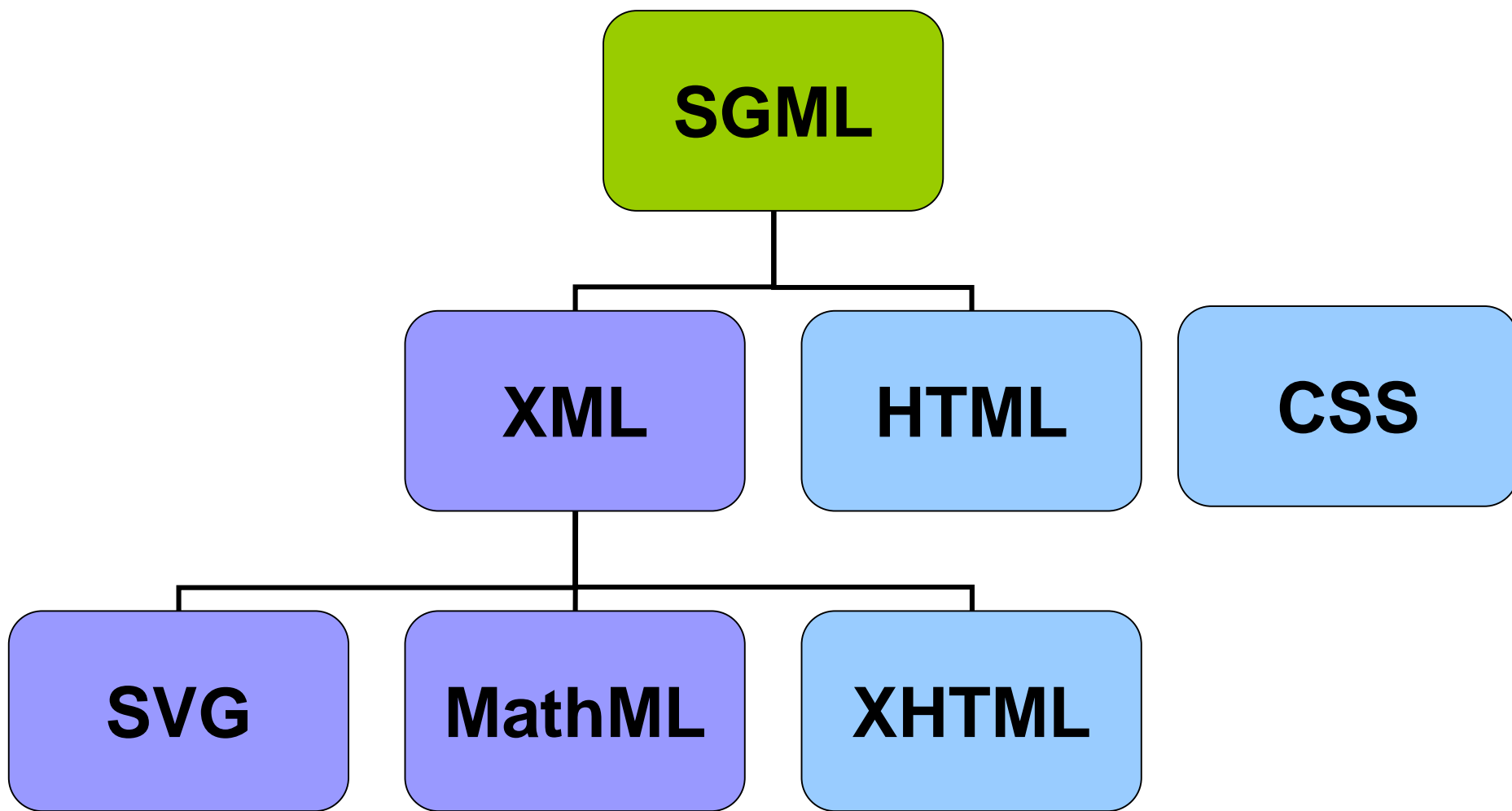




Декларативные языки

Декларативные языки





Язык XML

XML (eXtensible Markup Language)

XML был разработан *World Wide Web Consortium* (W3C) в 1996 году для представления и передачи **данных** в Интернет.



