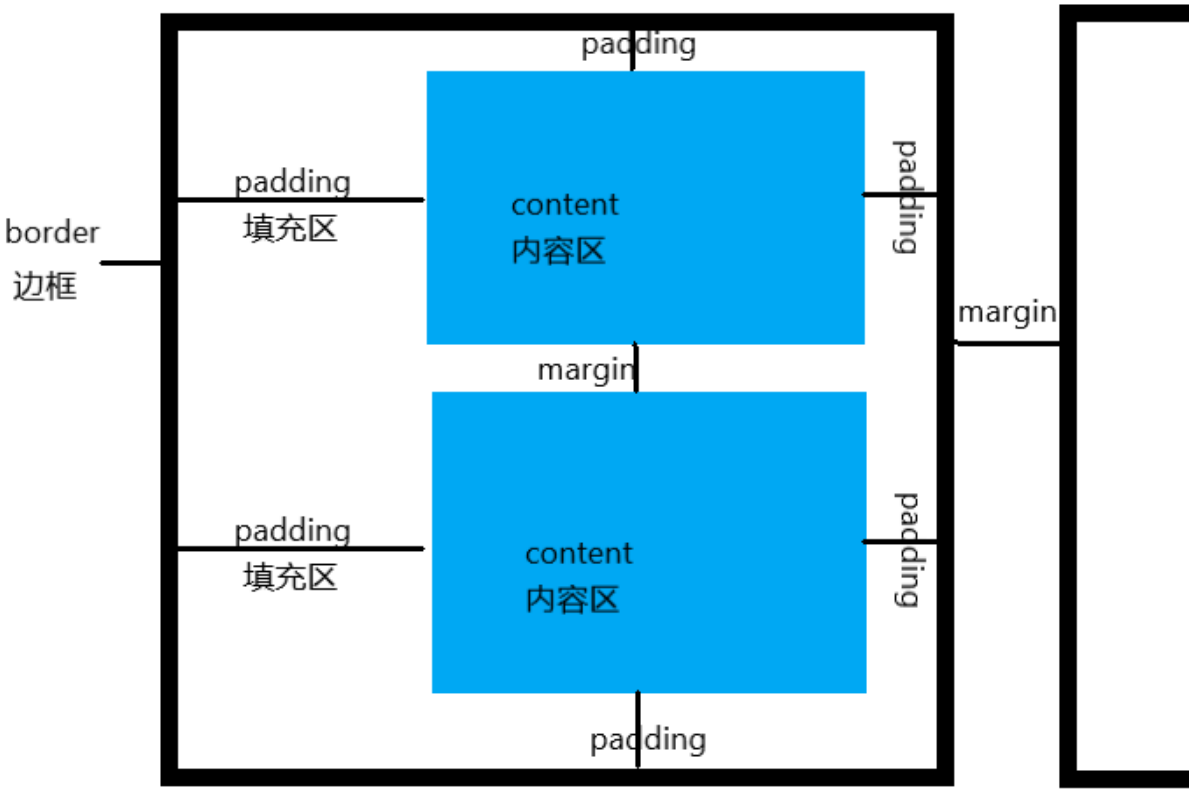
CSS部分

• 什么是盒模型

定义：盒模型规定网页元素之间怎么显示，盒模型包括（content(内容)，border(边框)，padding（填充区），margin(外边界区)）



• 行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？行内元素和块级元素有什么区别？

1. 行内元素： a , span , em , img , input , select , strong
2. 块级元素：div , p , ul , li , ol , dl , dt , dd , h1...h6
3. 空元素：没有内容的元素 img , hr , br , img , link , meta
4. 行内和块级元素的区别：

> - 行内元素是在同一排水平方向排列，块级是垂直方向排列
> - 块级元素可以包含行内元素，行内元素不能包含块级元素，只能包含文本或其他行内元素

> - 行内元素设置width，height，margin上下，padding上下无效，可设置line-height

简述src和href的区别

1. href表示超文本引用，用来建立当前连接与文档之间的连接，通常用于link，a 标签； src指向外部资源的地址，引入文件，主要用于iframe， img，script等标签
2. 检测到href链接，不会停止当前文档的处理，并行加载文档及css； 检测到src时，会暂停浏览器的渲染，直到加载完毕

什么是css Hack?

定义：由于不同的浏览器，不同浏览器对css的解析不同，导致页面渲染得不一样，这时就需要针对不同的浏览器去编写不同的css， hack本意为修改，css hack表示浏览器兼容 实例：ie6能识别'_'和'* '，ie7能识别' * '，而火狐两个都不能识别

什么叫优雅降级和渐进增强

1. 优雅降级：一开始构建完整的功能，到后期再兼容低版本的浏览器
2. 渐进增强：先满足最基本的功能，满足低版本的浏览器，然后再针对高浏览器进行交互及效果，更好的用户体验

区别：

> - 优雅降级从复杂开始，并试图减少用户体验的供给
> - 渐进增强从最基本开始，根据需求不同，增加不同的功能及交互，适应未来环境的需要
> - 降级意味功能衰退往回看，增强意味着往未来看，有着更好的用户体验，并保证根基处在安全地带

px和em的区别

px表示像素，是一个比较精确的值，但浏览器的缩放，会打破原本的布局，这时可以用em定义字体的大小，em是相对父元素的，1em等于一倍父元素的大小，但em存在一个问题，设置任何元素时，都有可能需要他的元素大小，比较麻烦，而rem是相对于根元素HTML的，比较固定

Http的状态码有哪些

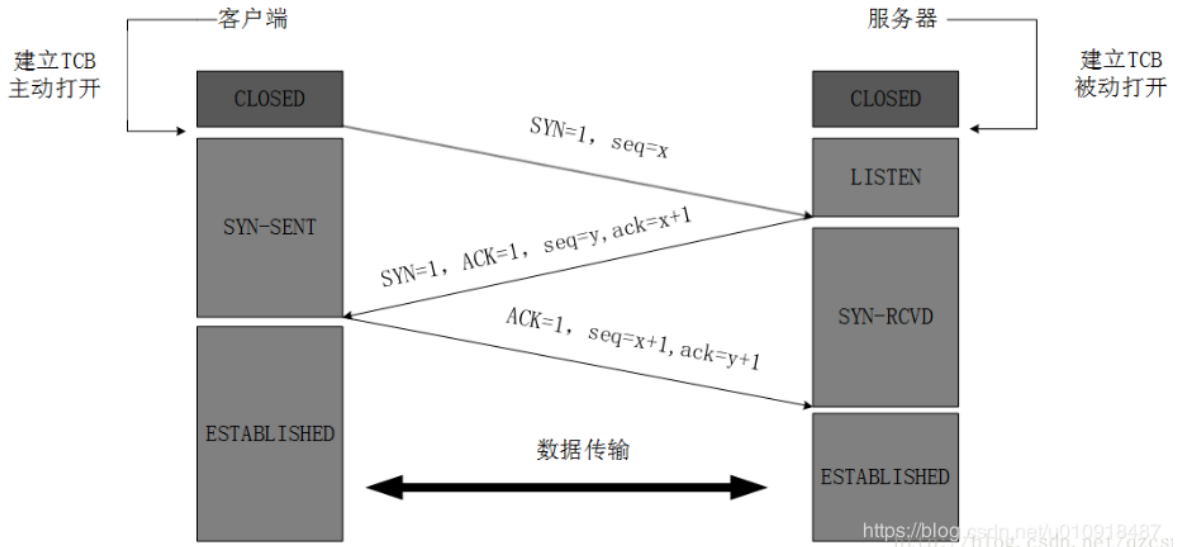
1. 200 请求成功
2. 301 网页或者资源被永久转移到其他url
3. 404 请求的网页或者地址不存在
4. 500 内部服务器错误
5. 401 请求用户信息

