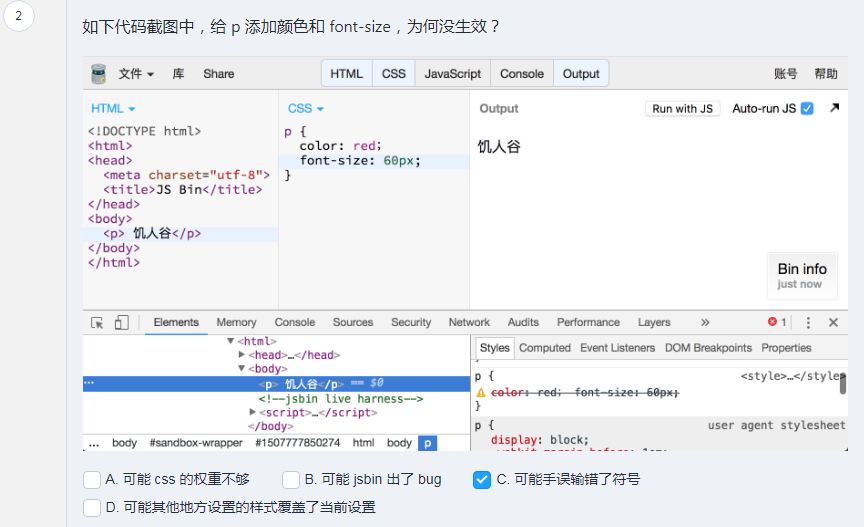
7.CSS基础与选择器

.header + p指 .header的同级下一个邻居p;  
.header ~ p指 .header的同级之后的所有邻居p





一．CSS 加载方式有几种？

1.内部引用，在html文档中在<style>标签里面写的css样式

<style>

bdoy{font-size:14px;}

</style>

2.外部引用，用<link>标签引用外部的css文件，将样式引用到html文档来。

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

3.在标签中使用，使用style属性将当前的标签样式改变。

<div style="color:red"></div>

4.@import引入样式

<style type="text/css" media="screen"> @import url("CSS文件"); </style>

二．@charset有什么作用？

@charset <charset>;

在外部样式表文件内使用。指定该样式表使用的字符编码。声明该css文件使用utf-8编码。当文件保存为utf-8编码时，其中的注释文字将正常显示。否则注释乱码。

<charset>：字符编码。如：@charset "utf-8";

该规则后面的分号是必需的，如果省略了此分号，会生成错误信息。

三．@import有什么作用？如何使用？

link和@import引入样式写法

link和@import引入CSS样式

link：<link rel="stylesheet" type="text/css" href="../main.css">

rel：规定当前文档与被链接文档之间的关系。

type：规定被链接文档的 MIME 类型。

href：规定被链接文档的位置。

@import：<style type="text/css" media="screen"> @import url("CSS文件"); </style>

link和@import的区别：

link是XHTML标签，除了加载CSS外，还可以定义RSS等其他事务；@import属于CSS范畴，只能加载CSS。

link引用CSS时，在页面载入时同时加载；@import需要页面网页完全载入以后加载。

link是XHTML标签，无兼容问题；@import是在CSS2.1提出的，低版本的浏览器不支持。

link支持使用Javascript控制DOM去改变样式；而@import不支持。

四．id 选择器和 class 选择器的使用场景分别是什么？

class使用场景：需要某些特定样式的标签则放在同一个class中，需要此样式的标签可再添加此类。（class不可被javascript中的GetElementByID函数所调用）。类选择器可以同时控制多个标签，只要赋予这些标签相同的类属性就可以实现。一个class类属性内可以有多个值。

id使用场景：1、根据提供的唯一id号快速获取标签对象，如：document.getElementById(id) ；2、用于充当label标签for属性的值：示例：<label for='userid'>用户名：</label>，表示单击此label标签时，id为userid的标签获得焦点。id选择器能且仅能控制一个标签。因为每一个id值在一个页面中都是唯一的，就像人的身份证号码一样，绝对不会出现第二个相同的id。

五．CSS选择器常见的有几种？

1、类别选择器

类别选择器根据类名来选择，前面以“.”来标志。在HTML中，元素可以定义一个class的属性，并且多个元素都可以重复应用这个属性。

2、标签选择器

一个完整的HTML页面是有很多不同的标签组成，如：body，div，p，ul，li，而标签选择器，则是决定哪些标签采用相应的css样式。

3、ID选择器

ID选择器可以为标有特定ID的HTML元素制定特定的样式。根据元素ID来选择元素，具有唯一性，这意味着同一ID在同一文档页面中只能出现一次。例如，你将一个元素的id取值为“name”，那么在同一页面你就不能再将其他元素id取名为“name”了。前面以“#”号来标志。

4、通用选择器

用过“\*”号来表示。同时通用选择器还可以和后代选择器组合。但是如果你在p标签里嵌套了一个p标签，就会出现浏览器不能解析的情况，因此要避免这种情况发生。

5、后代选择器

后代选择器也称为包含选择器，用来选择特定元素或元素组的后代，将对父元素的选择放在前面，对子元素的选择放在后面，中间加一个空格分开。后代选择器中的元素不仅仅只能有两个，对于多层子线后代关系，可以有多个空格加以分开。

