

2. HTML + CSS + JavaScript

本节目标

1. 认识 HTML 的基本结构, 学习常用的 HTML 标签.
2. 掌握 CSS 基本语法规则和CSS选择器的各种用法, 熟练使用CSS的常用属性.
3. 了解什么是JavaScript, 学习JavaScript的常见操作, 以及使用JQuery完成简单的页面元素操作.

1. HTML

1.1 HTML基础

1.1.1 什么是HTML

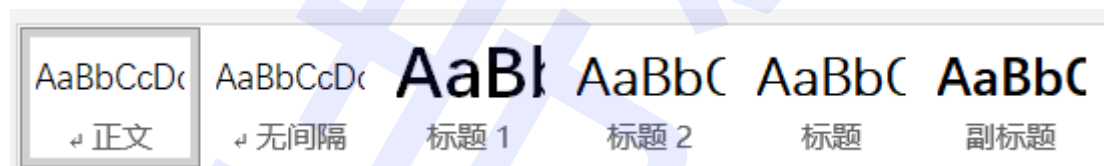
HTML(Hyper Text Markup Language), 超文本标记语言.

超文本: 比文本要强大. 通过链接和交互式方式来组织和呈现信息的文本形式. 不仅仅有文本, 还可能包含图片, 音频, 或者自己已经审阅过它的学者所加的评注、补充或脚注等等.

标记语言: 由标签构成的语言

HTML的标签都是提前定义好的, 使用不同的标签, 表示不同的内容.

类似飞书文档, Word文档



如果选中文本, 点击标题1, 就会使用标题1的样式来显示文本, 上述标题1就是一个"标签"

比如下方代码:

```
1 <h1>我是一级标题</h1>
2 <h2>我是二级标题</h2>
3 <h3>我是三级标题</h3>
```

经过浏览器解析后的效果如下:

我是一级标题

我是二级标题

我是三级标题

上面代码中的<h1> <h2> <h3> 就是标签

学习HTML 主要就是学习标签.

1.1.2 认识 HTML 标签

HTML 代码是由 "标签" 构成的.

形如:

```
1 <h3>我是三级标题</h3>
```

- 标签名 (body) 放到 < > 中
- 大部分标签成对出现. <h1> 为开始标签, </h2> 为结束标签.
- 少数标签只有开始标签, 称为 "单标签".
- 开始标签和结束标签之间, 写的是标签的内容.
- 开始标签中可能会带有 "属性". id 属性相当于给这个标签设置了一个唯一的标识符(身份证号码).

```
1 <h3 id="myId">我是三级标题</h3>
```

1.1.3 HTML 文件基本结构

```
1 <html>
2   <head>
3     <title>第一个页面</title>
4   </head>
5   <body>
6     hello world
7   </body>
```

```
8 </html>
```

- html 标签是整个 html 文件的根标签(最顶层标签)
- head 标签中写页面的属性.
- body 标签中写的是页面上显示的内容
- title 标签中写的是页面的标题.

1.1.4 标签层次结构

- 父子关系
- 兄弟关系

```
1 <html>
2   <head>
3     <title>第一个页面</title>
4   </head>
5   <body>
6     hello world
7   </body>
8 </html>
```

其中:

- head 和 body 是 html 的子标签(html 就是 head 和 body 的父标签)
- title 是 head 的子标签. head 是 title 的父标签.
- head 和 body 之间是兄弟关系.

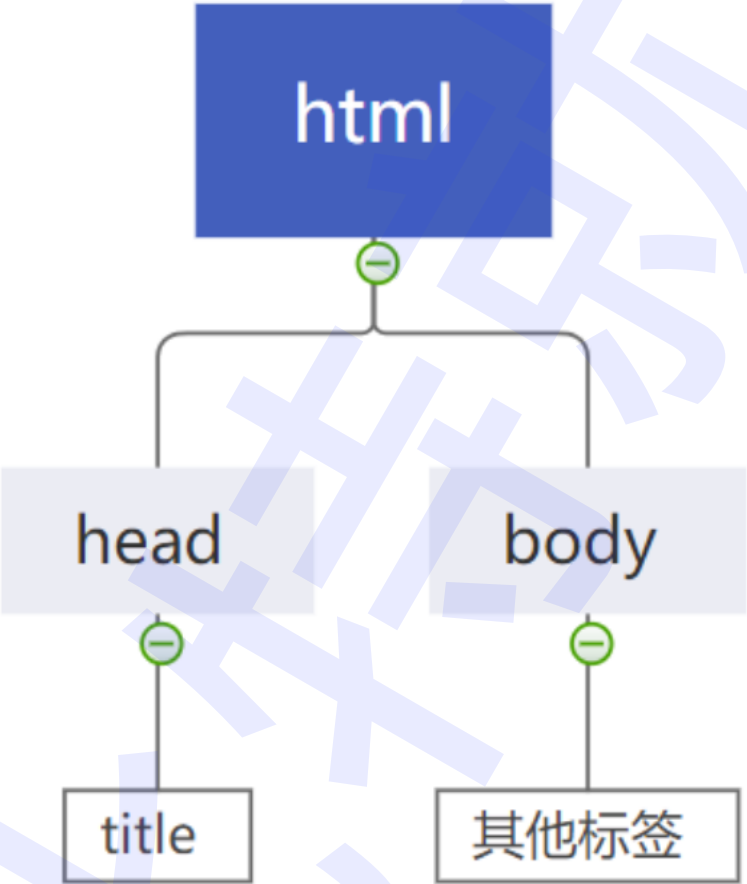
可以使用 chrome 的开发者工具查看页面的结构.

