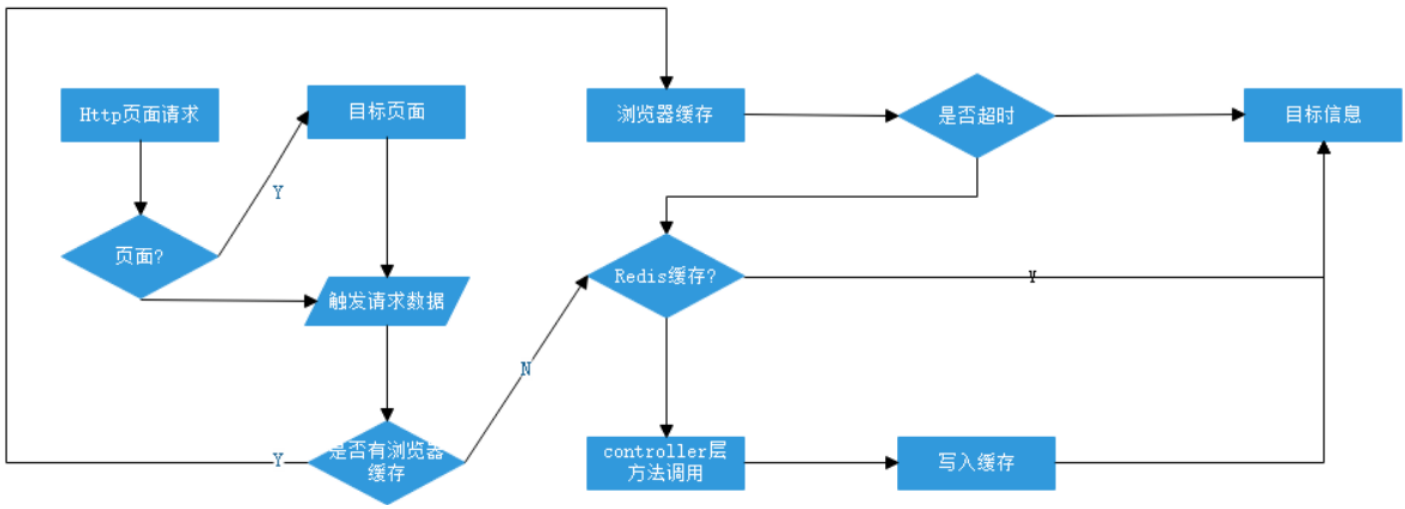
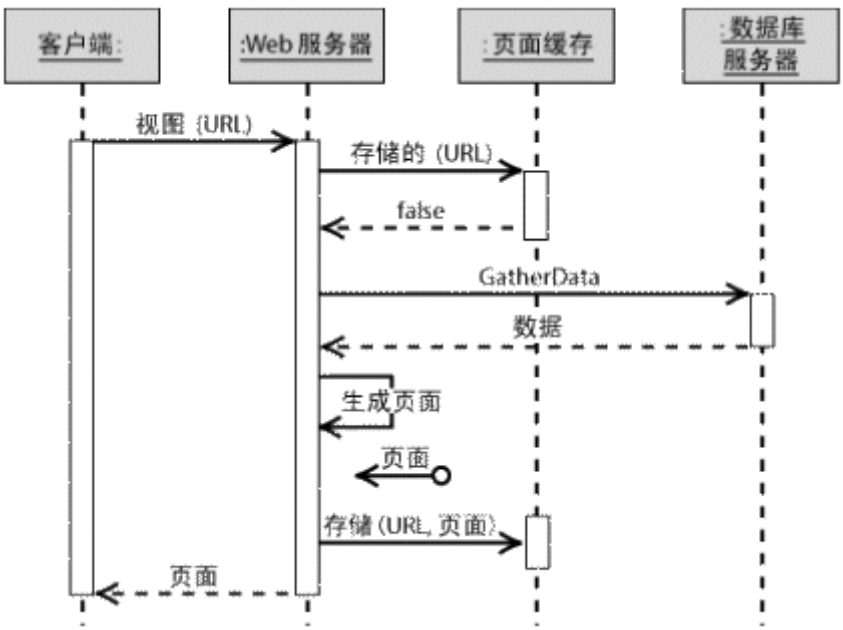
前端

页面缓存





静态资源：

图片（.jpg, .png, .svg等）

视频（.mp4, .avi等）

字体文件（.ttf, .woff等）

模板文件（.html）

样式文件（.css）和脚本文件（.js）：

背景JS和样式表文件，这些文件很少变化，适合使用长强缓存来减少加载时间

计算机目录结构

绝对路径

绝对路径是文件或目录在文件系统中的完整路径，它从根目录开始一直到目标文件或目录的路径。在不同操作系统下，绝对路径的表示方法有所不同：

- **Windows**:

```
C:\Users\Username\Documents\file.txt
```

- **Unix/Linux**:

```
/home/username/documents/file.txt
```

绝对路径具有唯一性和确定性，不受当前工作目录影响，因此无论从哪里访问，路径都是固定的。

相对路径

相对路径是相对于当前工作目录或者另一文件的路径。使用相对路径可以简化文件的引用，特别是当文件和执行文件（如程序或脚本）在同一文件系统层级内时，使用相对路径更为方便。

