第 2 章

HTML 5

本章导读

从本章开始主要带领读者来学习前端的 HTML 5 的知识以及在面试和笔试中常见的问题。本章首先告诉读者要掌握的重点知识有哪些，然后教会读者应该如何更好地回答这些问题，最后总结了一些企业的面试及笔试中的真题。

知识清单

本章要点（已掌握的在方框中打钩）

- ☐ HTML 5 基础
- ☐ 网页标签
- ☐ 常用超链接
- ☐ 行内元素和块级元素

2.1 HTML 5 基础

本节主要讲解前端内容中 HTML 5 的基本结构、HTML 的发展史、HTML 5 的优势以及 W3C 的标准等基础知识内容。读者需要牢牢掌握这些基础知识，才能在面试及笔试中应对自如。

2.1.1 基本结构

HTML 5 的基本结构主要分为头部区域、主体区域以及结束区域。

HTML 5 是一个以.html 为扩展名的文件，下面是一个 HTML 5 的基本结构：

```
<!DOCTYPE html>          <!--用于声明、告诉浏览器这是一个 HTML 文档-->
<html lang="zh-cn">      <!--HTML 文档开始 | lang="zh-cn"中文声明-->
<head> </head>           <!--头部区域-->
<body> </body>           <!--主体区域-->
</html>                  <!-- HTML 文档结束-->
```

1. 头部区域

HTML 5 的头部区域 head 用于定义一些网页的初始化工作，如文档编码格式、网页的 title 标题以及引入 CSS 和 JavaScript 文件等。

```
<head>
    <meta charset="utf-8">   <!--文档编码格式-->
    <meta name="description" content="我是一个网页描述:) ">   <!--网页描述-->
    <title>我是标题</title>   <!-- 网页标题 -->
</head>
```

其中，title 标签定义的网页标题显示在浏览器顶部窗口的标签栏，而 meta 中的 name 为 description 定义的是网页描述不可见的，主要用于告诉搜索引擎的爬虫机器人当前页面主要的内容，适当的时候搜索引擎会给出相关词汇的搜索排名，类似的标签还有 meta 中的 keyword 和 author。

2. 主体区域

HTML 5 的主体区域 body 是浏览器的视窗区域，在这个区域写入的任何内容，都会显示在浏览器中。

```
<body>
    <p>当前是一个段落内容</p>
    <span>主体内容</span>
</body>
```

3. 标签形式

每个以<>符号组成的元素都是一个 HTML 5 标签，分为单标签和双标签。

```
<!--双标签-->
<html>   <!--< >标签开始-->
    <head>…</head>     <!--双标签内部可以包含下级标签-->
    <body>…</body>     <!--双标签内部可以包含下级标签-->
</html>   <!--</ >标签结束-->
<!--单标签-->
<br/>   <!--自闭合-->
<hr/>   <!--自闭合-->
```

4. 标签属性

标签的形式除了有单标签和双标签之外，还包括了属性，通常是作为标签功能的补充。

```
<html lang="zh-cn">   <!-- lang=" "就是标签的属性 | 而 zh-cn 称为属性值-->
<!-- 一个标签可以包含多属性 -->
<a href="http://jumooc.com" style="color: red;">聚慕课教程</a>
```

5. 严格类型

HTML 5 并不是严格类型的语言，拥有一定的容错机制，当我们把标签写错的时候，浏览器并不会直接报错不显示了，而是根据自己的理解去解析错误的内容。

```
<hr/>     <!--单标签的自闭合写成<hr>也能正确显示-->
<p </p>  <!--忘记写了一个“>”号可能也会解析正确，也可能段落的内容被错位显示-->
```

6. 代码注释

HTML 5 用于注释标签的是<!--我是注释-->。

```
<!--这是一段注释内容-->
```

2.1.2 HTML 的发展史

HTML 的英文全称是 Hypertext Marked Language，即超文本标记语言。HTML 是由 Web 的发明者 Tim Berners-Lee 和同事 Daniel W. Connolly 于 1990 年创立的一种标记语言，它是标准通用化标记语言 SGML 的应用。用 HTML 编写的超文本文档称为 HTML 文档，它能独立于各种操作系统平台（如 UNIX 和 Windows 等）。使用 HTML 语言，将所需要表达的信息按某种规则写成 HTML 文件，通过专用的浏览器来识别，并将这些 HTML 文件“翻译”成可以识别的信息，即现在所见到的网页。

自 1990 年以来，HTML 就一直被用作 WWW 的信息表示语言，使用 HTML 语言描述的文件需要通过 WWW 浏览器显示出效果。HTML 是一种建立网页文件的语言，通过标记式的指令（Tag），将影像、声音、图片、文字动画、影视等内容显示出来。事实上，每一个 HTML 文档都是一种静态的网页文件，这个文件里面包含了 HTML 指令代码，这些指令代码并不是一种程序语言，只是一种排版网页中资料显示位置的标记结构语言，易学易懂，非常简单。HTML 的普遍应用就是带来了超文本的技术，即通过单击鼠标从一个主题跳转到另一个主题，从一个页面跳转到另一个页面，与世界各地主机的文件链接。超文本传输协议规定了浏览器在运行 HTML 文档时所遵循的规则和进行的操作。HTTP 协议的制定使浏览器在运行超文本时有了统一的规则和标准。

万维网（World Wide Web）上的一个超媒体文档称之为一个页面（page）。作为一个组织或者个人在万维网上放置开始点的页面称为主页（Homepage）或首页，主页中通常包括指向其他相关页面或其他节点的指针（超级链接）。所谓超级链接，就是一种统一资源定位器（Uniform Resource Locator，URL）指针，通过激活（点击）它，可使浏览器方便地获取新的网页。这也是 HTML 获得广泛应用的最重要的原因之一。在逻辑上将视为一个整体的一系列页面的有机集合称为网站（Website 或 Site）。

网页的本质就是超级文本标记语言，通过结合使用其他的 Web 技术（如脚本语言、公共网关接口、组件等），可以创造出功能强大的网页。因而，超级文本标记语言是万维网编程的基础，也就是说万维网是建立在超文本基础之上的。超级文本标记语言之所以称为超文本标记语言，是因为文本中包含了所谓“超级链接”。

HTML 历史上有以下版本。

- HTML 1.0：在 1993 年 6 月作为互联网工程工作小组（IETF）工作草案发布。
- HTML 2.0：1995 年 11 月作为 RFC 1866 发布，在 RFC 2854 于 2000 年 6 月发布之后被宣布已经过时。
- HTML 3.2：1997 年 1 月 14 日，W3C 推荐标准。
- HTML 4.0：1997 年 12 月 18 日，W3C 推荐标准。
- HTML 4.01：1999 年 12 月 24 日，W3C 推荐标准。
- HTML 5：HTML 5 是公认的下一代 Web 语言，极大地提升了 Web 在富媒体、富内容和富应用等方面的能力，被喻为终将改变移动互联网的重要推手。

2.1.3　HTML 5 的优势

学习前端知识都需要学习 HTML 5，并知道使用它的优势是什么。下面将对它的优势进行介绍。

（1）跨平台：跨平台技术在早期大多因为性能问题没有实现，中后期硬件能力增强后又成为主流，因为跨平台确实是适合大众需要的。

（2）成本低：创业者融资并不容易，如何更高效地花钱非常重要。使用原生开发的 App 和使用 HTML 5 开发的 App 没什么区别，但是开发成本会高出一倍。

（3）快速迭代：移动互联网更新速度很快，谁对用户的需求满足得更快，谁的试错成本更低，谁就拥有巨大的优势。

（4）导流入口多：HTML 5 应用导流非常容易，超级 App（如微信朋友圈）、搜索引擎、应用市场、浏览器，到处都是 HTML 5 的流量入口。而原生 App 的流量入口只有应用市场。聪明的 HTML 5 开发者当然会玩转各种流量入口从而取得更强的优势。

（5）开源生态系统发达：HTML 5 前端是开放的正反馈循环生态系统，拥有大量的开源库，开发应用变得更轻松、更敏捷，当然这也体现在了快速迭代和成本下降上。

（6）开放的数据交换：HTML 是以 page 为单元开放代码的，无须专门开发 SDK，只要不混淆，就能与其他应用交互数据。开发者可以让手机搜索引擎很容易地检索到自己的数据，也更容易通过

