/\*

\*\*\*\*\*\* 元素视图属性

\* offsetWidth 水平方向 width + 左右padding + 左右border-width

\* offsetHeight 垂直方向 height + 上下padding + 上下border-width

\*

\* clientWidth 水平方向 width + 左右padding

\* clientHeight 垂直方向 height + 上下padding

\*

\* offsetTop 获取当前元素到 定位父节点 的top方向的距离

\* offsetLeft 获取当前元素到 定位父节点 的left方向的距离

\*

\* scrollWidth 元素内容真实的宽度，内容不超出盒子高度时为盒子的clientWidth

\* scrollHeight 元素内容真实的高度，内容不超出盒子高度时为盒子的clientHeight

\*

\*\*\*\*\*\* 元素视图属性结束

\*

\*\*\*\*\*\* Window视图属性（低版本IE浏览器[<IE9]不支持） 【自测包含滚动条，但网络教程都说不包含？？？】

\* innerWidth 浏览器窗口可视区宽度（不包括浏览器控制台、菜单栏、工具栏）

\* innerHeight 浏览器窗口可视区高度（不包括浏览器控制台、菜单栏、工具栏）

\* \*\*\*\*\* Window视图属性结束

\*

\*\*\*\*\*\* Document文档视图

\* （低版本IE的innerWidth、innerHeight的代替方案）

\* document.documentElement.clientWidth 浏览器窗口可视区宽度（不包括浏览器控制台、菜单栏、工具栏、滚动条）

\* document.documentElement.clientHeight 浏览器窗口可视区高度（不包括浏览器控制台、菜单栏、工具栏、滚动条）

\*

\* document.documentElement.offsetHeight 获取整个文档的高度（包含body的margin）

\* document.body.offsetHeight 获取整个文档的高度（不包含body的margin）

\*

\* document.documentElement.scrollTop 返回文档的滚动top方向的距离（当窗口发生滚动时值改变）

\* document.documentElement.scrollLeft 返回文档的滚动left方向的距离（当窗口发生滚动时值改变）

\*\*\*\*\*\* Document文档视图结束

\*

\*\*\*\*\*\* 元素方法

\* 1. getBoundingClientRect() 获取元素到body

\* bottom: 元素底边（包括border）到可视区最顶部的距离

\* left: 元素最左边（不包括border）到可视区最左边的距离

\* right: 元素最右边（包括border）到可视区最左边的距离

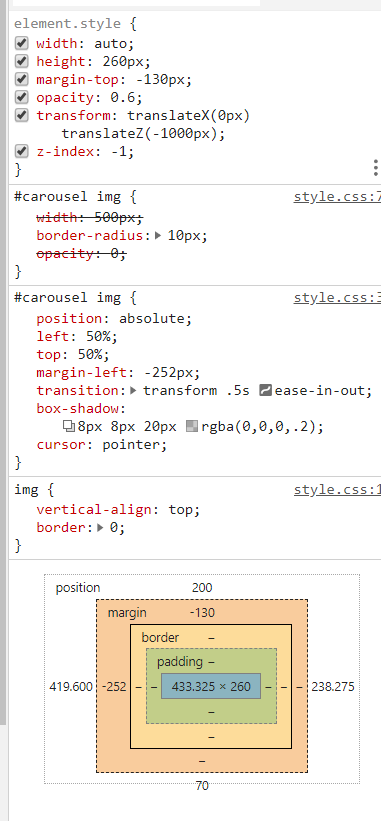
\* top: 元素顶边（不包括border）到可视区最顶部的距离

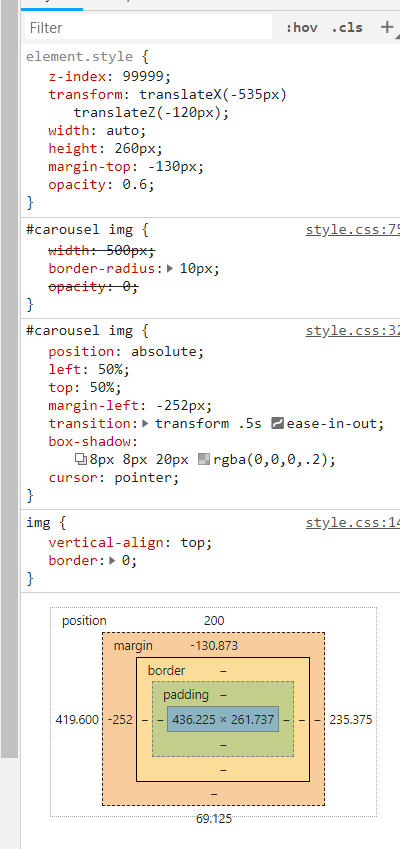
\* height: 元素的offsetHeight

\* width: 元素的offsetWidth

\* x: 元素左上角的x坐标

\* y: 元素左上角的y坐标





## 浏览器内核

渲染引擎Rendering Engine=布局引擎Layout Engine=排版引擎

=HTML解释器+CSS解释器+布局layout+网络等模块

|  |  |  |
| --- | --- | --- |
|  | 渲染引擎 | JS引擎 |
| IE | trident(<=IE10) | JScript(<IE9), Chakra(IE9+) |
| Edge | EdgeHTML | Chakra |
| Chrome | Webkit(很早很早以前),Chromium(扩展自Webkit), Blink | V8 |
| Safari | Webkit | JSCore/Nitro(4+) |
| Firefox | Gecko | SpiderMonkey(<3.0), TraceMonkey(<3/6), JaegerMonkey(4.0+) |
| Opera | Presto, blink |  |