一次完整的HTTP事务是怎么一个过程

1. 域名解析

2. 发起TCP 3次握手
3. 建立TCP连接后发起http请求
4. 服务器响应http请求，浏览器得到HTML代码
5. 浏览器解析HTML代码，并请求HTML代码中的资源
6. 浏览器对页面进行渲染并呈现给用户

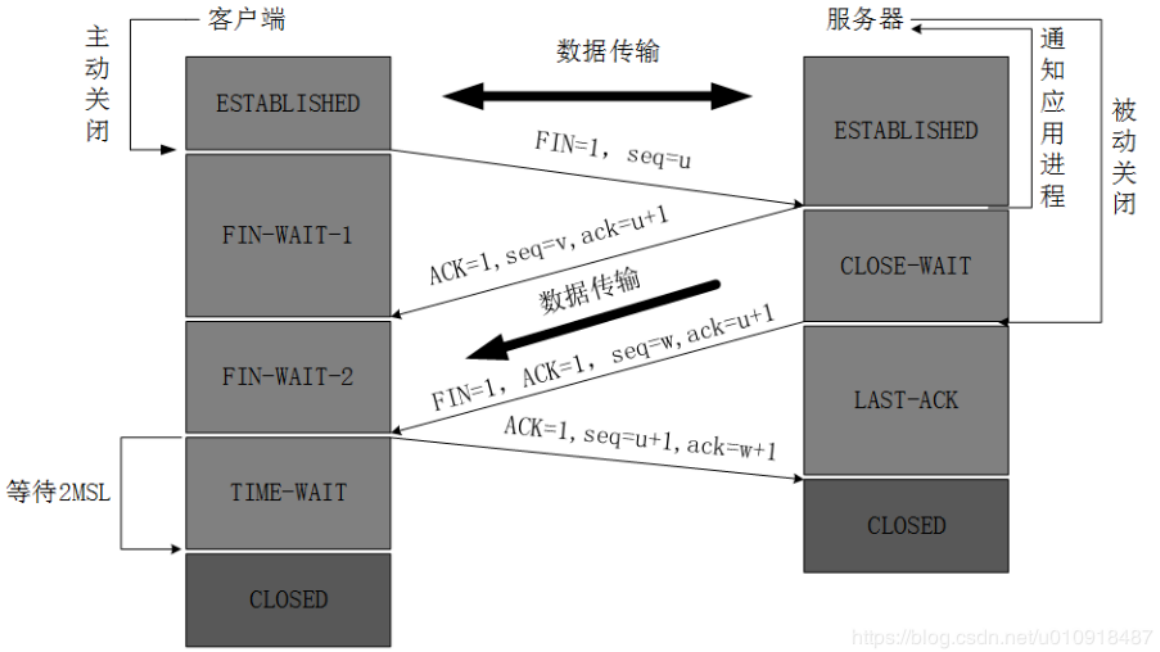
- tcp三次握手



1. 客户发送初始序号x和SYN=1的请求
2. 服务器发送请求表示SYN=1，发送确认标志ACK，发送自己的序号seq=y，发送客户端的确认序号ack=x+1
3. 客户端发送ACK确认号，发送自己的序号seq=x+1，及发送对方的序列号ack=y+1

> 相当于一个传声筒，A在这头联系B，B收到了A的信息给予答复，这时A把传过来的信息再传给B，B确认无误，tcp就连接成功状态，完成3次握手

- 四次挥手



1. 客户端发送释放fin=1,自己的序列号seq=u，进入fin-wait-1阶段
2. 服务器端接收，并返回确认标志ACK=1和客户的确认号ack=u+1，自己的序列号seq=v，进入close-wait状态
3. 客服端收到服务器信息确认后，进入fin-wait-2阶段，服务器发出释放信号fin=1，确认标志ACK=1，确认序号ack=u+1，自已的序号seq=w，进入last-ack阶段
4. 客服端发送确认标志ACK=1，ack=w+1和自己的序号seq=u+1，客服端进入time-wait状态，客户端经过2个最长报文段寿命后，客户端CLOSE；服务器收到确认后，立刻进入CLOSE状态。

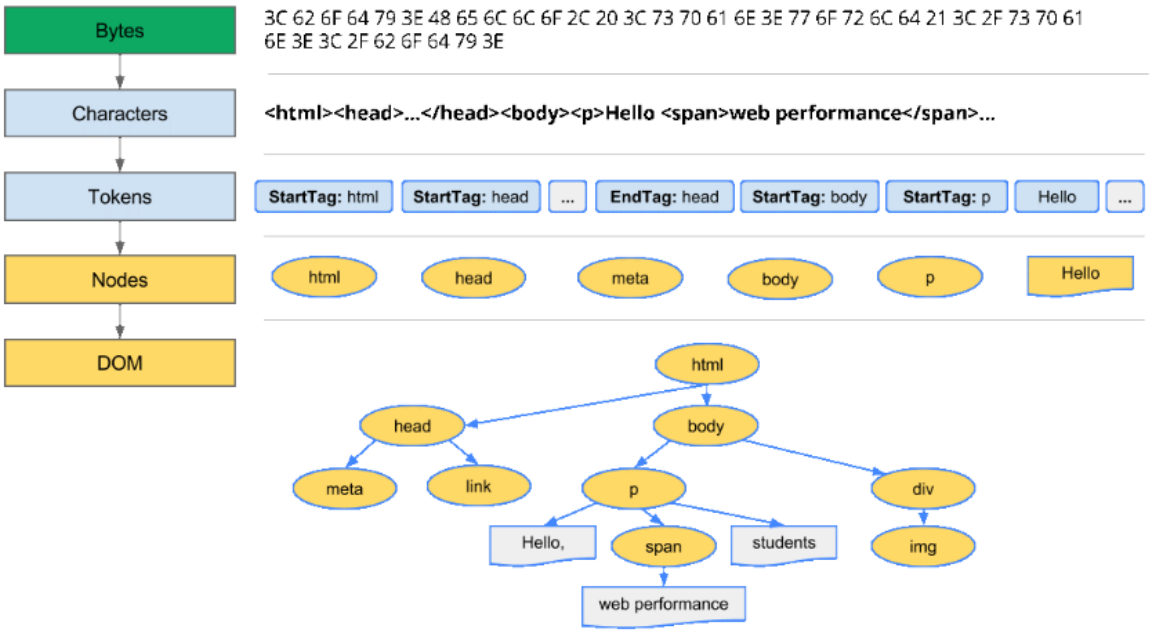
• HTTPS是如何实现加密

定义：HTTPS是加密的http，HTTPS是http+SSL（TLS），HTTP的传输层是TCP，是在tcp上直接通信的，而HTTPS是http先和SSL通信，再和SSL通信，相当于SSL嵌套在http和tcp直接

• 浏览器是如何渲染页面的?

