



真传X

IT前沿技术在线大学

www.zhenchuanx.com

前端模版 & webpack

想当年没有模版的时候

不尽流出了眼泪

比惨大会

```
1 container.innerHTML = '<div>' + word + '</div>'
```

```
1 var div = document.createElement('div')
2 var p = document.createElement('p')
3 p.innerHTML = 'hello world'
4 div.appendChild(p)
5 document.body.appendChild(div)
```

后来大家觉得好麻烦

所以 jQuery 的作者就做了个小东西

JavaScript Micro-Templating

```

// Simple JavaScript Templating
// John Resig - https://johnresig.com/ - MIT Licensed
(function(){
  var cache = {};

  this.tmpl = function tmpl(str, data){
    // Figure out if we're getting a template, or if we need to
    // load the template - and be sure to cache the result.
    var fn = !/\W/.test(str) ?
      cache[str] = cache[str] ||
        tmpl(document.getElementById(str).innerHTML) :

      // Generate a reusable function that will serve as a temp
      // generator (and which will be cached).
      new Function("obj",
        "var p=[],print=function(){p.push.apply(p,arguments);};

        // Introduce the data as local variables using with(){}
        "with(obj){p.push('" +

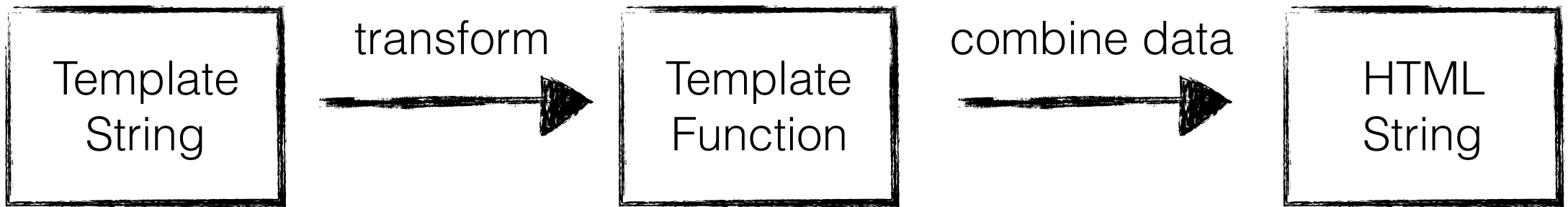
          // Convert the template into pure JavaScript
          str
            .replace(/[\r\t\n]/g, " ")
            .split("<%").join("\t")
            .replace(/((^|%)>[^\t]*)'/g, "$1\r")
            .replace(/\t=(.*?)%>/g, "', $1, '")
            .split("\t").join("'");")
            .split("%>").join("p.push('")
            .split("\r").join("\n")
          + "');}return p.join('');");

    // Provide some basic currying to the user
    return data ? fn( data ) : fn;
  };
})();

```

```
<script type="text/html" id="item_tmpl">
  <div id="<%=id%>" class="<%= (i % 2 == 1 ? " even" : "") %>">
    <div class="grid_1 alpha right">
      
    </div>
    <div class="grid_6 omega contents">
      <p><b><a href="/<%=from_user%>"><%=from_user%></a>:</b> <
    </div>
  </div>
</script>
```

```
var results = document.getElementById("results");
results.innerHTML = tmpl("item_tmpl", dataObject);
```

模版语言的出现，体现了神马？

DOM API 强大但不好维护，
前端做了那么多 View 相关的库本质上就是
为了屏蔽底层 DOM 操作

功能 VS 性能

功能：handlebars 强的的 Blocks 扩展能力

```
1 <div class="entry">
2   <h1>{{title}}</h1>
3   <div class="body">
4     {{#bold}}{{body}}{{/bold}}
5   </div>
6 </div>
```

```
1 Handlebars.registerHelper('bold', function(options) {
2   return new Handlebars.SafeString(
3     '<div class="mybold">'
4     + options.fn(this)
5     + '</div>');
6 });
```



<http://handlebarsjs.com/>

性能：以 doT 为代表

所以怎么做到性能比较好呢？

性能： doT 为代表

- 首先你得小
- 其次，你可以线下预编译，成为静态的 js 文件

```
var render = (function () {
  var cache =
    "var $out = [];\n"
    with ($data) {\n
      $out.push('<h3>');\n
      if (typeof content === 'string') {\n
        $out.push(content);\n
      }\n
      $out.push('</h3>');\n
    }\n
    return $out.join('');"

  return function (data) {
    var fn = new Function('$data', cache);
    return fn(data);
  }
})();
```


性能： doT 为代表

高性能JavaScript模板引擎原理解析

doT 源码

想一想🤔，只用正则替换有什么问题？

MVVM们 和 React 的冲击

DOM Template & JSX

DOM Template: 反正就是没有任何东西不能在DOM上被表征的

```
1  <div id="app-3">
2    <p v-if="seen">现在你看到我了</p>
3    <ol>
4      <li v-for="todo in todos">
5        {{ todo.text }}
6      </li>
7    </ol>
8  </div>
```

DOM Template

- 比较好静态分析
- 可以把模版直接插入DOM，不会有太奇葩的事情发生
- 不需要写词法分析，一般可以直接用现成的HTML parser

JSX: JS上可以写标签

```
1  var MyComponent=React.createClass({
2    getInitialState: function() {
3      return {clickNum: 0};
4    },
5    handleClick:function(){
6      var num=this.state.clickNum;
7      num++;
8      this.setState({clickNum:num});
9    },
10   render: function() {
11     return (
12       <div>
13         <h1 onClick={this.handleClick}>Hello {this.props.name}!</h1>
14         <h2 style={{color: 'red'}}>点击{this.props.name}次数: {this.state.clickNum}</h2>
15       </div>
16     );
17   }
18 });
19 ReactDOM.render(
20   <div>
21     <MyComponent name="张三" />
22     <hr/>
23     <MyComponent name="李四" />
24   </div>,
25   document.getElementById('example')
26 );
```

JSX

- JSX 打破了内容和逻辑分离这个 web 传统的分离方式
- 尽量减少了概念的引入
- 通过 component 化来拆解应用

Web Component

样式 (CSS)

内容 (HTML)

逻辑 (Javascript)

样式 (CSS)

内容 (HTML)

逻辑 (Javascript)

Component

custom element

```
1  class HelloElement extends HTMLElement {  
2    // Monitor the 'name' attribute for changes.  
3    static get observedAttributes() {return ['name']; }  
4  
5    // Respond to attribute changes.  
6    attributeChangedCallback(attr, oldValue, newValue) {  
7      if (attr == 'name') {  
8        this.textContent = `Hello, ${newValue}`;  
9      }  
10   }  
11 }  
12  
13 // Define the new element  
14 customElements.define('hello-element', HelloElement);
```

```
1  <hello-element name="Anita"></hello-element>
```

custom element

Hello, Anita

但我们依然要关注 DOM 实现细节



Polymer = MVVM + Web Component

面向未来的探索

反正 Polymer 我不准备讲，
因为目前还没有什么地方可以用

<https://www.polymer-project.org/>

大家可以自行体验下

晚上的时候我们聊聊一聊DSL的事情



IT前沿技术在线大学