相对路径分为几种类型：

1. **当前目录**:

```
./file.txt
```

1. 这表示当前目录下的 `file.txt` 文件。

2. **上级目录**:

```
../parent-directory/file.txt
```

1. 这表示当前目录的上一级目录中的 `parent-directory` 目录下的 `file.txt` 文件。

2. **其他目录**:

```
../../other-directory/file.txt
```

1. 这表示当前目录的上两级目录中的 `other-directory` 目录下的 `file.txt` 文件。

相对路径的实际效果取决于当前程序或脚本执行时所处的工作目录。因此，相对路径是相对于执行时的上下文环境来定义的。

实际应用

在编程中，特别是在处理文件系统时，通常会使用相对路径来引用文件。这样做可以增加代码的可移植性，因为代码不依赖于特定的绝对路径。例如，在一个项目中，如果需要访问项目根目录下的 `config.json` 文件，可以使用相对路径 `./config.json` 或 `../config.json`，而不需要硬编码完整的绝对路径。

URL

Uniform(统一) Resource Locators(管理器)

常见结构:

scheme://host.domain:port/path/filename

- scheme(方案) - 定义因特网服务的类型。最常见的类型是 http
- host - 定义域主机（http 的默认主机是 www）
- domain(域) - 定义因特网域名，比如 runoob.com
- :port - 定义主机上的端口号（http 的默认端口号是 80）
- path - 定义服务器上的路径（如果省略，则文档必须位于网站的根目录中）。
- filename - 定义文档/资源的名称

常见url scheme:

Scheme	访问	用于...
http	超文本传输协议	以 http:// 开头的普通网页。不加密。
https	安全超文本传输协议	安全网页，加密所有信息交换。
ftp	文件传输协议	用于将文件下载或上传至网站。
file		您计算机上的文件。

url字符编码

URL 只能使用 ASCII 字符集.

来通过因特网进行发送。由于 URL 常常会包含 ASCII 集合之外的字符，URL 必须转换为有效的 ASCII 格式。

URL 编码使用 "%" 其后跟随两位的十六进制数来替换非 ASCII 字符。

URL 不能包含空格。URL 编码通常使用 + 来替换空格。

例如:

```
1 <img src="/static/img/The%20King%20Of%20Ring.png" alt="Book 1">
```

HTML

Hyper(超)**T**ext **M**arkup (标记)**L**anguage:

HTML 是用来描述网页的一种语言。HTML 是一种在 Web 上使用的通用标记语言。HTML 允许你格式化文本，添加图片，创建链接、输入表单、框架和表格等等，并可将之存为文本文件，浏览器即可读取和显示

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>页面标题</title>
6 </head>
```

```
<body>

<h1>我的第一个标题</h1>
<p>我的第一个段落。</p>

</body>
</html>
```

<!DOCTYPE html> 声明为 HTML5 文档

<html> 元素是 HTML 页面的根元素

<head> 元素包含了文档的元（meta）数据，如 <meta charset="utf-8"> 定义网页编码格式为 utf-8（由于在大部分浏览器中直接输出中文会出现乱码，所以要在头部将字符声明为UTF-8）

<title> 元素描述了文档的标题

<body> 元素包含了可见的页面内容

<h1> 元素定义一个大标题

<p> 元素定义一个段落

常用标签(label)

标题

```
<h1>haha<h1>              中间放置想要展示的title
                            1-6逐渐变小
<h2><h2>
```

段落

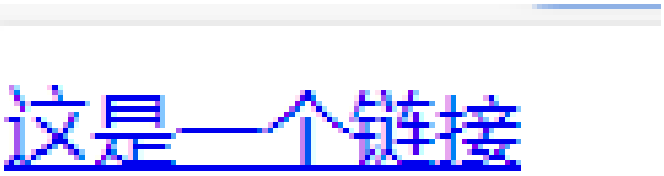
```
<p>this is a piece of words<p>
```

用于展示文本文件

链接

```
    <a href="https://www.runoob.com">这是一个链接</a>
```

最终在页面显示的结果:



作为一个附加元素加入到标签,中间内容会被特殊标记

图片

```
<img src="img.jpg" height="128" width="128"></div>
```

定义一个128*128的一个图片展示在页面上,src中指定资源路径

表格

```
<table>
  <thead>
    <tr>
      <th>列标题1</th>
      <th>列标题2</th>
      <th>列标题3</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>行1，列1</td>
      <td>行1，列2</td>
      <td>行1，列3</td>
    </tr>
    <tr>
      <td>行2，列1</td>
      <td>行2，列2</td>
      <td>行2，列3</td>
    </tr>
  </tbody>
</table>
```

- **tr**：tr 是 table row 的缩写，表示表格的一行。

- **td**：td 是 table data 的缩写，表示表格的数据单元格。
- **th**：th 是 table header的缩写，表示表格的表头单元格。

边框属性

