### cookie

\* cookie的设置

document.cookie = 'key1=value1';

document.cookie = 'key2=value2';

> 注意,后面的不会覆盖前面的,在ff下测试

\* cookie有效期的设置

```javascript

var oDate = new Date();

oDate.setDate(oDate.getDate() + 14);

document.cookie = "name=Tom;expires="+oDate;

```

\* 封装cookie相关的函数

```javascript

function setCookie(key,value,iDate){

var oDate = new Date();

oDate.setDate(oDate.getDate() + iDate);

document.cookie = key+"="+value+";expires="+oDate;

}

function getCookie(name){

var str = document.cookie;

var arr1 = str.split(";");

var result = "";

for(var i=0;i<arr1.length;i++){

var arr2 = arr1[i].split("=");

if(arr2[0].trim() == name.trim()){

result = arr2[1];

break;

}

}

return result;

}

function removeCookie(name){

setCookie(name,false,-1);

}

```

\* cookie的特点:

> \* cookie是document对象上的一个属性

> \* cookie的大小限制是4k

> \* cookie会存在于请求头中

> \* 主域名中的cookie会在子域名中存在

### Storage

\* localStorage 和 sessionStorage的api相同

> \* setItem(key,value) 设置

> \* getItem(key) 获取

> \* removeItem(key) 删除

> \* clear() 清除所有

> \* length 内容个数

> \* key(num) 获取key的值

\* localStorage 和 sessionStorage的存储时效不同

> \* localStorage 持久化

> \* sessionStorage 网页会话结束失效

\* localStorage 和 sessionStorage的存储容量不同

> \* localStorage 2-5Mb左右

> \* sessionStorage 2-10Mb左右,部分浏览器 没有限制

\* 注意点:

>\* Storage仅能存储字符串

>\* Storage的数据不能在不同的域中共享

### 拖拽

\* 被拖拽元素添加 draggable=true

\* 绑定拖拽事件

> 被拖拽元素相关

>> \* ondragstart 当元素开始拖动时触发

>> \* ondrag 被拖动元素拖动时触发

>> \* ondragend 拖拽时松开鼠标时触发

> 容器相关

>> \* ondrop 当放置被拖动元素时触发

>> \* ondragover 被拖动元素正在到放置到容器时触发,默认无法把元素放置到其他元素中,如果需要设置允许放置,我们必须阻止对元素的默认处理方式。

>> \* ondragenter 当被拖动元素进入容器时触发

>> \* ondragleave 当被拖动元素离开容器时触发

>> \* ondragend 被拖动元素在容器中松开鼠标时触发

>> \* ondrag 容器中的被拖动元素拖动时触发

\* dataTransfer对象

> \* 设置数据 dataTransfer.setData(key,val)

> \* 获取数据 dataTransfer.getData(key)

> \* 设置拖动的图标图片: dataTransfer.setDragImage(图片DOM节点,x,y)

> 注意:dataTransfer对象存在在原生的event对象上,如果用jQuery的event对象,需要使用jQuery的event对象的originalEvent

### 拖拽插件

\* jquery.pep.js 官网: http://pep.briangonzalez.org/

### SVG 介绍

SVG(Scalable Vector Graphics)可缩放矢量图形,一种用来描述二维矢量图形的XML语言

### SVG视口(viewport)

\* SVG视口是指可见区域的大小(画布的大小)

```javascript

<svg width='800' height='600'></svg>

```

> svg默认的宽高是300px\*150px

### SVG视区盒子(viewBox)

viewBox = 'x y width height'

//x:水平坐标 y:垂直坐标 width:宽度 height:高度

\* viewBox会根据坐标点和宽高来截取视口,把截取后的内容全屏

```javascript

<svg width='800' height='600' viewBox='0 0 80 60'>

<rect >

</svg>

```

### 图形

\* 直线

<line x1="0" y1="0" x2="300" y2="300" style="stroke:rgb(99,99,99);stroke-width:2"/>