6、子选择器

请注意这个选择器与后代选择器的区别，子选择器只是选择元素的直接后代，即仅仅作用于第一个后代。后代选择器通过空格来进行选择，而子选择器是通过“>”进行选择。

7、伪类选择器

即链接样式。a元素的伪类，存在4中不同的状态：link、visited、active、hover。

8、群组选择器

"，"多元素选择器。具有相同样式的标签分组显示，将具有多个相同属性的元素，合并群组进行选择，定义同样的css属性，这大大的提高了编码效率，同时也减少了css文件的体积。

9、相邻同胞选择器

如一个标题h1元素后面紧跟了两个段落p元素，我们想定位第一个段落p元素，对它应用样式，就可以使用相邻同胞选择器。两个元素之间用“+”进行连接。

六．伪类选择器有哪些？伪元素有哪些？

CSS 伪类用于向某些选择器添加特殊的效果。CSS 伪元素用于向某些选择器设置特殊效果。伪类的效果可以通过添加一个实际的类来达到，而伪元素的效果则需要通过添加一个实际的元素才能达到，这也是为什么他们一个称为伪类，一个称为伪元素的原因。

1.伪类选择器

E:first-child   匹配作为长子（第一个子女）的元素E

E:link  匹配所有未被点击的链接

E:visited   匹配所有已被点击的链接

E:active    匹配鼠标已经其上按下、还没有释放的E元素

E:hover 匹配鼠标悬停其上的E元素

E:focus 匹配获得当前焦点的E元素

E:lang(c)   匹配lang属性等于c的E元素

E:enabled   匹配表单中可用的元素

E:disabled  匹配表单中禁用的元素

E:checked   匹配表单中被选中的radio或checkbox元素

E::selection    匹配用户当前选中的元素

E:root  匹配文档的根元素，对于HTML文档，就是HTML元素

E:nth-child(n)  匹配其父元素的第n个子元素，第一个编号为1

E:nth-last-child(n) 匹配其父元素的倒数第n个子元素，第一个编号为1

E:nth-of-type(n)    与:nth-child()作用类似，但是仅匹配使用同种标签的元素

E:nth-last-of-type(n)   与:nth-last-child() 作用类似，但是仅匹配使用同种标签的元素

E:last-child    匹配父元素的最后一个子元素，等同于:nth-last-child(1)

E:first-of-type 匹配父元素下使用同种标签的第一个子元素，等同于:nth-of-type(1)

E:last-of-type  匹配父元素下使用同种标签的最后一个子元素，等同于:nth-last-of-type(1)

E:only-child    匹配父元素下仅有的一个子元素，等同于:first-child:last-child或 :nth-child(1):nth-last-child(1)

E:only-of-type  匹配父元素下使用同种标签的唯一一个子元素，等同于:first-of-type:last-of-type或 :nth-of-type(1):nth-last-of-type(1)

E:empty 匹配一个不包含任何子元素的元素，文本节点也被看作子元素

E:not(selector) 匹配不符合当前选择器的任何元素

2.伪元素选择器

E::first-line   匹配E元素内容的第一行

E::first-letter 匹配E元素内容的第一个字母

E::before   在E元素之前插入生成的内容

E::after    在E元素之后插入生成的内容

七．以下选择器分别是什么意思?

#header{

}

.header{

}

.header .logo{

}

.header.mobile{

}

.header p, .header h3{

}

#header a:hover{

}

#header .logo~p{

}

#header .logo+p{

}

#header .logo p{

}

#header .logo>p{

}

#header p.logo{

}

#header .logo.p{

}

#header input[checked]{

}

1. #header{ }  
   匹配ID为header的元素  
   2..header{ } 匹配Class包含header的元素 3..header .logo{  
   }匹配class标签为header的所有class名为logo的后辈元素 4..header.mobile{  
   }  
   匹配class标签为header的元素里的以mobile作class标签的元素  
   5..header p, .header h3{ }  
   同时匹配class标签为header的所有子后代p和class标签为header的所有子后代h3的元素  
   6.#header a:hover{ }  
   ID为header的元素的所有子后代a元素设置为鼠标悬停  
   7.#header .logo~p{ }  
   匹配ID为header的元素里所有子后代class名为logo的元素的邻近同级元素p(无论直接相邻与否)  
   8.#header .logo+p{ }  
   匹配ID为header的元素里所有子后代class名为logo的元素之后直接相邻的同级元素p  
   9.#header .logo p{ }  
   匹配ID为header的元素里所有子后代class名为logo的元素的所有子后代p  
   10.#header .logo>p{ }  
   匹配ID为header的元素里所有子后代class名为logo的元素的第一代后代元素p  
   11.#header p.logo{ }  
   匹配ID为header的元素里所有子后代p中class包含logo的元素  
   12.#header .logo.p{ }  
   匹配ID为header的元素里所有子后代class名为logo的元素中class包含p的元素  
   13.#header input[checked]{ }  
   匹配ID为header的元素里所有子后代具有checked属性的input元素