F12 或者右键审查元素, 开启开发者工具, 切换到 Elements 标签, 就可以看到页面结构细节.



标签之间的结构关系, 构成了一个 **DOM** 树

DOM 是 Document Object Mode (文档对象模型) 的缩写.



1.2 HTML 快速入门

1.2.1 开发工具

HTML 可以使用系统自带的记事本来编写, 但是非常不方便, 我们课程中使用前端专业的开发工具:

Visual Studio Code

Visual Studio Code (简称"VS Code") 是Microsoft在2015年4月30日Build开发者大会上宣布的一款跨平台源代码编辑器. 可以运行在Windows, macOS 和Linux上. 它具有对JavaScript, TypeScript和Node.js的内置支持, 并具有丰富的其他语言(例如C++, C#, Java, Python, PHP, Go).

- 官网: <https://code.visualstudio.com>

进行下载, 安装即可.

1.2.2 快速开发

在 VS Code中创建文件 `xxx.html`, 直接输入 `!`, 按 Enter 或 tab 键, 此时能自动生成代码的主体框架.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7 </head>
8 <body>
9
10 </body>
11 </html>
```

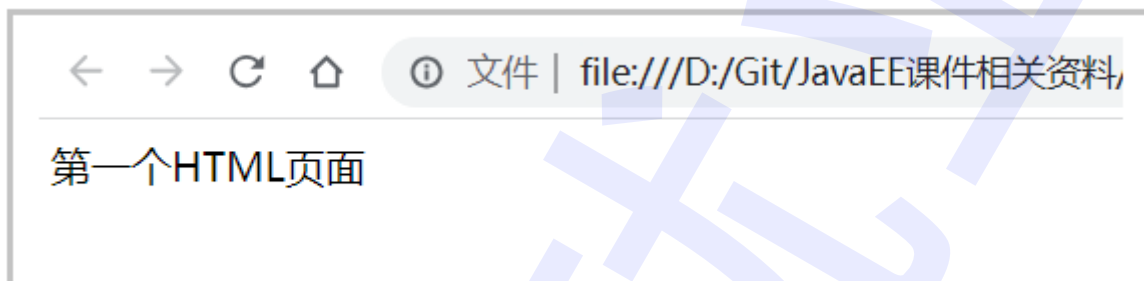
细节解释: (了解即可, 不必深究)

- `<!DOCTYPE html>` 称为 DTD (文档类型定义), 描述当前的文件是一个 HTML5 的文件.
- `<html lang="en">` 其中 lang 属性表示当前页面是一个 "英语页面". 这里暂时不用管. (有些浏览器会根据此处的声明提示是否进行自动翻译).
- `<meta charset="UTF-8">` 描述页面的字符编码方式. 没有这一行可能会导致中文乱码.
- `<meta name="viewport" content="width=device-width, initial-scale=1.0">`
 - `name="viewport"` 其中 viewport 指的是设备的屏幕上能用来显示我们的网页的那一块区域.
 - `content="width=device-width, initial-scale=1.0"` 在设置可视区和设备宽度等宽, 并设置初始缩放为不缩放. (这个属性对于移动端开发更重要一些).

在`<body></body>`标签中, 任意书写文字, 按Ctrl + s 保持文件, 通过浏览器访问即可. 如:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <title>Document</title>
7 </head>
8 <body>
9     第一个HTML页面
10 </body>
11 </html>
```

浏览器运行结果如下:



1.3 HTML常见标签

1.3.1 标题标签 h1-h6

有六个, 从 h1 - h6. 数字越大, 则字体越小.

```
1 <h1>hello</h1>
2 <h2>hello</h2>
3 <h3>hello</h3>
4 <h4>hello</h4>
5 <h5>hello</h5>
6 <h6>hello</h6>
```

1.3.2 段落标签: p

在HTML中, 段落, 换行符, 空格都会失效, 如果需要分成段落, 需要使用专门的标签

- p 标签表示一个段落.

```
1 <p>这是一个段落</p>
```

- 2 `<p>这是一个段落</p>`
- 3 `<p>这是一个段落</p>`

注意:

- p 标签描述的段落, 前面没有缩进. (未来 CSS 会学)
- 自动根据浏览器宽度来决定排版.
- html 内容首尾处的换行, 空格均无效.
- 在 html 中文字之间输入的多个空格只相当于一个空格.
- html 中直接输入换行不会真的换行, 而是相当于一个空格.

