Nunjucks

我们选择Nunjucks作为模板引擎。Nunjucks是Mozilla开发的一个纯JavaScript编写的模板引擎，**既可以用在Node环境下，又可以运行在浏览器端。但是，主要还是运行在Node环境下，因为浏览器端有更好的模板解决方案，例如MVVM框架。**

如果你使用过Python的模板引擎jinja2，那么使用Nunjucks就非常简单，两者的语法几乎是一模一样的，因为Nunjucks就是用JavaScript重新实现了jinjia2。

模板引擎就是基于模板配合数据构造出字符串输出的一个组件。比如下面的函数就是一个模板引擎：

```
function examResult (data) {
    return `${data.name}同学一年级期末考试语文${data.chinese}分，数学${data.math}分，位于年级第${data.ranking}名。`
}
```

输出HTML有几个特别重要的问题需要考虑：

转义

对特殊字符要转义，避免受到XSS攻击。比如，如果变量name的值不是小明，而是小明<script>...</script>，模板引擎输出的HTML到了浏览器，就会自动执行恶意JavaScript代码。

格式化

对不同类型的变量要格式化，比如，货币需要变成12,345.00这样的格式，日期需要变成2016-01-01这样的格式。

简单逻辑

模板还需要能执行一些简单逻辑

从上面的例子我们可以看到，虽然模板引擎内部可能非常复杂，但是使用一个模板引擎是非常简单的，因为**本质上我们只需要构造这样一个函数：**

```
function render(view, model) {
    // TODO:...
}
```

其中，**view是模板的名称（又称为视图）**，因为可能存在多个模板，需要选择其中一个。**model就是数据，在JavaScript中，它就是一个简单的Object。render函数返回一个字符串，就是模板的输出。**

```
const nunjucks = require('nunjucks');
```

```
function createEnv(path, opts) {
    var autoescape = opts.autoescape === undefined ? true : opts.autoescape,
    nocache = opts.nocache || false,
    watch = opts.watch || false,
    throwOnUndefined = opts.throwOnUndefined || false,
    env = new nunjucks.Environment(
        new nunjucks.FileSystemLoader(path, {
            nocache: nocache,
            watch: watch
    }), {
        autoescape: autoescape,
        throwOnUndefined: throwOnUndefined
    });
    if(opts.filters) {
        for(var f in opts.filters) {
            env.addFilter(f, opts.filters[f]);
        }
    }
    return env;
}

var env = createEnv("view", {
    watch: true,    // 当模板变化时重新加载。使用前请确保已安装可选依赖
chokidar。
    filters: {
        hex: function(n) {
            return '0x' + n.toString(16);
        }
    }
});

var s = env.render('index.html', {name: `htcs`});
```