Цели создания

- удобство чтения человеком
- проста в написании программ-обработчиков

ОСНОВНЫЕ части XML

- *пролог*
- *инструкция*
- *тэг*
- *элемент*
- *атрибут*
- комментарий

Пролог

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

<example>
    . . .
</example>
```



Тэги

<tag-name>

</tag-name>

<c:include>

<fmt:numberFormat>

Элементы

- открывающий тэг
- содержимого элемента
- закрывающий тэг

<login> admin </login>

<blocked>*/blocked>

Вложенность тегов

<products>

<sour>Уксус</sour>

<sweet>Мед</sweet>

</products>

Вложенность тегов

<products>

<sour>

Уксус

<sweet>

Вишневое варенье

</sour>

Мед

</sweet>

</products>

Вложенность тегов

<products>

<sour>

Уксус

<sweet>Вишневое варенье</sweet>

</sour>

<sweet>Мед</sweet>

</products>

Атрибуты

```
<user id="123">  
  <login>root</login>  
  <password hash="md5">  
    A3B8270CE12D98765AE23EFF45912BC  
  </password>  
  <role blocked="blocked">admin</role>  
  <role>user</role>  
  <last-session date="12.01.11"  
    login="13:01" logout="15:38"/>  
</user>
```

Комментарии

<txt>Это обыкновенный **<!-- это**
комментарий **-->текст</txt>**

<txt lang="ru<!-- неверный
комментарий**-->ssian"> текст </txt>**

<txt lang="ru" <!-- неверный
комментарий**--> >А и Б</txt>**

CDATA

<txt><![CDATA[

Это обыкновенный <!-- А это

Уже не комментарий -->текст

]]></txt>

Комментарии

```
<txt lang="ru<![CDATA[<!-- неверный  
Текст-->]]>ssian"> текст </txt>
```

Комментарии

```
<txt lang="ru<lt;!-- неверный  
Текст-->ssian"> текст </txt>
```


Спецсимволы XML

Символ	Обозначение
<	&lt;
>	&gt;
&	&amp;
'	&apos;
"	&quot;



Недостатки XML

- избыточность текстового представления информации
- более сложная обработка, по сравнению с двоичными файлами
- отсутствие встроенной поддержки типов данных
- трудность представления не иерархических типов данных

Области применения

- Форматы файлов

- ☐ XHTML

- ☐ MathML

- ☐ SVG

- ☐ docx



Области применения

■ Протоколы

- ☐ XMPP

- ☐ RSS

- ☐ SOAP

- ☐ AJAX



Проверка правильности

- **well-formed**

- **valid**

- **DTD – Document Type Definition**

- **XSD – XML Schema Definition**

Well-formed XML документ

- Только один корневой элемент

`<books>`

...

`</books>`

- Каждому открывающемуся тегу соответствует закрывающийся

`<item>...</item>`

`<item />`

Well-formed XML документ

- Должна быть соблюдена последовательность закрытия тегов

```
<p>There is <b>simple <i>HTML</b>
tags</i> example</p>
```

There is **simple** *HTML tags* example

Well-formed XML документ

- Атрибутам всегда присваивается некоторое значение, которое обязательно заключается в кавычки

`<option selected>`

`<option selected="selected">`



Valid XML документ

- DTD (Document Type Definition)
- XSD (XML Schema Definition)

Расположение DTD

- Внутри XML