1.3.3 换行标签: br

想要完成换行的话, 也可以通过`
`标签来实现.

br 是 break 的缩写. 表示换行.

- br 是一个单标签(不需要结束标签)
- br 标签不像 p 标签那样带有一个很大的空隙.
- `
` 是规范写法. 不建议写成 `
`

- 1 这是一个br标签`
`
- 2 这是一个br标签`
`
- 3 这是一个br标签`
`

展示结果:

这是一个br标签
这是一个br标签
这是一个br标签

观察`
`标签和`<p>`标签的区别

这是一个段落

这是一个段落

p标签

这是一个段落

这是一个br标签

这是一个br标签

这是一个br标签

br标签

1.3.4 图片标签: img

img 标签必须带有 src 属性, 表示图片的路径.

```
1 
```

此时要把 rose.jpg 这个图片文件放到和 html 中的同级目录中.

img 标签的其他属性

- width/height: 控制宽度高度. 高度和宽度一般改一个就行, 另外一个会等比例缩放. 否则就会图片失衡.
- border: 边框, 参数是宽度的像素. 但是一般使用 CSS 来设定.

```
1 
```

注意:

1. 属性可以有多个, 不能写到标签之前
2. 属性之间用空格分割, 可以是多个空格, 也可以是换行.
3. 属性之间不分先后顺序
4. 属性使用 "键值对" 的格式来表示.

关于目录结构:

对于一个复杂的网站, 页面资源很多, 这种情况可以使用目录把这些文件整理好.

1. **相对路径:** 以 html 所在位置为基准, 找到图片的位置.
 - 同级路径: 直接写文件名即可 (或者 ./)
 - 下一级路径: image/1.jpg

- 上一级路径: ../image/1.jpg

2. 绝对路径: 一个完整的磁盘路径, 或者网络路径. 例如

- 磁盘路径 `D:/rose.jpg` (最好使用 `/`, 不要使用 `\`)
- 网络路径

```
https://img2.baidu.com/it/u=3359649605,3626874592&fm=253&fmt=auto&app=138&f=JPEG?w=500&h=752
```

1.3.5 超链接: a

- href: 必须具备, 表示点击后会跳转到哪个页面.
- target: 打开方式. 默认是 `_self`. 如果是 `_blank` 则用新的标签页打开.

```
1 <a href="http://www.baidu.com">百度</a>
```

链接的几种形式:

- 外部链接: href 引用其他网站的地址

```
1 <a href="http://www.baidu.com">百度</a>
```

- 内部链接: 网站内部页面之间的链接. 写相对路径即可.

在一个目录中, 先创建一个 1.html, 再创建一个 2.html

```
1 <!-- 1.html -->
2 我是 1.html
3 <a href="2.html">点我跳转到 2.html</a>
4
5 <!-- 2.html -->
6 我是 2.html
7 <a href="1.html">点我跳转到 1.html</a>
```

- 空链接: 使用 `#` 在 href 中占位.

```
1 <a href="#">空链接</a>
```

1.4 表格标签

- table 标签: 表示整个表格
- tr: 表示表格的一行
- td: 表示一个单元格
- thead: 表格的头部区域
- tbody: 表格的主体区域.

table 包含 tr, tr 包含 td

表格标签有一些属性, 可以用于设置大小边框等. 但是一般使用 **CSS 方式** 来设置.

这些属性都要放到 table 标签中.

- align 是表格相对于周围元素的对齐方式. align="center" (不是内部元素的对齐方式)
- border 表示边框. 1 表示有边框(数字越大, 边框越粗), "" 表示没边框.
- cellpadding: 内容距离边框的距离, 默认 1 像素
- cellspacing: 单元格之间的距离. 默认为 2 像素
- width / height: 设置尺寸.

注意, 这几个属性, vscode 都提示不出来.

```
1 <table align="center" border="1" cellpadding="20" cellspacing="0" width="500"
  height="500">
2   <tr>
3     <td>姓名</td>
4     <td>性别</td>
5     <td>年龄</td>
6   </tr>
7   <tr>
8     <td>张三</td>
9     <td>男</td>
10    <td>10</td>
11  </tr>
12  <tr>
13    <td>李四</td>
14    <td>女</td>
15    <td>11</td>
16  </tr>
17 </table>
```

1.5 表单标签

表单是让用户输入信息的重要途径.

