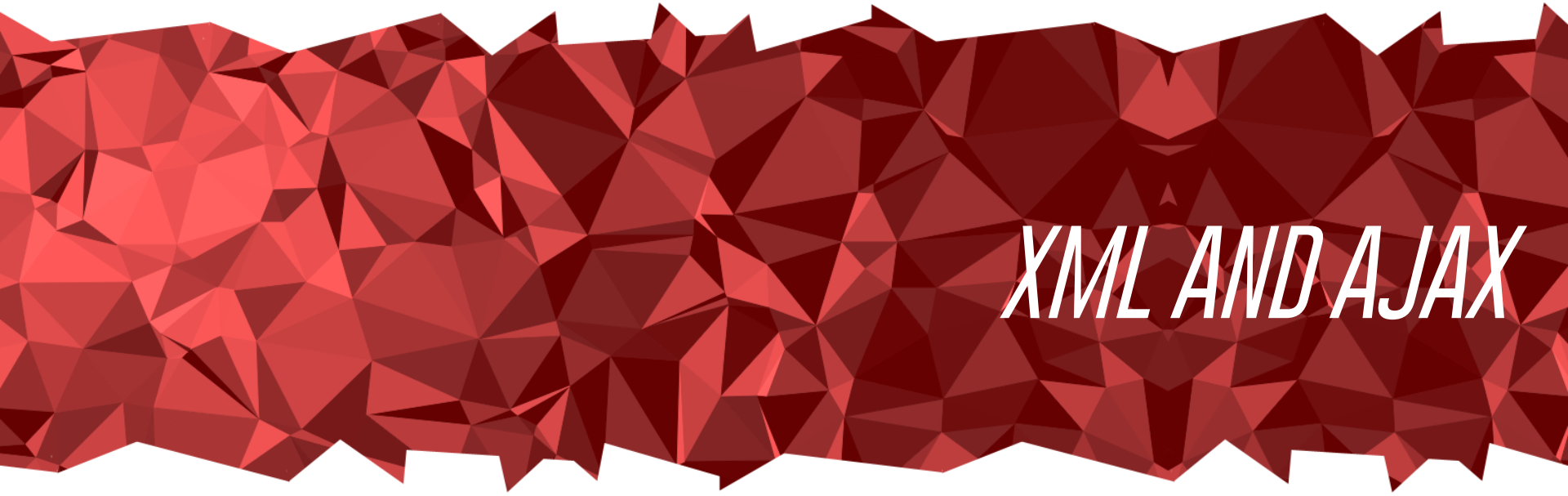


# XML AND AJAX

CMSC 126
WEB ENGINEERING
SECOND SEMESTER 2015 – 2016

EARL FRANCIS VALDEHUESA

# EXTENSIBLE MARKUP LANGUAGE

## EXTENSIBLE MARKUP LANGUAGE (XML)

- XML is a text-based markup language used to store and organize data, rather than specifying how to display it – a "skeleton" for markup languages.

### BAD WAY TO STORE DATA

```
My note:
BEGIN
  TO: Tove
  FROM: Jani
  SUBJECT: Reminder
  MESSAGE (english):
    Hey there,
    Don't forget to call me this weekend!
END
```
XML

# EXTENSIBLE MARKUP LANGUAGE

## SYNTAX

```XML
<element  attribute="value">content</element>
```

## LANGUAGES WRITTEN IN XML SPECIFY:

- NAMES OF TAGS – IN XHTML: H1, DIV, IMG, ETC.
- NAMES OF ATTRIBUTES – IN XHTML: ID/CLASS, SRC, HREF, ETC.
- RULES ABOUT HOW THEY GO TOGETHER

## SELF-DESCRIBING DATA

- USED TO PRESENT COMPLEX DATA IN HUMAN-READABLE FORM

# EXTENSIBLE MARKUP LANGUAGE

## ANATOMY OF AN XML FILE

```xml
<?xml version="1.0" encoding="UTF-8"?>   <!-- XML prolog -->
<note>                                   <!-- root element -->
  <to>Tove</to>
  <from>Jani</from>                      <!-- element ("tag") -->
  <subject>Reminder</subject>            <!-- content of element -->
  <message language="english">           <!-- attribute and its value -->
    Don't forget me this weekend!
  </message>
</note>
```

XML

- BEGINS WITH AN <?XML … ?> HEADER TAG
- HAS A SINGLE ROOT ELEMENT
- TAG, ATTRIBUTE, AND COMMENT SYNTAX IS JUST LIKE XHTML

# USES OF XML

XML DATA COMES FROM MANY SOURCES ON THE WEB

- WEB SERVERS STORE DATA AS XML FILES
- DATABASES SOMETIMES RETURN QUERY RESULTS AS XML
- WEB SERVICES USE XML TO COMMUNICATE

XML IS THE DE FACTO STANDARD FOR EXCHANGE OF DATA

XML LANGUAGES ARE USED FOR MUSIC, MATH, VECTOR, ETC.