之前JS引擎也被集成在内核中，但是JS引擎越来越独立，慢慢浏览器内核就只有渲染引擎了。

## 盒子模型

Box盒子是CSS布局的对象和基本单位

* Block-level Box 块级元素

display: block/list-item/table

* Inline-level Box 行内级元素

display: inline/inline-block/inline-table

* flex container 弹性容器

display: flex生成块级box-level弹性容器框

display: inline-flex生成行内级inline-level弹性容器框

* grid container 栅格容器

display: grid/inline-grid

每个块级元素生成一个块级盒参与BFC，每个行内元素生成一个行内级盒参与IFC

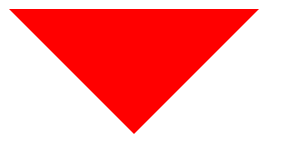
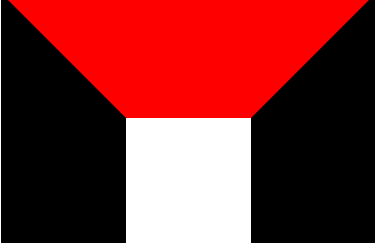
width

height height和width是内容区的宽度，不含paadding和其他的

padding

border

margin

* IE盒模型！它和别人不一样，它的height和width就包括了padding和border!
* body的margin默认8px
* background-color的颜色包括内容区和padding和border！border设置成dashed就可以看出来
* padding: 30px 20px 10px 5px 上右下左顺时针
* 想想怎么搞出来的
* background-image从 padding 边缘的左上角起到 border 的右下角边缘止。

background-clip: border-box; // 背景延伸到边框外沿（但是在边框之下）

background-clip: padding-box; // 边框下面没有背景，即背景延伸到内边距外沿。

background-clip: content-box; // 背景裁剪到内容区 (content-box) 外沿。

background image 的绘制中有两个因素决定了绘图区域

* background positioning area。background-origin 属性决定了这个相对定位位置，默认为 padding-box。所以默认的背景图片绘制是从 padding box 的左上顶点开始的。
* background painting area。background-clip 属性决定了绘制区间，默认为 border-box。所以在 background-repeat: repeat 的情况下：

