真传X

IT前沿技术在线大学

www.zhenchuanx.com

模板引擎和webpack

模板引擎

```
<div id="container">
    我们正处于刀耕火种的石器时代
</div>
```

DOM API

```
var newTxt = '石器时代需要自己撸工具，摩擦摩擦，似魔鬼的步伐...';
var container = document.getElementById('container');
var title = document.createElement('H1');
var txt = document.createTextNode(newTxt);
title.appendChild(txt);
container.replaceChild(desc, container.childNodes[0]);
```

罗里吧嗦。。。

innerHTML

```
var newTxt = '石器时代需要自己撸工具，摩擦摩擦，似魔鬼的步伐...';
var template = '<H1>' + newTxt + '</H1>';
var container = document.getElementById('container');
container.innerHTML=template;
```

可读性和可维护性差

苍天饶过谁?

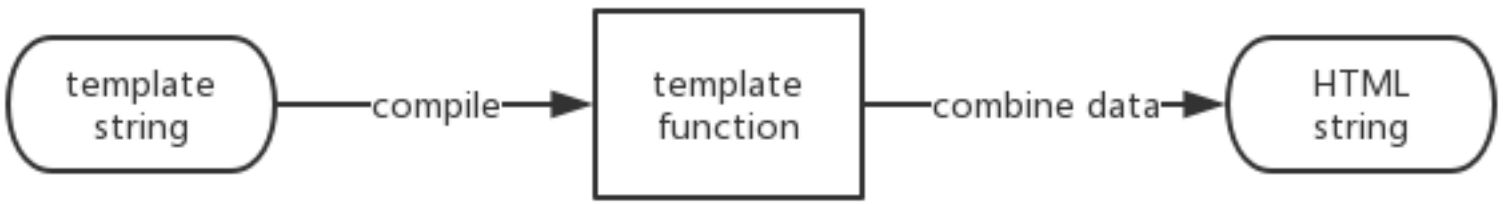
- DOM API操作繁琐
- innerHTML可读性和可维护性差

于是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 template
      // generator (and which will be cached).
      // 等价于var fn = function(obj) {}
      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("\\'")
      + "');}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">
        <img class="righted" src="<%=profile_image_url%>"/>
      </div>
      <div class="grid_6 omega contents">
        <p><b><a href="/<%=from_user%>"><%=from_user%></a>:</b> <%=text%></p>
      </div>
    </div>
</script>
<script>
  var dataObject = {
    id: 1,
    i: 1,
    profile_image_url: 'https://www.google.com.hk/images/branding/googlelogo/2x/g
    from_user: 'Bob',
    text: 'hello world'
  }
  var results = document.getElementById("container");
  results.innerHTML = tmpl("item_tmpl", dataObject);
</script>
```



出现模板引擎的本质原因是：

DOM API虽然强大灵活，但设计得很难用

这些模板引擎又可以细分为2类：

功能 VS 性能

功能：handlebars强大的Blocks

```
<div class="entry">
    <h1>{{title}}</h1>
    <div class="body">
        {{#bold}}{{body}}{{/bold}}
    </div>
</div>
```

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

http://handlebarsjs.com/

如何做到性能好？

1. 体积小
2. 支持线下预编译，成为静态JS文件

性能：体积小

- https://github.com/olado/doT/blob/master/doT.min.js
- https://github.com/jashkenas/underscore/blob/master/underscore.js#L1525

性能：预编译

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

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