POPULAR USE: RSS FOR NEWS FEEDS AND PODCASTS

# PROS AND CONS OF XML

## PROS

- EASY TO READ (FOR HUMANS AND COMPUTERS)
- STANDARD FORMAT MAKES AUTOMATION EASY
- DON'T HAVE TO "REINVENT THE WHEEL" FOR STORING NEW TYPES OF DATA
- INTERNATIONAL, PLATFORM-INDEPENDENT, OPEN/FREE STANDARD
- CAN REPRESENT ANY GENERAL KIND OF DATA

## CONS

- BULKY SYNTAX/STRUCTURE MAKES FILES LARGE; CAN DECREASE PERFORMANCE
- CAN BE HARD TO "SHOEHORN" DATA INTO A GOOD XML FORMAT

# SYNTAX

## EXAMPLE XML DOCUMENT

```
<?xml version="1.0" encoding="UTF-8"?>
<lecture>
   <class>CMSC 126</class>
   <instructor>Frae Valdehuesa</instructor>
</lecture>
```

## KINDS OF INFORMATION

**MARKUP** Like <lecture>, <class>, and <instructor>.

**CHARACTER DATA** Like *CMSC 126*, and *Frae Valdehuesa*.

# SYNTAX

## SYNTAX RULES

### XML DECLARATION

- The XML document can optionally have an XML declaration – it is case sensitive and must begin with <?xml … ?>.

### TAGS AND ELEMENTS

- An XML file is structured by several XML elements (a.k.a. tags, or nodes). XML elements' names are enclosed by triangular brackets < >.

**ELEMENT SYNTAX** Each XML element needs to be closed either with start or with end elements, or just a singleton.

**NESTED ELEMENTS** An XML element can contain multiple children, but the children elements must not overlap.

**ROOT ELEMENT** An XML document can only have one root element.

# SYNTAX RULES

## XML TEXT

- All XML files should be saved as Unicode UTF-8 or UTF-16 files to avoid character encoding problems – some characters are reserved by the XML syntax itself.

## COMMENTS

- The syntax for writing comments in XML is similar to that of HTML - <!-- *comment* -->.

## WHITESPACE

- XML does not truncate multiple whitespaces in a document – contrary to to HTML, where it truncates them into one single whitespace.

# XML DOCUMENTS

## XML DOCUMENT

- A basic unit of XML information composed of elements and other markup in an orderly package – contains wide variety of data.

## PARTS OF AN XML DOCUMENT

### DOCUMENT PROLOG SECTION

- The document prolog comes at the top of the document, before the root element – XML declaration and document type declaration.

### DOCUMENT ELEMENTS SECTION

- Document elements are the building blocks of XML – divide the document into a hierarchy of sections, each serving specific purpose.

# XML DECLARATION

## XML DECLARATION

- XML declaration contains details that prepare an XML processor to parse the XML document.

## XML DECLARATION SYNTAX

```
<?xml version="version_number" encoding="declaration"
        standalone="standalone_status" ?>
```

**VERSION** Specifies the version of the XML standard usd.

**ENCODING** Defines the character encoding used.

**STANDALONE** Informs the parser whether the document relies on the information from an external source for its content.

# NAMESPACES

## NAME CONFLICTS

### SOLVING THE NAME CONFLICT USING A PREFIX

- Name conflicts in XML can easily be avoided using a name prefix.

## XML NAMESPACES

### XMLNS ATTRIBUTE

- When using prefixes in XML, a so-called namespace for the prefix must be defined – the namespace is defined by the xmlns attribute in the start tag (xmlns:*prefix*="URI").

### DEFAULT NAMESPACES

- Defining a default namespace for an element saves us from using prefixes in all the child elements (Syntax: xmlns="namespaceURI").

# XML DOCUMENT TYPES

## VALID XML DOCUMENTS

- A valid XML document is not the same as a well-formed XML document – it must conform to a document type definition.

## DOCUMENT TYPE DEFINITIONS

### DOCUMENT TYPE DEFINITION

- The original Document Type Definition (DTD) – used to verify the data you receive from the outside world is valid.