## http协议

hypertext transfer protocol 超文本传输协议

* http协议无状态

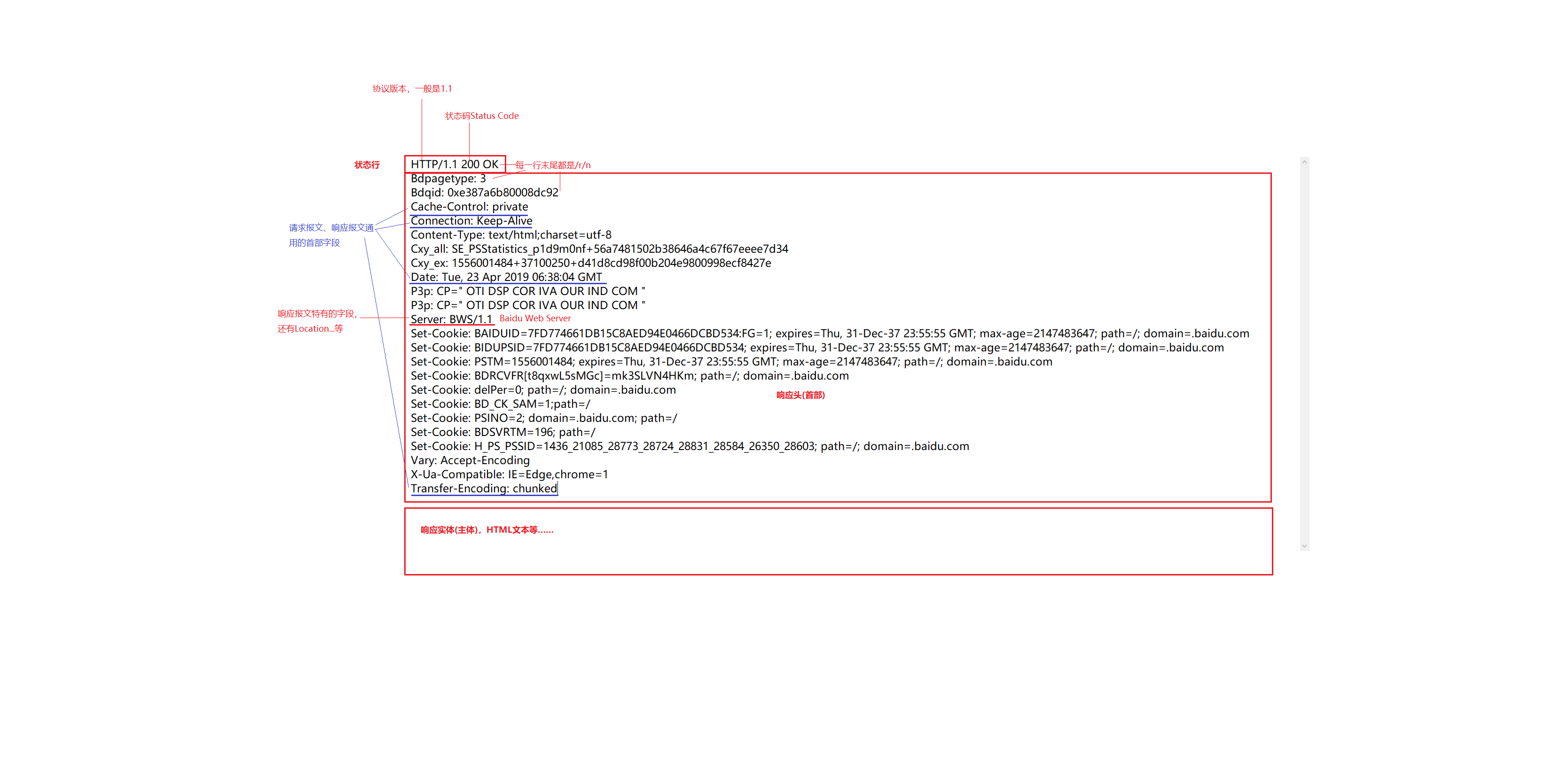
CR Carriage Return \r 回车

LF Linefeed \n 换行

CRLF \r\n 回车并换行

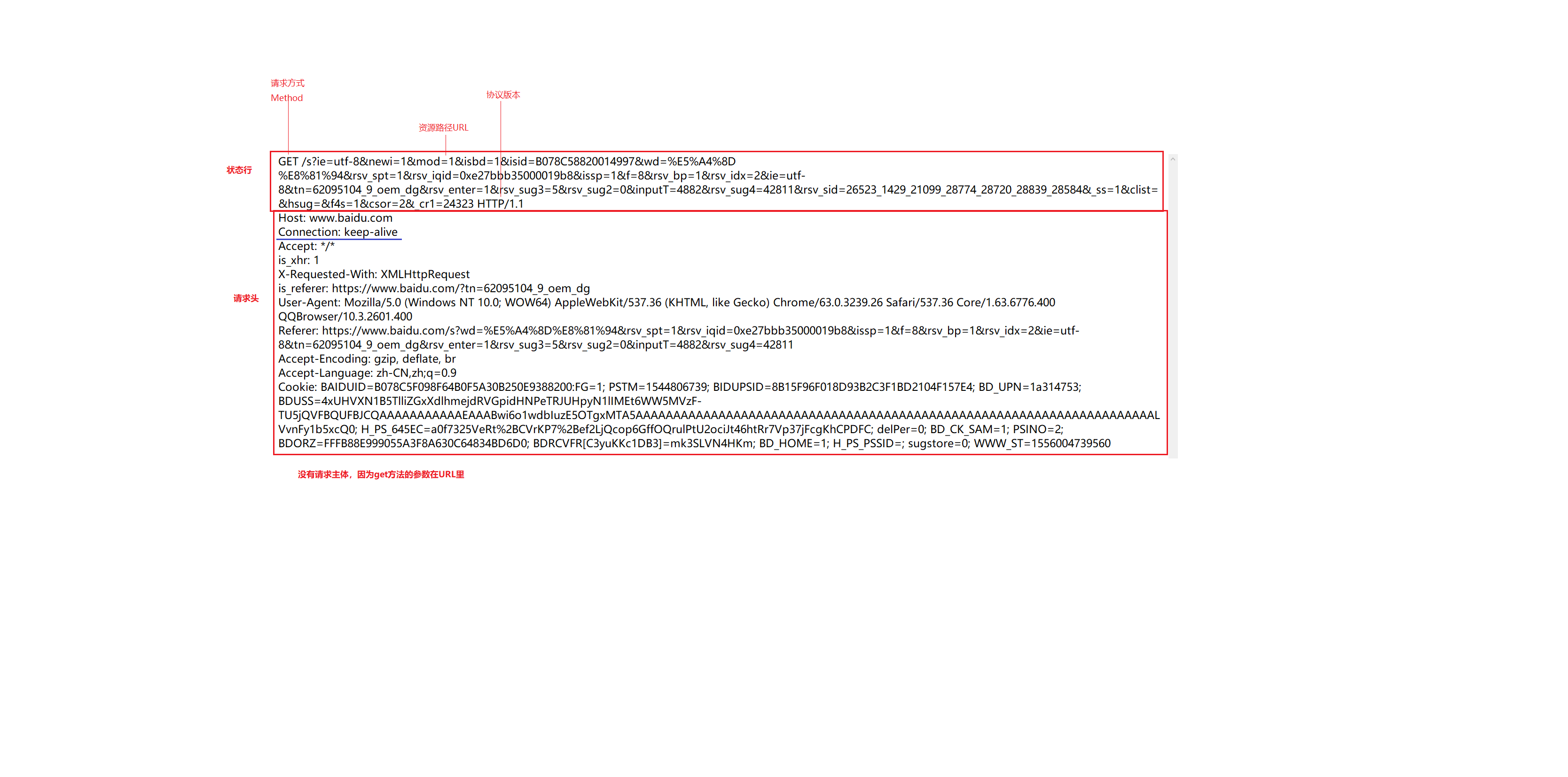
Windows采用两个字符换行，即CRLF

Unix/Linux/Mac OS X采用单个字符LF换行

响应报文

* 200---OK/请求已经正常处理完毕
* 301---/请求永久重定向
* 302---/请求临时重定向
* 304---/请求被重定向到客户端本地缓存
* 400---/客户端请求存在语法错误
* 401---/客户端请求没有经过授权
* 403---/客户端的请求被服务器拒绝，一般为客户端没有访问权限
* 404---/客户端请求的URL在服务端不存在
* 500---/服务端永久错误
* 503---/服务端发生临时错误