分成两个部分:

- 表单域: 包含表单元素的区域. 重点是 form 标签.
- 表单控件: 输入框, 提交按钮等. 重点是 input 标签.

1.5.1 form 标签

```
1 <form action="test.html">
2     ... [form 的内容]
3 </form>
```

描述了要把数据按照什么方式, 提交到哪个页面中.

| 关于 form 需要结合后端代码来进一步理解. 后面再详细研究.

1.5.2 input 标签

各种输入控件, 单行文本框, 按钮, 单选框, 复选框.

- type(必须有), 取值种类很多, button, checkbox, text, file, image, password, radio 等.
- name: 给 input 起了个名字. 尤其是对于 单选按钮, 具有相同的 name 才能多选一.
- value: input 中的默认值.
- checked: 默认被选中. (用于单选按钮和复选按钮)

下面介绍一些常用的类型:

1. 文本框

```
1 <input type="text">
```

2. 密码框

```
1 <input type="password">
```

3. 单选框

```
1 性别:
```

```
2 <input type="radio" name="sex">男
3 <input type="radio" name="sex" checked="checked">女
```

注意: 单选框之间必须具备相同的 name 属性, 才能实现 多选一 效果.

4. 复选框

```
1 爱好:
2 <input type="checkbox"> 吃饭 <input type="checkbox"> 睡觉 <input
   type="checkbox"> 打游戏
```

5. 普通按钮

```
1 <input type="button" value="我是个按钮">
```

当前点击了没有反应. 需要搭配 JS 使用(后面会重点研究).

```
1 <input type="button" value="我是个按钮" onclick="alert('hello')">
```

6. 提交按钮

```
1 <form action="test.html">
2   <input type="text" name="username">
3   <input type="submit" value="提交">
4 </form>
```

提交按钮必须放到 form 标签内. 点击后就会尝试给服务器发送请求

1.5.3 select 标签

下拉菜单

- option 中定义 selected="selected" 表示默认选中.

```
1 <select>
2   <option>北京</option>
3   <option selected="selected">上海</option>
4 </select>
```

1.5.4 textarea 标签

```
1 <textarea rows="3" cols="50">
2
3 </textarea>
```

文本域中的内容, 就是默认内容, 注意, 空格也会有影响.

rows 和 cols 也都不会直接使用, 都是用 css 来改的.

1.6 无语义标签: div&span

div 标签, division 的缩写, 含义是 分割

span 标签, 含义是跨度

就是两个盒子. 用于**网页布局**

- div 是独占一行的, 是一个大盒子.
- span 不独占一行, 是一个小盒子.

```
1 <div>
2   <span>咬人猫</span>
3   <span>咬人猫</span>
4   <span>咬人猫</span>
5 </div>
6 <div>
7   <span>兔总裁</span>
8   <span>兔总裁</span>
9   <span>兔总裁</span>
10 </div>
11 <div>
12   <span>阿叶君</span>
13   <span>阿叶君</span>
14   <span>阿叶君</span>
15 </div>
```

1.7 综合练习: 用户注册

用户注册界面

用户注册

用户名	<input type="text" value="请输入用户名"/>
手机号	<input type="text" value="请输入手机号"/>
密码	<input type="text" value="请输入密码"/>
<input type="button" value="注册"/>	已有账号? 登录

提示:

- 使用表格进行整体布局
- 使用各种 input 标签实现页面中的输入控件

```
1 <h1>用户注册</h1>
2 <table>
3   <tr>
4     <td>用户名</td>
5     <td><input type="text" placeholder="请输入用户名"></td>
6   </tr>
7   <tr>
8     <td>手机号</td>
9     <td><input type="text" placeholder="请输入手机号"></td>
10  </tr>
11  <tr>
12    <td>密码</td>
13    <td><input type="text" placeholder="请输入密码"></td>
14  </tr>
15 </table>
16 <div>
17   <input type="button" value="注册">
18   <span>已有账号? </span><a href="#">登录</a><br/>
19 </div>
```

2. CSS

2.1 CSS介绍

2.1.1 什么是CSS?

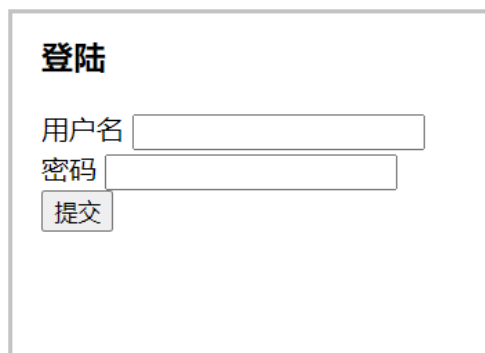
CSS(Cascading Style Sheet), 层叠样式表, 用于控制页面的样式.