```
<table border="1">
    <tr>
        <td>Row 1, cell 1</td>
        <td>Row 1, cell 2</td>
    </tr>
</table>
```

列表

有序(unorder list)

显示的内容有一定顺序,前面有顺序化表示

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

无序(order list)

标题前面没有顺序化的提示,可能用一些图案替换

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

区块

区块元素

换行,独立区块

HTML 元素是块级元素，它可用于组合其他 HTML 元素的容器。

 元素没有特定的含义。除此之外，由于它属于块级元素，浏览器会在其前后显示折行。

如果与 CSS 一同使用， 元素可用于对大的内容块设置样式属性。

 元素的另一个常见的用途是文档布局。它取代了使用表格定义布局的老式方法。使用 <table> 元素进行文档布局不是表格的正确用法。<table> 元素的作用是显示表格化的数据。

内联元素

不换行

HTML 元素是内联元素，可用作文本的容器

 元素也没有特定的含义。

当与 CSS 一同使用时， 元素可用于为部分文本设置样式属性

表单

用于获取用户输入的信息,可以传递信息

- `<form>` 元素用于创建表单，`action` 属性定义了表单数据提交的目标 URL，`method` 属性定义了提交数据的 HTTP 方法（这里使用的是 "post"）。
- `<label>` 元素用于为表单元素添加标签，提高可访问性。
- `<input>` 元素是最常用的表单元素之一，它可以创建文本输入框、密码框、单选按钮、复选框等。`type` 属性定义了输入框的类型，`id` 属性用于关联 `<label>` 元素，`name` 属性用于标识表单字段。
- `<select>` 元素用于创建下拉列表，而 `<option>` 元素用于定义下拉列表中的选项。

```
<form action="/" method="post">
    <!-- 文本输入框 -->
    <label for="name">用户名:</label>
    <input type="text" id="name" name="name" required>

    <br>

    <!-- 密码输入框 -->
    <label for="password">密码:</label>
    <input type="password" id="password" name="password" required>

    <br>

    <!-- 单选按钮 -->
    <label>性别:</label>
    <input type="radio" id="male" name="gender" value="male" checked>
    <label for="male">男</label>
```

```
    <input type="radio" id="female" name="gender" value="female">
    <label for="female">女</label>

    <br>

    <!-- 复选框 -->
    <input type="checkbox" id="subscribe" name="subscribe" checked>
    <label for="subscribe">订阅推送信息</label>

    <br>

    <!-- 下拉列表 -->
    <label for="country">国家:</label>
    <select id="country" name="country">
        <option value="cn">CN</option>
        <option value="usa">USA</option>
        <option value="uk">UK</option>
    </select>

    <br>

    <!-- 提交按钮 -->
    <input type="submit" value="提交">
</form>
```

INPUT

1. 文本输入框

```
htmlCopy Code
<input type="text" id="username" name="username" placeholder="请输入用户名">
```

- `type="text"`：指定输入框类型为文本输入框。
- `id` 和 `name` 属性：分别用于标识该输入框的唯一标识符和在表单提交时用作字段名称。
- `placeholder` 属性：在输入框为空时显示的灰色提示文本。

2. 密码输入框

```
htmlCopy Code
<input type="password" id="password" name="password" placeholder="请输入密码">
```

- `type="password"`：指定输入框类型为密码输入框，输入内容会被隐藏显示。

3. **单选框**

```
htmlCopy Code
<input type="radio" id="male" name="gender" value="male"><label for="male">男性</label><input type="radio" id="female" name="gender" value="female"><label for="female">女性</label>
```

- `type="radio"`：创建单选按钮，通过 `name` 属性进行分组，使用户只能选择其中一个选项。

4. **复选框**

```
htmlCopy Code
<input type="checkbox" id="subscribe" name="subscribe" checked><label for="subscribe">订阅电子邮件</label>
```

- `type="checkbox"`：创建复选框，用户可以选择一个或多个选项。
- `checked` 属性：默认情况下选中复选框。

5. **下拉列表**

```
htmlCopy Code
<select id="country" name="country"><option value="china">中国</option><option value="usa">美国</option><option value="uk">英国</option></select>
```

- `<select>` 元素用于创建下拉列表框。
- `<option>` 元素用于定义选项。

6. **提交按钮**

```
htmlCopy Code
<input type="submit" value="提交">
```

- `type="submit"`：创建提交按钮，用于提交表单数据。

7. **文件上传**

```
htmlCopy Code
```

```
2 <input type="file" id="avatar" name="avatar">
```

- `type="file"`：创建文件上传按钮，允许用户选择本地文件。

8. **隐藏输入域**

```
1 htmlCopy Code
2 <input type="hidden" id="user_id" name="user_id" value="123">
```

- `type="hidden"`：创建隐藏的输入域，不会在用户界面中显示，但在提交表单时会发送到服务器。

如何更换页面图标

CSS

简介

Cascading(层叠) Style Sheets

主要分id选择器和class选择器

id选择器用.定义

class选择器用#定义

创建样式表

外部样式表(External style sheet)

在另一个文件内定义所需要的样式,这样就可以直接多次应用,减少代码量,便于修改,降低代码耦合性.

```
1 <head>
2 <link rel="stylesheet" type="text/css" href="mystyle.css">
3 </head>
4
5 hr {color:sienna;}
6 p {margin-left:20px;}
7 body {background-image:url("/images/back40.gif");}
```

内部样式表(Internal style sheet)

与外部样式表稍有不同,就是不将文件写在外面,而是写在一个html页面当中,方便查看.一般当代码量小的时候可以用.

