关于HTML和HTML5的面试复习。

1、HTML5的新增特性

- **语义化标签**：`header`、`nav`、`footer`、`section`、`aside`、`article`、`details`、`summary`、`dialog`（对话框）

- **媒体标签**：`audio`、`video`
- **canvas绘图**
- **拖放**：即抓取对象以后拖到另一个位置。设置元素可拖放。
  - `<img draggable="true"/>`
- **DOM查询操作**
  - document.querySelector()
  - document.querySelectorAll()

    他们选择的对象可以是标签，可以是类（需要加点），可以是ID（需要加#）
- **web存储**：`localStorage`、`sessionStorage`
- **表单**
  - **input输入类型增多。**color(颜色选取)、date、datetime（UTC时间）、datetime-local（日期时间无时区）、email、month、number、range（一定范围内数字值）、search、tel（电话号码，这样就不用做正则了）、time、url、week
  - **新增表单元素。**datalist（输入域选项列表）、Keygen（验证用户）、output（不同类型输出）
  - **新增表单属性。**placehoder（提示语）、required（Boolean，不能为空）、pattern（正则）。min/max、step（合法数字间隔）、height/width（image高宽）、autofocus（Boolean，自动获取焦点）、mutiple（Boolean，多选）

2、meta和title

> meta是HTML语言head区的一个辅助性标签，位于文档的头部，不包含任何内容。标签的属性定义了与文档相关的名称/值对。
>
> meta可提供相关页面的元信息（meta-information），比如针对搜索引擎和更新频度的描述和关键词。

- `元数据`：用来构建html文档的基本结构，以及如何处理文档向浏览器提供信息和指示，本身不是文档内容，但是提供了后面文档的信息，也就是说提供了网站的信息。

meta的属性有两种：name和http-equiv

- `name`属性主要用于描述网页，对应于content（网页内容）配合使用。不能与charset、http-equiv配合使用。
- `http-equiv`：可以用作http头部的一些作用，通过定义这个属性改变用户代理等行为。
- 常用属性：charset（声明字符编码）、content（通常配合name或者http-equiv使用）

```
<head>
    <titlt>这是一个例子</titlt>
    <meta name="keywords" content="描述网站的关键词，以逗号隔开，用于SEO搜索">
    <meta name="description" content="当前网页的描述">
    <meta name="application name" content="当前页所属web应用系统的名称">
    <meta name="author" content="当前网页作者的名">
    <meta name="copyright" content="版权信息">
    <meta http-equiv="refresh" content="5; http://www.baidu.com"> //重定向
</head>
```

> SEO：搜索引擎优化。

title是网站的标题，有助于SEO搜索引擎优化。

3、前端需要注意哪些SEO（搜索引擎优化）

1. 合理的 `title`、`description`、`keywords`：搜索对这三项的权重逐个减小，`title`值强调重点即可，重要关键词出现不要超过2次，而且要靠前，不同页面 `title`要有所不同； `description`把页面内容高度概括，长度合适，不可过分堆砌关键词，不同页面 `description`有所不同；`keywords`列举出重要关键词即可。
2. **语义化的html代码，符合W3C规范**：语义化代码让搜索引擎容易理解网页
3. **重要内容HTML代码放在最前**：搜索引擎抓取html顺序是从上到下，有的搜索引擎对抓取长度有限制，保证内容一定会被抓取。
4. **重要内容不要用JS输出**：爬虫不会爬取js执行内容
5. **少用** `iframe`：搜索引擎不会抓取iframe中的内容
6. 非装饰性图片必须加alt（到时候图片可以慢慢渲染，先有文字描述该图片）
7. 提高网站速度：网站速度是搜索引擎排序的一个重要指标

4、< img> 的title和alt的区别

- `title`通常当鼠标滑到元素上时显示
- `alt`是img标签的特有属性，是图片内容的等价描述，用于图片无法加载显示、读屏器阅读图片。可以提高图片搞可访问性，除了纯装饰图片外都必须设置有意义的值，搜索引擎会重点分析

5、src和href的区别

- `href`指向网络资源的位置，建立当前文档和资源的连接，一般用于超链接
- `src`将资源放入当前文档，在请求src资源的时候会将指向的资源下载并应用到文档中，比如js、图片等元素。

