README

基于 virtual-dom 的前端框架

框架分为两部分

- `mcore` 模板引擎（Core), 无依赖，支持（IE6+)
- `mcoreApp` SPA（单页应用）框架，依赖 `jQuery` , `mcore` 支持（IE6+）

> *注：开发时，依赖 webpack 及 [h2svd-loader]*

webpack 配置

```
//webpack.config.js
var webpack = require('webpack');

// h2svd-loader
require('h2svd-loader');

module.exports = {
    ...
    module: {
        loaders: [
            {// 引入 html 转 mcore virtual-dom 的 loader
                test: /\/tpl\/.*(\.html)$/,
                loader: 'h2svd-loader'
            }
        ]
    }
};
```

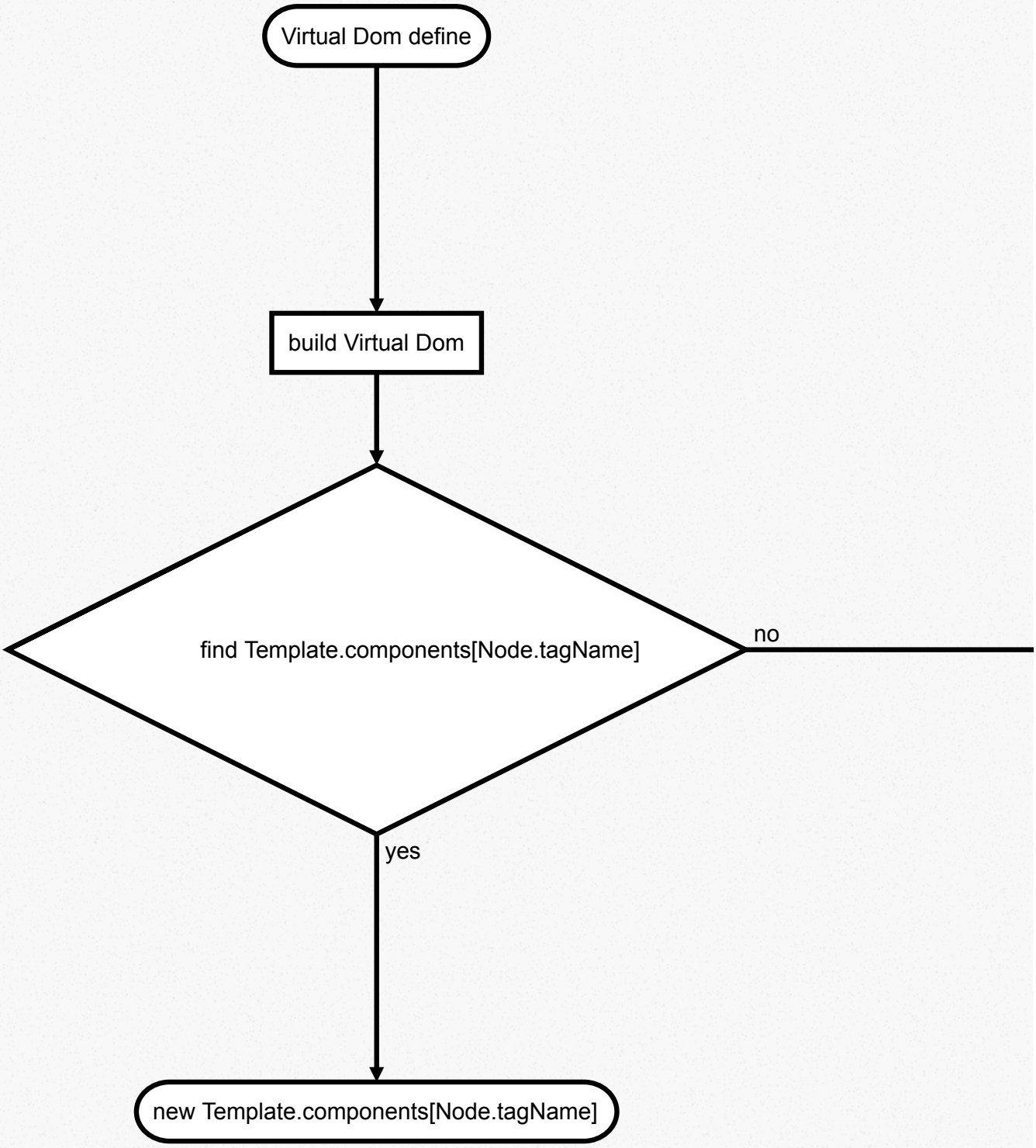
mcore 模板引擎

原理

渲染流程


Template
@render Html, scope
Html -> Virtual Dom define 依赖：h2svd-loader(开发时，通过 webpack 实现)
new virtual dom = (Virtual Dom define)(scope)
是否存在 old virtual dom
no
yes
diff VirtualDom
应用变更到 dom

component 实现流程



> *diff VirtualDom 时，component 当成没有子树的 Node (只 diff 属性), component 的 dom 变更，由 component 自身维护*

DEMO

> *模板中，变量都在 `scope` 名字空间下 （事件除外）*

```
<!-- ./tpl/test.html -->
<ul>
  <li mc-for="v , k in scope.list" mc-on-click="showIx(v)">
    <span mc-data-ix="k + 1">{v.name}</span>
  </li>
</ul>
```

```
# demo.coffee
{Template} = require 'mcore'

# init tpl
tpl = new Template()

# bind click event
tpl.showIx = (v, el, event)->
    console.log v, el, event

 # render
tpl.render require('./tpl/test.html'),
    list: [
       {name : 'ok1'}
       {name : 'ok2'}
    ]
 , -> # rendered
    document.body.appendChild tpl.refs
```

binders 自定义属性

```
{Template} = require 'mcore'

Template.binders.color = (el, value)->
    el.style.color = value
```

```
<button mc-color="scope.color">Apply</button>
```

components 自定义组件

```
{Template, Component} = require 'mcore'
```

```coffee
class Time extends Component
    init: ->
        @on 'rendered', =>
            setTimeout =>
                @set 'time', new Date()
            , 1000

        @render require('./tpl/tagTime.html'),
            time: new Date()

Template.components.time = Time
```

```
<!-- ./tpl/tagTime.html -->
{scope.time}
```

```
<time></time>
```

formatters 过滤函数

```coffee
{Template} = require 'mcore'

Temaplat.formatters.formNow = (value)->
    moment(value).formNow()

Template.formatters.toString = (value)->
    String value or ''
```

```
<span>{scope.date | formNow | toString }</span>
```