跨应用协作来满足最终用户需求。

（7）持续交付：很多人有这样的体会，一个原生应用上线 Appstore，突然有一个大问题，只好尽快修复，然后静静等待一段时间进行审核。正是因为这段间隔时间，用户可能会大量流失，等新应用被审核上线了，用户可能已经卸载了该应用。但是使用 HTML 5 没有这些问题，可以实时更新，有问题立即响应。

（8）对最终用户的三大优势：大幅降低使用门槛；实时更新、提高体验；跨应用的使用体验。

上文对 HTML 5 开发语言做了详细的介绍和优势说明，希望大家在学习之后可以对 HTML 5 有一个大概认识，未来的开发技术市场将会由 HTML 5 占领。

2.1.4 W3C 标准

W3C 标准主要由 DOCTYPE、名字空间（NameSpace）、定义语言编码、CSS 定义等内容组成。

1. DOCTYPE

DOCTYPE 是 DocumentType（文档类型）的简写，用来说明使用的 XHTML 或者 HTML 是什么版本。其中的 DTD（如 xhtml1-transitional.dtd）叫作文档类型定义，包含文档的规则，浏览器根据定义的 DTD 来解释页面的标识，并展现出来。要建立符合标准的网页，DOCTYPE 声明是必不可少的关键组成部分；除非 XHTML 确定了一个正确的 DOCTYPE，否则标识和 CSS 都不会生效。

在 XHTML 1.0 中提供了三种 DTD 声明可供选择。

- 过渡的（Transitional）：要求非常宽松的 DTD，它允许继续使用 HTML 4.01 的标识（但是要符合 xhtml 的写法）。

完整代码如下所示：

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

- 严格的（Strict）：要求严格的 DTD，不能使用任何表现层的标识和属性。

完整代码如下所示：

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

- 框架的（Frameset）：专门针对框架页面设计使用的 DTD，如果页面中包含框架，需要采用这种 DTD。

完整代码如下所示：

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

☆**注意**☆　DOCTYPE 声明必须放在每一个 XHTML 文档最顶部，在所有代码和标识之上。

2. 名字空间（NameSpace）

通常在 HTML 4.0 的代码中有“xmlns”。这个“xmlns”是 XHTML NameSpace 的缩写，叫作“名字空间”声明。XHTML 是 HTML 向 XML 过渡的标识语言，它需要符合 XML 文档规则，因此也需要定义名字空间。又因为 XHTML 1.0 不能自定义标识，所以它的名字空间都相同，目前阶段只需要参照它的代码即可。

3. 定义语言编码

为了被浏览器正确解释和通过 W3C 代码校验，所有的 XHTML 文档都必须声明其所使用的编码语言，一般使用 gb2312（简体中文）。制作多国语言页面，也有可能用 Unicode、ISO-8859-1、UTF-8 等，根据所需来进行定义。

☆**注意**☆　如果忘记了定义语言编码，可能就会出现这样的情况：在 DW（dreamweaver）做完

一个页面，第二次打开时所有的中文变成了乱码。

4. CSS 定义

为保证各浏览器的兼容性，在写 CSS 时都需要写上数量单位，代码如下所示：

```
错误: .space_10 {padding-left:10}
正确: .space_10 {padding-left:10px}
```

5. 不要在注释内容中使用“--”

“--”只能发生在 XHTML 注释的开头和结束，也就是说，在内容中它们是不起任何作用的。

6. 所有标签的元素和属性的名字都必须使用小写

与 HTML 不一样，XHTML 对大小写是敏感的，所以在书写的时候需要注意大小写。

2.2　网页标签

HTML 是简单的文本标签语言，一个 HTML 网页文件都是由元素构成的，元素由开始标签、结束标签、属性和元素的内容 4 部分构成。下面介绍基本标签、图像标签和链接标签。

2.2.1　基本标签

基本标签是经常使用到的，可以说只要写前端页面就需要和其打交道。

1. <!DOCTYPE>：定义文档类型

<!DOCTYPE html>表示文档是 HTML。

代码如下所示：

```
<!DOCTYPE html>
<html>
<head>
    <title>文档标题</title>
</head>
<body>
    这里书写文档的内容
</body>
</html>
```

2. <html>：定义 HTML 文档

它是 HTML 文档的根元素。但是 HTML 5 可以对它进行省略。浏览器会以特殊的方式来使用标题，并且通常把它放置在浏览器窗口的标题栏或状态栏上。同样，当把文档加入用户的链接列表或收藏夹或书签列表时，标题将成为该文档链接的默认名称。

☆**注意**☆　<title>标签是<head>标签中唯一要求包含的东西。

代码如下所示：

```
<!DOCTYPE HTML>
<html>
<head>
    这是文档的头部
</head>
<body>
    这是文档的主体内容
</body>
</html>
```

3. <head>：定义 HTML 文档的头部

HTML 5 可以省略这个元素。

代码如下所示：

```
<head>
    这是文档的头部
</head>
```

4. <title>：定义 HTML 页面的标题

代码如下所示：

```
<!DOCTYPE HTML>
<html>
<head>
    <title>我的第一个 HTML 页面</title>
</head>
<body>
    <p>这是一个段落</p>
    <h1>这是一级标题</h1>
</body>
</html>
```

5. <body>：定义 HTML 的页面主体部分

该标签可以指定 id、class、style 等核心属性，还可以指定 onload、onunload、onclick 等事件属性。

代码如下所示：

```
<!DOCTYPE HTML>
<html>
<head>
    <title>文档的标题</title>
</head>
<body>
    文档的内容
</body>
</html>
```

6. <style>：定义引用样式

代码如下所示：

```
<!DOCTYPE HTML>
<html>
<head>
    <style type="text/css">
       h1 {color:blue}
       p {color:yellow}
    </style>
</head>
<body>
    <h1>我是蓝色</h1>
    <p>我是黄色</p>
</body>
</html>
```

7. <h1>～<h6>：定义标题 1 到标题 6

代码如下所示：

```
<!DOCTYPE HTML>
<html>
<body>
    <h1>标题 1</h1>
    <h2>标题 2</h2>
```

```
    <h3>标题 3</h3>
    <h4>标题 4</h4>
    <h5>标题 5</h5>
    <h6>标题 6</h6>
    <p>
        请仅仅把标题标签用于标题文本。
        不要仅仅为了产生粗体文本而使用它们。
        请使用其他标签或 CSS 代替。
    </p>
</body>
</html>
```

8. <p>：定义段落

该元素会自动在其前后创建一些空白。浏览器会自动添加这些空间，也可以在样式表中规定。该标签也可以指定 id、class、style 等核心属性。

代码如下所示：

```
<!DOCTYPE HTML>
<html>
<body>
    <p>这是段落。</p>
    <p>这是段落。</p>
    <p>这是段落。</p>
    <p>段落元素由 p 标签定义。</p>
</body>
</html>
```

9.
：插入一个换行

代码如下所示：

```
<!DOCTYPE HTML>
<html>
<body>
    <p>
    我<br />要<br />换<br />行<br />结束。
    </p>
</body>
</html>
```

10. <hr>：定义水平线

代码如下所示：

```
<!DOCTYPE HTML>
<html>
<body>
    <p>标签 hr 定义水平线：</p>
    <hr />
    <p>这是段落。</p>
    <hr />
    <p>这是段落。</p>
    <hr />
    <p>这是段落。</p>
</body>
</html>
```

11. <!--……-->：定义注释

代码如下所示：

```
<!DOCTYPE html>
<html>
<body>
```

```
    <!--这是一段注释，不会在浏览器中显示。-->
    <p>这是一段普通的段落</p>
</body>
</html>
```

12. ：定义文档中的 division/section

代码如下所示：

```
<!DOCTYPE html>
<html>
<body>
    <div>
        <p>这是一段普通的段落</p>
        <h1>标题 1…</h1>
    </div>
</body>
</html>
```

13. ：与 标签基本相似

该标签内容默认不会换行。

代码如下所示：

```
<!DOCTYPE html>
<html>
<body>
    <span>
        <p>这是一段普通的段落</p>
        <h1>标题 1…</h1>
    </span>
</body>
</html>
```

☆**注意**☆ <p>标签是块标签， 标签也是块标签，是内联标签。这三个都是容器标签，不同的是<p>标签自带的有段落间距，标签与 标签基本相似，该标签内容默认不会换行。