6、对HTML语义化的理解

> 语义化是指根据内容的结构化（内容语义化），选择合适的标签（代码语义化）。通**俗来讲就是用正确的标签做正确的事。**

语义化的优点：

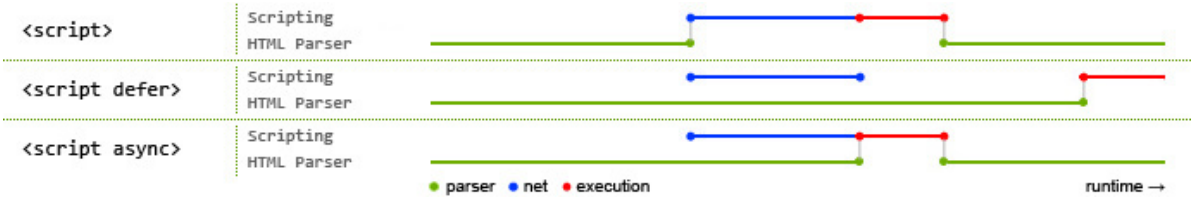
- 对机器友好，有语义的文字表现力丰富，更适合搜索引擎的爬虫爬取有效信息，有利于SEO。
- 对开发者友好，使用语义类标签增强了可读性，结构更加清晰，开发者能清晰的看出网页的结构，便于团队的开发和维护。

常见的语义化标签：

```
<header></header>  头部
<na></na>  导航栏
<section></section> 区块
<main></main> 主要区域
<article></article>  主要内容
<aside></aside> 侧边栏
<footer></footer>  底部
```

7、script标签中defer和async的区别

> 如果没有`defer`或`async`属性，浏览器会立即加载并执行相应的脚本。他不会等待后续加载的文档元素（就是即使这个文档元素没有加载完也要停下来），读取到就会开始加载和执行，这样就会阻塞后续文档的加载。defer是推迟的意思，async是异步的意思。



蓝色代表js脚本的加载时间；红色表示js脚本的执行时间；绿色表示HTML解析时间

共同点：

defer和async都是异步加载外部的脚本文件，他们不会阻塞HTML的解析。

不同点：

- 执行顺序不同。
  - 多个带async属性的标签，不能保证加载的顺序；
  - 多个带defer属性的标签，按照加载顺序执行。
- 脚本是否并行执行。
  - async属性，脚本文件加载完后立即执行脚本，停止对HTML的解析，脚本执行完后在继续执行HTML的解析。
  - defer属性，即脚本文件加载完后，要等整个html文档加载完后才开始执行。

8、文档生明（Doctype）和<!Doctype html>有何作用?

- **文档声明Doctype**的作用是为了告诉浏览器，当前 html 文档使用什么版本的HTML来写的，这样浏览器才能按照声明的版本来正确解析。
- **<!Doctype html>的作用**：`<!doctype html>`的作用是让浏览器进入标准模式，使用最新的html5标准来解析渲染页面，如果不写，浏览器会进入**混杂模式**，我们需要避免此类情况发生。

关于混杂模式和严格模式：

- 严格模式又称为标准模式，指浏览器按照`W3C`标准解析代码
- 混杂模式又称怪异模式、兼容模式，是指浏览器用自己的方式解析代码。混杂模式通常模拟老式浏览器的行为，以防止老站点无法工作。

9、对于web work的理解

- JavaScript采用的是单线程模式，也就是说所有任务只能在一个线程上完成，一次只能做一件事。
- web work的作用，就是为JavaScript创造多线程环境，允许主线程创建worker线程，将一些任务分配给worker线程处理。 在主线程运行的同时，worker线程在后台运行，两者互不干扰，好处是一些计算密集型或高延迟的任务，被worker线程负担了，主线程就会很流畅，不会被阻塞或拖慢。
- worker线程一旦被创建成功，就会始终运行，不会被主线程上的活动打断。这样有利于随时响应主线程的通信。但是，也会造成worker比较耗费资源，不应该过度使用，而且一旦使用完毕，就应该关闭。

如何创建web worker：

1. 检测浏览器对于web worker的支持性
2. 创建web worker文件（js，回调函数等）
3. 创建web worker对象

10、canvas和svg的区别

svg

svg可缩放矢量图形是基于可扩展标记语言xml描述的2D图形语言，SVG基于XML意味着SVG DOM中的每个元素都是可用的，可以为某个元素附加JavaScript事件处理器。在SVG中，每个被绘制的图形均被视为对象。如果SVG对象的属性发生变化，那么浏览器能够自动重现图形。