1. 根据HTML文档构建DOM树和CSSOM树及加载js，如果构建途中遇到js，要等js加载完毕，再继续构建dom，HTML文档中皆是节点。DOM构建过程：读取HTML文档-->将字节转换成字符--

>确定tokens（标签）-->将tokens转换成节点-->节点构建DOM树



2. 构建渲染树，由DOM树和CSSOM树合并而成，但不必需要某一个加载完成或者另一个加载完成才开始合并渲染树，三者并行构建，并无先后条件，也并非完全独立，形成一边加载，一边解析，一边渲染的工作现象
3. 页面的重绘与重排

> - 重绘（repaint）:页面的一部分需要重绘，渲染树节点发生变化，但不影响节点的位置及大小，如字体颜色或者某个标签的背景颜色发生变化
> - 重排（reflow）：也称回流，如标签的宽高，内外边距，，导致节点位置发生变化，此时需要浏览器重新生成渲染树，重新布局，即重排

- 如何减少和避免重排？（减少对渲染树的操作）

1. css避免

- 避免使用table布局
- 避免设置多层内联样式
- 将动画效果设置在position的absolute或者flexd的元素上
- 避免使用css表达式
- 使用css3硬件加速，可以让transform、opacity、filters等动画效果不会引起回流重绘

2. js避免

- 避免频繁操作样式，最好一次性重写style属性，或者将样式列表定义为class并一次性更改class属性
- 避免频繁操作dom，创建一个documentFragment，在上面进行所有的dom操作，在添加到文档中
- 可以使用display:none，先隐藏元素，进行修改过后再显示出来，因为display在dom上的操作不会引发重绘和回流

- ## 浏览器的内核有哪些？分别有什么代表的浏览器

1. Trident内核：IE，搜狗高速浏览器
2. Gecko内核：火狐浏览器
3. Wekit内核：safair浏览器，曾经的谷歌浏览器
4. Presto内核：opear 欧朋浏览器
5. Blink内核：现在的谷歌浏览器内核，opear现已改用blink浏览器

- ## 页面导入样式时，@import和link有什么区别?

1. link属于XHTML标签，除了可以记载css，还能定义RSS，定义rel连接属性等作用，import是css提供的，只能加载css
2. 页面加载时，link会同步加载，@import会等页面加载完再加载 3.@import是css2.1提出的，在ie5上可以使用，而link无兼容问题
3. link支持js控制dom改变样式，而@import不可以

- ## 如何优化图像，图像格式的区别

- 优化图像

1. 不用图片，尽量使用css3代替，如半透明、圆角，阴影、半透明等
2. 使用矢量图SVG代替位图，可缩放，位置小
3. 按照HTTP协议设置合理的缓存
4. 使用恰当的图片格式，jpeg适用于内容，修饰图片适用于无损压缩的PNG，动画建议使用video元素和视频格式，或者用SVG动画取代
5. 用css或者JavaScript实现预加载

```
/**
css和JavaScript预加载
*/
<div id="preload1">index</div>
<div id="preload2">index</div>
<div id="preload3">index</div>

function preloader(){
    if(document.getElementById){
        document.getElementById("preload-01").style.background =
'url(http://domain.tld/image-01.png) no-repeat -9999px -9999px';
        document.getElementById("preload-02").style.background =
'url(http://domain.tld/image-01.png) no-repeat -9999px -9999px';
        document.getElementById("preload-03").style.background =
'url(http://domain.tld/image-01.png) no-repeat -9999px -9999px'
    }
}
function AddLoaderEvent(func){
    var oldonload = window.onload;
    if(typeof window.onload != 'function'){
        window.onload = func
```

```
    }else{
        window.onload = function(){
            if(oldonload){
                oldonload()
            }
            func()
        }
    }
}
AddLoaderEvent(preloader)

/**
js预加载
*/

var prodList = [
'https://ss1.bdstatic.com/70cFuXSh_Q1YnxGkpoWK1HF6hhy/it/u=2996011778,1200189855&fm=26&gp=0.jpg',
'https://ss1.bdstatic.com/70cFuXSh_Q1YnxGkpoWK1HF6hhy/it/u=2996011778,1200189855&fm=26&gp=0.jpg',
'https://ss1.bdstatic.com/70cFuXSh_Q1YnxGkpoWK1HF6hhy/it/u=2996011778,1200189855&fm=26&gp=0.jpg'
]

function preloader(){
    if(document.images){
      for(let i = 0 ; i < prodList.length ; i ++){
         var o = document.createElement('object')
         o.data = prodList[i]
         document.body.appendChild(o)
      }
    }
}
function AddLoaderEvent(func){
    var oldonload = window.onload;
    if(typeof window.onload != 'function'){
        window.onload = func
    }else{
        window.onload = function(){
            if(oldonload){
                oldonload()
            }
            func()
        }
    }
}
AddLoaderEvent(preloader)
```

- 图像格式的区别

1. 矢量图：图标字体，如SVG，font-awesome
2. 位图：JPG（JPEG），GIF，PNG

区别：

- GIF：是一种无损，8位图片格式，支持动画、压缩等特性，适用于色彩简单色彩少的图片，如logo，各种小图标icons等
- JPG：是大小与质量相平衡的压缩图片格式，适用于允许微失真的色彩丰富的照片，不适合做色彩单调的图片
- PNG：PNG可以分为三种格式，PNG8，PNG24，PNG32，后面的数字代表最多可以索引和存储的颜色值，PNG8支持索引透明和alpha透明，PNG24不支持透明，

优缺点：

- 保证图片不失真的情况下使用压缩图片
- 对于高保真较复杂的图像，PNG虽然无损压缩，但不适合用在web页面上

列举你了解Html5. Css3 新特性

HTML5提供了新的元素来创建更好的页面结构：

<article>	定义页面独立的内容区域。
<aside>	定义页面的侧边栏内容。
<bdi>	允许您设置一段文本，使其脱离其父元素的文本方向设置。
<command>	定义命令按钮，比如单选按钮、复选框或按钮
<details>	用于描述文档或文档某个部分的细节
<dialog>	定义对话框，比如提示框
<summary>	标签包含 details 元素的标题
<figure>	规定独立的流内容（图像、图表、照片、代码等等）。
<figcaption>	定义 <figure> 元素的标题
<footer>	定义 section 或 document 的页脚。
<header>	定义了文档的头部区域
<mark>	定义带有记号的文本。
<meter>	定义度量衡。仅用于已知最大和最小值的度量。
<nav>	定义导航链接的部分。
<progress>	定义任何类型的任务的进度。
<ruby>	定义 ruby 注释（中文注音或字符）。
<rt>	定义字符（中文注音或字符）的解释或发音。
<rp>	在 ruby 注释中使用，定义不支持 ruby 元素的浏览器所显示的内容。
<section>	定义文档中的节（section、区段）。
<time>	定义日期或时间。
<wbr>	规定在文本中的何处适合添加换行符。