2.2.2 图像标签

图像标签是经常使用的一种标签元素，只要在页面中需要链接图片等都需要使用到图像标签，下面将会介绍图像标签。

1. <img/>：一个闭合的标签，用来定义图像

主要属性如下。

- src：图片的路径，要使用相对路径。
- alt：当图片无法打开时显示的文字。
- title：鼠标指针悬停在图片上时显示的文字。
- width：图片的宽。
- height：图片的高。

宽高可以设置为固定的像素，也可以称为百分比，按电脑屏幕的百分比显示。一般情况下都会设置宽高，避免因图片无法打开造成的错位等问题。

height 和 width 属性有一种隐藏的特性，就是无须指定图像的实际大小，也就是说，这两个值可以比实际的尺寸大一些或小一些。浏览器会自动调整图像，使其适应预留空间的大小。使用这种方法就可以很容易地为大图像创建其缩略图，以及放大很小的图像。

☆**注意**☆ 不管最终显示的尺寸到底是多大，浏览器还是必须要下载整个文件，而且，如果没有保持其原来的宽度和高度比例，图像会发生扭曲。

代码如下所示：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>图像 img 测试</title>
</head>
<body>
    <!--src 图片路径 alt 无法打开时显示的文字，title 鼠标指针悬停时显示的文字，width height 宽高-->
    <img src="01.jpg" alt="这是一幅画" title="路边开的花" width="600" height="600"/>
</body>
</html>
```

2. <map>：定义图像映射

代码如下所示：

```
<!DOCTYPE html>
<html>
<body>
    <p>请点击图片，将它们放大。</p>
    <img src="01.jpg" border="0" usemap="#planetmap" alt="Planets" />
    <map name="planetmap" id="planetmap">
    <area shape="circle" coords="180,139,14"
       href ="/example/html/venus.html" target ="_blank"alt="Venus" />
    <area shape="circle" coords="129,161,10"
       href ="/example/html/mercur.html" target ="_blank" alt="Mercury" />
    <area shape="rect" coords="0,0,110,260"
    href ="/example/html/sun.html" target ="_blank" alt="Sun" />
</map>
<p>
    <b>注释：</b>img 元素中的"usemap"属性引用 map 元素中的"id"或"name"属性（根据浏览器），所以同时向 map 元素添加了"id"和"name"属性。
</p>
</body>
</html>
```

3. <area>：定义图像映射

该属性的使用在上面的案例中已有展现。

4. <canvas>：定义图形

代码如下所示：

```
<!DOCTYPE HTML>
<html>
<body>
    <canvas id="myCanvas">canvas tag</canvas>
    <script type="text/javascript">
        var canvas=document.getElementById('myCanvas');
        var ctx=canvas.getContext('2d');
        ctx.fillStyle='#FF0000';
        ctx.fillRect(0,0,80,100);
    </script>
</body>
</html>
```

5. <figcaption>：定义 figure 元素的标题

代码如下所示：

```
<!DOCTYPE html>
<html>
<body>
    <p>这是一个段落</p>
    <figure>
```

```
        <img src="01.jpg" alt="The Pulpit Rock" width="310" height="228">
        <figcaption>图片</figcaption>
    </figure>
</body>
</html>
```

6. <figure>：定义媒介内容的分组，以及它们的标题

代码如下所示：

```
<!DOCTYPE HTML>
<html>
<body>
    <p>这是一个女孩</p>
    <figure>
        <p>女孩</p>
        <p>拍摄者：聚慕课，拍摄时间：2019 年 10 月</p>
        <img src="01.jpg" width="350" height="234" />
    </figure>
</body>
</html>
```

2.2.3 链接标签

链接标签可以让用户使用链接来链接一个网站、一个图片等。下面对链接标签进行介绍。

1. <a>：定义锚

（1）常用属性如下。

- href：最重要的属性，指定链接的目标。如果不使用 href 属性，则不能使用 hreflang、media、rel、target 以及 type 属性。
- target：规定在何处打开目标 URL，可与 iframe 标签中的 name 属性配合使用（仅在 href 属性存在时使用）。
- type：规定目标 URL 的 mime 类型（仅在 href 属性存在时使用）。
- rel：规定当前文档与目标 URL 之间的关系（仅在 href 属性存在时使用）。

（2）类别：文本级标签。

（3）作用：定义超链接，用于从一个页面链接到另一个页面。

（4）说明：未被访问的链接带有下画线而且是蓝色的，已被访问的链接带有下画线而且是紫色的，活动链接带有下画线而且是红色的。通常在当前浏览器窗口中显示被链接页面，除非规定了其他 target，标签既可以是超链接，也可以是锚。在 HTML 5 中，<a>标签是超链接，但是假如没有 href 属性，它仅仅是超链接的一个占位符。

（5）全局属性：支持。

（6）事件属性：支持。

<a>标签的几种常用用法，代码如下所示：

```
<!--普通超链接-->
<a href="http://www.jumooc.com/" target="_blank" rel="search">点击我</a>
<!--图像超链接-->
<a href="http://www.jumooc.com/" target="_blank" rel="search">
    <img src="XXX.jpg" alt="这张图片可能有问题"/>
</a>
<!--锚点定位-->
<a href="#跳转位置对应标签的 id 值">
<!--返回顶部-->
<a href="#">
<!--
```

```
    失效超链接：
    使用 javascript:void(0)的目的是为了阻止 a 链接的默认行为，方便让 js 绑定事件不受干扰
-->
<a href="javascript:void(0)">失效超链接</a>
```

2. <link>：定义文档与外部资源的关系

（1）常用属性如下。

- href：定义被链接文档的位置。
- type：规定被链接文档的 mime 类型。
- rel：必需的，定义当前文档与被链接文档之间的关系。

（2）作用：定义文档与外部资源的关系。

（3）说明：该标签最常见的用途是链接样式表（CSS），它只能存在于<head>部分，不过它可出现任何次数。

（4）全局属性：支持。

（5）事件属性：支持。

代码如下所示：

```
<!DOCTYPE HTML>
<html>
<head>
    <link rel="stylesheet" type="text/css" href="test.css">
</head>
<body>
    <h1>我通过外部样式表进行格式化。</h1>
    <p>我也是</p>
</body>
</html>
```

3. <nav>：定义导航链接

（1）类别：容器级标签。

（2）作用：定义导航链接的部分。

（3）说明：语义化标签，更有利于代码的阅读和搜索引擎的识别，并不是所有的 HTML 文档都要使用到<nav>标签。<nav>标签只是作为标注一个导航链接的区域（代替 DIV 布局中的 ）。

（4）全局属性：支持。

（5）事件属性：支持。

☆**注意**☆　如果文档中有“前后”按钮，则应该把它放到<nav>标签中。

代码如下所示：

```
<!DOCTYPE html>
<html>
<body>
 <nav id="nav">
    <ul>
        <li>标题一</li>
        <li>标题二</li>
        <li>标题三</li>
        <li>标题四</li>
    </ul>
 </nav>
</body>
</html>
```

2.3 常用超链接

根据前面介绍的前端基础知识，本节将介绍一些页面间链接和锚链接。

2.3.1 页面间链接

网页设计中，链接标记<a>用于实现超级链接。<a>标记是成对标记，主要属性如下。

- href：定义链接文件的地址。
- target：定义链接目标的位置。

（1）指向电子邮件的链接。在 HTML 页面中，可以建立 E-mail 链接。当浏览者单击链接后，系统会启动默认的本地邮件服务系统发送邮件。

```
例如：<a href="XXX@qq.com">与我联系</a>。
```

（2）指向站点内部文件的链接。站点内部文件的链接是指在同一个网站内部不同的 HTML 页面之间的链接关系。在建立网站内部文件的链接时，要明确哪个页面是当前页，哪个页面是链接的目标文件。内部链接一般采用相对路径。

```
例如：<a href="index.html">动物森林</a>。
```

（3）指向站点外部文件的链接。站点外部链接是指跳转到当前网站外部，与其他网站中的页面或元素之间的链接关系。这种链接的 URL 地址一般采用绝对路径，要有完整的 URL 地址，包括协议名、主机名、文件所在主机上的位 N 路径及文件名。

