Cette case est courante et on peut y taper un nombre entre 1 et 4 ou une espace (utilisez la fonction `keypress` sur la fenêtre et `String.fromCharCode(event.which)` pour récupérer la chaîne correspondant à la clé tapée au clavier). Pour remplacer un nombre, il suffit de taper un nouveau nombre dans la cellule courante ou une espace pour l'effacer. Si l'utilisateur viole une contrainte en tapant un nombre qui entre en conflit avec des nombres existants, les cases des nombres en conflit sont indiquées en rouge (ici elles apparaissent en gris foncé). Par exemple, la dernière case de la deuxième rangée dans la figure suivante:

Construction d'un sudoku 4 x 4

Cliquez pour sélectionner une cellule et ensuite taper un nombre entre 1 et 4 pour l'entrer dans cette cellule

	3		
			1
		2	
4	2	3	1

Lorsque l'utilisateur pense avoir complété sa grille, il appuie sur le bouton `terminé` et le système indique combien de cellules sont en conflit et les indique dans la grille, comme dans cet exemple.

Construction d'un sudoku 4 x 4

Cliquez pour sélectionner une cellule et ensuite taper un nombre entre 1 et 4 pour l'entrer dans cette cellule

4	<i>3</i>	1	2
1	2	3	4
3	4	<i>2</i>	1
3	1	4	3

3 cases en conflit

- (10) (a) Écrivez le code HTML utilisant le fichier CSS suivant, nommé `sudoku4x4.css`, pour créer la page web de la première figure.

```

table {
  border-collapse: collapse;
  margin: 10px;
}

tr:nth-child(odd) { border-top: 2px solid #999; }
tr:nth-child(even) { border-bottom: 2px solid #999; }

td {
  border: 1px solid #999;
  font-size: 15pt;
  font-weight: bold;
  height: 25px;
  text-align: center;
  width: 25px;
}

td:nth-child(odd) { border-left: 2px solid #999; }
td:nth-child(even) { border-right: 2px solid #999; }
td.readonly {
  color: blue;
  font-style: italic;
}

td.current { background-color: lightGray; }
td.error { background-color: red; }

```

- (20) (b) Écrivez le code Javascript, en supposant que *jQuery* est déjà chargé, qui permet de gérer l'interaction avec l'utilisateur. Expliquez comment lier ce code, qui doit être dans un fichier distinct de la page web, à la page HTML précédente.

Vous pouvez supposer l'existence de la fonction `verifierConflits(td)` qui pour un élément DOM `td` retourne un `Array` contenant les autres éléments `td` de la table qui ont la même valeur que celle dans `td` et qui sont dans la même ligne, colonne

ou coin. Un tableau vide est retourné lorsqu'il n'y a pas de cases en conflit. **Vous n'avez pas à écrire le code de cette fonction.**

Les fonctions *jQuery* sont rappelées en annexe.

Cet examen comporte 6 questions pour un total de 100 points.