可以通过哪些方法优化css3 animation渲染

1. 尽可能多的利用硬件能力，如使用3D变形来开启GPU加速：
2. 尽可能少的使用box-shadows与gradients

- **列举几个前端性能方面的优化**

1. 减少http请求（合并js和css，使用base64图片）
2. 减少资源体积（压缩jcss、图片，gzip压缩）
3. 缓存（CDN缓存，DNS缓存，http缓存）
4. 优化网页渲染（css放在头部，js放在底部或者异步，避免内联样式）
5. DOM操作优化（

- 避免在document上直接频繁操作DOM
- 使用classname代替内联样式
- 尽量使用css动画
- 使用position等于absolute或者flexed
- 适当使用canvas动画
- 尽量避免使用css表达式 ）

3. **操作细节注意**：

- 避免图片或者frame使用空src
- 在css属性为0时，去掉单位
- 禁止图像缩放
- 正确的css前缀的使用
- 移除空的css规则
- 对于css中可继承的属性，如font-size，尽量使用继承，少一点设置
- 缩短css选择器，多使用伪元素等帮助定位

- **如何实现同一个浏览器多个标签页之间的通信**

1. 使用localStorage
2. 使用cookie+setInterval

- **浏览器的存储技术有哪些**

1. cookie
2. localStorage
3. sessionStorage

区别：

1. 与服务器通信：cookie数据诗句在同源的http的请求中携带，session和local仅在本地保存
2. 数据生命期限：一般由服务器生成，可设置生命期限，在浏览器生成，默认浏览器关闭后失效，local不删除永久保存，session仅在当前对话生效，关闭浏览器或者关闭会话删除

3. 存放数据大小：cookie为4K左右，Storage一般为5M
4. 易用性：cookie源生接口不友好，需要自己封装，Storage源生接口友好，可再对其进行封装对Array和Object更加友好

• css定位方式（position）

1. static：默认值，没有定位，忽略left，right，top或者z-index声明
2. relative：生成相对定位元素，可以设置top，left或者z-index声明
3. absolute：生成绝对定位元素，相对于static以外的第一个元素进行定位
4. fixed：生成绝对定位元素，相对于浏览器窗口定位

relative和absolute的区别：

1. relative脱离文档流，但文档流这种位置依然存在，而absolute脱离文档流，文档流的位置不存在
2. relative无论父元素是何种定位，都是相对最近的父元素定位，absolute是相对于其最近的定义为absolute或relative或fixed的父层，如果找不到就根据body定位

• 出浏览器兼容性问题

1. 不同浏览器标签默认的{margin:0;padding:0}不一样

- 解决方案：给每个标签都加{margin:0;padding:0}，如 body,h1,h2,h3,ul,li,input,div,span,a,form …… { margin:0; padding:0; }

2. 块级元素设置float后又有横向的margin时，ie6中显示margin比设置的大，导致最后一块被顶到下一行

- 解决方案：在设置float的标签样式中加入display：inline

3. 在设置标签高度过小时（小于10px），在ie6、7，遨游浏览器中高度超出自己设置的高度

- 解决方案：设置overflow：hidden或者设置line-height小于设置的高度

4. 行内间距问题，行内元素设置display：block后采用float布局，又有横向的margin，ie6的间距比设置的间距大

- 解决方案：在display：block加入display：inline;display:table

5. 图片默认有间距，几个图片放在一起时，有些浏览器会默认有间距

- 解决方案：设置float：left

6. 设置min-height不兼容

- 解决方案：加上height：auto !important;height:200px;overflow:visible

7. 边距重叠问题：当相邻两个元素都设置了margin边距时，margin将取最大值，舍弃最小值

- 解决方案：为了让边距不重叠，给子元素设置一个父元素，并设置父元素的overflow：hidden

- **垂直上下居中的方法**

1. 第一种

```
<style>
    .box{
        width: 100px;
        height: 100px;
        border: 1px solid green;
        position: absolute;
        top:0;
        left:0;
        right:0;
        bottom:0;
        margin:auto
    }
</style>
<div class='box'></div>
```

2. 第二种

```
<style>
    .box{
        width: 100px;
        height: 100px;
        border: 1px solid green;
        position: absolute;
        top:50%;
        left:50%;
        tranform:tranlate(-50%,-50%)
    }
</style>
<div class='box'></div>
```

3. 第三种

```
<style>
    .box{
        width: 100px;
        height: 100px;
        border: 1px solid green;
        position: absolute;
        top:50%;
        left:50%;
        margin-left:-50px;
        margin-top:-50px
    }
</style>
<div class='box'></div>
```

- **响应式布局原理**

1. 媒体查询（移动端首先使用min-width，PC端首先使用max-width）
2. 百分比布局
3. rem布局

- **清除浮动的方法**

1. clear：both（不建议使用，添加无意义标签，语义化差）
2. 给父元素加overflow：hidden（不建议使用，当内容增多时，不会自动换行，导致内容被隐藏，无法显示出要溢出的元素）
3. 使用after伪元素清除

```
.fahter {
          width: 400px;
          border: 1px solid deeppink;
          /* overflow: hidden; */
      }

      .big {
          width: 200px;
          height: 200px;
          background: darkorange;
          float: left;
      }

      .small {
          width: 120px;
          height: 120px;
          background: darkmagenta;
          float: left;
      }

      .footer {
          width: 900px;
          height: 100px;
          background: darkslateblue;
      }

       .clearfix:after{
          content:"",
          display:block,
          height:0,
          clear:both,
          visibility:hidden
  }
       .clearfix:{
          *zoom:1,
       }

<div class="fahter clearfix">
```

```
        <div class="big">big</div>
        <div class="small">small</div>
        <div class="clear ">额外标签法</div>
    </div>
    <div class="footer"></div>
```

4. 使用伪元素after和before

```
.clearfix::after, .clearfix::before{
    content:"";
    display:table
}
..clearfix::after{
    clear:both
}
.clearfix{
    *zoom:1
}
```

- ## http协议和tcp协议

定义：HTTP（超文本传送协议），规定浏览器向服务器请求及服务器怎么把数据传给浏览器，浏览器和服务器之间的请求和响应之间的交互；TCP是传输层协议，所以说要使用的时候必须先建立TCP连接，是一对一的连接

理解：TCP是传输层协议，定义数据传输和连接方式的规范，三次握手的包不包含数据，三次握手建立成功过后客户端和服务器才开始传输数据；HTTP是超文本传输协议，是应用层协议，定义数据内容的规范，HTTP协议中的数据是基于TCP传递的；就好比TCP是路路上跑的车，HTTP是车里面的人，路上可以使用三轮车、大车、或者轿车去承载人，每辆车载的人也不一样

- ## 一次js请求，一般会有哪些缓存处理

1. DNS缓存：短时间内多次访问某个网站，在限定时间内，不用多次访问DNS服务器
2. CDN缓存：内容分发网络（人们在就近的代售点取火车票，不用非得去火车站排队）
3. 浏览器缓存：浏览器在用户磁盘上，对最新请求过得文档进行刷新
4. 服务器缓存：将最近访问的web页面和对象保存在离用户最近的系统中，加快访问速度

- ## 如何对网站的文件和资源进行优化

1. js的api使用cdn地址
2. 精简优化js和css
3. gzip压缩js和css
4. 使用css sprites合并图片（很多小图片合并成一张图片）
5. 优化网址图片