\* 矩形

<rect x='100' y='100' width='400' height='300' fill="#ff0">

### canvas

canvas是一个画布,用来在网页上绘制图像

### canvas基础

\* 创建canvas

```javascript

<canvas></canvas>

```

> canvas默认的宽高是300px\*150px

\* 画布的大小

> \* 在canvas标签中设置的width和height是指画布大大小,不用带单位

> \* 在css中设置的width和height是指画布拉伸缩放后的大小

> \* 一般建议在js中设置画布的大小

\* 画直线

```javascript

cxt.moveTo(0,0);//指定起始坐标

cxt.lineTo(100,100);//直线终点坐标

cxt.lineTo(100,200);

cxt.strokeStyle='#f00';//指定描边的样式

cxt.lineWidth = 10;//设置线条的宽度

cxt.closePath();//封闭路径

cxt.stroke(); //描边

cxt.fillStyle = 'rgba(0,255,0,0.5)';//设置填充的颜色

cxt.fill();//闭合路径后填充

```

\* cxt.beginPath();重新开始路径,绘制时不会绘制之前的路径

\* 圆形

```javascript

cxt.arc(圆心x,圆心y,半径,起始角度,终止角度,是否是逆时针(true为逆时针))

```

> 3点钟时0PI 6点钟是0.5PI 9点钟是1PI 12点钟是1.5PI

\* 矩形

cxt.strokeRect(x,y,宽,高)

cxt.fillRect(x,y,宽,高)

\* 平移坐标原点

cxt.translate(x,y);

\* 旋转坐标系

cxt.rotate(弧度) 以顺时针为方向旋转

\* 缩放变换

cxt.scale(x轴比例,y轴比例);

> 所有的变换都是针对坐标系的

> 各种变换之间是相互叠加的

\* save和restore环境

cxt.save(); 保存之前的环境

cxt.restore();恢复到上一次保存时的环境

\* 线性渐变(LinearGradient)

var linearGradient = cxt.createLinearGradient(起始x,起始y,终止x,终止y);

linearGradient.addColorStop(百分比,颜色)

\* 径向渐变(RadialGradient)

var radialGradient = createRadialGradient(起始圆心x,起始圆心y,起始圆半径,终止圆心x,终止圆心y,终止圆半径);

\* 文本

\* 图像绘制

\* 图形画刷

\* 剪辑区

\* 绘制阴影

\* 绘制曲线

> \* 圆弧: context.arc(圆心x,圆心y,半径,起始角度,终止角度,是否逆时针)

> \* 二次样条曲线: context.quadraticCurveTo(qcpx,qcpy,qx,qy)

>> http://blogs.sitepointstatic.com/examples/tech/canvas-curves/quadratic-curve.html

> \* 贝塞尔曲线: context.bezierCurveTo(cp1x,cp1y,cp2x,cp2y,x,y)

>> http://blogs.sitepointstatic.com/examples/tech/canvas-curves/bezier-curve.html

### 推荐插件

\* http://echarts.baidu.com/

\* https://antv.alipay.com/zh-cn/index.html

\* http://www.chartjs.org/

\* https://www.highcharts.com

\* https://d3js.org/

CSS进阶

### Less介绍

Less 是一门 CSS 预处理语言，它扩展了 CSS 语言，增加了变量、Mixin、函数等特性，使 CSS 更易维护和扩展。

### Less编译

\* 浏览器

```javascript

<!--

1. 引入.less文件,注意rel="stylesheet/less"

2. 在less文件后面引入编译less的js文件,可以从http://lesscss.cn/中下载

-->

<link rel="stylesheet/less" type="text/css" href="less/001.less">

<script src="js/less.js" type="text/javascript"></script>

```

\* 客户端

> http://koala-app.com/

\* 服务端

### Less语法

\* 注释

```css

//我是注释一 编译被删除

/\*我是注释二 编译被保留\*/

```

\* 变量

@变量名:值

\* 混合(Mixin)

> \* 基本混合,直接把选择器写在另外一个选择中

> \* 带参数

> \* 带参数有默认值

\* 匹配模式

选择符合条件的混合模式,根据传入的参数来决定使用哪个混合

\* 嵌套

在选择器中嵌套选择器

\* 运算

+-\*/

\* 避免编译

属性:~'css的值'

\* !important关键字

在混合模式后添加!important关键字,会为混合模式中的所有样式添加!important

\*混合参数做为属性和设置计算后的属性值

.set(@attr,@val){

@{attr}:unit(@val/10,px);