特点：

- 不依赖分辨率
- 支持事件处理器
- 最适合带有大型渲染区域的应用程序（比如谷歌地图）
- 复杂度高会减慢渲染速度（任何过度使用DOM的应用都不快）
- 不适合游戏应用

canvas

canvas是画布，通过JavaScript来绘制2D图形，是逐像素进行渲染的。其位置发生改变，就会重新绘制。

特点：

- 依赖分辨率
- 不支持事件处理器
- 弱的文本渲染能力‘
- 能够以.png或.jpg格式保存结果图像
- 最适合图像密集型的游戏，其中的许多对象会被频繁重绘。

> 注：矢量图，也称为面向对象的图像或绘图图像，在数学上定义为一系列由线连接的点。矢量文件中的图形元素称为对象。每个对象都是一个自成一体的实体，它具有颜色、形状、轮廓、大小和屏幕位置等属性。

11、head标签有什么作用？其中什么标签比不可少?

- < head >标签用于定义文档的头部，它是所有头部元素的容器。< head >中的元素可以引用脚本、指示浏览器在哪里找到样式表、提供元信息等。
- 文档的头部描述了文档的各种属性和信息，包括文档的标题、在web中的位置以及和其他文档的关系等。绝大多数文档头部包含的数据不会真正作为内容显示给读者。
- 这些标签可以包含在head里：`<base>\<link>\<meta>\<script>\<style>\<title>`
    - 其中`<title>`定义文档的标题，它是head部分中唯一必需的元素。

12、iframe的优点和缺点?

`<iframe>`是HTML内联框架元素，它能够将另一个HTML页面嵌入当前页面。（即创建一个行内框架）

优点：

- 用来加载速度较慢的内容（如广告）
- 可以使脚本可以并行下载
- 可以跨子域通信

缺点：

- iframe会阻塞主页面的onload事件
- 无法被一些搜索引擎识别
- 会产生很多页面，不容易管理

13、title与h1、b与strong、i与em的区别?

- `title`属性没有明确意义只是表示是个标题，`h1`则表示层次明确的标题，对页面信息的抓取有很大的影响
- `strong`标签有语义，是起到加重语气的效果，加重字符的语气是通过加粗实现的，搜索引擎更侧重`strong`标签；而`b标签`只是一个简单的加粗标签，没有任何语义。
- `i`内容展示为斜体，`em`表示强调的文本

14、行内元素有哪些？块级元素有哪些？空（void）元素有哪些?

- 行内元素：`a、b、span、img、input、select、strong`
- 块级元素：`div、ul、li、ol、dl、dt、dd、h1-h6、p`
- 空元素，即没有内容的元素。空元素是在开始标签中关闭的，也就是空元素没有闭合标签。
    - `br\hr\img\input\link\meta\area';`

15、HTML5 drag API元素拖拽

- `dragstart`：事件主体是被拖拽元素，在开始 拖放 被拖放元素时触发
- `drag`：事件主体是被拖拽元素，在正在 拖放 被拖放元素时触发
- `dragenter`：事件主体是目标元素，在被拖放元素放入某元素时触发
- `dragover`：事件主体是目标元素，在被拖放在某元素内移动时触发
- `dragleave`：事件主体是目标元素，在被拖放元素移出目标元素时触发
- `drop`：事件主体是目标元素，在目标元素完全接受被拖放元素时触发
- `dragend`：事件主体是被拖放元素，在整个拖放操作结束时触发

```
<style>
    .contain {
        width: 100px;
        height: 100px;
        margin-bottom: 20px;
        background: lightblue;
    }

    .el {
        width: 50px;
        height: 50px;
        background: lightcoral;
    }
</style>
<body>
    <div style="background-color: aqua;">hhhhh</div>
    <div class="contain"></div>
    <div class="el" draggable="true"></div> <!-- 使元素能够被拖拽 -->
    <script>
        /**
         * 将要拖拽的元素设置允许拖拽，赋予dragstart事件将id转换成数据保存
```

```
        *为容器添加dragover属性阻止浏览器默认事件，允许元素放置，赋予drop放置的位置
        **/
        const contain = document.querySelector('.contain')
        const el = document.querySelector('.el')
        el.addEventListener('dragstart', (e) => { // 当元素被拖拽时触发
            console.log(e.target); // 被拖拽元素
            e.dataTransfer.setData('message', 'hello')
        })
        contain.addEventListener('dragover', (e) => {
            e.preventDefault()
            console.log("move")
        })
        contain.addEventListener('drop', (e) => {
            e.preventDefault()
            console.log(e.dataTransfer.getData('message'));
        })
    </script>
</body>
```