

# Шаблонизация на клиенте

Степан Резников  
разработчик интерфейсов

Я.Субботник, Москва, 3 июля 2010 года

Больше веб-приложений  
работающих без перезагрузки

Чаще возникает задача  
шаблонизации на клиенте

# Преимущества шаблонизации на клиенте:

## Уменьшение трафика

Посылаем чистые данные — JSON или XML вместо HTML

## Уменьшение нагрузки на сервер

Не нужно накладывать шаблон на сервере

Не нужно на каждый запрос доставать вспомогательные данные



*Задача:*

В JavaScript сгенерить блок (HTML),  
заполнить его имеющимися данными  
и вставить в DOM.

# Данные

```
var data = {  
  url: "/test/",  
  src: "test.gif",  
  width: 60,  
  height: 30,  
  caption: "Трам-парам!"  
};
```

# Хотим получить вот такой HTML

```
<div class="preview">  
  <p class="image">  
    <a href="/test/">  
        
    </a>  
  </p>  
  <p class="caption">Трам-парам!</p>  
</div>
```

В каком виде хранить шаблон?

Как вставить данные в шаблон?



# DOM API

`createElement`

`appendChild`

громоздко

шаблон размазан тонким слоем по JS-коду

# Конкатенация кусочков шаблона вперемешку с данными

```
var result = '<div class="preview"><p class="image"><a href="'  
+ data.url + "'></a></p><p class="caption">'  
+ data.caption + '</p></div>';
```

# Простейший способ шаблонизации в JS



*Douglas Crockford*

# Метод supplant

```
String.prototype.supplant = function(obj) {  
  return this.replace(/{\{([\^}]*)\}/g,  
    function(a, b) {  
      var r = obj[b];  
      return typeof r === 'string' || typeof r === 'number' ? r : a;  
    }  
  );  
};
```

# Метод supplant

```
var data = {  
  url: "/test/",  
  src: "test.gif",  
  width: 60,  
  height: 30,  
  caption: "Трам-парам!"  
};  
var template = '<div class="preview"><p class="image">  
<a href="{url}"></a></p><p class="caption">{caption}</p></div>';  
  
var result = template.supplant(data);
```

# Метод `supplant`. Подстановка данных в сообщение для пользователя

```
var data = {  
  from: 'Madrid',  
  to: 'Barcelona',  
  number: '78A'  
};
```

```
var template = 'Take train {number} from {from} to {to}.';
```

```
var result = template.supplant(data);
```

# JS-шаблонизаторы

Micro-Templating    jTemplates

PURE    JST (TrimPath)

EJS (Embedded JavaScript)

JSONT    JBST    Jemplate

...

# JavaScript-шаблонизатор Micro-Templating

~20 строк, 1,2 КБ

<http://ejohn.org/blog/javascript-micro-templating/>



*John Resig*

# Micro-Templating

```
<script type="text/html" id="test_template">
  <div class="preview">
    <p class="image">
      <a href="<%=url%>">
        " height="<%=height%>"/>
      </a>
    </p>
    <p class="caption"><%=caption%></p>
  </div>
</script>
```

```
<script type="text/javascript">
  var result = tmpl("test_template", data);
</script>
```

# Micro-Templating

```
<script type="text/html" id="tpl_comments">  
  <ul class="comments">  
    <% for (var i = 0, len = items.length; i < len; i++) { %>  
      <%= tpl("tpl_comment", items[i]) %>  
    <% } %>  
  </ul>  
</script>
```

```
<script type="text/html" id="tpl_comment">  
  <li class="comment">  
    <div class="content"><%=content%></div>  
    <div class="author"><%=author%>, <%=datetime%></div>  
  </li>  
</script>
```

# Micro-Templating: компилирование шаблона в функцию

```
var cache = {  
  test_template: function (obj) {  
    var p = [];  
    with (obj) {  
      p.push('<div class="preview"><p class="image"><a href="", url, "">  
<img src="", src, "" width="", width, "" height="", height, ""/></a></p>  
<p class="caption">', caption, '</p></div>');  
    }  
    return p.join("");  
  }  
};
```

Как выбрать  
JS-шаблонизатор?

# Синтаксис вставки данных

Micro-Templating

```

```

Microsoft Ajax 4.0 template engine

```

```

jTemplates

```

```

# Язык выражений

Micro-Templating

```
<% for (var i = 0, len = items.length; i < len; i++) { %>
```

jTemplates

```
{#foreach $T.items as item}
```

# Расположение шаблонов

## Micro-Templating

```
<script type="text/html" id="template">  
  ...  
</script>
```

## jTemplates

```
<textarea id="template" style="display:none">  
  ...  
</textarea>
```

Подгрузка шаблонов из отдельного файла

Читабельность

Производительность

Вложенные шаблоны

XSLT на клиенте

XSLT (eXtensible Stylesheet Language Transformations) — часть спецификации XSL, задающая язык преобразований XML-документов. Спецификация XSLT является рекомендацией W3C.