- ## 谈谈你对web标准以及W3C的理解

W3C标准是强调一个网址的结构、样式、行为，从而达到结构清晰，分工明确，易于修改和理解。

- HTML：作为整个页面的骨架，结构要良好，易于SEO，使用闭合标签，标签要小写，不允许随意嵌套
- css：负责页面样式，美化页面的布局
- JavaScript：使页面更加动态化、交互性

如何让Chrome浏览器显示小于12px的文字

1. transform：scale()

```
.p{
    font-size:10px;//要变成8像素
    transform:scale(0.8)
}
```

如何实现页面每次打开时清除本页缓存

1. 设置HTTP头信息

```
<HEAD>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">
<META HTTP-EQUIV="Expires" CONTENT="0">
</HEAD>
```

说明：HTTP头信息’ Expires ‘和’ Cache-Control ‘为服务器提供了一个控制浏览器和代理服务器的缓存机制，Expires告诉代理服务器缓存页面何时将过期，Cache-Control告诉浏览器不缓存任何页面

2. 在需要打开的url后面加一个随机的参数

增加参数前：url=test/test.jsp

增加参数后：url=test/test.jsp?ranparam=random()

3. ajax方法，加上请求头If-Modified-Since和Cache-Control，或直接加cache:false

```
$.ajax({
     url:'www.haorooms.com',
     dataType:'json',
     data:{},
     cache:false
     beforeSend :function(xmlHttp){
        xmlHttp.setRequestHeader("If-Modified-Since","0");
        xmlHttp.setRequestHeader("Cache-Control","no-cache");
     },
```

```
    success:function(response){
        //操作
    }
    async:false
});
```

- ## 什么是Virtual DOM,为何要用Virtual DOM

Virtual DOM：虚拟节点，通过js的Object对象，模拟DOM中节点，然后再通过特定的render方法渲染成真实的DOM节点，提高重绘性能。

- ## 伪元素和伪类的区别

定义：伪类是向某些选择器添加特殊效果；伪元素是将特殊效果加到某些选择器上

区别：

1. 伪类使用单：，伪元素可以使用单：也可以是双：
2. 伪类可以叠加使用，没有上限只要不互斥（p:last-child:hover）；伪元素在一个选择器中只能出现一次，并且只能出现在末尾
3. 伪元素要配合content来使用，不会出现在DOm中，不能通过JS获取，仅在css的渲染层加入

- ## HTTP请求方式?

0. OPTIONS
1. HEAD
2. GET
3. POST
4. PUT
5. DELETE

- GET和POST的区别?

1. GET浏览器回退/刷新是无害的，POST会被重新提交请求
2. 书签：GET能被收藏为书签，POST不能
3. 缓存：GET能被缓存，POST不能
4. GET只能进行URL编码，而POST可以支持多种编码方式
5. GET通过URL传递，有长度限制；POST放在请求头的对象中且长度没有限制，所以POST比GET更安全，GET的参数暴露
6. GET产生一个TCP数据包，而POST产生两个。因为GET请求把请求头和data一起发送，服务器响应200；而POST先把请求头发给服务器，服务器响应100，再把data发过去，服务器响应200

- ## 前端怎么注意SEO（搜索引擎的优化）？

1. 合理的title，description，keyboards，

2. 语义化HTML代码，符合W3C标准，让搜索引擎理解页面
3. 重要的HTML放在前面，搜索引擎顺序是从上到下，有的会有长度限制，保证重要内容一定会被抓取
4. 尽量少用iframe，搜索引擎不会抓取iframe里面的内容
5. 非装饰的图片必须加alt
6. 提高网址速度

- ## img的title和alt有什么区别?

1. alt表示图片无法显示时的替代信息，给搜索引擎看的，只能用于img，area，input标签中，title表示给改元素提供建议信息，鼠标悬停的时候显示，

- ## HTML语义化?

- 作用

1. 页面结构化
2. 利于浏览器解析和搜索引擎的优化
3. 提高代码的重复利用性及可维护性

- 语义化内容

1. 标题语义化
2. 图片语义化（加alt和title，alt给搜索引擎看，title给用户看，增加figcaption图注文字）
3. 表单语义化（用fieldest标签包裹起来，并用legend标签说明表单用途，label的for属性和input的id绑定起来，说明文本和input标签绑定起来）
4. 表格语义化

- ## HTML5的离线储存怎么使用，工作原理能不能解释一下?

定义：在离线状态下，可以正常访问站点和网页，有网的情况下用户机器上的缓存文件

原理：基于新建的.appcache文件的缓存机制，将应用程序安装到应用程序缓存中，先创建一个清单，包含所有应用程序的依赖和URL列表，通过在HTML的标签中设置manifest属性，指向该清单文件即可

- ## iframe的优缺点?

- 优点

1. 能把页面原封不动的嵌入进来
2. 网页头部的底部一样的风格时，可以用iframe来嵌套，增加代码可重用性
3. 遇到加载缓慢的第三方图标，可以用iframe来加载

- 缺点

1. 会阻塞主页面的onload事件
2. 不利于SEO
3. iframe会增加服务器的http请求

- ## Doctype作用? 严格模式与混杂模式如何区分？它们有何意义?

Doctype：<!DOCTYPE>叫做文件类型定义，定义在第一行，不是HTML标签，告诉浏览器该文件的类型，让浏览器知道应该用哪个规范来解析文档

区分严格和混杂模式：

1. 严格模式：又称标准模式，指按W3C的标准解析代码
2. 混杂模式：又称怪异模式或者兼容模式，指浏览器按自己的方式解析代码
3. 浏览器解析用严格还是混杂模式，与网页的DTD有关

> dtd的生命有三种选择，过渡性的（Transitional），严格的（strict），框架的（Frameset）
> HTML 4.01 Strict： <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

> HTML 4.01 Transitional： <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

> HTML 4.01 Frameset： <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

1、过渡的：一种要求不很严格的，允许在html中使用html 4.01的标识(符合xhtml语法标准)，过渡的dtd写法如下：

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

2、严格的：一种要求严格的dtd，不允许使用任何表现层的标识和属性，严格的dtd写法如下：

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

3、框架的：一种专门针对框架页面所使用的dtd，当页面中含有框架元素时，就要采用这种dtd，写法如下：

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

• HTML全局属性(global attribute)有哪些?

1. class：为元素设置类标识，可通过class属性获取元素
2. data-*:为元素增加自定义属性
3. id：元素id，文档内唯一
4. draggable：设置元素是否可以拖动
5. lang：元素内容的语言
6. style：行内样式
7. title：元素相关的建议信息
8. contextmenu：自定义鼠标右键弹出菜单内容

• SVG和Canvas的区别?