Bonne chance

B.4. RELAX NG

Table B.4. Table 3.3, Section 3.3

Compact Syntax (RNC)	XML Syntax (RNG)
<pre>{ default? namespace id=URI datatypes id=URI }* { start=pattern id=pattern }*</pre>	<pre><grammar> { <start> pattern </start> <define name="NCName">pattern+</define> }* </grammar></pre>
Patterns	
element <i>QName</i> «{ » <i>pattern</i> «}»	<element name=" <i>QName</i> "> <i>pattern</i> +</element>
attribute <i>QName</i> «{ » <i>pattern</i> «}»	<attribute name=" <i>QName</i> "> <i>pattern</i> +</attribute>
<i>pattern</i> { «, » <i>pattern</i> }+	<group name=" <i>QName</i> "> <i>pattern</i> +</group>
<i>pattern</i> { «& » <i>pattern</i> }+	<interleave name=" <i>QName</i> "> <i>pattern</i> +</interleave>
<i>pattern</i> { « » <i>pattern</i> }+	<choice name=" <i>QName</i> "> <i>pattern</i> +</choice>
<i>pattern</i> «?»	<optional name=" <i>QName</i> "> <i>pattern</i> +</optional>
<i>pattern</i> «*»	<zeroOrMore name=" <i>QName</i> "> <i>pattern</i> +</zeroOrMore>
<i>pattern</i> «+»	<oneOrMore name=" <i>QName</i> "> <i>pattern</i> +</oneOrMore>
mixed «{ » <i>pattern</i> «}»	<mixed name=" <i>QName</i> "> <i>pattern</i> +</mixed>
<i>id</i>	<ref name=" <i>NCName</i> " />
empty	<empty />
text	<text />
data <i>TypeValue</i>	<value { name=" <i>NCName</i> " }?> <i>string</i> +</value>
data <i>TypeValue</i> «{ » { <i>id=value</i> }* «}»	<data { type=" <i>NCName</i> " }?> { <param name=" <i>NCName</i> "> <i>string</i> </param> }* </data>

B.6. XSLT

Table B.6. Table 5.1, Section 5.1

<pre><xsl:stylesheet> xsl:import*, (declaration/xsl:variable/xsl:param)* </xsl:stylesheet></pre>
<pre><xsl:template match="pattern" name="QName"> xsl:param*, sequence-constructor* </xsl:template></pre>
<pre><xsl:param name="QName" select="expression"> sequence-constructor </xsl:param></pre>
<pre><xsl:apply-templates select="expression"> (xsl:sort*/xsl:with-param)* </xsl:apply-templates></pre>
<pre><xsl:call-template name="Qname"/></pre>
<pre><xsl:with-param name="QName" select="expression"> sequence-constructor </xsl:with-param></pre>
<pre><xsl:function name="QName"> xsl:param*, sequence-constructor* </xsl:function></pre>
<pre><xsl:value-of select="expression"> sequence-constructor </xsl:value-of></pre>
<pre><xsl:variable name="QName" select="expression"> sequence-constructor </xsl:variable></pre>
<pre><xsl:copy> sequence-constructor </xsl:copy></pre>
<pre><xsl:copy-of select="expression"/></pre>
<pre><xsl:if test="expression"> sequence-constructor </xsl:if></pre>
<pre><choose> xsl:when*, xsl:otherwise? </choose></pre>
<pre><xsl:when test="expression"> sequence-constructor </xsl:when></pre>
<pre><xsl:otherwise> sequence-constructor </xsl:otherwise></pre>
<pre><xsl:for-each select="expression"> xsl:sort*, sequence-constructor </xsl:for-each></pre>
<pre><xsl:for-each-group select="expression" group-by="expression"> xsl:sort*, sequence-constructor </xsl:for-each-group></pre>

<pre><xsl:sort select="expression" data-type="{string}"> sequence-constructor </xsl:sort></pre>
<pre><xsl:element name="{string}"> sequence-constructor </xsl:element></pre>
<pre><xsl:text> character data </xsl:text></pre>
<pre><xsl:attribute name="{string}" select="expression"> sequence-constructor </xsl:attribute></pre>
<pre><xsl:attribute-set name="QName" use-attribute-sets="Qnames"> xsl:attribute* </xsl:attribute-set></pre>
<pre><xsl:message> sequence-constructor </xsl:message></pre>



Selectors
 Basics
 #id
 element selector
 .class
 *
 selector1
 selector2
 :first
 :last
 :not(selector)
 :even
 :odd
 :eq(index)
 :gt(index)
 :lt(index)
 :header
 :animated
 :focus

Core
 jQuery function
 \$
 jQuery(selector [, context] | element | elementArray | jQuery Object) accepts
 fn beforeSend(jqXHR, config)
 fn complete(jqXHR, status)
 str contentType
 map converters
 obj, str data
 bool global = true
 bool ifModified = false
 fn jsonpCallback
 bool processData = true
 map statusCode
 bool traditional
 str url = curr. page
 fn xhr
 str dataType ∈ {xml, json, script, html}
 fn error(jXHR, status, errorThrown)
 fn success(data, status, jqXHR)
 jQuery.ajaxSetup(options)