CSS 能够对网页中元素位置的排版进行像素级精确控制, 实现美化页面的效果. 能够做到页面的样式和结构分离.

CSS 可以理解为"东方四大邪术" 之化妆术.

对页面的展示进行"化妆"

CSS前 -> CSS修饰后



登陆

用户名

密码



登陆

用户名

密码

2.1.2 基本语法规范

选择器 + {一条/N条声明}

- 选择器决定针对谁修改 (找谁)
- 声明决定修改啥. (干啥)
- 声明的属性是键值对. 使用 ; 区分键值对, 使用 : 区分键和值.

```
1 <style>
2   p {
3       /* 设置字体颜色 */
4       color: red;
5       /* 设置字体大小 */
6       font-size: 30px;
7   }
8 </style>
9
10 <p>hello</p>
```

注意:

- CSS 要写到 style 标签中(后面还会介绍其他写法)

- style 标签可以放到页面任意位置. 一般放到 head 标签内.
- CSS 使用 /* */ 作为注释. (使用 ctrl + / 快速切换).

2.1.3 引入方式

CSS有3种引入方式, 语法如下表格所示:

引入方式	语法描述	示例
行内样式	在标签内使用style属性, 属性值是css属性键值对	<code><div style="color:green">绿色</div></code>
内部样式	定义<style>标签, 在标签内部定义css样式	<code><style> h1 {...} </style></code>
外部样式	定义<link>标签, 通过href属性引入外部css文件	<code><link rel="stylesheet" href="[CSS文件路径]"></code>

3种引入方式对比:

1. 内部样式会出现大量的代码冗余, 不方便后期的维护, 所以不常用. (课堂上为了方便讲解, 使用该方式)
2. 行内样式, 只适合于写简单样式. 只针对某个标签生效. 缺点是不能写太复杂的样式.
3. 外部样式, html和css实现了完全的分离, 企业开发常用方式.

2.1.4 规范

样式大小写

虽然 CSS 不区分大小写, 我们开发时统一使用小写字母

空格规范

- 冒号后面带空格
- 选择器和 { 之间也有一个空格.

2.2 CSS选择器

CSS 选择器的主要功能就是选中页面指定的标签元素. 选中了元素, 才可以设置元素的属性.

就好比 SC2, War3 这样的游戏, 需要先选中单位, 再指挥该单位行动.

CSS选择器主要分以下几种:

1. 标签选择器
2. class选择器
3. id选择器

4. 复合选择器

5. 通配符选择器

我们通过代码来学习CSS选择器的使用.

```
1 <div class="font32">我是一个div, class为font32</div>
2 <div class="font32">我是一个div, class为font32</div>
3 <div><a href="#">我是一个div</a></div>
4 <ul>
5     <li>aaa</li>
6     <li>bbb</li>
7     <li><a href="#">ccc</a></li>
8 </ul>
9 <ol>
10    <li>1111</li>
11    <li>2222</li>
12    <li>3333</li>
13 </ol>
14 <button id="submit">提交</button>
```

1. 标签选择器

```
1 /* 选择所有的a标签, 设置颜色为红色 */
2 a {
3     color: red;
4 }
5
6 /* 选择所有的div标签, 设置颜色为绿色 */
7 div {
8     color: green;
9 }
```

2. 类选择器

```
1 /* 选择class为font32的元素, 设置字体大小为32px */
2 .font32 {
3     font-size: 32px;
4 }
```

- 一个类可以被多个标签使用, 一个标签也能使用多个类(多个类名要使用空格分割, 这种做法可以让代码更好复用)

3. ID选择器

```
1 /* 选择id为submit的元素, 设置颜色为红色 */
2 #submit {
3     color: red;
4 }
```

- id 是唯一的, 不能被多个标签使用 (是和 类选择器 最大的区别)

4. 通配符选择器

```
1 /* 设置页面所有元素, 颜色为红色*/
2 * {
3     color: red;
4 }
```

5. 复合选择器

```
1 /*只设置 ul标签下的 li标签下的 a标签, 颜色为红色*/
2 ul li a {
3     color: blue;
4 }
```

1. 以上三个标签选择器 `ul` `li` `a` 中的任意, 都可以替换成类选择器, 或者id选择器, 可以是任意选择器的组合, 也可以是任意数量选择器的组合
2. 不一定是相邻的标签, 也可以是"孙子"标签
3. 如果需要选择多种标签, 可以使用 `,` 分割, 如 `p, div { }` 表示同时选中p标签和div标签. 逗号前后可以是以上任意选择器, 也可以是选择器的组合.