```
 <head>
<style>
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```

内联样式表(Internal style sheet)

内联样式表就是在所需定义的标签上直接添加所需的样式,同时,也是优先级最高的,适用于特定的样式修改,或者代码量比较小的情况.

```
<p style="color:sienna;margin-left:20px">这是一个段落。</p>
```

样式表优先级

越具体,越靠近的优先级越高:

（内联样式）Inline style > （内部样式）Internal style sheet >（外部样式）External style sheet > 浏览器默认样式

优先级顺序

下列是一份优先级逐级增加的选择器列表：

- 通用选择器（*）
- 元素(类型)选择器
- 类选择器
- 属性选择器
- 伪类
- ID 选择器
- 内联样式

颜色

CSS中，颜色值通常以以下方式定义:

- 十六进制 - 如："#ff0000"
- RGB - 如："rgb(255,0,0)"
- 颜色名称 - 如："red"

必要的时候可以通过彩色器来获取颜色:

https://www.runoob.com/tags/html-colorpicker.html

背景

CSS 属性定义背景效果:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
<style>
h1
{
        background-color:#6495ed;
}
p
{
        background-color:#e0ffff;
}
div
{
        background-color:#b0c4de;
}
</style>
</head>

```

```
<body>

<h1>CSS background-color 实例!</h1>
<div>
该文本插入在 div 元素中。
<p>该段落有自己的背景颜色。</p>
我们仍然在同一个 div 中。
</div>

</body>
</html>
```

常用指令:

```
body
{
background-image:url('gradient2.png');
}                          //设置背景图片

body
{
background-image:url('gradient2.png');
background-repeat:repeat-x;    //水平方向平铺
}

body
{
background-image:url('img_tree.png');
background-repeat:no-repeat;    //让这个图片作为背景只出现一次,
}

body
{
background-image:url('img_tree.png');
background-repeat:no-repeat;
background-position:right top;    //使图片出现在右上角
}


```

详细背景属性:

background	简写属性，作用是将背景属性设置在一个声明中。
background-attachment	背景图像是否固定或者随着页面的其余部分滚动。
background-color	设置元素的背景颜色。
background-image	把图像设置为背景。
background-position	设置背景图像的起始位置。
background-repeat	设置背景图像是否及如何重复。

文本

属性	描述
color	设置文本颜色
direction	设置文本方向。
letter-spacing	设置字符间距
line-height	设置行高
text-align(对齐)	对齐元素中的文本
text-decoration	向文本添加修饰
text-indent	缩进元素中文本的首行
text-shadow	设置文本阴影
text-transform	控制元素中的字母
unicode-bidi	设置或返回文本是否被重写
vertical-align	设置元素的垂直对齐
white-space	设置元素中空白的处理方式
word-spacing	设置字间距

与文本相关的字体也有很多设置:

```
1 body {font-size:100%;}
2 h1 {font-size:2.5em;}
3 h2 {font-size:1.875em;}
4 p {font-size:0.875em;}
```

这样设置可以使得在所有浏览器中都显示相同的大小,以及在计算机缩小放大的时候也可以保证不冲突

链接

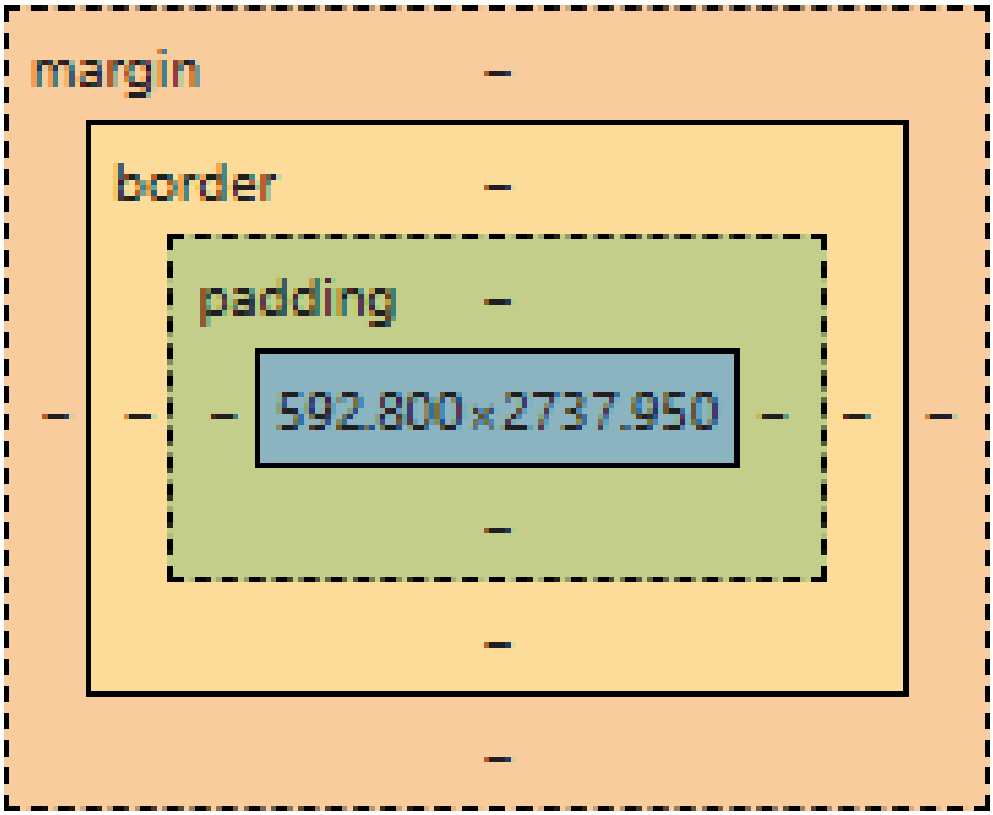
css中主要讲链接的样式修饰,可以简单的设置链接的状态,这样链接可以随鼠标的变化显示出不同的颜色.

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
<style>
a:link {background-color:#B2FF99;}     /* 未访问链接 */
a:visited {background-color:#FFFF85;} /* 已访问链接 */
a:hover {background-color:#FF704D;}    /* 鼠标移动到链接上 */
a:active {background-color:#FF704D;}  /* 鼠标点击时 */
</style>
</head>

<body>
<p><b><a href="/css/" target="_blank">这是一个链接</a></b></p>
<p><b>注意:</b> hover必须在:link和 a:visited之后定义才有效.</p>
<p><b>注意:</b>active必须在hover之后定义是有效的.</p>
</body>
</html>
```