请求报文



一次完整的http事务

* 域名解析
* 浏览器先后在浏览器缓存、操作系统缓存、host文件、DNS服务器等找域名对应的IP
* 建立TCP连接

三次握手

* 1） Client首先发送一个连接试探，ACK=0 表示确认号无效，SYN = 1 表示这是一个连接请求或连接接受报文，同时表示这个数据报不能携带数据，seq = x 表示Client自己的初始序号（seq = 0 就代表这是第0号包），这时候Client进入syn\_sent状态，表示客户端等待服务器的回复
* 2） Server监听到连接请求报文后，如同意建立连接，则向Client发送确认。TCP报文首部中的SYN 和 ACK都置1 ，ack = x + 1表示期望收到对方下一个报文段的第一个数据字节序号是x+1，同时表明x为止的所有数据都已正确收到（ack=1其实是ack=0+1,也就是期望客户端的第1个包），seq = y 表示Server 自己的初始序号（seq=0就代表这是服务器这边发出的第0号包）。这时服务器进入syn\_rcvd，表示服务器已经收到Client的连接请求，等待client的确认。
* 3） Client收到确认后还需再次发送确认，同时携带要发送给Server的数据。ACK 置1 表示确认号ack= y + 1 有效（代表期望收到服务器的第1个包），Client自己的序号seq= x + 1（表示这就是我的第1个包，相对于第0个包来说的），一旦收到Client的确认之后，这个TCP连接就进入Established状态，就可以发起http请求了
* 浏览器发送请求
* Web服务器应答
* Web服务器关闭连接

## OSI七层模型

open system interconnect开放式系统互联

物理层、数据链路层、网络层、传输层、会话层、表示层、应用层

TCP/IP 五层模型

物理层、数据链路层、网络层、传输层、应用层

应用层

http ftp DNS

传输层

TCP UDP

网络层

IP 路由器

链路层

物理层

# HTML

## Attribute和Property

attribute和property的区别

attribute 由 HTML 定义。property 由 DOM (Document Object Model) 定义。

* getAttribute, setAttribute是获取、更改attribute的方法
* eleName.propertyName来获取、修改property

attribute的值只能是字符串。在标签里边写的东西都是给attribute赋值

* 对于class、id、name

<button class="hello">nihao</button>

var a=document.getElementsByTagName('button')[0]

a.className

a.getAttribute('class')

a.classList.add('shijie')

a.setAttribute('class', 'temp')

不能a.class获取class属性，回避JavaScript关键字。用a.className查看

class的attribute和property，不管其中的一个怎么改，另一个也跟着改，这俩始终一样。id和name也是如此

* 对于value

<input type="text" value="Bob">

value 这个 property 被初始化为 “Bob”。

当在输入框中输入 “Sally” 时，DOM 元素的 value 这个 property 变成了 “Sally”。 但是value 这个 attribute 保持不变。

var a=document.getElementsByTagName('input')[0]

a.value='Alice' //value property被改变，输入框内容变为Alice

a.getAttribute('value') //返回Bob

* 对于disabled、hidden

一个button，默认情况下disabled属性值为false，可以点击。

<button disabled>b</button>

添加 disabled 这个 attribute，将 disabled 这个 property 初始化为 true，按钮被禁用。不论disabled attribute的值为"false"、"true"还是空串""，disable property都是true。只有用

a.removeAttribute('disabled')

删掉disabled attribute，disabled property就变成了false，可以点击。

<button disabled="false"></button>

上面这种写法是错误的，或者说没用，即使写成这样仍被禁用

a.disabled=false

控制台输入这一句按钮也能用，同时disabled attribute不改变，可以是"niubi", "true"等，说明真正决定按钮能不能按的是disable property，而不是disable attribute。

* 有些 HTML attribute 没有对应的 property，如 colspan。
* 有些 DOM property 没有对应的 attribute，如 textContent。

## 表单

<form>

First name:<br>

<input type="text" name="firstname" readonly="readonly"><br>

//单选框

<input type="text" name="sex" placeholder="man">Male<br>

<input type="text" name="name1" disabled="disabled" >//禁用的input元素

</form>

<form>

<input type="radio" name="sex" value="male" checked>Male

<br>

<input type="radio" name="sex" value="female">Female

</form>

<input type="text" name="n1" oninput="f1()">

<script type="text/javascript">

function f1(){

alert('fff')

}

</script>

submit不跳转的方法：

form标签加属性target="rfFrame"

form外

<iframe id="rfFrame" name="rfFrame" src="about:blank" style="display:none;"></iframe>

* input可以直接设置width和height

placeholder 属性提供提示信息。该提示会在输入字段为空时显示，并会在字段获得焦点时消失。

# CSS

## >+~选择器

* >子选择器
* +相邻选择器

h1+p {

background-color:Red;

}

<h1>11111111111111</h1>

<p>22222222222222</p>

选中这个p标签。紧接着h1之后的p

* ~匹配选择器/兄弟选择符

h1~p {

background-color:Red;

}

<h1>11111111111111</h1>

<p>22222222222222</p>

<p>22222222222222</p>

<p>22222222222222</p>

选中这些个所有的P标签。必须是h1之后的，所有同级元素

## 权重

行间>内联>外部

<div style=""></div>

<style type="text/css">

div{

}

</style>

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

* background-color: #000 !important; 权重最高

## nth-of-type()

* p:nth-of-type(2){

background:#ff0000;

}

规定属于其父元素的第二个 p 元素的每个 p

* p:nth-of-type(odd){

background:#ff0000;

}

p:nth-of-type(even){

background:#0000ff;

}

奇数p和偶数p是两种不同的颜色

* 使用公式 (an + b)

第an+b(n=0,1,2,3…)个p被选中。从第一个p开始

p:nth-of-type(3n+0){

background:#ff0000;

