前端知识

文档修订记录

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 版本  编号 | \*变化  状态 | 简要说明  （变更内容和变更范围） | 日期 | 变更人 | 批准日期 | 批准人 |
| V1.0 | 建立 | 前端知识 |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

\*变化状态：建立，修改，增加，删除

文档审批信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 审批人 | 角色 | 审批日期 | 签字 | 备注 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**版权说明**

Copyright **©** 2017 xxxx版权

保留所有的权利

**目录**

一、 兼容篇 5

二、 性能优化篇 8

三、 适配篇 9

1.PC端 9

2.移动端 9

四、 打包篇 10

1. Requirejs 10

2. Webpack 10

五、 组件篇 11

步骤 12

六、 CSS预编译篇 13

七、 框架篇 13

# 兼容篇

1.浏览器默认的基本样式的初始值不一致，需要统一处理，建议使用淘宝的reset.css，代码如下

body, h1, h2, h3, h4, h5, h6, hr, p, blockquote, dl, dt, dd, ul, ol, li, pre, form, fieldset, legend, button, input, textarea, th, td { margin:0; padding:0; }

body, button, input, select, textarea { font:12px/1.5tahoma, arial, \5b8b\4f53; }

h1, h2, h3, h4, h5, h6{ font-size:100%; }

address, cite, dfn, em, var { font-style:normal; }

code, kbd, pre, samp { font-family:couriernew, courier, monospace; }

small{ font-size:12px; }

ul, ol { list-style:none; }

a { text-decoration:none; }

a:hover { text-decoration:underline; }

sup { vertical-align:text-top; }

sub{ vertical-align:text-bottom; }

legend { color:#000; }

fieldset, img { border:0; }

button, input, select, textarea { font-size:100%; }

table { border-collapse:collapse; border-spacing:0; }

2.浮动ie产生的双倍距离问题

在IE6下，如果对元素设置了浮动，同时又设置了margin-left或margin-right，margin值会加倍。

代码如下：

#box{ float:left; width:10px; margin:0 0 0 10px;}

此时IE下会产生20px的距离，解决方案是在float的标签样式中加入\_display:inline,将其转换为行内属性

3.渐进识别的方式，从总体中逐渐排除局部

用”\9”，识别ie9以下的情况

用”+”，识别ie6，ie7的情况

用”\_”，识别ie6的情况

举例 css

.back-color {

background-color: #000; /\*所有识别\*/

.background-color\9: #000; /\*IE6、7、8识别\*/

+background-color: #000; /\*IE6、7识别\*/

\_background-color: #000; /\*IE6识别\*/

}

4.怪异模式问题：漏写DTD声明，firefox仍然会按照标准模式来解析网页，但在IE中会触发怪异模式。为了避免带来不必要的麻烦，可以直接使用<doctype html>

对标准模式和怪异模式的理解：

标准模式：浏览器按W3C标准解析执行代码

怪异模式：使用浏览器自己的方式解析执行代码，而每个浏览器的执行的方式是不一样的

举个例子：

标准模式下，!important在IE6下不认识，在IE7、IE8、Firefox、Chrome等浏览器中认识；但怪异模式中，IE6/7/8都不认识!important

最严格模式 应该使用XHTML1.0，具体 声明如下：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

如果是接手遗留网页，没有DTD声明，并有很多标签已被废除，建议使用XHTML兼容模式，声明如下：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

判定浏览器是标准模式还是怪异模式，可以使用以下判断：

方法一：alert(window.document.compatMode);//BackCompat 怪异模式;CSS1Compat 标准模式

方法二：jQuery的方法：alert($.boxModel) alert($.support.boxModel)

以下大部分针对原生开发来说，若只是用jquery大可不用了解

5.IE支持currentStyle，Firefox使用getComputeStyle

样式表分为三种，内嵌样式（inline style）：写在tag里面的，内嵌样式只对所有的Tag有效；内部样式（internal style sheet）是写在HTML里面的，只对所在的网页有效；外部样式表（External style sheet）写在.css文件里，网页内引用css文件。

在js中，引用是通过document.getElementById(id).style.XXX方式获取XXX值，但是只能取到通过内嵌方式的样式值。

例子如下：

<style>

#mydiv {

Width: 300px

}

</style>

Js中调用：

Var mydiv = document.getElementById(‘mydiv’)

If(mydiv.currentStyle) {

Var width = mydiv.currentStyle[‘width’]

} else if(window.getComputeStyle) {

Var width = window.getComputeStyle(mydiv, null)[‘width’]