盒子模型(Box Model)

实际例子:



- **Margin(外边距)** 清除边框外的区域，外边距是透明的。
- **Border(边框)** 围绕在内边距和内容外的边框。
- **Padding(内边距)** 清除内容周围的区域，内边距是透明的。
- **Content(内容)** 盒子的内容，显示文本和图像。

```
div {
    width: 300px;
    border: 25px solid green;
    padding: 25px;
    margin: 25px;
}
```

虽然定义这个块的宽度为300,但是实际其实为450,要加入边框,边距,填充,向外逐渐变大.

定位(position)

主要控制元素的位置关系:

- static:HTML 元素的默认值，即没有定位，遵循正常的文档流对象。

- fixed:

元素的位置相对于浏览器窗口是固定位置。

即使窗口是滚动的它也不会移动：

- relative:相对定位元素的定位是相对其正常位置
- absolute:绝对定位的元素的位置相对于最近的已定位父元素，如果元素没有已定位的父元素，那么它的位置相对于<html>:
- sticky:绝对定位的元素的位置相对于最近的已定位父元素，如果元素没有已定位的父元素，那么它的位置相对于<html>,这个的效果很不错,留下实例代码:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
<style>
div.sticky {
  position: -webkit-sticky;
  position: sticky;
  top: 0;
  padding: 5px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;
}
</style>
</head>
<body>

<p>尝试滚动页面。</p>
<p>注意: IE/Edge 15 及更早 IE 版本不支持 sticky 属性。</p>

<div class="sticky">我是粘性定位!</div>

<div style="padding-bottom:2000px">
  <p>滚动我</p>
  <p>来回滚动我</p>
  <p>滚动我</p>
  <p>来回滚动我</p>
  <p>滚动我</p>
  <p>来回滚动我</p>
</div>

</body>
</html>
```

这句话就可以一直显示在屏幕上,在很多编译器上就有应用这个效果.

Overfloat

这个效果可以做一些滚动条,可以设置滚动框的样式,可以在一些商业网页使用.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
<style>
#overflowTest {
    background: #4CAF50;
    color: white;
    padding: 15px;
    width: 80%;
    height: 100px;
    overflow: scroll;
    border: 1px solid #ccc;
}
</style>
</head>
<body>

<div id="overflowTest">
<p>这里的文本内容是可以滚动的，滚动条方向是垂直方向。</p>
<p>这里的文本内容是可以滚动的，滚动条方向是垂直方向。</p>
<p>这里的文本内容是可以滚动的，滚动条方向是垂直方向。</p>
<p>这里的文本内容是可以滚动的，滚动条方向是垂直方向。</p>
<p>这里的文本内容是可以滚动的，滚动条方向是垂直方向。</p>
<p>这里的文本内容是可以滚动的，滚动条方向是垂直方向。</p>
</div>

</body>
</html>
```

Float

用于定义在块中图片的相对位置,可以使图片出现在想要的位置,并且图片不会因为浏览器的放大以及缩放而调整间距,更灵活的展示在页面当中,是一个常用的技术.

```
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
<style>
.thumbnail
{
        float:left;
        width:110px;
        height:90px;
        margin:5px;
}
</style>
</head>

<body>
<h3>图片库</h3>
<p>试着调整窗口,看看当图片没有足够的空间会发生什么。</p>
<img class="thumbnail" src="/images/klematis_small.jpg" width="107" height="90">
<img class="thumbnail" src="/images/klematis2_small.jpg" width="107" height="80">
<img class="thumbnail" src="/images/klematis3_small.jpg" width="116" height="90">
<img class="thumbnail" src="/images/klematis4_small.jpg" width="120" height="90">
<img class="thumbnail" src="/images/klematis_small.jpg" width="107" height="90">
<img class="thumbnail" src="/images/klematis2_small.jpg" width="107" height="80">
<img class="thumbnail" src="/images/klematis3_small.jpg" width="116" height="90">
<img class="thumbnail" src="/images/klematis4_small.jpg" width="120" height="90">
</body>
</html>
```

同时可以用另一个关键词clear来使得控制浮动机制,使其按章理想的方式运行.

```
img
{
    float:right;
```

```
}
```

Margin(边缘) margin: auto

这个可以使元素居中对齐,使页面的主体更加具有标识性,可作为主要的题目使用.