}

### 移动基础

\* 像素

>> \* 设备的物理像素,屏幕的物理像素,任何设备的物理像素是固定的,例如电脑屏幕的分辨率是1366 X 768,表示在电脑屏幕上有1366 X 768个小格子,一个小格子代表一个物理像素

>> \* CSS像素,又称逻辑像素,是为WEB开发者创造出来的一个抽象层

>>> \* 只有在页面不缩放的情况下,一个CSS像素才等于一个物理像素

>>> \* 如果页面放大,根据放大的比例,多个物理像素合成一个CSS像素

>>> \* 如果页面缩小,根据缩小的比例,一个物理像素分成多个CSS像素

>>> \* 因为像素的合成,当网页放大后,CSS定义的像素并没有改变,但是视觉效果上变大了

\* DPI/PPI

> \* DPI（Dot Per Inch）每英寸的墨汁点，多用于印刷行业

> \* PPI（Pixels Per Inch）屏幕每英寸的物理像素数量，即单位英寸的物理像素密度

> \* 1 英寸(inch)=25.4毫米(millimetres)=2.54厘米

<img src="img/ppi.png">

\* PPI值越高,一个实际1英寸上容纳的物理像素就越多,就能够展示更多的画面细节,也就意味着画面更细腻

\* DPR

> \* 手机分辨率很高,但手机屏幕尺寸有限,所以可以让手机屏幕上的多个物理像素合成一个CSS像素来更好的显 示内容

> \* DPR(device Pixel Ratio) 设备像素比 = 设备物理像素 / CSS像素,这个DPR就决定了多少个物理像素合成一个CSS像素

> \* 以iPhone6为例

>> \* iPhone6 的分辨率为1334X750

>> \* iPhone6 的CSS像素为667X375

>> \* 所以iPhone6 的DPR为 1336/667 = 2,即2个物理像素对应一个css像素

>> \* DPR的设置是根据人体手臂的长度来设定的,也就是说将 1334X750的物理像素合成667X375的CSS像素,视觉效果是最好的

\* 视口 viewport

```html

<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=0">

```

>\* width,设置视口的宽度,表示把移动设备的宽缩放为多少个CSS像素,为一个正整数,或字符串"device-width",device-width表示缩放到设备合成后的CSS像素

>\* initial-scale 设置页面的初始缩放值,为一个数字,可以带小数

>\* minimum-scale 允许用户的最小缩放值,为一个数字,可以带小数

>\* maximum-scale 允许用户的最大缩放值,为一个数字,可以带小数

>\* height,设置视口的高度

>\* user-scalable,是否允许用户进行缩放,值为"no"或"yes", no 代表不允许,yes代表允许

>\* 这些属性可以同时使用，也可以单独使用或混合使用，多个属性同时使用时用逗号隔开

### 屏幕尺寸适配

\* 像素px,绝对单位,由于单位固定

> \* 无法做屏幕的适配

\* 百分比%,相对于父元素的宽高

> \* 可以实现布局的适配,但是字体不容易适配

\* em,当前元素字体的大小,1em=当前元素的font-size

> \* 可以根据不同的屏幕来设置元素的font-size,再利用em来适配,但是元素的font-size可以被继承,同时js和css经常会修改font-size,如果用em来适配的话容易造成混乱

> \* 注意:font-size最小值是12px;

\* 1vw=当前设备宽度的1%,vh=当前设备高度的1%

> \* 可以适配布局和文字,但兼容性不好

\* rem,根元素字体的大小,默认值是16px

> \* 可以根据不同的屏幕来设置根元素的font-size,再利用rem来适配

> \* 根元素的font-size需要动态设置,一般有JS和媒体查询两种方法来动态设置

### 响应式布局(Responsive)

\* 媒体类型 (media type)

> \* 媒体类型是从CSS2和HTML4开始,可以根据设备的类型来决定使用什么样式

> \* 使用方法一

```html

<link rel="stylesheet" media="print" href="css/print.css" >

<link rel="stylesheet" media="screen" href="css/screen.css" >

```

> \* 使用方法二

```html

<style type="text/css">

@media screen{

h1{

color: blue;

}

}

@media print{

h1{

color: red;

}

}

</style>

```

> \* 常见的媒体类型有(还有其他的但基本已被抛弃,所以在此不在列举)