2.3 常用CSS

接下来学习一些常见的css样式

准备如下html

```
1 <div class="text1">我是文本1</div>
```

2.3.1 color

color: 设置字体颜色

```
1 .text1{
2     color: red;
3 }
```

颜色有如下几种表达方式:

- 英文单词,如red,blue
- rgb代码的颜色 如rgb(255,0,0)
- 十六进制的颜色 如#ff00ff

2.3.2 font-size

font-size: 设置字体大小

```
1 .text1{
2     font-size: 32px;
3 }
```

2.3.3 border

border: 边框

边框是一个复合属性,可以同时设置多个样式,不分前后顺序,浏览器会根据设置的值自动判断

```
1 .text1{
2     border: 1px solid purple;
3 }
```

以上border属性的对应设置的维度分别为边框粗细, 边框样式, 边框颜色.

也可以拆开来设置

样式	说明	取值
border-width	设置边框粗细	数值
border-style	设置边框样式	<ul style="list-style-type: none">• dotted : 点状

		<ul style="list-style-type: none"> • solid : 实线 • double : 双线 • dashed: 虚线
border-color	设置边框颜色	同 color

`border: 1px solid purple;` 就等同于以下代码:

```
1 .text1 {
2     /* border: 1px purple solid; */
3     border-width: 1px;
4     border-style: solid;
5     border-color: purple;
6 }
```

2.3.4 width/height

width: 设置宽度

height: 设置高度

只有块级元素可以设置宽高

块级元素是HTML标签的一种显示模式, 对应的还有行内元素

常见块级元素: h1-h6, p, div 等

常见行内元素: a span

块级元素独占一行, 可以设置宽高

行内元素不独占一行, 不能设置宽高

改变显示模式

使用 display 属性可以修改元素的显示模式.

- `display: block` 改成块级元素 [常用]
- `display: inline` 改成行内元素 [很少用]

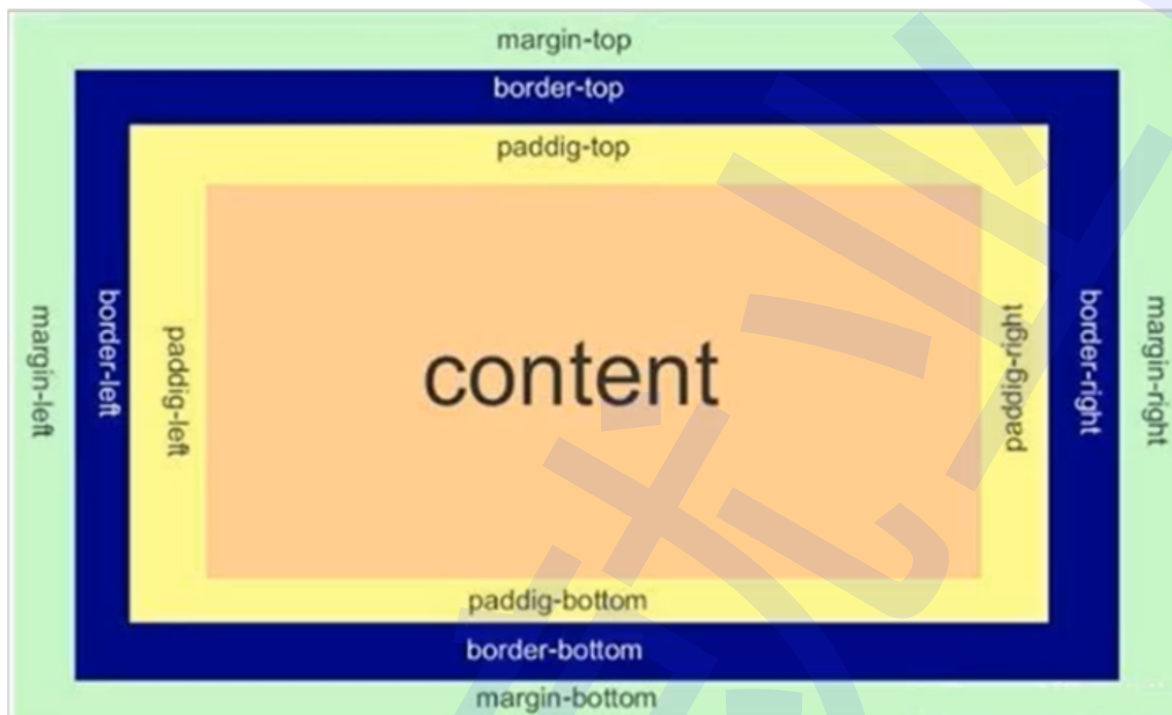
```
1 .text1 {
2     width: 200px;
```

```
3     height: 100px;
4 }
```

2.3.5 padding

padding: 内边距, 设置内容和边框之间的距离.