### XML SCHEMA

- An XML-based alternative to DTD – written in XML and more powerful than DTD because it is extensible to additions (support data types and namespaces).

# XML DOCUMENT TYPES

## DOCUMENT TYPE DEFINITION (DTD)

### INTERNAL DTD DECLARATION

- If the DTD is declared inside the XML file, it must be wrapped inside the <!DOCTYPE> definition.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE root [ ... ]>
```

### EXTERNAL DTD DECLARATION

- If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE root SYSTEM "file.dtd">
```

# XML DOCUMENT TYPES

## BUILDING BLOCKS OF XML DOCUMENTS

### ELEMENTS

- Elements are the main building blocks of both XML and HTML documents.

### ATTRIBUTES

- Attributes provide extra information about elements – always placed inside the opening tag of an element and always come in name/value pairs.

### ENTITIES

- Some characters have a special meaning in XML, like (<) that defines the start of an XML tag. Entities are expanded when a document is parsed by an XML parser.

### PCDATA & CDATA

- PCDATA is text that will be parsed by a parser, while CDATA will not be parsed.

# XML DOCUMENT TYPES

## DECLARING ELEMENTS

```
<!ELEMENT element-name category>
<!ELEMENT element-name (element-content)>
```

**EMPTY ELEMENTS** Empty elements are declared with the category keyword EMPTY.

**PARSED CDATA** Elements with only parsed character data are declared with #PCDATA inside parentheses.

**ANY CONTENTS** Elements declared with the category keyword ANY can contain any combination of parsable data.

**CHILDREN** Elements with one or more children are declared with the name of the children elements inside parentheses.

**OCCURENCES** Elements can be specified how many times they can occur using quantifiers (+, *, ?).

# XML DOCUMENT TYPES

## DECLARING ATTRIBUTES

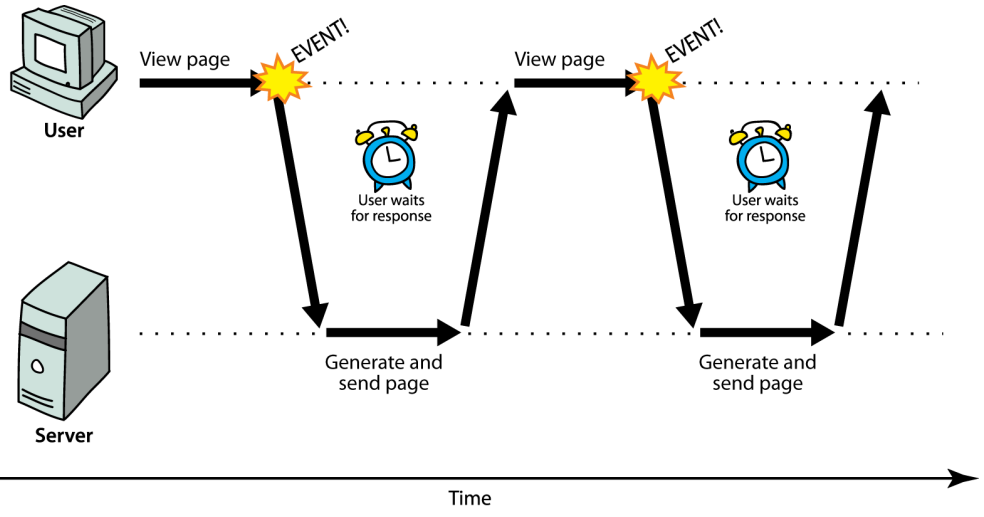
```
<!ATTLIST element-name attribute-name
   attribute-type attribute-value>
```

| TYPE | DESCRIPTION |
|---|---|
| CDATA | The value is character data. |
| (*en1*\|*en2*\|..) | The value must be one from an enumerated list. |
| ID | The value is a unique id. |
| IDREF (S) | The value is the id of another element (or a list of other ids). |
| NMTOKEN (S) | The value is a valid XML name (or a list of valid XML names). |
| ENTITY (IES) | The value is an entity (or a list of entities. |