```
var render = function ($data) {
    var content = $data.content, $out = '';
    $out += '<h3>';

    if (typeof content === 'string') {
        $out += content;
    }

    $out += '</h3>';

    return $out;
};
```

性能

- 高性能JavaScript模板引擎原理解析
- doT源码

MVVM和react.js提供了更好的方案

MVVM: DOM template

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

配合双向绑定，DOM template具有强大灵活的表征能力

React.js: JSX

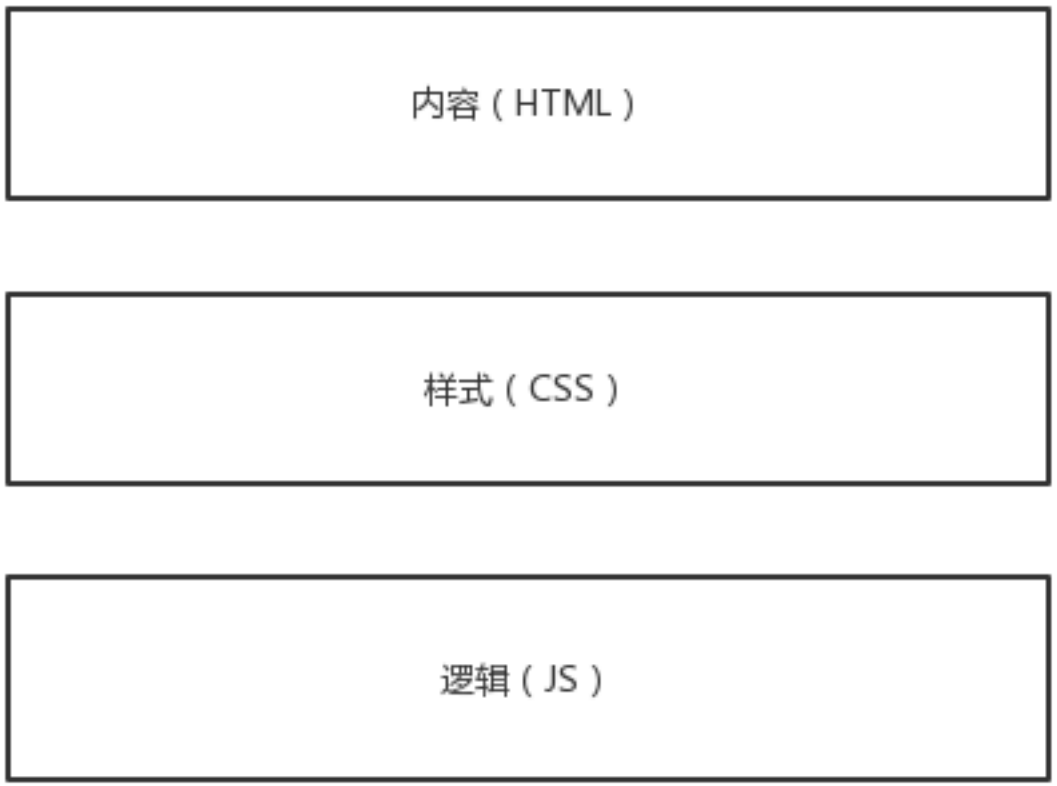
```
export default class Online extends Component {
    constructor() {
        super();
        Report.init(this);
    }
    wording() {…
    }