AJAX
 Low-Level Interface
 jqXHR jQuery.ajax(options, [settings])
 bool async = true
 bool cache = true
 map contents
 obj context
 bool crossDomain
 fn dataFilter(data, type)
 map headers
 str jsonp
 str jsonpScriptCharset
 num timeout
 str type = "GET"
 str username
 Global Ajax Event Handlers
 jQuery.getJSON(url [, data] [, fn(data, status, jqXHR)])
 jQuery.getScript(url [, fn(data, status)])
 jQuery.post(url [, data] [, fn(data, status)] [, type])

Shorthand Methods
 \$
 load(url [, data] [, fn(responseText, textStatus, jqXHR)])
 jqXHR
 jQuery.get(url [, data] [, fn(data, status, jqXHR)])
 jqXHR
 jQuery.getJSON(url [, data] [, fn(data, status, jqXHR)])
 jqXHR
 jQuery.getScript(url [, fn(data, status)])
 jqXHR
 jQuery.post(url [, data] [, fn(data, status)] [, type])
 Global Ajax Event Handlers
 \$
 ajaxComplete(fn(event, XHR, options))
 \$
 ajaxError(fn(event, XHR, options, errorThrown))
 \$
 ajaxSend(fn(event, XHR, options))
 \$
 ajaxStart(fn ())
 \$
 ajaxStop(fn ())
 \$
 ajaxSuccess(fn(event, XHR, options))
 Miscellaneous
 str .serialize()
 [obj] .serializeArray()
 str .jQuery.param(obj, [traditional])

Basic Filters
 :contains(text)
 :empty
 :has(selector)
 :parent
 :eq(index)
 :gt(index)
 :lt(index)
 :header
 :animated
 :focus

jQuery Object Accessors
 jQuery.each(fn(index, element))
 num size(), length
 str str
 selector selector
 \$ eq(index)
 jQuery.error(str)
 [el] .get(index)
 num num
 index(), .index(selector | element)
 jQuery.pushStack(elements, [name, args])
 arr .toArray()

Interoperability
 jQuery.noConflict([extreme])

Events
 Page Load
 \$.ready(fn())
 Event Handling
 \$ on(events [, selector] [, data], handler) 1.7+
 \$ on(events-map [, selector] [, data]) 1.7+
 \$ off(events [, selector] [, handler]) 1.7+
 \$ off(events-map [, selector]) 1.7+
 \$.bind(type [, data], fn(eventObj))
 \$.bind(type [, data], false)
 \$.bind(array)
 \$.unbind([type] [, fn])
 \$.one(type [, data], fn(eventObj))
 \$.trigger(event [, data])
 obj .triggerHandler(event [, data])
 \$.delegate(selector, type, [data], handler)
 \$.undelegate([selector, type, [handler]] | selector, events | namespace

Live Events
 \$.live(eventType [, data], fn())
 \$.die(), .die(eventType | fn())

Child Filters
 :nth-child(expr)
 :first-child
 :last-child
 :only-child

Attributes
 Attribut
 str attr(name | name , value)
 \$ attr(name, val | map | name, fn(index, attr))
 \$ removeAttr(name)
 \$ prop(name)
 \$ removeProp(name)
 Class
 \$.addClass(class | fn(index, class))
 bool .hasClass(class)
 \$.removeClass([class] | fn(index, class))
 \$.toggleClass(class [, switch] | fn(index, class) [, switch])

Positioning
 obj .offset()
 \$.offset(coord | fn(index, coord))
 \$.offsetParent()
 obj .position()
 int .scrollTop()
 \$.scrollLeft()
 int .scrollLeft()
 \$.scrollLeft()

Interaction Helpers
 \$.hover(fn(eventObj), fnOut(eventObj))
 \$.toggle(fn(eventObj), fn2(eventObj) [, ...])