**VALUE** The attribute-value can be *value*, #REQUIRED, #IMPLIED, or #FIXED *value*.

# WEB COMMUNICATION

## SYNCHRONOUS WEB COMMUNICATION

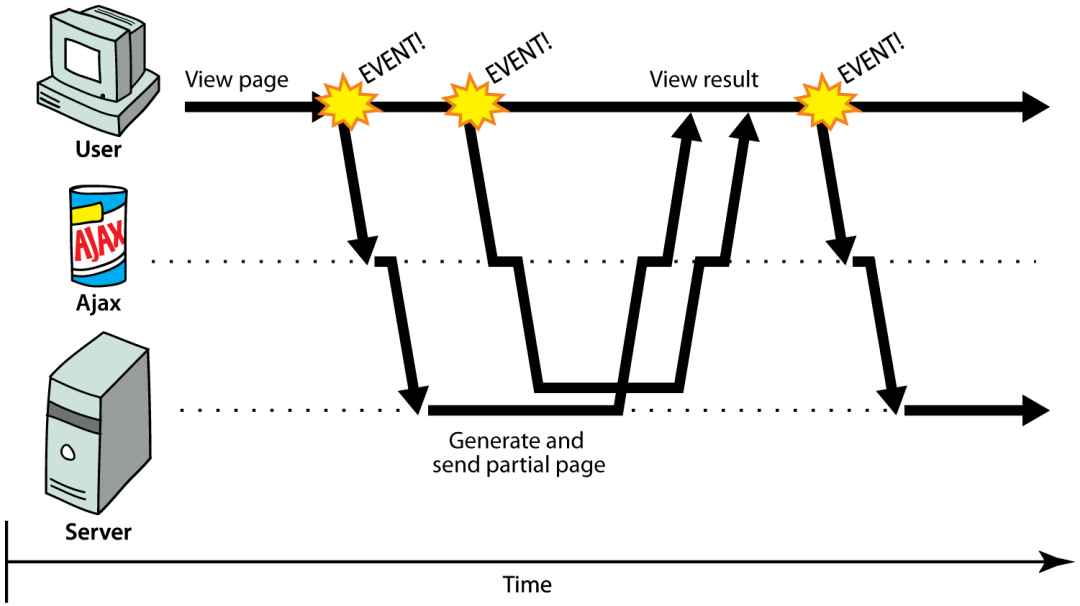



### SYNCHRONOUS

- User must wait while new pages load – the typical communication pattern used in web pages (click, wait, refresh).

# WEB COMMUNICATION

## ASYNCHRONOUS WEB COMMUNICATION




## ASYNCHRONOUS

- User can keep interacting with page while data loads – communication pattern made possible by Ajax.

# WEB APPLICATIONS AND AJAX

## WEB APPLICATION

- A dynamic web site that mimics the feel of a desktop application – presents a continuous user experience rather than disjoin pages.
  - GMAIL
  - GOOGLE MAPS
  - GOOGLE DOCS
  - FLICKR
  - A9

## AJAX

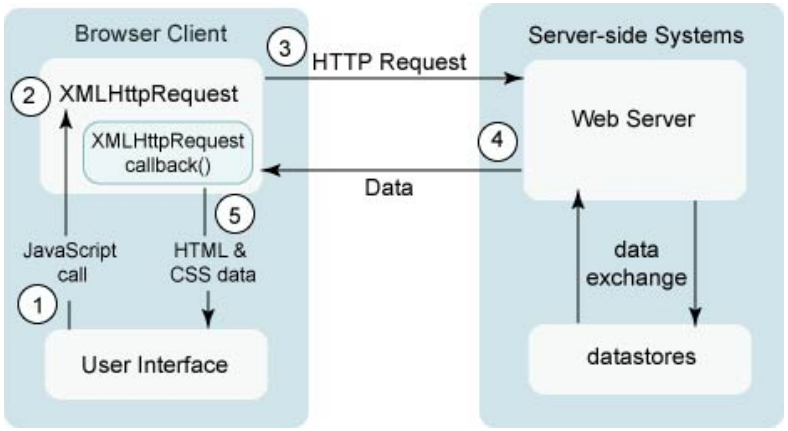

- Asynchronous JavaScript and XML (AJAX) is not a programming language, but a particular way of using JavaScript.
  - DOWNLOADS DATA FROM A SERVER IN THE BACKGROUND
  - ALLOWS DYNAMICALLAY UPDATING A PAGE WITHOUT MAKING THE USER WAIT
  - AVOIDS THE "CLICK-WAIT-REFRESH" PATTERN

# XMLHTTPREQUEST

## XMLHTTPREQUEST

- JavaScript includes an XMLHttpRequest object that can fetch files from a Web server – it can do this asynchronously (in the background, transparent to user).