//或用 var width = document.defaultView.getComputeStyle(mydiv, null).width

}

6.IE中使用innerText，ff中使用textContent

7.滤镜：

IE：filter: alpha(opacity = num),

Ff：-moz-opacity: num

8.事件：

IE：使用attachEvent

Ff：使用addEventListener

9.鼠标：

IE：使用event.clientX

Ff：使用event.pageX

10.回调函数目标：

IE：使用event.srcElement

Ff：使用event.target

以下针对ie8的兼容性处理：

11.IE8/9不支持Array.indexOf，解决方法重写

if (!Array.prototype.indexOf)

{

Array.prototype.indexOf = function(elt /\*, from\*/)

{

var len = this.length >>> 0;

var from = Number(arguments[1]) || 0;

from = (from < 0) ? Math.ceil(from) : Math.floor(from);

if (from < 0)

from += len;

for (; from < len; from++)

{

if (from in this && this[from] === elt)

return from;

}

return -1;

};

}

IE8不支持trim方法，解决方法重写

if(!String.prototype.trim){

var rtrim = /^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g;

function trim( text ) {

return text == null ?

"" :

( text + "" ).replace( rtrim, "" );

}

String.prototype.trim = function(){return trim(this);}

}

12.IE8不支持canvas，解决方案，引入excanvas.js，路径：SSC\开发2组\工作报告\其他报告\交接报告\谢磊\补充\excanvas.js

注意点：

1. 必须在canvas标签加载完成后再引入excanvas.js
2. 与jquery混合使用的时候必须使用jquery1.7以上的版本

13.IE8不支持媒体查询，解决方案引入response.min.js，路径：SSC\开发2组\工作报告\其他报告\交接报告\谢磊\补充\ respond.min.js

14.IE8不支持css3样式，解决方案，引入PIE.htc，路径：SSC\开发2组\工作报告\其他报告\交接报告\谢磊\补充\ PIE-1.0.0.zip

能解决border-radius、box-shadow、gradient等问题

使用方式：

.pie\_radius{

width:360px;

height:200px;

background-color:#34538b;

-moz-border-radius:8px;

-webkit-border-radius:8px;

border-radius:8px;

position:relative;

behavior:url(pie.htc);//pie.htc的路径为你在工程放的路径

}

注意，meta中要提供content-Type新属性，” text/x-component”,否则behavior会被忽略

15.IE8中的text-align:center文字标签居中，但元素并不居中，解决方案：在子元素中加入css属性：margin: 0 auto

# 性能优化篇

代码层面：避免使用css表达式，避免使用高级选择器，通配选择器。

缓存利用：缓存Ajax，使用CDN，使用外部js和css文件以便缓存，添加Expires头，服务端配置Etag，减少DNS查找等

Etag：当发送一个服务器请求时，浏览器首先会进行缓存过期判断。浏览器根据缓存过期时间判断缓存文件是否过期：

情景一：若没有过期，则不向服务器发送请求，直接使用缓存中的结果，此时我们在浏览器控制台中可以看到 200 OK(from cache) ，此时的情况就是完全使用缓存，浏览器和服务器没有任何交互的

情景二：若已过期，则向服务器发送请求，此时请求中会带上设置的文件修改时间和Etag；然后，进行资源更新判断。服务器根据浏览器传过来的文件修改时间，判断自浏览器上一次请求之后，文件是不是没有被修改过；根据Etag，判断文件内容自上一次请求之后，有没有发生变化：

情形一：若两种判断的结论都是文件没有被修改过，则服务器就不给浏览器发index.html的内容了，直接告诉它，文件没有被修改过，你用你那边的缓存吧—— 304 Not Modified，此时浏览器就会从本地缓存中获取index.html的内容。此时的情况叫协议缓存，浏览器和服务器之间有一次请求交互

情形二：若修改时间和文件内容判断有任意一个没有通过，则服务器会受理此次请求，之后的操作一样

只有get请求会被缓存，post请求不会

Etag由服务器端生成，客户端通过If-Match或者说If-None-Match这个条件判断请求来验证资源是否修改。常见的是使用If-None-Match。请求一个文件的流程可能如下：

====第一次请求===

1.客户端发起 HTTP GET 请求一个文件

2.服务器处理请求，返回文件内容和一堆Header，当然包括Etag(例如"2e681a-6-5d044840")(假设服务器支持Etag生成和已经开启了Etag).状态码200

====第二次请求===

客户端发起 HTTP GET 请求一个文件，注意这个时候客户端同时发送一个If-None-Match头，这个头的内容就是第一次请求时服务器返回的Etag：2e681a-6-5d0448402.服务器判断发送过来的Etag和计算出来的Etag匹配，因此If-None-Match为False，不返回200，返回304，客户端继续使用本地缓存

Expires和Cache-control

Expires要求客户端和服务端的时钟严格同步。HTTP1.1引入Cache-Control来克服Expires头的限制。如果max-age和Expires同时出现，则max-age有更高的优先级