同时也可以使文本进行居中显示.

```
.center {
    margin: auto;
    width: 50%;
    border: 3px solid green;
    padding: 10px;
}

.center {
    text-align: center;
    border: 3px solid green;
}

img {
    display: block;
    margin: auto;
    width: 40%;
}
```

使用absolute可以指定位置调整元素位置

```
body {
    margin: 0;
    padding: 0;
}

.right {
    float: right;
    width: 300px;
    background-color: #b0e0e6;
}
```

css组合选择符

- 后代选择器(以空格 分隔)
- 子元素选择器(以大于 > 号分隔)
- 相邻兄弟选择器（以加号 + 分隔）
- 普通兄弟选择器（以波浪号 ~ 分隔）

后代选择器( )

```
div p
{
  background-color:yellow;
}
```

当p在div当中的时候执行变黄的指令

子选择器(>)

```
div>p
{
  background-color:yellow;
}
```

p只能在div下才能变黄

相邻兄弟选择器(+)

```
div+p
{
  background-color:yellow;
}
```

选择紧接在另一元素后的元素，且二者有相同父元素,

因为相邻,所以具有相同父类元素,故只会选取第一个后面的元素,位于 后面的第一个元素

后继兄弟选择器(~)

```
div~p
{
  background-color:yellow;
}
```

选取 之后的所有相邻元素<q>,使其实现指定样式

导航栏

Html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Apple Style Cinema Navigation</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
<nav class="apple-style-nav">
  <ul>
    <li><a href="#" class="active">首页</a></li>
    <li><a href="#">电影</a></li>
    <li><a href="#">影院</a></li>
    <li><a href="#">票房</a></li>
    <li><a href="#">活动</a></li>
    <li><a href="#">关于</a></li>
  </ul>
</nav>

<div class="content">
  <!-- 页面内容 -->
</div>
</body>
</html>
```

Css:

```
* {
```

```
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Body styles */
body {
    font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
    background-color: #f5f5f5;
}

/* Navigation styles */
.apple-style-nav {
    background-color: #f0f0f0;
    padding: 10px 0;
    border-bottom: 1px solid #ccc;
    text-align: center;
}

.apple-style-nav ul {
    list-style-type: none;
    padding: 0;
    margin: 0;
}

.apple-style-nav li {
    display: inline-block;
    margin-right: 20px;
}

.apple-style-nav a {
    text-decoration: none;
    color: #333;
    font-size: 16px;
    padding: 10px 15px;
    transition: color 0.3s ease;
}

.apple-style-nav a.active,
.apple-style-nav a:hover {
    color: #007aff; /* 苹果蓝色 */
}

/* Content area styles (placeholder) */
.content {
    padding: 20px;
    text-align: center;
```

```
49 }
```

JavaScript

简介

互联网脚本语言,可直接作用于网页,使网页实现一些动态效果.

js语句

和Go类似的创建变量方式

也是用var关键字创建变量

js输出

- 使用 **window.alert()** 弹出警告框。
- 使用 **document.write()** 方法将内容写到 HTML 文档中。
- 使用 **innerHTML** 写入到 HTML 元素。
- 使用 **console.log()** 写入到浏览器的控制台。

语法

语句用;分割

声明关键字:var

注释://

数据类型:

```
var length = 16;                                // Number 通过数字字面量赋值
var points = x * 10;                            // Number 通过表达式字面量赋值
var lastName = "Johnson";                       // String 通过字符串字面量赋值
var cars = ["Saab", "Volvo", "BMW"];            // Array  通过数组字面量赋值
var person = {firstName:"John", lastName:"Doe"};  // Object 通过对象字面量赋值
```

注: JavaScript 中，如果对一个数字（如16）和一个字符串（如"Volvo"）进行加法操作，JavaScript 会将数字转换为字符串，然后进行字符串拼接。

所以，表达式 `16 + "Volvo"` 的结果是一个字符串 `"16Volvo"`。

函数:

```
function myFunction(a, b) {
            return a * b;

// 返回 a 乘以 b 的结果
}
```

大小写:区分

字符集;Unicode

数据类型

值类型(基本类型)

值类型(基本类型)：字符串（String）、数字(Number)、布尔(Boolean)、空（Null）、未定义（Undefined）、Symbol

注：*Symbol 是 ES6 引入了一种新的原始数据类型，表示独一无二的值。*

引用数据类型(对象类型)

引用数据类型（对象类型）：对象(Object)、数组(Array)、函数(Function)，还有两个特殊的对象：正则（RegExp）和日期（Date）

JavaScript 拥有动态类型

JavaScript 拥有动态类型。这意味着相同的变量可用作不同的类型：

```
var x;
        // x 为 undefinedvar
 x = 5;
        // 现在 x 为数字
var x = "John";       // 现在 x 为字符串
```

字符串

用于存储文本,单引号双引号都可以