```
例如:<a href="http//www.jumooc.com">聚慕课链接<a>。
```

（4）指向本页面中特定部分的链接。一个页面的内容较长，不能在一个窗口中满屏全部显示时，可使用指向页内的超级链接进行跳转，可以先定义锚点，再设置锚点的超级链接，过程如图 2-1 所示。

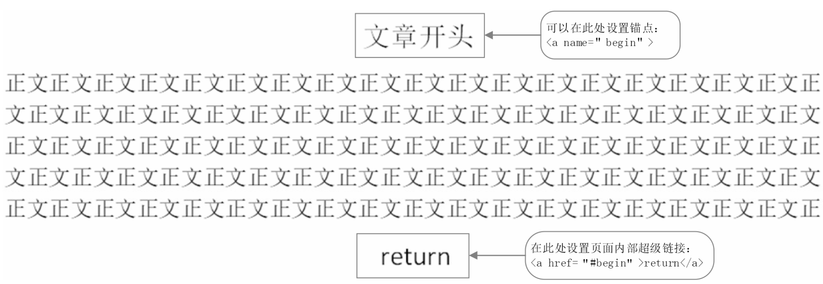

图 2-1 设置锚点

（5）指向空链接。空链接是指未指定链接对象的链接，可用于向页面中的对象附加行为。

```
例如：<a href="#">首页<a>。
```

2.3.2 锚链接

锚链接实际上就是链接文本，又叫锚文本，可以理解为关键词上带超链接的一种链接方式，且这个词本身可以作为对指向目标页面的内容概述。但是单页站锚链接与锚文本有所不同。

一般来讲，网站页面中增加的锚链接都和页面本身的内容有一定的必然联系。例如，行业网站上一般会增加一些同行网站的链接或者一些做网站建设的知名设计网站的链接。

下面介绍锚链接的几种方式。

1. 图片锚链接

代码语法如下所示：

```
<a href="链接的地址" target="_blank" >
    <img scr="图片地址 url" width="图片宽度" height="图片高度" alt="图片说明" border="0">
</a>
```

2. 文字锚链接

代码语法如下所示：

```
<a href="链接的地址" target="_blank">输入文字</a>
```

3. LOGO 锚链接

代码语法如下所示：

```
<img src="图片 logo 地址" border="0">
```

☆**注意**☆　把以下代码加在已经存在的面板里（想选哪种方式就加哪段代码，不要全部都加）。

链接的样式代码如下所示：

（1）使链接变色。

```
<style type="text/css">
a { text-decoration: none; color: #51bfe0}
a:hover { color: #3399FF }
</style>
```

（2）使链接字体变粗。

```
<style type="text/css">
a { text-decoration: none; color: #51bfe0}
a:hover {font-weight: bold }
</style>
```

（3）触到链接时出现虚线。

```
<style type="text/css">
a { text-decoration: none; color: #51bfe0}
a:hover {border-bottom:1px dashed #51bfe0 }
</style>
```

（4）超链接的位置移动。

```
<style type="text/css">
a { text-decoration: none; color: #51bfe0}
a:hover { position: relative; left:1px; top:1px; }
</style>
```

（5）给链接添加背景色。

```
<style type="text/css">
a { text-decoration: none; color: #51bfe0}
a:hover { background-color: #CCFFFF;}
</style>
```

锚链接代码如下所示：

```
<!DOCTYPE html>
<html>
<body>
    <p><a href="#">查看链接</a>
    </p>
    <h2>链接 1</h2>
    <p>我是链接 2</p>
    <h2>链接 2</h2>
```

```
</body>
</html>
```

2.4 行内元素和块级元素

在学习或者使用前端框架的时候也会经常遇到行内元素和块级元素，如果不能够详细地了解它们，很容易出现写的代码或者样式没有效果的情况。下面就来了解一下行内元素和块元素之间的定义以及区别。

1. 行内元素

行内元素和其他元素都在一行上。行高及外边距和内边距不可改变。宽度就是其文字或图片的宽度，不可改变。内联元素只能容纳文本或者其他内联元素。

对行内元素，需要注意以下几点。

- 设置宽度 width 无效。
- 设置高度 height 无效，可以通过 line-height 来设置。
- 设置 margin 时，只有左右 margin 有效，上下无效。
- 设置 padding 时，只有左右 padding 有效，上下无效。注意元素范围是增大了，但是对元素周围的内容是没影响的。

通过 display 属性对行内元素和块级元素进行切换，常见的值如表 2-1 所示。

表 2-1 基本数据类型

值	描　述
block	此元素将显示为块级元素，此元素前后会带有换行符
inline	默认。此元素会被显示为内联元素，元素前后没有换行符
inline-block	行内块元素

常见的 HTML 行内元素如表 2-2 所示。

表 2-2 常见的 HTML 行内元素

标　签	描　述	标　签	描　述
<a>	定义锚	<embed>	定义外部交互内容或插件
<abbr>	定义缩写	<i>	定义斜体字
<acronym>	定义只取首字母的缩写	<img>	定义图像
<b>	定义粗体字	<input>	定义输入控件
<bdo>	定义文字方向	<kbd>	定义键盘文本
<big>	定义大号文本	<label>	定义 input 元素的标注
 	定义简单的折行	<map>	定义图像映射
<button>	定义按钮（push button）	<mark>	定义有记号的文本
<cite>	定义引用（citation）	<object>	定义内嵌对象
<code>	定义计算机代码文本	<progress>	定义任何类型的任务的进度
<command>	定义命令按钮	<q>	定义短的引用

续表

标　签	描　述	标　签	描　述
<dfn>	定义项目	<samp>	定义计算机代码样本
<del>	定义被删除文本	<select>	定义选择列表（下拉列表）
	定义强调文本	<small>	定义小号文本
<textarea>	定义多行的文本输入控件		定义强调文本
<time>	定义日期/时间	<sup>	定义上标文本
<tt>	定义打字机文本	<sub>	定义下标文本
<var>	定义文本的变量部分		定义文档中的节
<video>	定义视频	<wbr>	定义可能的换行符

行内元素代码如下所示：

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>行内元素案例测试</title>
   <style type="text/css">
     span {
       width: 120px;
       height: 120px;
       margin: 1000px 20px;
       padding: 50px 40px;
       background: lightblue;
       }
   </style>
</head>
<body>
   <i>不会自动换行</i>
   <span>行内元素</span>
</body>
</html>
```

行内元素运行效果如图 2-2 所示。

图 2-2　行内元素运行效果

2. 块级元素

块级元素会占据新的一行。块级元素中可以包含块级元素和行内元素。块级元素可以设置宽高，并且宽度、高度以及外边距、内填充都可随意控制。宽度没有设置时，默认为 100%。

对块级元素，需要注意以下几点。

- 总是在新的一行开始。
- 行高以及外边距和内边距都可控制。
- 宽度缺省时，是它的容器的 100%，除非设定一个宽度。
- 可以容纳内联元素和其他块元素。

常见的 HTML 块级元素如表 2-3 所示。

表 2-3　常见的 HTML 块级元素

标　签	描　述	标　签	描　述
<address>	定义地址	<dd>	定义列表中项目的描述
<article>	定义文章		定义文档中的节
<aside>	定义页面内容之外的内容	<dl>	定义列表
<audio>	定义声音内容	<dt>	定义列表中的项目
<blockquote>	定义长的引用	<details>	定义元素的细节
<canvas>	定义图形	<fieldset>	定义围绕表单中元素的边框
<caption>	定义表格标题	<figcaption>	定义 figure 元素的标题
<figure>	定义媒介内容的分组，以及它们的标题	<footer>	定义 section 或 page 的页脚
<form>	定义供用户输入的 HTML 表单	<h1>to<h6>	定义 HTML 标题
<header>	定义 section 或 page 的页眉	<hr>	定义水平线
<legend>	定义 fieldset 元素的标题	<menu>	定义命令的列表或菜单
	定义列表的项目	<meter>	定义预定义范围内的度量
<nav>	定义导航链接	<noframes>	定义针对不支持框架的用户的替代内容
<noscript>	定义针对不支持客户端脚本的用户的替代内容		定义有序列表
<output>	定义输出的一些类型	<p>	定义段落
<pre>	定义预格式文本	<section>	定义 section
<table>	定义表格	<tbody>	定义表格中的主体内容
<td>	定义表格中的单元	<tfoot>	定义表格中的表注内容（脚注）
<th>	定义表格中的表头单元格	<thead>	定义表格中的表头内容
<time>	定义日期/时间	<tr>	定义表格中的行
	定义无序列表		