请求数量：合并样式和脚本，使用css图片精灵，初始首屏之外的图片资源按需加载，静态资源延迟加载。

请求带宽：压缩文件，开启GZIP

代码层面：

少用全局变量

用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能

用setTimeout来避免页面失去响应

缓存DOM节点查找的结果

避免使用CSS Expression

避免全局查询

避免使用with(with会创建自己的作用域，会增加作用域链长度)

多个变量声明合并

避免图片和iFrame等的空Src。空Src会重新加载当前页面，影响速度和效率

尽量避免写在HTML标签中写Style属性

移动端：

尽量使用css3动画，开启硬件加速

适当使用touch事件代替click事件

避免使用css3渐变阴影效果

可以用transform: translateZ(0)来开启硬件加速

不滥用Float。Float在渲染时计算量比较大，尽量减少使用

不滥用Web字体。Web字体需要下载，解析，重绘当前页面，尽量减少使用

合理使用requestAnimationFrame动画代替setTimeout

CSS中的属性（CSS3 transitions、CSS3 3D transforms、Opacity、Canvas、WebGL、Video）会触发GPU渲染，请合理使用。过渡使用会引发手机过耗电增加

同样pc端的也适用

# 适配篇

## 1.PC端

最常用的就是自适应，解决方案，使用媒体查询，监控不同的分辨率下去动态改变属性样式，公司已经采用了该方式处理

## 2.移动端

移动端适配方案有三种

1. 固定高度，宽度自适应

垂直方向用定值，水平方向用百分比、定值、flex都行。

- 以小宽度作为参照是因为如果布局满足了小宽度的摆放，当屏幕变宽时，简单的填充空白就可以了；而如果反过来就可能造成“挤坏了”，考虑 header 区域，左测 logo 右测横向 nav 的情况。

- 需要小宽度的布局，又需要大宽度的图像，这是一个矛盾点。

- 320px 过于窄小，不利于页面的设计；只能设计横向拉伸的元素布局，存在很多局限性。

- 兼容性较好。

该方案使用的是完美视口：

<meta name="viewport" content="width=device-width,initial-scale=1">

1. 固定宽度，viewport缩放

设计图、页面宽度、viewport width使用一个宽度，浏览器帮我们完成缩放。单位使用px即可。

- 开发简单 缩放交给浏览器，完全按视觉稿切图。

- 还原精准 绝对等比例缩放，可以精准还原视觉稿（不考虑清晰度的情况下）。

- 测试方便 在PC端即可完成大部分测试，手机端只需酌情调整一些细节（比如图标、字体混合排列时，因为字体不同造成的对齐问题）

这种方法需要根据屏幕宽度来动态生成viewport，生成的 viewport 基本是这样：

<meta name="viewport" content="width=640,initial-scale=0.5,maximum-scale=0.5,minimum-scale=0.5,user-scalable=no">

1. Rem做宽度，viewport缩放

依照某特定宽度设定 rem 值（即 html 的 font-size），页面任何需要弹性适配的元素，尺寸均换算为 rem 进行布局；当页面渲染时，根据页面有效宽度进行计算，调整 rem 的大小，动态缩放以达到适配的效果。利用该方案，还可以根据 devicePixelRatio 设定 initial-scale 来放大 viewport，使页面按照物理像素渲染，提升清晰度

- 清晰度高，能达到物理像素的清晰度。

- 能解决 DPR 引起的“1像素”问题。

- 向后兼容较好，即便屏幕宽度增加、PPI 增加该方案依旧适用。

强烈推荐第三种，第一种最为简单

# 打包篇

## Requirejs

目前公司内部的无论是pc端还是移动端使用的打包方式都是requirejs。

介绍下移动端部分，代码路径：SSC\开发2组\工作报告\其他报告\交接报告\谢磊\手机组件\edcui\_mobile

总结公司组件的好处：

- 定义的引入方式支持指定引入顺序

- 依赖提前加载在script标签内的ext\_js\_depend中定义，这个属性内的js是按顺序引入

- 无需提前加载的在ext\_js中定义

- 支持按模块按需加载，开放了一个属性edcui\_module，该属性可以指定需要加载的组件

- 支持css在script标签中定义，用‘css!’做标识

- 支持指定皮肤，用属性theme做皮肤的更换

- 支持打包和非打包两种模式，分别在生产环境和开发环境使用

## 2. Webpack

WebPack 是一个模块打包工具，可以使用WebPack管理模块依赖，并编绎输出模块们所需的静态文件。它能够很好地管理、打包Web开发中所用到的HTML、Javascript、CSS以及各种静态文件（图片、字体等），让开发过程更加高效。对于不同类型的资源，webpack有对应的模块加载器。webpack模块打包器会分析模块间的依赖关系，最后生成了优化且合并后的静态资源