```
var
carname="Volvo XC60";
```

```
var
carname='Volvo XC60';
```

数字

存储数字,不同于传统语言,没有整形和浮点数之分

```
var x1=34.00;      //使用小数点来写
var x2=34;
           //不使用小数点来写
```

数组

可以为数字数组,也可以为字符串数组,存储字符串

```
var cars=["Saab","Volvo","BMW"];
```

对象

类似于C语言里的结构体,Go里的结构体,不过这里叫做对象

使用键值对

```
var person={
firstname : "John",
lastname  : "Doe",
id        :  5566
};

name=person.lastname;
name=person["lastname"];
```

Undefined 和 Null

Undefined 这个值表示变量不含有值。

可以通过将变量的值设置为 null 来清空变量。

声明变量(new)

```
```

```
var carname=new String;
var x=      new Number;
var y=      new Boolean;
var cars=   new Array;
var person= new Object;
```

对象属性

标准对象定义方法:

```
var person = {
    firstName:"John",
    lastName:"Doe",
    age:50,
    eyeColor:"blue"
};


//访问对象方法
person.lastName;

//对象方法
对象的方法定义了一个函数，并作为对象的属性存储。
对象方法通过添加 () 调用 (作为一个函数)。
该实例访问了 person 对象的 fullName() 方法:
name = person.fullName();
```

函数

```
function functionname()
{
    // 执行代码
}
```

再html页面中,当执行到script脚本时运行

事件

HTML 事件可以是浏览器行为，也可以是用户行为。

以下是 HTML 事件的实例：

- HTML 页面完成加载
- HTML input 字段改变时
- HTML 按钮被点击

通常，当事件发生时，你可以做些事情。

在事件触发时 JavaScript 可以执行一些代码。

HTML 元素中可以添加事件属性，使用 JavaScript 代码来添加 HTML 元素。

```
<button onclick="getElementById('demo').innerHTML=Date()">现在的时间是?</button>
```

正则表达式(字符串匹配)

正则表达式是由一个字符序列形成的搜索模式。

当你在文本中搜索数据时，你可以用搜索模式来描述你要查询的内容。

正则表达式可以是一个简单的字符，或一个更复杂的模式。

正则表达式可用于所有文本搜索和文本替换的操作。

```
var str = "Visit Runoob!";
var n = str.search(/Runoob/i);        //i 不区分大小写

----------------------------------------------------
6

```

JavaScript 表单验证

通过js可以实现一些表单验证的功能

```
function validateForm() {
    var x = document.forms["myForm"]["fname"].value;
    if (x == null || x == "") {
        alert("需要输入名字。");
        return false;
```

```
    }
}
```

Json

Java**S**cript **O**bject **N**otation

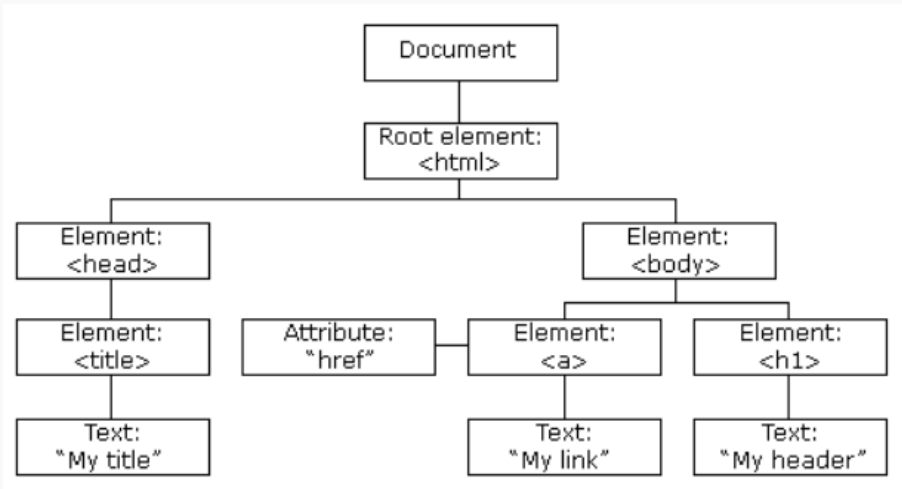
JSON 是一种轻量级的数据交换格式。

```
{"sites":[
    {"name":"Runoob", "url":"www.runoob.com"},
    {"name":"Google", "url":"www.google.com"},
    {"name":"Taobao", "url":"www.taobao.com"}
]}
```

DOM

Document Object Model

当网页被加载时，浏览器会创建页面的文档对象模型



功能:获取html文档的内容,并进行设置或者修改

代码:

```
<script>

```

```
<img

var light=document,getElementById("ID")


</script>
```

可对元素进行修改属性.

首先获取元素ID

在<scri>中进行修改

属性title.innerHTML="内容"

BOM

Browser **O**bject **M**odel (BOM)

将浏览器的各个部分封装成不同对象

浏览器对象

*window窗口对象

创建

方法

弹出框

alert()

显示出一段消息和一个确认按钮的警告框

Confirm

显示带有一段信息以及确认按钮,取消按钮的警告框

```
Var bool=comfirm("你要推出吗?")