}

指定下标是 3 的倍数的所有 p 元素的背景色

ul>li:nth-child() span

1.内联元素不能包含块元素，它只能包含其他的内联元素

2.一些块元素不可以包含另一些块元素

p中不要包含任何其他块级元素

## list-style

ul{

list-style:square inside url('/i/arrow.gif');

}

就是这三个属性一起写

ul 和li都能

1. list-style-type
2. list-style-position
3. list-style-image

1.list-style-type属性

|  |  |
| --- | --- |
| none | 无标记。 |
| disc | 默认。标记是实心圆。 |
| circle | 标记是空心圆。 |
| square | 标记是实心方块。 |
| decimal | 标记是数字。 |
| decimal-leading-zero | 0开头的数字标记。(01, 02, 03, 等。) |
| lower-roman | 小写罗马数字(i, ii, iii, iv, v, 等。) |
| upper-roman | 大写罗马数字(I, II, III, IV, V, 等。) |
| lower-alpha | 小写英文字母The marker is lower-alpha (a, b, c, d, e, 等。) |
| upper-alpha | 大写英文字母The marker is upper-alpha (A, B, C, D, E, 等。) |
| lower-greek | 小写希腊字母(alpha, beta, gamma, 等。) |
| lower-latin | 小写拉丁字母(a, b, c, d, e, 等。) |
| upper-latin | 大写拉丁字母(A, B, C, D, E, 等。) |
| hebrew | 传统的希伯来编号方式 |
| armenian | 传统的亚美尼亚编号方式 |
| georgian | 传统的乔治亚编号方式(an, ban, gan, 等。) |
| cjk-ideographic | 简单的表意数字 |

2）list-style-position

|  |  |
| --- | --- |
| inside | 列表项目标记放置在文本以内，且环绕文本根据标记对齐。 |
| outside | 默认值。保持标记位于文本的左侧。列表项目标记放置在文本以外，且环绕文本不根据标记对齐。 |

**display:none**隐藏，在文档布局中不再给它分配空间，它各边的元素会合拢，就当他从来不存在。

visibility:hidden隐藏，但是在文档布局中仍保留原来的空间。

* li标签里边有个span,而且span是inline-block的话，span的line-height会继承li的line-height, 如果此时li有padding，那么就会发生问题

padding

**li**

**span**

line-height

变长了！

解决方法:span的line-height属性设置为normal

* 子元素浮动之后父元素的高度无法被撑起，解决方法

父元素加属性overflow:hidden

* 实战学的标准开头

1. \*{
2. **padding**: 0px;
3. **margin**: 0px;
4. **text-decoration**: none;  //取消a标签下划线
5. **list-style**: none;
6. }
7. html, body{
8. **width**: 100%;
9. }

## CSS font属性

至少要指定字体大小font-size和字体系列font-family

未设置的属性会使用其默认值

* font-style 规定字体样式

normal

italic 斜体

oblique 倾斜

如果一个字体没有倾斜字体样式，则italic无效，oblique有效，拉斜了

* font-variant 规定字体异体

font-variant设置小型大写字母的字体显示文本，这意味着所有的小写字母均会被转换为大写，但是所有使用小型大写字体的字母与其余文本相比，其字体尺寸更小

取值：normal small-caps

* font-weight 规定字体粗细

normal bold bolder lighter

100 200 300 400 500 600 700 800 900

定义由粗到细的字符。400 等同于 normal，而 700 等同于 bold。

* font-size/line-height 规定字体尺寸和行高
* font-family 规定字体系列

font:14px/1.3 'Segoe UI',Arial, sans-serif;

有逗号的值都是并列；网上的一般都只设置size和family

不设置html的css属性时，body就是根节点，整个页面的属性就是body的属性

设置html的css属性后，根节点变为html

min-height

可能取值：

length 定义元素的最小高度。默认值是 0。

% 定义基于包含它的块级对象的百分比最小高度

## shadow

* box-shadow: 10px 10px 5px #888888 inset;

内阴影 外阴影

h-shadow 必需。水平阴影的位置。允许负值。

v-shadow 必需。垂直阴影的位置。允许负值。

blur 可选。模糊距离。上面的5px

spread 可选。阴影的尺寸。

color 可选。阴影的颜色

inset 可选。将外部阴影 (outset) 改为内部阴影。默认outset

* text-shadow: h-shadow v-shadow blur color;

text-shadow: 5px, 5px, 3px, #000;

文字阴影

可以设置多个阴影，用逗号隔开

## transition

transition:all 0.25s linear;

div{

width:100px;

height:100px;

transition:width 2s;

-moz-transition:width 2s; /\* Firefox 4 \*/

-webkit-transition:width 2s; /\* Safari and Chrome \*/

-o-transition:width 2s; /\* Opera \*/

}

div:hover

{

width:300px;

}

* transition-property 规定设置过渡效果的 CSS 属性的名称
* transition-duration 规定完成过渡效果需要多少秒或毫秒。必须设置
* transition-timing-function 规定速度效果的速度曲线
* linear 匀速
* ease 慢速开始和结束
* ease-in 先慢后快（等于 cubic-bezier(0.42,0,1,1)）。
* ease-out 先快后慢（等于 cubic-bezier(0,0,0.58,1)）。
* ease-in-out 慢速开始和结束（等于 cubic-bezier(0.42,0,0.58,1)）。
* transition-delay 定义过渡效果何时开始

默认值transition: all 0 ease 0

这玩意就是跟:hover :active :focus配合

transition: transform 0.1s, opacity 0.3s;

.moved {

transition: transform 2s;