```
<!DOCTYPE root [ . . . ]>
```

- Локально в отдельном файле

```
<!DOCTYPE root SYSTEM "File.dtd">
```

- Во внешнем Интернет-ресурсе

```
<!DOCTYPE root  
PUBLIC "идентификатор" "адрес">
```

Синтаксис DTD

- **<!ELEMENT ...>**
- **<!ATTLIST ...>**
- **<!ENTITY ...>**



Описание элементов

<!ELEMENT элемент (содержимое)>



Описание элементов

<!ELEMENT user (login,password)>

<user>

<login>...</login>

<password>...</password>

</user>

Описание элементов

- **<!ELEMENT tag (#PCDATA)>**
- **<!ELEMENT tag EMPTY>**
- **<!ELEMENT tag ANY>**

Описание элементов

- `<!ELEMENT tag (subtag)>`
- `<!ELEMENT tag (subtag?)>`
- `<!ELEMENT tag (subtag*)>`
- `<!ELEMENT tag (subtag+)>`



Описание элементов

**<!ELEMENT person
(name,birthday?,email*,phone+)>**

Описание элементов

<person>

<name>Иванов</name>

<birthday>01.02.2003</birthday>


<email>ivanov@tut.by</email>

<email>ivanov@mail.ru</email>

<phone>123-45-67</phone>

<phone>987-65-43</phone>

</person>



Описание элементов

<!ELEMENT text

(#CDATA | paragraph* | url)>

Описание атрибутов

<!ELEMENT file EMPTY>

<!ATTLIST file

path CDATA #REQUIRED

encoding (UTF8|cp1251)

>

Описание атрибутов

<!ATTLIST элемент

атрибут тип [опции]

атрибут тип [опции]

атрибут тип [опции]

>



Типы атрибутов

CDATA

ID

IDREF

IDREFS

(значение1 | значение2 | . . . | значениеN)



Опции атрибутов

#REQUIRED

#IMPLIED [значение по умолчанию]

#FIXED [значение]



XML-анализаторы

- SAX-парсеры
- DOM-парсеры
- StAX-парсеры



XMLStreamReader

```
InputStream in =  
    new FileInputStream("test.xml");
```

```
XMLInputFactory factory =  
    XMLInputFactory.newInstance();
```

```
XMLStreamReader xmlStreamReader =  
    factory.createXMLStreamReader(in);
```


XMLStreamReader

```
while(xmlStreamReader.hasNext()) {  
    int event = xmlStreamReader.next();  
    switch(event) {  
        case XMLStreamReader.START_ELEMENT:  
            // обработка элемента  
            break;  
        case XMLStreamReader.END_ELEMENT:  
            // обработка элемента  
            break;  
    }  
}
```

XMLStreamReader

```
case XMLStreamReader.START_ELEMENT: {
    String tagName = xmlStreamReader.getLocalName();
    if("user".equals(tagName)) {
        user = new User();
        String id =
            xmlStreamReader.getAttributeValue(0);
        user.setId(id);
    } else if("login".equals(tagName)) {
        String text =
            xmlStreamReader.getElementText();
        user.setLogin(text);
    }
}
```

XMLStreamReader

```
case XMLStreamReader.END_ELEMENT: {  
    String tagName =  
        xmlStreamReader.getLocalName();  
    if("user".equals(tagName)) {  
        users.add(user);  
    }  
    break;  
}
```



XMLStreamReader

int next()

String getLocalName()

String getElementText()

String getAttributeValue(int index)

**String getAttributeValue(String namespace,
String name)**



XMLStreamWriter

```
OutputStream out =  
    new FileOutputStream("test.xml");  
  
XMLOutputFactory factory =  
    XMLOutputFactory.newInstance();  
  
XMLStreamWriter xmlStreamWriter =  
    factory.createXMLStreamWriter(out);
```



XMLStreamWriter

```
xmlStreamWriter
```

```
.writeStartDocument("UTF-8", "1.0");
```

XMLStreamWriter

```
xmlStreamWriter.writeStartElement("users");  
for(User user : user) {  
    xmlStreamWriter.writeStartElement("user");  
    xmlStreamWriter.writeAttribute("id",  
                                    user.getId());  
    xmlStreamWriter.writeStartElement("login");  
    xmlStreamWriter  
        .writeCharacters(user.getLogin());  
    xmlStreamWriter.writeEndElement();  
    xmlStreamWriter.writeEndElement();  
}  
xmlStreamWriter.writeEndElement();
```



XMLStreamWriter

```
xmlStreamWriter.writeEndDocument();
```

```
xmlStreamWriter.close();
```

```
out.close();
```




XMLStreamWriter

```
void writeStartDocument(String encoding,  
                        String version)  
void writeStartElement(String tagName)  
void writeAttribute(String name,  
                    String value)  
void writeCharacters(String text)  
void writeEndElement()  
void writeEndDocument()
```