内容默认是顶着边框来放置的. 用 padding 来控制这个距离



```
1 .text1 {
2     padding: 20px;
3 }
```

padding也是一个复合样式, 可以对四个方向分开设置

- padding-top
- padding-bottom
- padding-left
- padding-right

2.3.6 外边距

margin: 外边距, 设置元素和元素之间的距离.

```
1 .text1 {
2     margin: 20px;
```

margin也是一个复合样式, 可以给四个方向都加上外边距

- margin-top
- margin-bottom
- margin-left
- margin-right

3. JavaScript

3.1 初识 JavaScript

3.1.1 JavaScript 是什么

JavaScript (简称 JS), 是一个脚本语言, 解释型或即时编译型的编程语言. 虽然它是作为开发Web页面的脚本语言而出名, 但是它也被用到了很多非浏览器环境中.

发展历史

JavaScript 之父 布兰登·艾奇 (Brendan Eich)

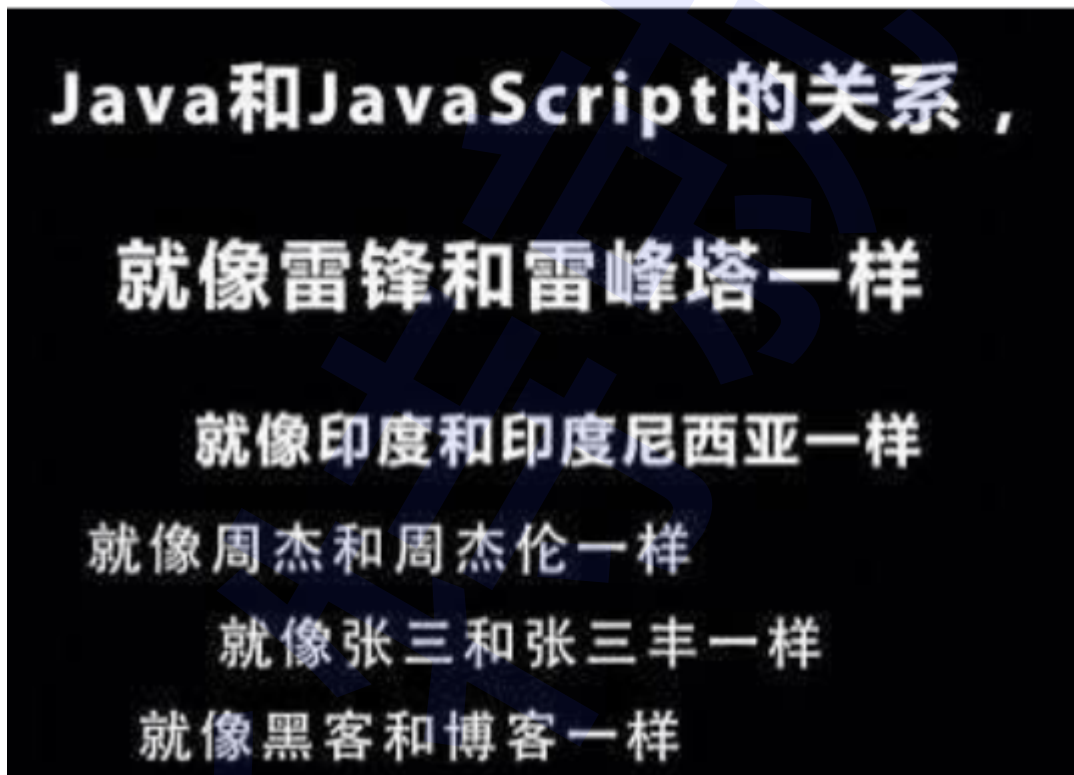


JavaScript 最初只是为了完成简单的表单验证(验证数据合法性), 结果后来不小心就火了.

1. 1994年，网景公司（Netscape）发布了Navigator浏览器0.9版。这是历史上第一个比较成熟的网络浏览器，轰动一时。但是，这个版本的浏览器只能用来浏览，不具备与访问者互动的能力。网景公司急需一种网页脚本语言，使得浏览器可以与网页互动
2. 1995年Sun公司将Oak语言改名为Java, 正式向市场推出, 并大肆宣传: 一次编写，到处运行。网景公司心动了，决定与Sun公司结成联盟。它允许Java程序以applet（小程序）的形式，直接在浏览器中运行, 甚至考虑直接将Java作为脚本语言嵌入网页, 但是这样会让HTML网页太复杂了, 不得不放弃
3. 1995年4月, Brendan Eich 入职了网景公司
4. 1995年5月，网景公司做出决策，决定开发一门新的语言, 新的语言要"看上去与Java足够相似", 但是比Java简单。Brendan Eich被指定为该语言的设计师。
5. 对Java一点兴趣都没有的Brendan Eich, 为了应付公司安排的任务, 只用10天时间就把Javascript设计出来了。(由于设计时间太短, 语言的一些细节考虑得不够严谨，导致后来很长一段时间，Javascript 写出来的程序混乱不堪)

最初在网景公司, 该语言命名为 LiveScript, 当时网景公司认为, Java作为当时最流行的编程语言, 带有 "Java" 的名字有助于这门新生语言的传播, 将 LiveScript 命名为 JavaScript.