定义：都允许在浏览器创建图形，canvas依赖JavaScript来绘制2D图形，SVG依赖XML绘制的2D图形的语言，canvas逐像素进行渲染，当其位置发生变化，会重新进行绘制；SVG是基于XML，SVG DOM中的每个像素都可以用，可以为某个像素增加JavaScript事件，在SVG中，每个绘制的图形称为对象，当对象的属性发生变化时，浏览器能够自动重现图形

区别：

1. SVG是用XML描述的2D图形，canvas是JavaScript绘制的2D图形
2. SVG不依赖分辨率，canvas依赖分辨率
3. SVG支持事件处理器，canvas不支持
4. SVG适合大型的渲染区域的应用程序（如谷歌地图），canvas弱的文本渲染能力
5. SVG复杂度高会减慢渲染速度，canvas能够以.jpg或者.png格式保存图形
6. SVG不适合游戏应用，canvas适合图像密集型的游戏，其中许多对象会被频繁重绘

- **如何在页面上实现一个圆形的可点击区域?**

1. img通过usemap映射到<map>的circle的

```
<img src='img.png' width='1366' height='500' border='1' usemap='#map' />
<map id='map' name='map'>
    <area shape='circle' coords='100,100,100' href='#></area>
</map>
```

2. 通过css

```
.circle{
    width:100px;
    height:100px;
    border-radius:50%;
    bcakground:dimgray;
    cursor:pointer;
    position:absolute;
    left:50px;
    top:50px;
}

<img src='img.png' width='1366' height='500' border='1' usemap='#map' />
<div class='circle'></div>
```

3. 通过js， 两点之间距离的公式 |AB| = $|AB| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$

```
<img src='img.png' width='1366' height='500' border='1' usemap='#map' />
<div class='circle'></div>

document.onlick = function(e){
    var r = 50 ; //半径
    var xLeft = 100 ;
    var yTop = 100
    var x1 = e.clientX;
    var y1 = e.clientY
    //<!--Math.abs():绝对值  Math.sqrt():平方根  Math.pow(x,y) ：x的y次方-->
    var dom = document.getElementsByClassName('disc')[0]
     dom.style.cssText=`background:'dimgray';width:${x1};height:${y1};border-
radius:50%;position:absolute;top:50px;left:50px;line-height:100px;
            text-align:center;color:white`
```

```
    var len = Math.abs(Math.sqrt(Math.pow(x1-xLeft,2)+Math.pow(y1-yTop,2)))
    if(len<=50){
        console.log("Inner");
    }else{
        console.log("Outer");
    }

}
```

- **网页验证码是干嘛的？用来解决什么安全问题？**

用来区分是人为操作还是机器操作，因为机器无法准确识别图片的内容

> 防止恶意破解密码、刷票、论坛灌水； 有效防止黑客对某人特定的账户用特定的程序进行不断的登录尝试

- **CSS选择器有哪些？哪些属性可以继承？CSS优先级算法如何计算？**

- css选择器类型

选择符类型	例子	描述
通用选择器	*	
类选择器(.class)	.intro	选择class='intro'的所有元素
ID选择器(#id)	#first	选择id='first'的所有元素
标签选择器(element)	div	选择所有 标签
后代选择器(element element)	div p	选择元素为 内部的所有 标签
子选择器(ele > ele)	div > p	选择父元素为 内部的所有 标签
群组选择器(ele,ele)	div,p	选择所有 和 标签
相邻同胞选择器(ele + ele)	div + p	选择连接在 之后的所有

选择符类型	例子	描述
伪类选择器(:first-child , :link , :hover , :focus等)	a:link , a:first-child , a:hover	选择所有未被访问的链接，，鼠标指针位于其上链接
伪元素选择器(:first-letter,:first-line,:before,:after, :lang(language))	p:first-letter,p:first-line,p:before,p:after,p:lang(it)	选择每个标签的首字母、首行、每条元素前插入内容、每条元素后插入内容、选择带有'it'开头的lang属性值的每个元素
属性选择器([attribute],[attribute = value])	[target=_blank]	

- 继承问题

0. 可继承样式

> 所有元素可继承：visibility,cursor
>
> 终极块级元素可继承：text-indent,text-align
>
> 内联元素：font-size/famliy/style/weight/,line-height,color
>
> 列表元素:list-style/style-type/style-position/style-image

2. 不可继承样式:width,height,margin,padding,display,top,float

- 优先算法问题

#id选择器加0100，.class选择器加0010，标签和伪元素加0001，通配符没有贡献加0000

1. 行内样式如(p style='height:100px')的优先级值为1000
2. !important的优先级值最高，可认为是10000
3. 按CSS代码中出现的顺序决定，后者CSS样式居上；（近水楼台 先得月）

- ## css3的新特性?

1. 实现圆角（border-radius），阴影（box-shadow），边框图片（border-image）
2. 文字特教（text-shadow），线性渐变（linear-gradient）
3. 旋转，缩放，定位，倾斜，rotate(),scale(),translate(),skew()
4. 媒体查询(@media)，flex布局

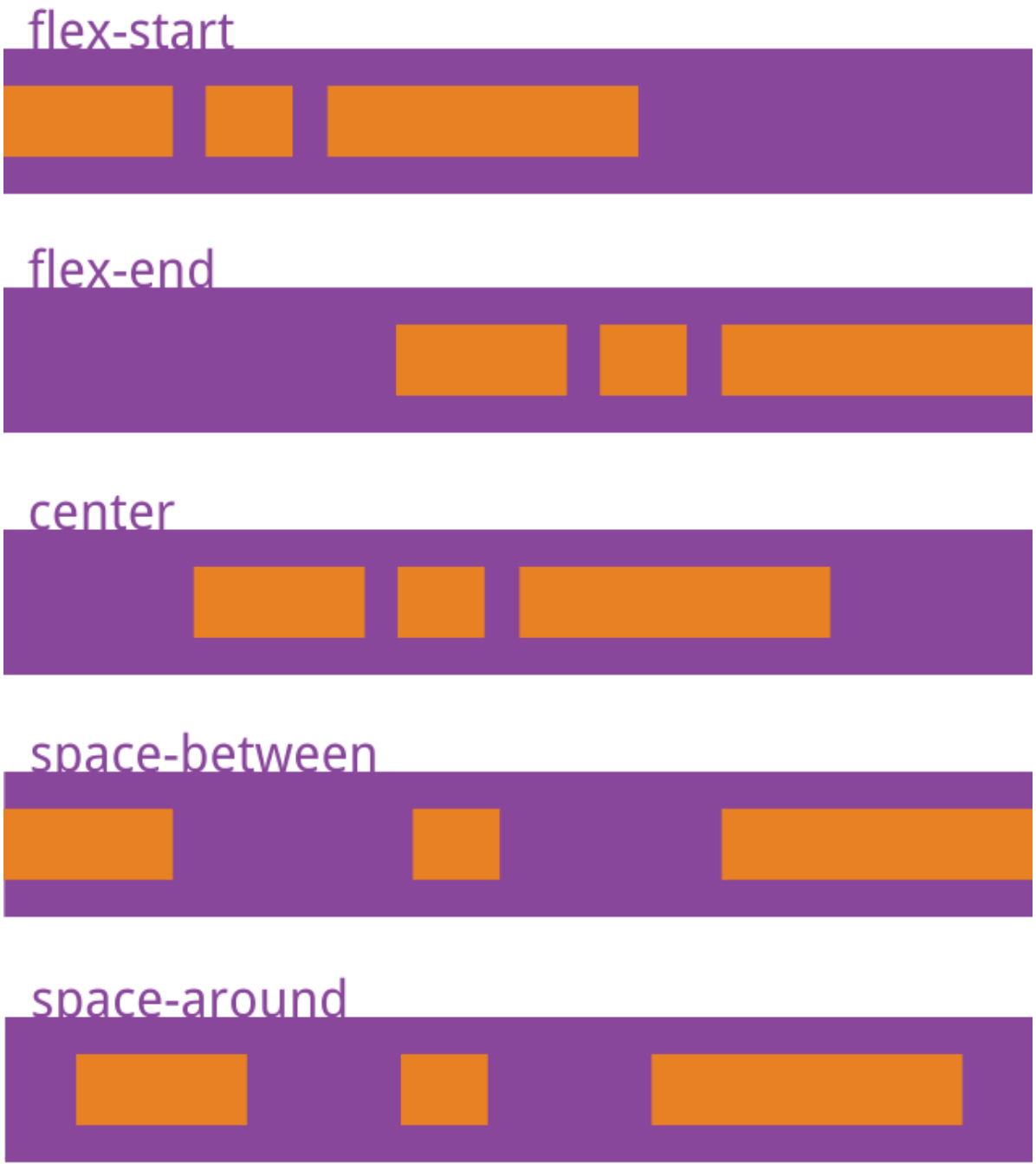
- ## 请解释一下CSS3的flexbox（弹性盒布局模型）,以及适用场景?