XSLT — набор шаблонов

Применяем XSLT к XML-документу, получаем другой XML, HTML или обычный текст.

Правила выбора данных из исходного XML пишутся на языке запросов XPath.

# Как это работает в хороших браузерах



```
// Создаем XSLT Processor
var xsltProcessor = new XSLTProcessor();

// Загружаем xsl-файл
var xhr = new XMLHttpRequest();
xhr.open("GET", "example.xsl", false);
xhr.send();

// Импортируем xsl-файл
xsltProcessor.importStylesheet(xhr.responseXML);
```

```
// Парсим XML из строки
var parser = new DOMParser();
var xml = parser.parseFromString("<data>
<url>/test/</url><src>test.gif</src><width>60</width>
<height>30</height><caption>Трам-парам!</caption>
</data>", "text/xml");

// Выполняем трансформацию
var fragment =
    xsltProcessor.transformToFragment(xml, document);

document.body.appendChild(fragment);
```

[www.stepanreznikov.com/client-side-xslt/01-fragment.html](http://www.stepanreznikov.com/client-side-xslt/01-fragment.html)

// Выполняем трансформацию

```
var doc = xsltProcessor.transformToDocument(xml);
```

```
var out = document.adoptNode(doc.documentElement);
```

ИЛИ

```
var out = document.importNode(doc.documentElement, true);
```

```
document.body.appendChild(out);
```

[www.stepanreznikov.com/client-side-xslt/02-document.html](http://www.stepanreznikov.com/client-side-xslt/02-document.html)



# ActiveXObject

MSXML2.DOMDocument.6.0 (или 3.0)

MSXML2.FreeThreadedDOMDocument.6.0 (или 3.0)

MSXML2.XSLTemplate.6.0 (или 3.0)

Также нужно реализовать `document.importNode`

[www.stepanreznikov.com/client-side-xslt/03-ie.html](http://www.stepanreznikov.com/client-side-xslt/03-ie.html)

# Преимущества XSLT

Есть предикаты в матчах

```
<xsl:template match="Brands"/>
```

```
<xsl:template match="Brands[Brand]">  
  <h2>Производители</h2>  
  <ul><xsl:apply-templates select="Brand"/></ul>  
</xsl:template>
```

```
<xsl:template match="Brands[count(Brand) > 10]">  
  <h2>Много производителей</h2>  
  <!-- Выводим бренды в три колонки -->  
</xsl:template>
```

# Преимущества XSLT

Есть вложенность в матчах

```
<xsl:template match="Brands/Brand">
```

```
...
```

```
</xsl:template>
```

```
<xsl:template match="SuperBrands/Brand">
```

```
...
```

```
</xsl:template>
```

# Преимущества XSLT

Есть моды

```
<xsl:template match="Brand" mode="navigation">
```

```
...
```

```
</xsl:template>
```

```
<xsl:template match="Brand" mode="promo-block">
```

```
...
```

```
</xsl:template>
```

# Преимущества XSLT

Есть оси

```
<xsl:apply-templates select="following-sibling::item"/>
```

ancestor

ancestor-or-self

child

descendant

descendant-or-self

following

following-sibling

parent

preceding

preceding-sibling

# Преимущества XSLT

Есть position(), first(), last()

```
<xsl:for-each select="item">  
  <xsl:value-of select="."/>  
  <xsl:if test="position() != last()">, </xsl:if>  
</xsl:for-each>
```

# Преимущества XSLT

Нет проблем с whitespace

Micro-Templating

```
  
<%=caption%>
```

XSLT

```
  
<xsl:value-of select="caption" />
```

Если нужно вывести пробел:

```
<xsl:text> </xsl:text> или <xsl:value-of select="' '>
```

Преимущества XSLT

Уже есть в браузере!

XSLT 1.0

# Производительность

200 писем с метками

	Micro-Templating	JST (TrimPath)	XSL
Firefox 3.6.4	4	12	10
IE 6	13	43	10
IE 7	13	42	10
Safari 4	2	5	25
Chrome 5	2	5	22
Opera 9.27	6	22	20
Opera 10.54	2	4	17

Оно вам  
надо?



# XSLT

2nd Edition

Programmer's Reference



Michael Kay

Спасибо за внимание!

Вопросы?

Степан Резников

[stepan@yandex-team.ru](mailto:stepan@yandex-team.ru)

twitter: @stepanvr