## A TYPICAL AJAX REQUEST




- CONTENTS OF THE FETCHED FILE CAN BE PUT INTO WEB PAGE USING THE DOM

# REQUEST

## SEND A REQUEST TO A SERVER

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

**OPEN** Specifies the type of request, accepts three parameters: *method* (GET or POST), *url*, and *async*.

**SEND** Sends the request to the server – send() for GET and send(*string*) for POST.

```
xhttp.onreadystatechange = function() {
  if (xhttp.readyState == 4 && xhttp.status == 200) {
    document.getElementById("demo").innerHTML = xhttp.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

# RESPONSE

## SERVER RESPONSE

**RESPONSETEXT** The responseText property returns the response as a string.

**RESPONSEXML** The responseXML property is used if the response from the server is XML, and you want to parse it as an XML object.

## ONREADYSTATECHANGE EVENT

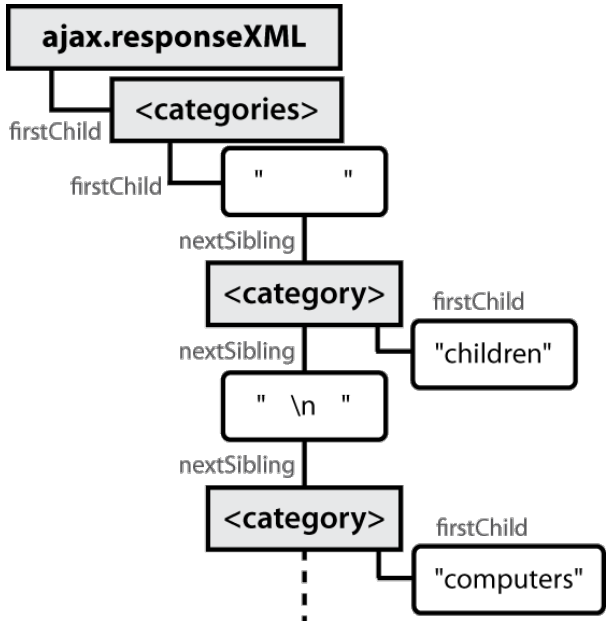
**ONREADYSTATECHANGE** Stores a function (or the name of a function) to be called automatically each time the readyState property changes.

**READYSTATE** Holds the status of the XMLHttpRequest. Changes from 0-4.

**CALLBACK FUNCTION** A function passed as a parameter to another function.

# XML DOM TREE STRUCTURE

```xml
<?xml version="1.0" encoding="UTF-8"?>
<categories>
  <category>children</category>
  <category>computers</category>
  ...
</categories>
```




# XML DOM TREE STRUCTURE

## XML DOM

### PROPERTIES

- firstChild, lastChild, childNodes, nextSibling, previousSibling, parentNode
- nodeName, nodeType, nodeValue, attributes

### METHODS

- appendChild, insertBefore, removeChild, replaceChild
- getElementsByTagName, getAttribute, hasAttributes, hasChildNodes

### CAUTION: CANNOT USE HTML-SPECIFIC PROPERTIES

# XML DOM TREE STRUCTURE

## NAVIGATING THE NODE TREE

- CAN ONLY USE STANDARD DOM METHODS/PROPERTIES IN XML DOM
- CAN'T USE ID OR CLASS TO GET SPECIFIC NODES

```
// returns all child tags inside node that use the given element
var elms = node.getElementsByTagName("tagName");
```
JS

- CAN'T USE INNERHTML TO GET THE TEXT INSIDE A NODE

```
var text = node.firstChild.nodeValue;
```
JS

- CAN'T USE ATTRIBUTENAME TO GET AN ATTRIBUTE'S VALUE

```
var attrValue = node.getAttribute("attrName");
```
JS

# XML DOM TREE STRUCTURE

## ANALYZING A FETCHED XML FILE USING DOM

```xml
<?xml version="1.0" encoding="UTF-8"?>
<employees>
  <lawyer money="5"/>
  <janitor name="Sue"><vacuumcleaner/></janitor>
  <janitor name="Bill">too poor</janitor>
</employees>
```

XML

## WE CAN USE DOM ON AJAX.RESPONSEXML

```js
// zeroth element of array of length 1
var employeesTag = ajax.responseXML.getElementsByTagName("employees")[0];

// how much money does the lawyer make?
var lawyerTag = employeesTag.getElementsByTagName("lawyer")[0];
var salary = lawyerTag.getAttribute("money");   // "5"

// array of 2 janitors
var janitorTags = employeesTag.getElementsByTagName("janitor");
var excuse = janitorTags[1].firstChild.nodeValue;  // " too poor "
```

JS