块级元素代码如下所示：

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>块级元素测试案例</title>
   <style type="text/css">
     div {
         width: 120px;
         height: 120px;
         margin: 50px 50px;
         padding: 50px 40px;
         background: lightblue;
       }
   </style>
</head>
<body>
   <i>自动换行</i>
   <div>块级元素</div>
   <div>块级元素</div>
</body>
</html>
```

块级元素运行效果如图 2-3 所示。

图 2-3　块级元素运行效果

下面介绍 HTML 中的行内元素、块状元素和行内块状元素之间的区别。

HTML 可以将元素分类方式分为行内元素、块状元素和行内块状元素三种。它们之间使用 display 可以进行相互转换。

- display:inline，转换为行内元素。
- display:block，转换为块状元素。
- display:inline-block，转换为行内块状元素。

代码如下所示：

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>测试案例</title>
   <style type="text/css">
     span {
       display: block;
       width: 120px;
       height: 30px;
       background: red;
     }
     div {
       display: inline;
       width: 120px;
       height: 200px;
       background: green;
     }
     i {
       display: inline-block;
       width: 120px;
       height: 30px;
       background: lightblue;
     }
   </style>
</head>
<body>
   <span>行内转块状</span>
   <div>块状转行内 </div>
   <i>行内转行内块状</i>
</body>
</html>
```

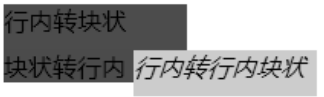

图 2-4　行内元素、块状元素以及行内块状元素运行效果

块级元素运行效果如图 2-4 所示。

2.5　面试与笔试试题解析

前端开发是创建 Web 页面或 App 等前端界面呈现给用户的过程，通过 HTML、CSS 和 JavaScript 以及衍生出来的各种技术、框架、解决方案，来实现互联网产品的用户界面交互。它从网页制作演变而来，名称上有很明显的时代特征。随着互联网技术的发展和 HTML 5、CSS 3 的应用，现代网页更加美观，交互效果显著，功能更加强大。下面将会介绍一些精选面试与笔试题解析，可以帮助读者了解和熟悉面试与笔试题类型。

2.5.1　怎样区分 HTML 5、HTML 和 XHTML

题面解析：本题主要考查应聘者对 HTML 5、HTML 和 XHTML 之间的熟练程度。看到此问题，

应聘者应该对三者的定义以及使用时的关键注意点来区分它们之间的不同之处。

解析过程：

了解 HTML 5、HTML 和 XHTML 之间的基本概念，然后对三者进行比较。

基本概念如下。

- HTML 5：是 HTML、XHTML 和 HTML DOM 的新标准。HTML 5 是最先由 WHATWG（Web 超文本应用技术工作组）命名的一种超文本标记语言，随后和 W3C 的 xhtml 2.0 相结合，产生现在最新一代的超文本标记语言。可以简单点理解成：HTML 5 = HTML + CSS 3 + html + JavaScript + API。
- HTML：超文本标记语言，是一种基本的 Web 网页设计语言。
- XHTML：可扩展超文本标记语言，是一种置标语言，表现方式与超文本标记语言（HTML）类似，不过语法上更加严格。从本质上来说，XHTML 是一个过渡技术，结合了部分 XML 的强大功能以及大多数 HTML 的简单特性。

特性区别如下。

- HTML 5：①用于绘画的 canvas 元素。②用于媒介回放的 video 和 audio 元素。③对本地离线存储有更好的支持。④新的特殊内容元素，如 article、footer、header、nav、section。⑤新的表单控件，如 calendar、date、time、email、url、search。⑥在语义上有很大的优势，提供了一些新的标签，如<header><article><footer>提供了语义化标签，可以更好地支持搜索引擎的读取，便于 SEO 的蜘蛛的爬行。
- HTML：①标识文本，如定义标题文本、段落文本、列表文本、预定义文本。②建立超链接，便于页面链接的跳转。③创建列表，把信息有序地组织在一起以方便浏览。④在网页中显示图像、声音、视频、动画等多媒体信息，把网页设计得更富冲击力。⑤可以制作表格，以便显示大量数据。⑥可以制作表单，允许在网页内输入文本信息，执行其他用户操作，方便信息互动。⑦没有体现结构语义化的标签，通常都是这样来命名的： ，表示网站的头部。
- XHTML：①要求正确嵌套。②所有元素必需关闭。③区分大小写。④属性值要用双引号。⑤用 id 属性代替 name 属性。⑥特殊字符的处理。⑦可以很好地处理各大浏览器的兼容问题。

2.5.2 请阐述你对 W3C 的理解

题面解析：本题是对 W3C 知识点的考查，应聘者在回答该问题时，不能按照定义直接背出来，而是要阐述自己对 W3C 概念的理解。

解析过程：

下面从 Web 的标准规范、脚本语言和样式等来进行阐述。

（1）Web 标准规范要求：书写标签必需闭合、标签小写、不乱嵌套，可提高搜索机器人对网页内容的搜索概率。

（2）建议使用外链 CSS 和 JavaScript 脚本，从而达到结构与行为、结构与表现的分离，提高页面的渲染速度，能更快地显示页面的内容。

（3）不需要变动页面内容，便可提供打印版本而不需要复制内容，提高网站易用性。遵循 W3C 制定的 Web 标准，能够使用户浏览者更方便，使网页开发者之间有更好的交流。

（4）样式与标签的分离，更合理的语义化标签，使内容能被更多的用户所访问和更广泛的设备所访问，使用更少的代码和组件，从而降低维护成本，且改版更为方便。

2.5.3　HTML 文档中的 DOCTYPE 有什么作用

题面解析：本题主要考查应聘者对 HTML 文档中的 DOCTYPE 的作用的了解程度，每次在编写前端页面的时候都会存在 DOCTYPE，细心的读者可能已经了解其作用，那么不了解的读者可以跟着这道面试题进行简单的了解。

解析过程：

（1）DOCTYPE 的定义。DOCTYPE 是 DocumentType 的简写，它并不是 HTML 标签，也没有结束标签。它是一种标记语言的文档类型声明，即告诉浏览器当前 HTML 是用什么版本编写的。

☆**注意**☆　DOCTYPE 的声明必须是 HTML 文档的第一行，位于 HTML 标签之前。大多数 Web 文档的顶部都有 DOCTYPE 声明，它是在新建一个文档时，由 Web 创作软件草率处理的众多细节之一。很少人会去注意 DOCTYPE，但在遵循标准的任何 Web 文档中，它都是一项必需的元素。DOCTYPE 会影响代码验证，并决定了浏览器最终如何显示用户的 Web 文档。

（2）DOCTYPE 的作用。DOCTYPE 在 Web 设计中用来声明文档类型。在所有 HTML 文档中规定 DOCTYPE 是非常重要的，这样浏览器就能了解预期的文档类型，告诉浏览器要通过哪一种规范（DTD）解析文档（如 HTML 或 XHTML 规范）。

2.5.4　DOCTYPE 文档类型有几种

题面解析：本题主要考查在使用页面编写代码时出现的 DOCTYPE 的相关内容，在不同的版本中规定的类型也是不同的。下面简单介绍在 HTML 标签中可以声明的三种类型。

解析过程：

标签可声明三种 DTD 类型，分别表示严格版本（Strict）、过渡版本（Transitional）以及基于框架（Frameset）的 HTML 文档。

HTML 4.01 规定了三种文档类型：Strict、Transitional 以及 Frameset。

XHTML 1.0 规定了三种 XML 文档类型：Strict、Transitional 以及 Frameset。

Standards（标准）模式（也就是严格呈现模式）用于呈现遵循最新标准的网页，Quirks（包容）模式（也就是松散呈现模式或者兼容模式）用于呈现为传统浏览器而设计的网页。

2.5.5　Quirks 模式是什么？它和 Standards 模式的区别

题面解析：本题主要考查 HTML 的 DTD 类型中 Quirks 模式和 Standards 模式，应聘者可以简单介绍 Quirks 和 Standards 的定义是什么，然后再回答两者之间的区别。