Event Helpers
 function ([data,] [fn])
 \$.blur,
 \$.change,
 \$.click,
 \$.dblclick,
 \$.error,
 \$.focus,
 \$.focusin,
 \$.focusout,
 \$.keydown,
 \$.keypress,
 \$.keyup,
 \$.load([data,] fn),
 \$.mousedown,
 \$.mouseenter,
 \$.mouseleave,
 \$.mousemove,
 \$.mouseout,
 \$.mouseover,
 \$.mouseup,
 \$.resize,
 \$.scroll,
 \$.select,
 \$.submit,
 \$.unload([data,] fn)

Forms
 Form Filters
 :enabled
 :disabled
 :password
 :checkbox
 :submit
 :image
 :reset
 :button
 :file

HTML, text
 str html()
 \$.html(val | fn(index, html))
 str text()
 \$.text(val | fn(index, html))

Height and Width
 int .height()
 \$.height(val | fn(index, height))
 int .width()
 \$.width(val | fn(index, height))
 int .innerHeight()
 int .innerWidth()
 int .outerHeight([includeMargin])
 \$.outerHeight(val | fn(index, outerHeight)) 1.8+
 int .outerWidth([includeMargin])
 \$.outerWidth(val | fn(index, outerWidth)) 1.8+

Effects
 \$.show([duration], [easing], [fn])
 \$.hide([duration], [easing], [fn])
 \$.toggle([showOrHide])
 \$.toggle(duration [, easing], [fn])

Sliding
 \$.slideDown(duration [, easing], [fn])
 \$.slideUp(duration [, easing], [fn])
 \$.slideToggle(duration [, easing], [fn])

Value
 str, arr .val()
 \$.val(val | fn())

Manipulation
 Inserting Inside
 \$.append(content | fn(index, html))
 \$.appendTo(target)
 \$.prepend(content | fn(index, html))
 \$.prependTo(target)
 Inserting Outside
 \$.after(content | fn())
 \$.before(content | fn())
 \$.insertAfter(target)
 \$.insertBefore(target)

Replacing
 \$.replaceWith(content | fn)
 \$.replaceAll(selector)

Fading
 \$.fadeIn(duration [, easing], [fn])
 \$.fadeOut(duration [, easing], [fn])
 \$.fadeTo([duration,] opacity [, easing], [fn])
 \$.fadeToggle([duration,] [, easing], [fn])

Custom
 \$.animate(params [, duration], [easing], [fn])
 \$.animate(params, options)
 \$.stop([queue] [, clearQueue] [, jumpToEnd]) 1.7+
 \$.delay(duration [, queueName])

Traversing
 Filtering
 \$.eq(index)
 \$.first()
 \$.last()
 \$.has(selector), .has(element)
 \$.filter(selector), .filter(fn(index))
 bool
 \$.is(selector | function(index) | jQuery object | element) 1.7+
 \$.map(fn(index, element))
 \$.not(selector), .not(elements), .not(fn(index))
 \$.slice(start [, end])

Tree traversal
 \$.children(selector)
 \$.closest(selector [, context] | jQuery object | element)
 arr .closest(selectors [, context]) removed
 \$.find(selector | jQuery object | element)
 \$.next(selector)
 \$.nextAll([selector])
 \$.nextUntil([selector])
 \$.offsetParent()
 \$.parent(selector)
 \$.parents(selector)
 \$.parentsUntil([selector])
 \$.prev(selector)
 \$.prevAll([selector])
 \$.prevUntil([selector])
 \$.siblings(selector)

Inserting Around
 \$.unwrap()
 \$.wrap(wrappingElement | fn)
 \$.wrapAll(wrappingElement | fn)
 \$.wrapInner(wrappingElement | fn)