>> screen 屏幕

>> print 打印机

>> all 所有

\* 媒体查询(media query)

> \* 媒体查询是由媒体类型(media type)和媒体特征(media features)共通决定使用什么样式

> \* 常见的媒体特征有

>> \* width, 浏览器窗口的宽度,可以有min/max前缀

>> \* height, 浏览器窗口的高度,可以有min/max前缀

>> \* device-width,分辨率宽度,可以有min/max前缀

>> \* device-height,分辨率高度,可以有min/max前缀

>> \* orientation,移动设备窗口的方向,portrait(纵向),landscape(横向)

> \* min/max的解释

>> \* (min-width:400px) 浏览器窗口的宽度最小是400px样式生效,换句话说是大于等于400px时样式生效

>> \* (max-width:800px) 浏览器窗口的宽度最大是800px样式生效,换句话说是小于等于800px时样式生效

> \* 使用方法一

```html

<link rel="stylesheet" href="css/blue.css">

<link rel="stylesheet" href="css/yellow.css" media="screen and (max-width: 500px)">

```

> \* 使用方法二

```html

<style>

@media screen and (min-width: 500px){

body{

background: yellow;

}

}

</style>

```

### 弹性布局(Flexible Box)

\* 把容器指定为弹性布局

> \* 块状元素 display: flex;

> \* 行内元素 display: inline-flex;

\* 指定了弹性布局的元素称为Flex容器(flex container),它的所有子元素自动成为容器成员,称为 Flex项目(flex item)

\* 容器属性

> \* flex-direction 决定项目的排列方向,有以下取值

>> \* row(默认值):水平方向,起点在左端

>> \* row-reverse:水平方向,起点在右端

>> \* column:垂直方向,起点在上沿

>> \* column-reverse:垂直方向,起点在下沿

> \* flex-wrap 决定项目一行排不下，如何换行,有以下取值

>> \* nowrap(默认值):不换行,元素的宽度按照指定款的的比例从新计算

>> \* wrap:换行,第一行在上方

>>> \* 换行后项目的垂直对齐方式由align-items/align-content来决定

>> \* wrap-reverse:换行,第一行在下方

> \* flex-flow:flex-direction属性和flex-wrap属性的简写形式,默认值为row nowrap。

> \* justify-content 决定项目的水平对齐方式,有以下取值

>> \* flex-start(默认值):左对齐

>> \* flex-end:右对齐

>> \* center:居中

>> \* space-between:两端对齐,项目之间的间隔都相等。

>> \* space-around:每个项目两侧的间隔相等。所以,项目之间的间隔比项目与边框的间隔大一倍。

> \* align-items/align-content 决定项目的垂直对齐方式,有以下取值

>> \* stretch(默认值):换行后,如果项目未设置高度或设为auto,将占满整个容器的高度

>> \* flex-start：起点对齐

>> \* flex-end：终点对齐

>> \* center：中点对齐

>> \* baseline: 项目的第一行文字的基线对齐

>> \* align-items/align-content的区别在于:

>>> \* align-item换行后会根据父容器的高度填充整个行

>>> \* align-content换行后按项目的的高度来填充行

\* 项目属性

> \* order 项目的排列顺序,数值越小,排列越靠前,默认为0

> \* flex-grow 项目的放大比例,默认为0,即如果存在剩余空间,也不放大

>> \* 如果所有项目的flex-grow属性都为1,则它们将等分剩余空间(如果有的话)

>> \* 如果一个项目的flex-grow属性为2,其他项目都为1,则前者占据的剩余空间将比其他项多一倍

> \* flex-shrink 项目的缩小比例,默认为1,即如果空间不足,该项目将缩小

>> \* 如果所有项目的flex-shrink属性都为1,当空间不足时,都将等比例缩小

>> \* 如果一个项目的flex-shrink属性为0,其他项目都为1,则空间不足时,前者不缩小

> \* flex-basis 定义了在分配多余空间之前,项目占据的大小,浏览器根据这个属性,计算是否有多余空间

>> \* 它的默认值为auto,即项目的本来大小

> \* flex 是flex-grow,flex-shrink,flex-basis的简写,默认值为0 1 auto,后两个属性可选

> \* align-self 允许单个项目有与其他项目不一样的对齐方式,可覆盖align-items属性