解析过程：

（1）Quirks 模式的定义。

在写程序时，程序员会经常遇到这样的问题：如何保证原来的接口不变，又提供更强大的功能，尤其是新功能不兼容旧功能时。IE 6 以前的页面一般都不会去写 DTD，所以 IE 6 就假定如果写了 DTD，就意味着这个页面将采用对 CSS 支持更好的布局，而如果没有，则采用兼容之前的布局方式。这就是 Quirks 模式（怪癖模式，诡异模式，怪异模式）。

（2）区别如下。

- 盒模型：在 W3C 标准中，如果设置一个元素的宽度和高度，指的是元素内容的宽度和高度；在 Quirks 模式下，IE 的宽度和高度还包含了 padding 和 border。
- 设置百分比的高度：在 Standards 模式下，一个元素的高度是由其包含的内容来决定的，如果父元素没有设置百分比的高度，子元素设置一个百分比的高度是无效的。

- 设置行内元素的高宽：在 Standards 模式下，给行内元素设置 width 和 height 都不会生效，而在 Quirks 模式下，则会生效。
- 设置水平居中：使用 margin:0 auto 在 Standards 模式下可以使元素水平居中，但在 Quirks 模式下却会失效。

2.5.6 HTTP 状态码

试题题面：在 HTTP 中，总共有 5 类状态码，请简单介绍一下这 5 类状态码。

题面解析：本题主要考查应聘者对状态码的了解。应聘者可以根据自己在项目练习时遇到的状态码进行回答。

解析过程：

通过 HTTP 状态码可以很方便地了解请求的状态。状态码是面试题中经常出现的，所以很有必要总结一下，对读者今后的面试会很有帮助。

HTTP 状态码分为 5 大类。

（1）以 1 开头：信息状态码，如表 2-4 所示。

表 2-4 信息状态码

状 态 码	含 义	描 述
100	继续	初始的请求已经接受，请客户端继续发送剩余部分
101	切换协议	要求服务器切换协议，服务器已确定切换

（2）以 2 开头：成功状态码，如表 2-5 所示。

表 2-5 成功状态码

状 态 码	含 义	描 述
200	成功	服务器已成功处理了请求
201	已创建	请求成功并且服务器创建了新的资源
202	已接受	服务器已接受请求，但尚未处理
203	非授权信息	服务器已成功处理请求，但返回的信息可能来自另一个来源
204	无内容	服务器成功处理了请求，但没有返回任何内容
205	重置内容	服务器处理成功，用户终端应重置文档视图
206	部分内容	服务器成功处理了部分 GET 请求

（3）以 3 开头：重定向状态码，如表 2-6 所示。

表 2-6 重定向状态码

状 态 码	含 义	描 述
300	多种选择	针对请求，服务器可执行多种操作
301	永久移动	请求的页面已永久跳转到新的 URL
302	临时移动	服务器目前从不同位置的网页响应请求，但请求仍继续使用原有位置来进行以后的请求
303	查看其他位置	请求者应当对不同的位置使用单独的 GET 请求来检索响应时，服务器返回此代码

续表

状　态　码	含　　义	描　　述
304	未修改	自从上次请求后，请求的网页未修改过
305	使用代理	请求者只能使用代理访问请求的网页
307	临时重定向	服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求

（4）以 4 开头：客户端错误状态码，如表 2-7 所示。

表 2-7　客户端错误状态码

状　态　码	含　　义	描　　述
400	错误请求	服务器不理解请求的语法
401	未授权	请求要求用户的身份验证
403	禁止	服务器拒绝请求
404	未找到	服务器找不到请求的页面
405	方法禁用	禁用请求中指定的方法
406	不接受	无法使用请求的内容特性响应请求的页面
407	需要代理授权	请求需要代理的身份认证
408	请求超时	服务器等候请求时发生超时
409	冲突	服务器在完成请求时发生冲突
410	已删除	客户端请求的资源已经不存在
411	需要有效长度	服务器不接受不含有效长度表头字段的请求
412	未满足前提条件	服务器未满足请求者在请求中设置的其中一个前提条件
413	请求实体过大	由于请求实体过大，服务器无法处理，因此拒绝请求
414	请求 URL 过长	请求的 URL 过长，服务器无法处理
415	不支持格式	服务器无法处理请求中附带媒体格式
416	范围无效	客户端请求的范围无效
417	未满足期望	服务器无法满足请求表头字段要求

（5）以 5 开头：服务端错误状态码，如表 2-8 所示。

表 2-8　服务端错误状态码

状　态　码	含　　义	描　　述
500	服务器错误	服务器内部错误，无法完成请求
501	尚未实施	服务器不具备完成请求的功能
502	错误网关	服务器作为网关或代理出现错误
503	服务不可用	服务器目前无法使用
504	网关超时	网关或代理服务器，未及时获取请求
505	不支持版本	服务器不支持请求中使用的 HTTP 协议版本

2.5.7 什么是 IP 地址

题面解析：本题主要考查应聘者是否了解协议，了解计算机，每台计算机都会有一个 IP 地址，甚至通过 IP 地址可以让两台计算机进行连接。

解析过程：

了解 IP 地址之前，先了解 IP 协议。

- IP 协议：了解 IP 地址，先了解 IP 协议。IP 协议是为计算机网络相互连接进行通信而设计的协议。IP 协议中有一个非常重要的内容，那就是给因特网上的每台计算机和其他设备都规定了一个唯一的地址，叫作“IP 地址”。
- IP 地址：IP 地址（Internet Protocol Address）是指互联网协议地址，又译为网际协议地址。IP 地址是 IP 协议提供的一种统一的地址格式，它为互联网上的每一个网络和每一台主机分配一个逻辑地址，以此来屏蔽物理地址的差异。

2.5.8 浏览器内核

试题题面：常见的浏览器内核有哪些？简单介绍一下你对浏览器内核的理解？

题面解析：本题主要考查应聘者对浏览器以及浏览器内核是否了解。建议应聘者可以关注一下这方面的知识点，做到应聘时游刃有余。

解析过程：

浏览器内核的英文是 Rendering Engine，中文翻译有很多种，如排版引擎、解释引擎、渲染引擎，现在流行称为浏览器内核。

浏览器内核就是用来渲染网页内容的，将开发者写的代码转换为用户可以看见的完美页面。由于会牵扯到排版问题，可能会出现排版错位等问题。有的是由于网站本身编写不规范，有的是由于浏览器本身的渲染不标准。现在有几个主流的排版引擎，因为这些排版引擎都有其代表的浏览器，所以常常会把排版引擎的名称和浏览器的名称混用，如常说的 IE 内核、Chrome 内核。因为一个完整的浏览器不会只有一个排版引擎，还有自己的界面框架和其他的功能相辅相成，而排版引擎本身也不可能实现浏览器的所有功能。

下面讲解几款主流的浏览器和浏览器内核，如表 2-9 所示。

表 2-9 主流的浏览器和浏览器内核

浏 览 器	描 述	浏览器内核	描 述
IE 浏览器（Internet Explorer）	诞生于 1994 年夏天，是现在主流的浏览器之一，使用 Trident 内核	Trident 内核	IE 浏览器以 Trident 作为内核引擎（遨游、世界之窗和腾讯 TT 都是 IE）。Trident 内核最慢
Chrome 浏览器	Google Chrome 是一款由 Google 公司开发的网页浏览器，具有速度快、不容易崩溃、灵活、更加安全等特点，同时还有很多各式各样的插件。以前使用 WebKit 内核，现在使用 Blink 内核	Blink 内核	2013 年，谷歌宣布从 WebKit 分支出来，创建了渲染引擎 Blink
Firefox 浏览器（火狐）	Mozilla 公司旗下浏览器，其内核 Gecko 是开源的，因此受到了很大的欢迎	Gecko 内核	开放源代码，以 C++编写的网页排版引擎，是跨平台的。FireFox 基于 Gecko 开发

续表

浏　览　器	描　　述	浏览器内核	描　　述
Safari 浏览器	苹果公司的 Safari 浏览器，使用的是 WebKit 内核	Webkit 内核（Safari 内核，Chrome 内核原型，开源）	WebKit 是一个开源项目，是现在流行的渲染引擎之一，很多浏览器都在使用它，比如苹果的 Safari
Opera 浏览器	Opera 浏览器是挪威 Opera Software ASA 公司制作的支持多页面标签式浏览的网络浏览器，是跨平台浏览器，可以在 Windows、Mac 和 Linux 三个操作系统平台上运行。Opera 浏览器创始于 1995 年 4 月	Blink 内核	Opera 浏览器跟随 Chrome 使用 Blink 内核