Settings
 bool jQuery.fx.off
 num jQuery.fx.interval

Settings
 bool jQuery.fx.off
 num jQuery.fx.interval

Miscellaneous
 \$.add(selector [, context] | elements | html)
 \$.andSelf()
 \$.contents()
 \$.end()

Removing
 \$.detach(selector)
 \$.empty()
 \$.remove(selector)
 Copying
 \$.clone([withDataAndEvents], [deepWithDataAndEvents])

Replacing
 \$.replaceWith(content | fn)
 \$.replaceAll(selector)

Settings
 bool jQuery.fx.off
 num jQuery.fx.interval

Settings
 bool jQuery.fx.off
 num jQuery.fx.interval

Event object
 event = {
 el .currentTarget,
 data .isDefaultPrevented(),
 bool .isImmediatePropagationStopped(),
 bool .isPropagationStopped(),
 str namespace,
 num pageX,
 num pageY,
 bool preventDefault(),
 str relatedTarget,
 obj result,
 stopImmediatePropagation(),
 stopPropagation(),
 el target,
 num timeStamp,
 str type,
 str which
 }

Deferred
 deferred object = {
 def .always(alwaysCallbacks [, alwaysCallbacks])
 def .done(doneCallbacks)
 def .fail(failCallbacks)
 bool .isRejected() deprecated
 bool .isResolved() deprecated
 str state() 1.8+
 def .notify(args) 1.7+
 def .notifyWith(context, [args]) 1.7+
 def .pipe([doneFilter] [, failFilter] [, progressFilter])
 def .progress(progressCallbacks) 1.7+
 def .reject([args])
 def .rejectWith(context, [args])
 def .resolve([args])
 def .resolveWith(context, [args])
 str state()
 def .then(doneCallbacks, failCallbacks [, progressCallbacks]) 1.7+
 def .promise([target])
 }

Callbacks
 callbacks object = { 1.7+
 and .add(callbacks)
 and .disable() deprecated
 and .empty() deprecated
 and .fire(arguments)
 bool .fired() deprecated
 and .fireWith([context] [, args])
 bool .has(callback)
 and .lock() deprecated
 bool .locked() deprecated
 and .remove(callbacks)
 and .removeWith(flags)
 and .CallCallbacks(flags)

Utilities
 Browser and Feature Detection
 obj .jQuery.support
 obj .jQuery.browser deprecated
 str .jQuery.browser.version deprecated
 bool .jQuery.boxModel deprecated
 Basic operations
 obj .jQuery.each(obj, fn(i, valueOrElement))
 obj .jQuery.extend([deep] target, obj1 [, objN])
 arr .jQuery.grep(arr, fn(el, i) [, invert])
 arr .jQuery.makeArray(obj)
 arr .jQuery.map(arrayOrObject, fn(el, i))
 num .jQuery.inArray(val, arr)
 arr .jQuery.merge(first, second)
 fn .jQuery.noop
 fn .jQuery.proxy(fn, scope | scope, name)
 arr .jQuery.unique(arr)
 str .jQuery.trim(str)
 obj .jQuery.parseJSON(str)

Data functions
 \$.clearQueue([name])
 \$.dequeue([name])
 \$.jQuery.dequeue([name])
 obj .jQuery.data(el, key)
 obj .data(el, key)
 \$.data(key, val | obj)
 \$.removeData([name]) [list] 1.7+
 [fn] .queue([name])
 \$.jQuery.queue([name])
 \$.queue([name,] fn(next))
 \$.jQuery.queue([name,] fn())
 \$.queue([name,] queue)
 \$.jQuery.queue([name,] queue)
 Test operations
 str .jQuery.type(obj)
 bool .jQuery.isArray(obj)
 bool .jQuery.isEmptyObject(obj)
 bool .jQuery.isFunction(obj)
 bool .jQuery.isPlainObject(obj)
 bool .jQuery.isWindow(obj)
 bool .jQuery.isNumeric(val) 1.7+