}

.moved2{

transform: perspective(500px) translateZ(200px);

}

m.addEventListener('click', (ev)=>{

let a=ev.target;

a.classList.add('moved2');

});

## transform

transform: translate(10px, 3px)

* translate(10px, 3px) 平移水平向右10px，垂直向下3px

translateX(-50%)

translateY(10%)

* scale(0.7, 1.5) 缩放比例
* rotate(30deg) 顺时针旋转角度
* rotateX(30deg) 沿着x轴3D旋转

rotateY(angle)

rotateZ(angle)

* skew(x-angle,y-angle) 定义沿着 X 和 Y 轴的 2D 倾斜转换

## after伪元素

p:after

{

content:"台词：-";

}

* :before和:after伪元素在页面中生成的元素在缺省情况下是“内联(inline)”
* content中的内容
* a:after { content: "↗"; }
* attr() 调用当前元素的属性，将图片的 Alt 提示文字或者链接的 Href 地址显示出来。

a:after { content:"(" attr(href) ")"; }

* url() uri() – 用于引用媒体文件。

h1::before { content: url(logo.png); }

* counter() – 调用计数器，可以不使用列表元素实现序号功能。

h2:before {

counter-increment: chapter;

content: "Chapter " counter(chapter) ". "

}

a:hover::before, a:hover::after { position: absolute; }

a:hover::before { content: "\5B"; left: -20px; }

a:hover::after { content: "\5D"; right: -20px; }

鼠标悬浮时添加左右方括号

* 大部分例子把before和after写成position: absolute，然后调整left、right、top或者bottom。absolute是相对于最近的有定位的父级元素定位的，所以必须把原来的元素设置成position: relative。这样看来，其实a和a::before是父子关系。嗯，就是父子关系

清除浮动

1.

.clearfix:after{content: "";display: block;clear: both;} <!--clearfix是预清浮动的父级元素-->

.clearfix{\*zoom:1;}

2.空div clear:both

鼠标指针变成

cursor:pointer; 一只手，指向连接类型

cursor:wait; 程序正忙

text 文本

default

a:link

a:visited

a:hover

a:active

未访问连接、已访问连接、悬浮、鼠标按下

## 渐变Gradients

* 线性渐变Linear Gradients

#grad1 {

height: 200px;

background: -webkit-linear-gradient(red, blue); /\* Safari 5.1 - 6.0 \*/

background: -o-linear-gradient(red, blue); /\* Opera 11.1 - 12.0 \*/

background: -moz-linear-gradient(red, blue); /\* Firefox 3.6 - 15 \*/

background: linear-gradient(red, blue); /\* 标准的语法（必须放在最后） \*/

}

/\*默认从上到下\*/

* background: linear-gradient(to right, red , blue);、

从左到右

* background: linear-gradient(to bottom right, red , blue);

从左上角到右下角

* background: linear-gradient(red, green, blue);

多个颜色节点均匀分布

* background: linear-gradient(to right, rgba(255,0,0,0.2), rgba(255,255,0,1));

最好使用透明度，否则看的有点难受

* background: repeating-linear-gradient(red, yellow 10%, green 40%);

效果：

green在40%的位置，red在一开始，yellow在10%的位置。后面重复上面的

* background: linear-gradient(90deg, #ffc700 0%, #e91e1e 100%);
* 径向渐变Radial Gradients

#grad {

background: -webkit-radial-gradient(red, green, blue); /\* Safari 5.1 - 6.0 \*/

background: -o-radial-gradient(red, green, blue); /\* Opera 11.6 - 12.0 \*/

background: -moz-radial-gradient(red, green, blue); /\* Firefox 3.6 - 15 \*/

background: radial-gradient(red, green, blue); /\* 标准的语法 \*/

}

最中间红，最外边蓝，一层一层向外变换

* background: radial-gradient(circle, red, yellow, green)

circle 表示圆形，ellipse 表示椭圆形。默认值是 ellipse

* background: radial-gradient(60% 55%, closest-side,blue,green,yellow,black);

closest-side

farthest-side

closest-corner

farthest-corner

我又看到linear-gradient写在background-image里边的

background: linear-gradient(160deg, red, yellow);

-webkit-background-clip: text;

什么效果你懂得

## animation

div{

**animation**: nihao 5s linear infinite alternate;

}

**@keyframes** nihao{

from{

background: red

}

to{

background: green

}

}

from to等同于0% 100%

@keyframes myfirst

{

0% {background: red; left:0px; top:0px;}

50% {background: blue; left:200px; top:200px;}

100% {background: red; left:0px; top:0px;}

}

background是纯色可以渐变，但是background是gradient的话就成了逐帧变换。可以渐变边框半径

* animation-name 动画名称
* animation-duration 一个周期所花费的秒或毫秒
* animation-timing-function 速度曲线。默认是 "ease".先快后慢
* linear
* ease 低速开始，然后加快，在结束前变慢
* ease-in低速开始
* ease-out 低速结束
* ease-in-out 动画以低速开始和结束。
* animation-delay 延迟
* animation-iteration-count 次数
* animation-direction 下一周期是否逆向播放。默认是 "normal"
* normal
* alternate 轮流反向播放。
* animation-play-state 动画是否正在运行或暂停

animation: nihao 2s linear infinite alternate forwards;