评语：#header .logo~p选择器解释错误

八．选择器的优先级是如何计算的？

选择器优先级由高到低分别为：！important>行内样式>ID选择器>类选择器>标签>通配符>继承>浏览器默认属性  
-!important > 作为style属性写在元素标签上的内联样式 >id选择器>类选择器>伪类选择器>属性选择器>标签选择器> 通配符选择器（\* 应少用）>浏览器自定义；  
-当比较多个相同级别的CSS选择器优先级时，它们定义的位置将决定一切。下面从位置上将CSS优先级由高到低分为六级：  
1、位于<head/>标签里的<style/>中所定义的CSS拥有最高级的优先权。  
2、第二级的优先属性由位于 <style/>标签中的 @import 引入样式表所定义。  
3、第三级的优先属性由<link/>标签所引入的样式表定义。  
4、第四级的优先属性由<link/>标签所引入的样式表内的 @import 导入样式表定义。  
5、第五级优先的样式有用户设定。  
6、最低级的优先权由浏览器默认。

-选择器注明的越具体，某些情况下则其优先级也越高；若重复写了则以最后的为准；

CSS3中选择器的优先级如下计算：

获取选择器中ID选择器的个数，赋值给a  
获取选择器中类选择器，属性选择器([type="text"])和伪类选择器(:hover)的个数，赋值给b  
获取选择器中标签(type)选择器和伪元素选择器(::before)的个数，赋值给c  
通用选择器(\*)不占优先级权重  
将abc三个数串联成一个基数绝对大的数字(既无论数字再大也不进位)。  
eg：

#btn.red //a=1,b=1,c=0 =>specificity = 110  
.big li //a=0,b=1,c=1 =>specificity = 11  
ul li //a=0,b=0,c=2 =>specificity = 2

如果两个specificity值一样呢？那自然是看样式表中出现的先后顺序，后者可以覆盖前者的样式。

注：写于HTML中行内的样式(style="width:100px")总会覆盖外部样式表的样式，具有绝对的优先级。

!important除外

当在一个样式声明上使用 !important 规则(虽然不鼓励用)时，该样式声明会覆盖CSS中任何其他的声明(内联样式除外)。

基于形式的优先级

<style>

#id{ color: "red";}  
[id="milk"]{ color: "green"}  
</style>

<div id="milk">牛奶</div>  
最终这个div的color仍然是 red，因为 [id="milk"]是以属性选择器来添加权重的。

九．运行如下代码，并对结果做出解释

<style>

.box:first-child {

color: red;

}

.box:first-of-type {

background: blue;

}

.box :first-child {

font-size: 30px;

}

.box :first-of-type {

font-weight: bold;

}

</style>

<div class="container">

<div class="box">div.box</div>

<p class="box">p.box</p>

<div class="box">div.box</div>

<div class="box">

<div class="item">div.item</div>

<p class="item">p.item</p>

</div>

<p class="box"></p>

</div>

1.\```.box:first-child {

color: red;

}\```

box:first-child——匹配box的长子（第一个子女）元素，box标签的第一个子元素为div.box，故将其颜色变为红色。

2.\```.box:first-of-type {

background: blue;

}\```

box:first-of-type——匹配父元素下使用同种标签的第一个子元素，box父元素下仅有div和p两种标签，此两种标签各自的长子元素分别为div.box和p.box。故将div.box和p.box的背景色变为蓝色。

3.\```.box :first-child {

font-size: 30px;

}\```

box空格：X——后代选择器，用空格分隔，匹配box元素所有的子后代元素X,而X:first-child——匹配作为长子（第一个子女）的元素；

故`.box :first-child`相当于先是针对box元素所有后代（包括div.box，p.box，div.box，及<div class="item">div.item</div><p class="item">p.item</p>），然后针对上一步筛选出的这四个里各自的长子元素（div.box和p.box和div.box均没有子元素,<div class="item">div.item</div><p class="item">p.item</p>的长子元素为div.item）。此两步最终筛选出div.item，将其字号变为30px。

4.\```.box :first-of-type {

font-weight: bold;

}\```

box空格：X——后代选择器，用空格分隔，匹配box元素所有的子后代元素X,而X:first-of-type——匹配父元素下使用同种标签的第一个子元素，故`.box :first-of-type`相当于先是针对box元素所有后代（包括div.box，p.box，div.box，及<div class="item">div.item</div><p class="item">p.item</p>），然后针对上一步括号里的四个对象里选取使用同种标签的第一个子元素（因为div.box没有子后代，p.box没有子后代，div.box没有子后代，使用div class标签的仅<div class="item">div.item</div>其长子后代为div.item，使用p class标签的仅<p class="item">p.item</p>其长子后代为p.item）。故两步下来得到的元素是div.item和p.item，将它们俩字体变为bold黑体。

评语.box:first-child和.box:first-of-type的选择器解释错误