Webpack存在两大特色：

1. code splitting
2. loader 可以处理各种类型的静态文件，并且支持串联操作

webpack 是以commonJS的形式来书写脚本滴，但对 AMD/CMD 的支持也很全面，方便旧项目进行代码迁移

webpack具有requireJs和browserify的功能，但仍有很多自己的新特性：

1.对 CommonJS 、AMD 、ES6的语法做了兼容

2.对js、css、图片等资源文件都支持打包

3.串联式模块加载器以及插件机制，让其具有更好的灵活性和扩展性，例如提供对CoffeeScript、ES6的支持

4.有独立的配置文件webpack.config.js

5.可以将代码切割成不同的chunk，实现按需加载，降低了初始化时间

6.具有强大的Plugin接口，大多是内部插件，使用起来比较灵活

7. webpack 使用异步 IO 并具有多级缓存。这使得 webpack 很快且在增量编译上更加快

# 组件篇

介绍我们公司开发组件的方式，以一个简单的例子为例，代码路径：SSC\开发2组\工作报告\其他报告\交接报告\谢磊\手机组件\edcui\_mobile\biui\js\nl-edc-ui\mobile\plugins\ com.newland.edc.ui.input.mobile.js

代码较为较短，复制出来讲解：

/\*\*

\* 微信版本

\* 文本框

\* \*/

(function($) {

var Input = function(input, config) {

}

$.fn.edcui\_input = function(options, params, target) {

if (!target) {

target = this;

}

if ((typeof options).toLowerCase() == "string") {

// 执行组件实例方法

return $.fn.edcui\_input.methods[options](params, target);

} else if ((typeof options).toLowerCase() == "object") {

params = options;

}

params = params || {};

params = $.extend({}, $.fn.edcui\_input.defaults, params);

return this.each(function() {

var self = this;

var $this = $(this);

var targetHTML = $this.prop("outerHTML"); // 目标节点自身HTML内容

if (!$this.attr("placeholder")) {

$this.attr("placeholder", placeholder);

} else {

params.placeholder = $this.attr("placeholder");

}

$this.data("edcuiInputConfig", params);

$this.data("targetHTML", targetHTML);

return $this;

});

}

/\*\*

\* 组件默认属性

\*/

$.fn.edcui\_input.defaults = {

tagName:"edcui\_input",

cssClass:"edcui\_input"

};

/\*\*

\*组件方法

\*/

$.fn.edcui\_input.methods = {

destroy: function(params, target) {

}

}

})($);

## 步骤

1. 创建一个匿名函数块，主要目的：a.为了防止全局变量的冲突；b.匿名函数块内的代码为自执行代码，不用依赖于别的函数
2. 定义方法$.fn.edcui\_input = function(options, params, target)，该方法在原型链上创建函数，可以直接用对象属性方式调用，需要传入三个参数，其中options参数如果是对象，表示为参数；若为string表示为方法名。params和target在使用的时候可以不定义，但代码里要给默认处理。该方法是必须的方法，不可少
3. 方法$.fn.edcui\_input.defaults表示默认属性，主要用来定义该组件总共有哪些外部可以设置的属性，可以通过$.extend({}, $.fn.edcui\_input.defaults, params)方式覆盖默认属性值
4. 方法$.fn.edcui\_input.methods定义一些外部可以直接调用的方法，比如setValue、getValue，调用的方式也和初始化的方式基本相同
5. 全部写完一个组件即完成
6. 使用上
   1. 初始化

$("#xxx").edcui\_input({

//因为input组件为基础组件，并没有外部参数，调用的时候不给参数即可

});

* 1. 调用方法

$("#xxx").edcui\_input("destroy");//表示调用destroy方式销毁实例

# CSS预编译篇

CSS预处理是现代处理css很好的方法，目前比较火的有SASS、LESS、POSTCSS等方式，个人建议可以多去了解这些，预处理可以减少很多写css上的成本以及更加规范合理

# 框架篇

当你拥有了很强的js基础时，可以通过阅读一些较好的框架源码，比如jquery、zepto；vue、react、angular。

前者框架重在直接处理真实dom，无论是绑定事件，渲染都以真实dom做处理，在性能方面较差，但同样存在优势即开发方便，入门快，节省了很多学习成本，社区中有很多较好用的插件可直接使用。

后者框架采用的是虚拟dom，用模板引擎渲染整个dom fragment，而且还对整个渲染机制都做了优化，在性能方面自然是较好，可以提高个人在前端方面的能力，但缺点就是要花时间学习，要有较强的js基础，生态链非常大，学习整个生态要花较长时间。

各有利弊，同学要结合个人的需要去学习，如果想成为一名很强悍的前端工程师，个人建议花点时间学习现代技术，可能不会使用到，但是思想一样可以借鉴。