浏览器内核的理解：渲染引擎（layout engineer 或 Rendering Engine）和 JavaScript 引擎。

（1）渲染引擎：负责取得网页的内容（HTML、XML、图像等）、整理讯息（如加入 CSS 等），以及计算网页的显示方式，然后会输出至显示器或打印机。浏览器的内核的不同对网页的语法解释会有不同，所以渲染的效果也不相同。所有网页浏览器、电子邮件客户端以及其他需要编辑、显示网络内容的应用程序都需要内核。

（2）JavaScript 引擎：解析和执行 JavaScript 来实现网页的动态效果。一开始，渲染引擎和 JavaScript 引擎并没有很明确的区分，后来 JavaScript 引擎越来越独立，内核就倾向于只指渲染引擎。

2.5.9　行内元素和块级元素

试题题面：行内元素有哪些？块级元素有哪些？行内元素和块级元素有什么区别？

题面解析：本题主要考查应聘者对基本的 HTML 语言是否了解，在平时编写前端页面代码的时候都会对行内元素和块级元素进行使用。应聘者在回答之前可以先回答行内元素和块级元素分别常用的有哪些，然后回答它们之间的区别。

解析过程：

行内元素大多数为描述性标记，如表 2-10 所示。

表 2-10　常见的行内元素

标　　签	描　　述	标　　签	描　　述
…	定义文档中的节	…	加粗
<a>…	链接	<img>	图片
 	换行	[…]	上标
…	加粗	_…	下标
…	斜体	…	斜体
~~…~~	删除线	<u>…</u>	下画线
<input>…</input>	文本框	<textarea>…</textarea>	多行文本
<select>…</select>	下拉列表		

块级元素大多数为结构性标记，如表 2-11 所示。

表 2-11　常见的块级元素

标　签	描　述	标　签	描　述
<address>…</address>	定义地址	…	地址文字
…	标题一级	<hr>	水平分割线
…	标题二级	…	段落
…	标题三级	<pre>…</pre>	预格式化
…	标题四级	<blockquote>…</blockquote>	段落缩进前后 5 个字符
…	标题五级	<marquee>…</marquee>	滚动文本
…	标题六级	…	无序列表
…	有序列表	<dl>…</dl>	定义列表
<table>…</table>	表格	<form>…</form>	表单
…	定义文档中的节		

行内元素和块级元素之间的区别如下。

- 块级元素会独占一行，其宽度自动填满其父元素宽度。行内元素不会独占一行，相邻的行内元素会排列在同一行里，在一行不能完全排列时，才会换行，其宽度随元素的内容而变化。
- 一般情况下，块级元素可以设置 width 和 height 属性，行内元素设置 width 和 height 无效。块级元素可以设置 margin 和 padding，行内元素的水平方向的 padding-left、padding-right、margin-left 和 margin-right 都产生边距效果，但是竖直方向的 padding-top、padding-bottom、margin-top 和 margin-bottom 都不会产生边距效果（水平方向有效，竖直方向无效）。

☆**注意**☆　块级元素即使设置了宽度，仍然是独占一行的。

2.5.10　link 和@import

试题题面：页面导入样式时，使用 link 和@import 有什么区别？

题面解析：本题主要考查应聘者对前端页面编写时使用链接 link 和导入包 import 之间的区别，应聘者可以先想什么情况下使用 link，什么情况下使用 import，然后从中找出它们之间的不同之处。

解析过程：

link 和@import 的区别有以下几点。

- 从属关系区别：@import 是 CSS 提供的语法规则，只有导入样式表的作用。link 是 HTML 提供的标签，不仅可以加载 CSS 文件，还可以定义 RSS、REL 连接属性等。
- 加载顺序区别：加载页面时，link 标签引入的 CSS 被同时加载，@import 引入的 CSS 将在页面加载完毕后被加载。
- 兼容性区别：@import 是 CSS 2.1 才有的语法，故只可在 IE 5+才能识别，link 标签作为 HTML 元素，不存在兼容性问题。
- DOM 可控性区别：可以通过 JavaScript 操作 DOM，插入 link 标签来改变样式；由于 DOM 方法是基于文档的，无法使用@import 的方式插入样式。

2.5.11　HTML 5 新特性和浏览器兼容

试题题面：HTML 5 中有哪些新特性？如何处理 HTML 5 新标签的浏览器兼容问题？

题面解析：本题主要考查应聘者对 HTML 5 的新特性的了解，以及编写代码时会遇到的浏览器有时不兼容的问题。

解析过程：

HTML 5 新特性有以下几点。

- 离线缓存，可以在关闭浏览器后再次打开时恢复数据，以减少网络流量。
- 音频、视频自由嵌入，多媒体形式更为灵活。
- 地理定位。地理位置定位，让定位和导航不再专属导航软件，地图也不用下载非常大的地图包，可以通过缓存来解决，较为灵活。
- Canvas 绘图，提升移动平台的绘图能力。使用 Canvas API 可以简单绘制热点图收集用户体验资料，支持图片的移动、旋转、缩放等常规编辑。
- 丰富的交互方式。提升互动能力，拖曳、撤销历史操作、文本选择等。
- 开发及维护成本低，这是相对于原生 App 开发来说的。更低的开发及维护成本使页面变得更小，减少了用户不必要的支出；而且，性能更好使耗电量更低。
- CSS 3 视觉设计师的辅助利器的支持。CSS 3 支持了字体的嵌入、版面的排版，以及最令人印象深刻的动画功能。
- HTML 5 调用手机摄像头和手机相册、通讯录等功能。

当在页面中使用 HTML 5 新标签时，可能会得到三种不同的结果。

- 结果 1：新标签被当作错误处理并被忽略，在 DOM 构建时会当作这个标签不存在。
- 结果 2：新标签被当作错误处理，并在 DOM 构建时，这个新标签会被构造成行内元素。
- 结果 3：新标签被识别为 HTML 5 标签，然后用 DOM 节点对其进行替换。

解决办法有以下两种。

方法一：实现标签被识别。

通过 document.createElement（tagName）方法即可让浏览器识别新标签，浏览器支持新标签后，还可以为新标签添加 CSS 样式。

方法二：JavaScript 解决方案。

使用 HTML 5 shim：

在<head>中调用以下代码：

```
    <script> src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>
```

当然也可以直接把这个文件下载到自己的网站上，但这个文件必须在 head 标签中调用。

使用 kill IE 6：

在</body>之前调用以下代码：

```
    <script src="http://letskillie6.googlecode.com/svn/trunk/letskillie6.zh_CN.pack.js">
</script>
```

2.5.12　如何实现浏览器内多个标签页之间的通信

题面解析：本题主要考查应聘者对数据库存储的知识是否了解。数据存储有本地和服务器存储两种方式，对前端开发来讲，只需要讲解用本地存储的方式来解决就好。当然如果知道服务器端的方式更好。本题的难易程度一般，只要能够说出思路就可以，至少说出两种解决方法。

解析过程：

实现浏览器内多个标签页之间的通信有以下两种方法。

（1）使用 localStorage：在一个标签页里面使用 localStorage.setItem(key,value)，可监听添加、修改、删除的动作，即可得到 localstorge 存储的值，实现不同标签页之间的通信。

语法代码如下所示：

```
<script type="text/javascript">
    $(function(){
      window.addEventListener("storage", function(event){
        console.log(event.key + "=" + event.newValue);
        });
    });
</script>
```

（2）使用 cookie+setInterval()：将要传递的信息存储在 cookie 中，每隔一定时间读取 cookie 信息，即可随时获取要传递的信息。

语法代码如下所示：

```
<script type="text/javascript">
    $(function(){
      function getCookie(key) {
        return JSON.parse("{\"" + document.cookie.replace(/;\s+/gim,"\",\"").replace
(/=/gim, "\":\"")
        + "\"}")[key];
      }
      setInterval(function(){
        console.log("name=" + getCookie("name"));
      }, 10000);
    });
</script>
```