基本概念：采用flex布局，称为flex容器，他所有的子元素自动成为容器成员。容器默认存在两根轴，水平的主轴和垂直的交叉抽，项目默认沿主轴排列

- 容器的属性

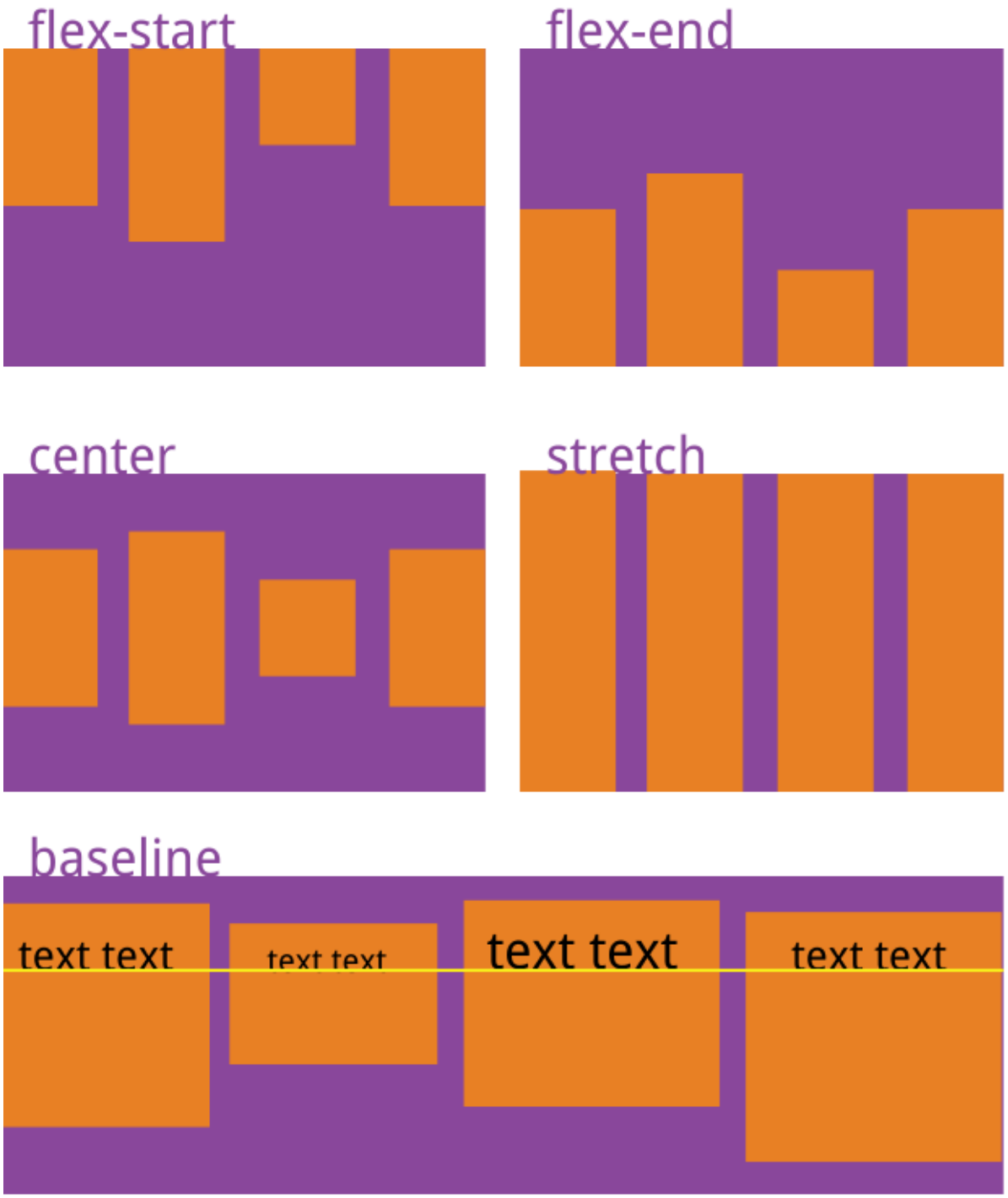
1. felx-direction:row(左→右)|row-reverse(右→左)|column(上→下)|column-reverse(下→上)，决定主轴的方向
2. flex-wrap：nowrap(不换行)，wrap(换行，第一行在上方)，wrap-reverse(换行，第一行在下方)
3. flex-flow：是flew-direction和flex-wrap的简写，可以使用他们的属性值
4. justify-content：定义在主轴上的对齐方式，flex-start(默认值，左对齐)，flex-end(右对齐)，center(居中)，space-between(两端对齐，项目之间的间隔相等)，space-around(每个元素的

两侧相等，项目两侧的距离比边框与边框的距离大一倍)