    downWording() {…
    }

    openApp() {…
    }

    render() {
        return (
            <section id="bar-container">
                <div className="download">
                    <i className="icon-logo"></i>
                    { this.props.appStatus !== -1 && this.props.anchorName &&
                        <span>{this.wording()}</span>
                    }
                    { this.props.appStatus !== -1 && this.props.anchorName &&
                        <div className="download-btn" onClick={this.openApp.bind(this)}>
                            {this.downWording()}
                        </div>
                    }
                </div>
            </section>
        )
    }
}
```

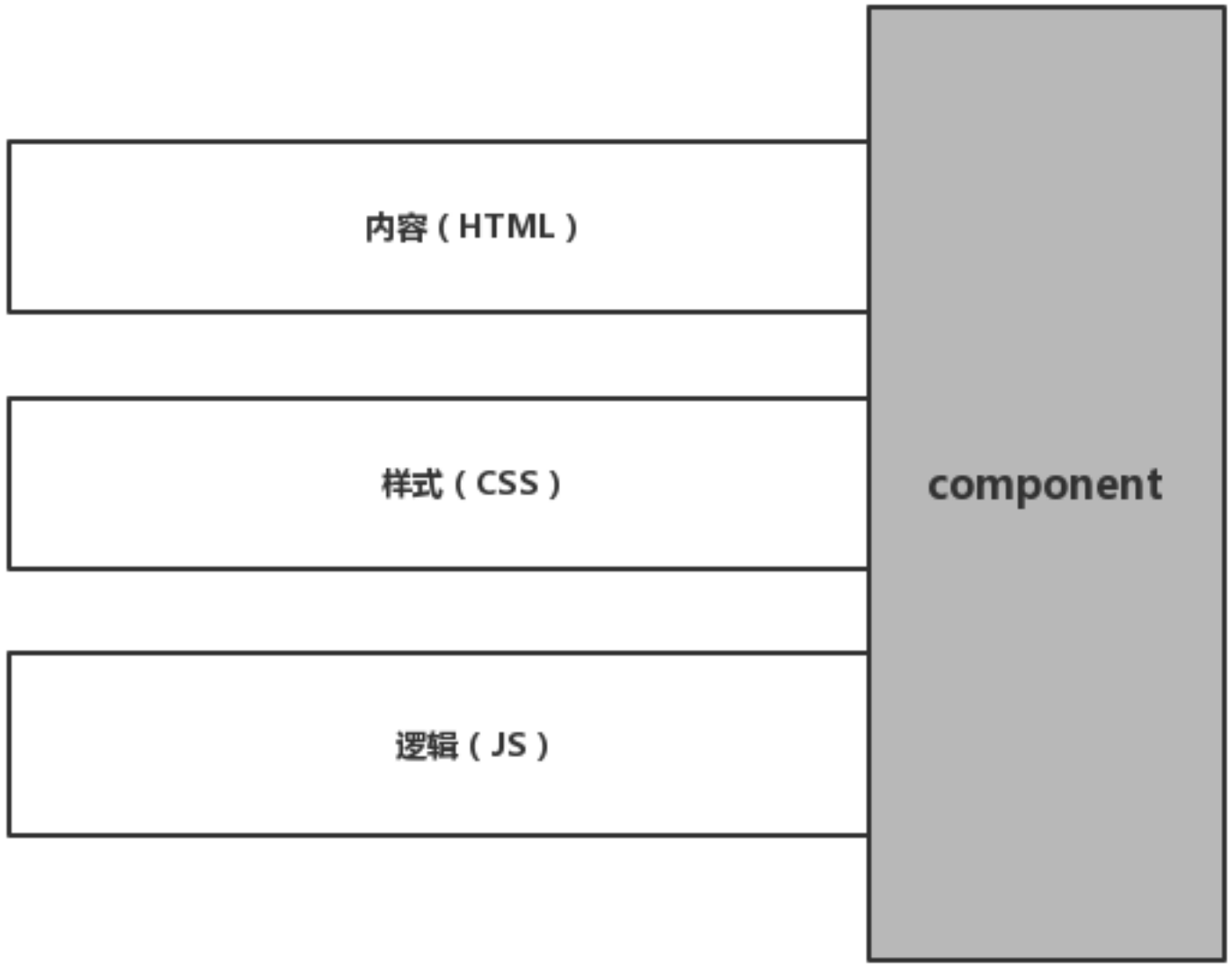
React.js: JSX

1.all in JS（逻辑+内容+样式（可选））

2.利于组件化（https://ant.design/components/button-cn/）

web component





```
<hello-element name="Bob"></hello-element>
<script>
    class HelloElement extends HTMLElement {
        static get observedAttributes() {
            return ['name'];
        }

        attributeChangedCallback(attribute, oldValue, newValue) {
            if(attribute === 'name') {
                this.textContent = `Hello ${newValue}`
            }
        }
    }

    customElements.define('hello-element', HelloElement);
</script>
```

实现

- https://github.com/Polymer/polymer
- https://github.com/ionic-team/stencil

webpack

概念

练习

IT前沿技术在线大学