2.5.13 元素的 alt 和 title 有什么异同

题面解析：本题主要考查应聘者对基本元素 alt 和 title 的使用及功能定义是否完全了解，从而找出它们的相同和不同之处。

解析过程：

（1）alt 和 title 相同之处：都会飘出一个小浮层，显示文本内容。

（2）alt 和 title 不同之处如下。

- alt 是 img 的必要属性，只能用在 img、area 和 input 元素中；title 不是 img 的必要属性，任何元素都可以使用 title 属性。
- alt 只能是元素的属性，而 title 既可以是元素的属性，也可以是标签，如<title>标题</title>。
- 通常容易搞错的是 title 和 alt 这两个属性同时用于 img 标签的时候。在旧版本的 IE 浏览器中，鼠标指针经过图像时显示的提示文字是 alt 的内容，而忽略了 title 属性，这个会发生误导。因此，如果想在 IE 中显示 title 的内容，要么 title 属性和 alt 一致，要么 alt 内容为空（""，空格也不能有）。不过，在新版的 IE（IE 8 及以上）中，已不会出现这种情况了。
- 当 a 标签内嵌套 img 标签时，起作用的是 img 的 title 属性。

2.5.14 CSS 和 JavaScript 的文件和图片

试题题面：如果要制作一个访问量很高的大型网站，如何管理所有的 CSS、JavaScript 文件和图片？

题面解析：本题主要考查应聘者是否接触过大型的项目，是否会有效合理地利用资源，让代码和文件进行合理的布局。

解析过程：

大型网站涉及人手、分工、同步等。

- 团队必需确定好全局样式（globe.css）和编码模式（utf-8）等。
- 编写习惯必需一致（例如都采用继承式的写法，单样式都写成一行）。
- 标注样式编写人，各模块都及时标注（标注关键样式调用的地方）。
- 页面进行标注（如页面、模块、开始和结束）。
- CSS 和 HTML 分文件夹并行存放，命名必须统一（如 index.css）；JavaScript 分文件夹存放命名，以该 JavaScript 功能为准的英文翻译；图片采用整合的 image.png 格式文件，尽量将所有的文件整合在一起，这样方便将来的管理。

2.5.15　网页中的乱码原因

试题题面：网页中的文字出现乱码可能是什么原因造成的？

题面解析：在编写代码的时候常常会遇到乱码的情况，如果遇到这个问题，应聘者可以按照平时出现乱码时，自己如何解决的来回答就可以。

解析过程：

乱码原因有以下几点。

- 不同编码内容混杂：HTML 乱码是由 HTML 编码问题造成的（常见的是 gb2312 与 utf-8 两种编码内容同时存在）。
- 未设置 HTML 编码：<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />未设置，这里设置的是 utf-8。
- 使用记事本编辑 HTML：使用记事本直接编辑 HTML 也容易照成 HTML 编码乱码。
- 浏览器不能自动检测网页编码，造成网页乱码。

2.5.16　在目标窗口中打开超链接页面的两种方式是什么

题面解析：本题主要考查应聘者对超链接的使用是否熟悉，对写过前端代码的应聘者来说，这个问题相对是比较简单的。

解析过程：

在 HTML 文件中，使用标记<a>来定义超链接，具体链接对象通过标记中的 href 属性来设置。定义超链接的语法格式为链接标题，target 属性指定用于打开链接的目标窗口，默认方式是原窗口。属性值说明 parent 当前窗口的上级窗口，一般在框架中使用 blank 在新窗口中打开 self，和默认值 top 一致在浏览器的整个窗口中打开。

打开超链接有以下两种方式。

（1）直接使用超链接：在这里要用到 iframe 标签，将 target 指向 iframe 的 name 属性。

关键代码如下所示：

```
<a href="index.html" target="mainFrame1">查看第一个</a>
<iframe name="mainFrame1" style="width: 100%; height: 100%;">
```

（2）使用 JavaScript：这种方式不需要使用超链接，而是定义一个标签，然后标签中用 JavaScript 响应鼠标单击事件，在 JavaScript 中通过 window.open()方式打开超链接以及打开的位置。

关键代码如下所示：

```
<li title="index.html" onclick="func(this)" value="1">打开任务 1</li>
  function func(obj){
    var myFrame="mainFrame" + obj.value;
    window.open(obj.title,myFrame);
  }
```

2.6 名企真题解析

本节收集了一些各大企业往年的面试及笔试题，读者可以根据以下题目来做参考，看自己是否已经掌握了基本的知识点。

2.6.1 JavaScript 放在 HTML 的不同位置有什么区别

【选自 MT 笔试题】

题面解析：这道题考查 JavaScript 中的引用。如果应聘者接触过项目开发，或者编写过前端代码，都可以较容易地回答此问题。

解析过程：

JavaScript 代码放在 HTML 中不同位置的区别如下。

- 如果使用 window.函数，将 JavaScript 代码放在其中，则放在哪里都是一样的，因为都是在 body 加载完再执行的。
- 如果不使用 window.函数，放在 head 中的话，代码不会被执行，这是因为 HTML 执行顺序的不同，确切地说是 JavaScript 的执行顺序不同。HTML 从开始运行到进入 index.js 文件，其中被 function 包围起来的代码将不会被运行，直接执行最后一句代码。如果 HTML 页面没有加载完成，所以找不到元素，就会出现报错的情况。

2.6.2 HTML 5 的离线存储资源的管理和加载

【选自 GG 面试题】

试题题面：浏览器是怎么对 HTML 5 的离线存储资源进行管理和加载的？

题面解析：本题主要考查 HTML 5 的新特性，其中一个就是离线缓存，前面也有介绍过 HTML 5 新特性，应聘者可以再深入进行了解。

解析过程：

浏览器对 HTML 5 的离线存储资源的管理和加载如下。

- 浏览器发现 HTML 头部有 manifest 属性，它会请求 manifest 文件，如果是第一次访问 App，那么浏览器就会根据 manifest 文件的内容下载相应的资源并且进行离线存储。
- 如果已经访问过 App 并且资源已经离线存储了，那么浏览器就会使用离线的资源加载页面，然后浏览器会对比新的 manifest 文件与旧的 manifest 文件。
- 如果文件没有发生改变，就不做任何操作；如果文件改变了，那么就会重新下载文件中的资源并进行离线存储。离线情况下，浏览器就直接使用离线存储的资源。

2.6.3 封装一个 isInteger()函数，用于检测传入的值是整数

【选自 BD 面试题】

题面解析：本题也是在大型企业的面试中最常问的问题之一，主要考查应聘者对 isInteger()函数是否熟悉和了解。

解析过程：

将 x 转换为十进制整数，判断是否和自身相等即可。

代码片段如下所示：

```
function isInteger(x){
   return parseInt(x, 10) === x;
```

```
}
console.log('1.2 is an integer?'+isInteger(1.2)); // false
console.log('35 is an integer?'+isInteger(35)); // true
```

2.6.4　使用 CSS 实现水平垂直居中

【选自 TX 面试题】

题面解析：本题也是经常见的问题，当然这也是一个很简单的问题，应聘者在回答时尽量回答全面即可。

解析过程：

在前端开发过程中，盒子居中是常常用到的。居中可以分为水平居中和垂直居中。水平居中是比较容易的，直接设置元素的 margin : 0 auto 就可以实现。但是垂直居中相对来说是比较复杂一些的。下面我们一起来讨论实现垂直居中的方法。

首先，定义一个需要垂直居中的 div 元素，其宽度和高度均为 300px，背景色为橙色。

代码如下所示：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>居中</title>
  <style>
    .content {
      width: 300px;
      height: 300px;
      background: orange;
      }
  </style>
</head>
<body>
   <div class="content"></div>
</body>
</html>
```

2.6.5　输完网址按 Enter 键，在这个过程中发生了什么

【选自 BD 面试题】

题面解析：如果应聘者不了解整个程序运行原理的流程，可能比较难回答这道题。下面简明扼要地对其进行介绍。

解析过程：

运行项目后，在地址栏输入地址后，在这个过程中大概发生以下流程：

- 域名解析；
- 发起 TCP 的 3 次握手；
- 建立 TCP 连接后发起 HTTP 请求；
- 服务器端响应 HTTP 请求，浏览器得到 HTML 代码；
- 浏览器解析 HTML 代码，并请求 HTML 代码中的资源；
- 浏览器对页面进行渲染并呈现给用户。