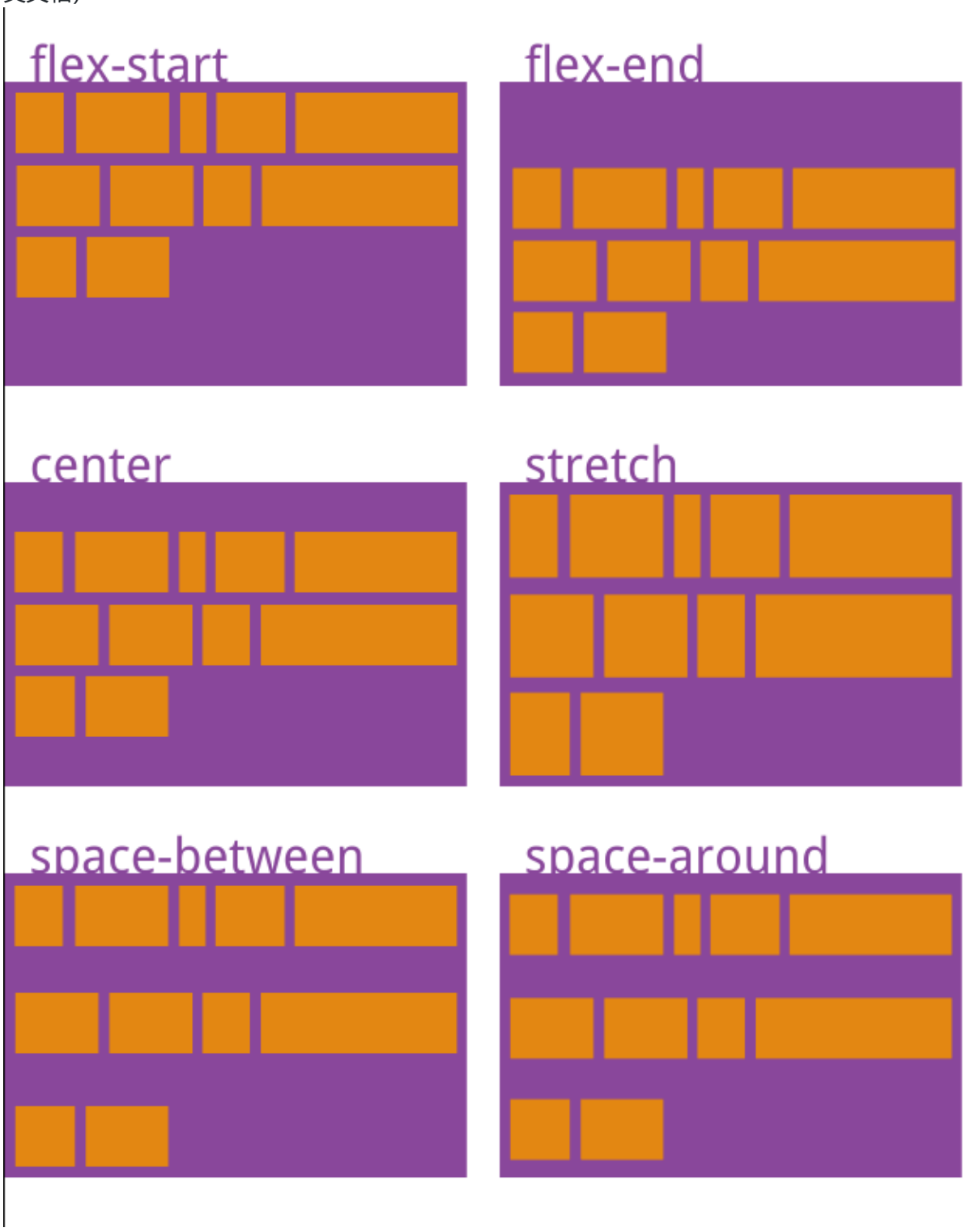
5. align-items：定义元素在交叉轴上如何对齐，flex-start（交叉的起点对齐），flex-end（交叉的终点对齐），center（交叉的中点对齐），baseline（项目的第一行文字的基线对

齐）,stretch(默认值)



6. align-content：多跟轴线的对齐方式，如果只有一根轴线，该属性不起作用，flex-start（与交叉轴的起点对齐），flex-end（与交叉轴的终点对齐），center（与交叉轴的中点对齐），space-between（与交叉轴的两端对齐，轴线之间饿间隔平均分布），space-around（每根轴线的两侧相等，轴线之间的间隔比轴线与边框的距离大一倍），stretch（默认值，轴线占满整个

交叉轴)



- **用纯CSS创建一个三角形的原理是什么?**

原理：均分原理，建立一个矩形，想四个角划分为4个部分，给矩形的宽度和高度设置为0，设置边框的高度，需要显示的加颜色，不需要显示的透明

```
.box{
    width:0;
    height:0;
    border:50px solid transparent;
```

```
    border-bottom:50px solid red;
}
```

2. 使用css旋转正方形

```
.entily{
    width: 100px;
    height: 100px;
    border: 1px solid rebeccapurple;
    overflow: hidden;
}
.entily::after{
    content: '';
    display: block;
    text-align: center;
    line-height: 50px;
    width: 50px;
    height: 50px;
    border: 1px solid violet;
    margin: 24px;
    transform: rotate(45deg);
    margin-top: 74px;
}
<div class="entily"></div>
```

- ## 为什么初始化css样式?

1. 浏览器差异，不同浏览器有些标签的默认值不一样，不初始化css样式浏览器页面差异大
2. 提高代码质量，不初始化页面样式弄完会很糟糕

- ## CSS里的visibility属性有个collapse属性值？在不同浏览器下以后什么区别?

1. 在普通标签中，和visibility:hidden没有什么区别，元素空间位置还在
2. 但使用在表格上面，和display：none一样，元素空间被释放，且在ie7以下使用visibility：collapse没有效果

- ## display:none与visibility：hidden的区别?

1. 设置display后内存空间释放，而visibility内存空间还在

2. visibility具有继承性，给父元素设置visibility，子元素会消失，但给子元素设置visibility：visible子元素又会显示出来

- ## position跟display、overflow、float这些特性相互叠加后会怎么样

display定义了生成的框的类型，position规定元素的定位类型，float是一种布局方式，定义元素在哪个方向浮动；优先机制：position：absolute/fixed优先级最高，有他们在，display需要调整，float不起作用，float或者absolute定位的元素只能是块级元素或者表格

- ## 对BFC规范(块级格式化上下文：block formatting context)的理解?

定义：创建了BFC就是创建了一个盒子，只有Block-Level box可以参与创建，规定内部如何布局，盒子里面的内容不受外界影响也不影响外界

特性：内部box会在垂直方向一个接着一个放；垂直方向的距离有margin决定，且同一个BFC的相邻的box的margin会发生叠加；

触发BFC：float定位（除了none），overflow除visible的值，position（值为absolute/fixed）

- ## css优化，提高性能的方法?

1. 合并css文件
2. 减少css嵌套，最好三层以下
3. 建立公共样式类
4. 不用css表达式
5. 使用csssprite，把一些小图标合成大的图片，减少http请求
6. gzip压缩，

- ## 浏览器是怎样解析CSS选择器的?

从上到下，从右到左解析

- ## 在网页中的应该使用奇数还是偶数的字体？为什么呢?

奇偶都可以，浏览器都可以解析，不过习惯写偶数，布局换算时比较方便

- ## 元素竖向的百分比设定是相对于容器的高度吗?

```
.main{
   height: 200px;
   width: 300px;
   background: gray
}
```

```
.set{
    height:20%;//相对于父元素的高度
    margin-top:20%;//相对于父元素的宽度
}
<div class='main'>
    <p class='set'></p>
</main>
```

- **什么是响应式设计？响应式设计的基本原理是什么？如何兼容低版本的IE?**

响应式设计就是一个网站能够兼容多个终端，而不是为每一个终端做一个特定的版本

原理：通过媒体查询来兼容不同分辨率的终端

兼容低版本的ie，页面头部必须有meta声明的viewport

> <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">

- **如果需要手动写动画，你认为最小时间间隔是多久，为什么?**

多数显示器饿频率显示为60HZ，即1秒刷新一次，所以理论上最小间隔为1/60 * 1000 = 16.7ms

- **有一个高度自适应的div，里面有两个div，一个高度100px，希望另一个填满剩下的高度**

```
.outer{
    height:100%;
    position:relative
}
.A{
    height:100px;
}
.B{
    position:absolute;
    top:100px;
    left:0;
    bottom:0;
}
<div class="outer">
    <div class="A"></div>
    <div class="B"></div>
</div>
```