if(bool){
}

```

```
6 确认 true
7 取消 false
```

Prompt

显示可以提示用户输入的对话框

```
1 var msg=prompt("显示的内容")
2
3 alert(msg)
```

打开关闭有关的方法

```
1 var open=document.getelementById(open)
2 open.onclick=function(){
3     open("wangzhi")
4 }
5
6 可实现点击按钮实现跳转
```

属性

特点

window对象可以直接应用,

window引用可以被省略

*History历史记录对象

screen显示器屏幕

*Location地址栏

Navigitar

Project

首页作为展示,其中包含了html,css,js

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Skyrim BookStore</title>
    <link type="text/css" rel="stylesheet" href="static/css/style.css">
</head>
<body>
<h1>Welcome To Skyrim Cinema</h1>

<div class="search-container">
    <input type="text" placeholder="Search" required>
    <input type="button" value="Search">
```

```
</div>

    <div class="book-container">
        <div class="book-item">
            <div class="book-info">
                <img src="/static/img/The%20King%20Of%20Ring.png" alt="Book 1">
                <p>Title of Book 1</p>
                <p>Author Name</p>
                <p>Summary or description of the book.</p>
            </div>
            <img src="/static/img/The%20King%20Of%20Ring.png" alt="Book 1">
        </div>
        <div class="book-item">
            <div class="book-info">
                <img src="/static/img/The%20King%20Of%20Ring.png" alt="Book 1">
                <p>Title of Book 1</p>
                <p>Author Name</p>
                <p>Summary or description of the book.</p>
            </div>
            <img src="/static/img/The%20King%20Of%20Ring.png" alt="Book 1">
        </div>
        <div class="book-item">
            <div class="book-info">
                <img src="/static/img/The%20King%20Of%20Ring.png" alt="Book 1">
                <p>Title of Book 1</p>
                <p>Author Name</p>
                <p>Summary or description of the book.</p>
            </div>
            <img src="/views/static/img/The%20King%20Of%20Ring.png" alt="Book
1">
        </div>
        <div class="book-item">
            <div class="book-info">
                <img src="/views/static/img/The%20King%20Of%20Ring.png"
alt="Book 1">
                <p>Title of Book 1</p>
                <p>Author Name</p>
                <p>Summary or description of the book.</p>
            </div>
            <img src="/views/static/img/The%20King%20Of%20Ring.png" alt="Book
1">
        </div>
        <div class="book-item">
            <div class="book-info">
                <img src="/views/static/img/The%20King%20Of%20Ring.png"
alt="Book 1">
                <p>Title of Book 1</p>
```

```
                <p>Author Name</p>
                <p>Summary or description of the book.</p>
            </div>
            <img src="/views/static/img/The%20King%20Of%20Ring.png" alt="Book 1">
        </div>
        <div class="book-item">
            <div class="book-info">
                <img src="/views/static/img/The%20King%20Of%20Ring.png" alt="Book 1">
                <p>Title of Book 1</p>
                <p>Author Name</p>
                <p>Summary or description of the book.</p>
            </div>
            <img src="/views/static/img/The%20King%20Of%20Ring.png" alt="Book 1">
        </div>
    </div>



<div class="user-section">
    <a href="http://localhost:8080/login" id="login">Login</a> <!-- Placeholder link, replace "#" with actual login page URL -->
    <a href="/regist" id="register">Register</a> <!-- Placeholder link, replace "#" with actual registration page URL -->
</div>

<script src="/static/js/scripts.js"></script>
</body>
</html>
```

CSS

```
body {
    font-family: 'Helvetica', sans-serif;
    background-color: #f5f5f5;
    margin: 20px;
    padding: 20px;
}
h1 {
    color: #333;
    text-align: center;
}
.search-container {
```

```
    margin-bottom: 20px;
    text-align: center;
}
.search-container input[type="text"] {
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
    width: 200px;
}
.search-container input[type="button"] {
    padding: 10px 20px;
    border: none;
    background-color: #007aff;
    color: white;
    border-radius: 5px;
    cursor: pointer;
}

.book-container {
    display: flex;
    justify-content: space-around;
}

.book-item {
    width: 150px;
    text-align: center;
    margin-bottom: 40px; /* 调整书籍之间的间距 */
    padding-bottom: 20px; /* 留出简介部分的空间 */
    border-bottom: 1px solid #ccc; /* 可选：添加书籍项之间的分隔线 */
    position: relative; /* 让子元素的绝对定位以其相对定位 */
}

.book-info {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    opacity: 0;
    background-color: rgba(255, 255, 255, 0.9);
    transition: opacity 0.3s ease;
    padding: 10px;
    box-sizing: border-box;
    border-radius: 5px;
    text-align: left;
}

```

```css
.book-item img {
    width: 100%;
    height: auto;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

.user-section {
    position: absolute;
    top: 20px; /* Adjust top position as needed */
    right: 20px; /* Adjust right position as needed */
}

.user-section a {
    margin-left: 10px;
    text-decoration: none;
    color: #333;
}
.book-item p {
    margin-top: 10px;
    font-size: 14px;
    color: #666;
}
.book-item:hover .book-info {
    opacity: 1;
}
```

JavaScript

```javascript
document.getElementById('login').addEventListener('click', function() {
    // Code to handle login action
    alert('Login clicked! Implement login functionality here.');
});

document.getElementById('register').addEventListener('click', function() {
    // Code to handle registration action
    alert('Register clicked! Implement registration functionality here.');
});
```