@keyframes nihao {

0%,

100% {border-radius: 63% 37% 54% 46% / 55% 48% 52% 45%;}

14% {border-radius: 40% 60% 54% 46% / 49% 60% 40% 51%;}

28% {border-radius: 54% 46% 38% 62% / 49% 70% 30% 51%;}

42% {border-radius: 61% 39% 55% 45% / 61% 38% 62% 39%;}

56% {border-radius: 61% 39% 67% 33% / 70% 50% 50% 30%;}

70% { border-radius: 50% 50% 34% 66% / 56% 68% 32% 44%;}

84% {border-radius: 46% 54% 50% 50% / 35% 61% 39% 65%;}

}

这段代码很强，边框就像一个~~卵蛋~~气泡一样动个不停

* user-select

.box{

-webkit-user-select: none;

-moz-user-select: none;

-ms-user-select: none;

user-select: none;

}

是否可以被选中

可能取值：

* none:文本不能被选择
* text:可以选择文本
* all：当所有内容作为一个整体时可以被选择。如果双击或者在上下文上点击子元素，那么被选择的部分将是以该子元素向上回溯的最高祖先元素。
* Element:可以选择文本，但选择范围受元素边界的约束
* outline: none;

border外边一圈轮廓

* text-indent:50px;

首行缩进

max-width:

max-height:

10%: 定义基于包含它的块级对象的百分比最大宽度

100px: 最大宽度值

## background

* background-position:

默认值：0% 0%。

* top left

top center

top right

center left

center center

center right

bottom left

bottom center

bottom right

如果仅规定了一个关键词，那么第二个值将是"center"

* x% y%

第一个值是水平位置，第二个值是垂直位置。

左上角是 0% 0%。右下角是 100% 100%。

如果仅规定了一个值，另一个值将是 50%。

* xpos ypos

第一个值是水平位置，第二个值是垂直位置。

左上角是 0 0。单位是像素 (0px 0px) 或任何其他的 CSS 单位。

如果仅规定了一个值，另一个值将是50%。

可以混合使用 % 和 position 值。

* background-attachment: fixed;

默认值scroll，背景图片随滚动条滚动

fixed不随屏幕滚动条滚动

* background-size:80px 60px;
* length
* 80% 20%

以父元素的百分比来设置背景图像的宽度和高度

如果只设置一个值，则第二个值会被设置为 "auto"，按比例缩放

* cover

把背景图像扩展至足够大，以使背景图像完全覆盖背景区域。按比例缩放的

背景图像的某些部分也许无法显示在背景定位区域中。

* contain

把图像图像扩展至最大尺寸，以使其宽度和高度完全适应内容区域，保证全部显示

我觉得这个很好用，按比例缩放，还尽量充满容器

* background-repeat: repeat-y;

repeat-x

no-repeat

让图片铺满整个屏幕，网上很多教程都是给body设置背景图片，这我觉得这样不好，既不符合元素的逻辑关系（铺满屏幕的照片在所有元素之上），又是在body上做文章，不具有可移植性

我这样实现：在body末尾加一个div.popBigPhoto

.popBigPhoto{

width: 100%;

height: 100%;

background-image: url('../images/pic1.jpg');

background-attachment: fixed;

background-position: center center;

background-size: contain;

background-color: rgba(0, 0, 0, 0.6);/\*设置背景灰色蒙版\*/

background-repeat: no-repeat;

position: fixed;

z-index: 100;

top: 0;

left: 0;

display: none;

}

document.querySelector("p");

参数是CSS选择器，只选择符合条件的第一个，如果要选择所有，用querySelectorAll()方法

获取文档中 class="example" 的第一个 <p> 元素:

document.querySelector("p.example");

获取文档中有 "target" 属性的第一个 <a> 元素：

document.querySelector("a[target]");

width: 50vw;

浏览器100vw表示的就是浏览器的视窗宽度(Viewport)

## BFC块级格式化上下文

Block formatting context

box是CSS布局的基本单位

* block-level box

display 属性为 block, list-item, table 的元素，会生成 block-level box。并且参与 block fomatting context；

* inline-level box

display 属性为 inline, inline-block, inline-table 的元素，会生成 inline-level box。并且参与 inline formatting context；

BFC(Block formatting context)直译为"块级格式化上下文"。它是一个独立的渲染区域，只有Block-level box参与，它规定了内部的Block-level Box如何布局，并且与这个区域外部毫不相干。

BFC的一个最重要的效果是，让处于BFC内部的元素与外部的元素相互隔离，使内外元素的定位不会相互影响。

BFC布局规则：

* 内部的Box会在垂直方向，一个接一个地放置。
* Box垂直方向的距离由margin决定。属于同一个BFC的两个相邻Box的margin会发生重叠
* BFC的区域不会与float元素重叠。如果上面一个div是float，下面一个div是overflow:hidden转成BFC，这两个div不会重叠（但是如果下面的div是position：fixed就没有这种效果了，也不知道为啥，咱也不敢问）
* BFC容器里面的子元素不会影响到外面的元素。
* 计算BFC的高度时，浮动元素也参与计算。一般的div不计算。所以加上overflow:hidden就能撑开高度

生成BFC

* 根元素
* float: left/right
* position: absolute/fixed
* display:flex????????display: inline-flex
* display: inline-block?????
* overflow: hidden

单独一个div不是BFC

## filter

div{

filter:blur(4px);

}

* blur(4px) 高斯模糊
* brightness(35%) 亮度。小数或百分数表示。1的时候是原图。可以大于1.
* contrast(0.3) 对比度
* grayscale(0.3) 灰度图像。默认1，就是黑白图像
* drop-shadow(3px 5px pink) 外阴影。参数同box-shadow，除了不能加inset
* hue-rotate(180deg) 色相旋转
* invert(100%) 反转
* opacity(0.3) 透明度
* saturate(0.5) 饱和度

## 层叠顺序Stacking Level/Order

垂直于电脑屏幕的堆叠顺序，由低到高

* 背景和边框
* 负的z-index
* 正常流式布局，无position定位的元素
* 浮动元素float
* inline-block元素
* z-index:0
* 正的z-index

.float {

float: left;

margin-right: -50px;

}

.inline-block {

display: inline-block;

}

<div class="inline-block">#divA display:inline-block</div>

<div class="float"> #divB float:left</div>

所以inline-block元素是肯定摞在float元素上方的，不管他俩顺序如何

特定的属性会使元素形成**Stacking Context堆叠上下文**

* 根元素 (HTML)
* z-index 值不为 "auto"的 绝对/相对定位
* 一个 z-index 值不为 "auto"的 flex 项目 (flex item)，即：父元素 display: flex|inline-flex
* opacity 属性值小于 1 的元素
* transform 属性值不为 "none"的元素
* mix-blend-mode 属性值不为 "normal"的元素
* filter值不为“none”的元素
* perspective值不为“none”的元素
* isolation 属性被设置为 "isolate"的元素
* position: fixed
* 在 will-change 中指定了任意 CSS 属性，即便你没有直接指定这些属性的值

如果上面代码中，inline-block和.float都增加了opacity:0.9，则生成了Stacking Context，谁先写谁被压在下面

## -webkit-box-reflect

倒影|镜像

只有webkit内核的浏览器可以，qq浏览器可以，edge不行

-webkit-box-reflect: left;

left,right,above,below。分别是左面，右边，上，下的镜像

div::after {

content: "";

position: absolute;

top: 100%;

left: 0;

right: 0;

bottom: -100%;

background-image: inherit;

transform: rotateX(180deg);

}

这段代码很牛逼，实现了倒影效果，而且对于不同的图片不需要改代码。

## :root :target :empty :not伪类选择器

:root{

background-color: #F40;

}

匹配文档树的根元素，即<html>

div:empty{

display:none;

}

没有子元素的元素，文本也算子元素。比如<div> </div>，中间有个空格，div:empty都不能选中。

:target伪类，用于选取当前活动的目标元素。URL 末尾带有锚名称 #，就可以指向文档内某个具体的元素。这个被链接的元素就是目标元素(target element)。它需要一个 id 去匹配文档中的 target

<div id="n1">feafea</div>

<a href="#n1">nihaoshijie</a>

这段代码。点击a以后，url变成了<file:///E:/tempDirectory/a.html#n1>，这时，如果有这段

#n1:target{

background-color: red;

}

则n1的背景颜色变成red。代表n1被链接以后发生的变化

## box-sizing

* box-sizing: content-box;

默认。width和height不包括padding和border

* box-sizing: padding-box;

width包括padding

* box-sizing: border-box;

width包括border和padding

# HTML5

**dataset**属性存取data-\*自定义属性的值

* dataset 属性是HTML5 JavaScript API的一部分，用来返回一个所有选择元素 data- 属性的DOMStringMap对象
* 使用这种方法时，去掉data- 前缀
* data- 属性名如果包含了连字符，例如data-date-of-birth ，连字符将被去掉，并转换为驼峰式的命名dataOfBirth

<div id="user" data-id="123" data-name="helloWorld" data-date-of-birth>你好世界</div>

var a = document.querySelector('#user');

a返回

DOMStringMap {

dateOfBirth:""

id:"123"

name:"helloWorld"

\_\_proto\_\_:DOMStringMap

}

* dataset.id

返回'123'

* dataset.dataOfBirth

返回""

* 设置data-date-of-birth的值.

a.dataset.dateOfBirth = '1985-01-05';

* 增加

a.dataset.attr1='15522'

* 删除

delete a.dataset.id

配合querySelectorAll使用更佳

选择所有包含 'data-flowering' 属性的元素

document . querySelectorAll ( '[data-flowering]' ) ;

选择所有包含 'data-text-colour' 属性值为red的元素

document . querySelectorAll ( '[data-text-colour="red"]' )

# 设计模式

## 创建型模式：工厂模式

简单工厂

public class aaa {

public static void main(String[] args) {

CarFactory.createCar("trunk").run();

}

}

interface Car{

public void run();

}

class NormalCar implements Car{

@Override

public void run() {

System.out.println("normal car runs");

}

}

class Trunk implements Car{

@Override

public void run() {

System.out.println("trunk runs");

}

}

class CarFactory{

public static Car createCar(String carType) {

if(carType.equals("normal")) {

return new NormalCar();

}else if(carType.equals("trunk")) {

return new Trunk();

}

throw new IllegalArgumentException("请输入车类型");

}

}

可扩展的工厂

public class aaa {

public static void main(String[] args) {

CarFactory ccc=new TrunkFac();

ccc.createCar().run();

}

}

interface Car{

void run();

}

class NormalCar implements Car{

@Override

public void run() {

System.out.println("normal car runs");

}

}

class Trunk implements Car{

@Override

public void run() {

System.out.println("trunk runs");

}

}

interface CarFactory{

Car createCar();

}

class NormalCarFac implements CarFactory{

@Override

public Car createCar() {

return new NormalCar();

}

}

class TrunkFac implements CarFactory{

@Override

public Car createCar() {

return new Trunk();

}

}

想生产别的汽车创建别的类实现CarFactory接口就行了

## 开闭原则

就是不要出现这样的代码

if(a==1){

}else if(a==2){

}else if(a==3){

}

而是设计一个接口，然后有不同的实现

这样就 对扩展开放，对修改封闭