其实 Java 和 JavaScript 之间的语法风格相去甚远.



JavaScript 发展历史可以参考阮一峰大神的博客

http://www.ruanyifeng.com/blog/2011/06/birth_of_javascript.html

JavaScript 和 HTML 和 CSS 之间的关系



- HTML: 网页的结构(骨)
- CSS: 网页的表现(皮)
- JavaScript: 网页的行为(魂)

HTML



CSS



JavaScript



3.1.2 JavaScript快速上手

1. 在HTML文件中, 写如下代码

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Document</title>
8 </head>
```



```
9
10 <body>
11     <script>
12         alert("Hello JavaScript!")
13     </script>
14
15 </body>
16
17 </html>
```

2. 运行浏览器

/test.html

此网页显示
Hello JavaScript!

确定

3.1.3 引入方式

JS有3种引入方式，语法如下表格所示：

引入方式	语法描述	示例
行内样式	直接嵌入到 html 元素内部	<code><input type="button" value="点我一下" onclick="alert('haha')"></code>
内部样式	定义<script>标签，写到 script 标签中	<code><script> alert("haha"); </script></code>
外部样式	定义<script >标签，通过src属性引入外部js文件	<code><script src="hello.js"></script></code>

3种引入方式对比:

1. 内部样式会出现大量的代码冗余, 不方便后期的维护, 所以不常用. (课堂上为了方便讲解, 使用该方式)
2. 行内样式, 只适合于写简单样式. 只针对某个标签生效. 缺点是不能写太复杂的JS.
3. 外部样式, html和js实现了完全的分离, 企业开发常用方式.

3.2 基础语法

JavaScript 虽然一些设计理念和 Java 相去甚远, 但是在基础语法层面上还是有一些相似之处的.

有了 Java 的基础很容易理解 JavaScript 的一些基本语法.

3.2.1 变量

创建变量(变量定义/变量声明/变量初始化), JS声明变量有3种方式

关键字	解释	示例
var	早期JS中声明变量的关键字, 作用域在该语句的函数内	var name = 'zhangsan';
let	ES6 中新增的声明变量的关键字, 作用域为该语句所在的代码块内	let name = 'zhangsan';
const	声明常量的, 声明后不能修改	const name = 'zhangsan';

注意事项:

1. JavaScript 是一门动态弱类型语言, 变量可以存放不同类型的值(动态), 比如:

```
1 var name = 'zhangsan';
2 var age = 20;
```

随着程序的运行, 变量的类型可能会发生改变. (弱类型)

```
1 var a = 10;    // 数字
2 a = "hehe";   // 字符串
```

Java是静态强类型语言, 在变量声明时, 需要明确定义变量的类型. 如果不强制转换, 类型不会发生变化.

2. 变量名命名规则:

- a. 组成字符可以是任何字母、数字、下划线 (_) 或美元符号 (\$)

- b. 数字不能开头
 - c. 建议使用驼峰命名
3. `+` 表示字符串拼接, 也就是把两个字符串首尾相接变成一个字符串.
4. `\n` 表示换行

3.2.2 数据类型

虽然js是弱数据类型的语言, 但是js中也存在数据类型, js中的数据类型分为: 原始类型 和 引用类型, 具体有如下类型

数据类型	描述
number	数字. 不区分整数和小数.
string	字符串类型. 字符串面值需要使用引号引起来, 单引号双引号均可.
boolean	布尔类型. true 真, false 假
undefined	表示变量未初始化. 只有唯一的值 undefined.

使用typeof函数可以返回变量的数据类型

```
1 <script>
2     var a = 10;
3     console.log(typeof a); //number
4
5     var b = 'hello';
6     console.log(typeof b); //string
7
8     var c = true;
9     console.log(typeof c); //boolean
10
11    var d;
12    console.log(typeof d); //undefined
13
14    var e = null;
15    console.log(typeof e); //null
16 </script>
```

3.2.3 运算符

JavaScript 中的运算符和 Java 用法基本相同

运算符类型	运算符
算术运算符	<code>+</code> , <code>-</code> , <code>*</code> , <code>/</code> , <code>%</code>
自增自减运算符	<code>++</code> , <code>--</code>
赋值运算符	<code>=</code> , <code>+=</code> , <code>-=</code> , <code>*=</code> , <code>/=</code> , <code>%=</code>
比较运算符	<code><</code> , <code>></code> , <code><=</code> , <code>>=</code> , <code>!=</code> , <code>!==</code> <code>==</code> 比较相等(会进行隐式类型转换) <code>===</code> 比较相等(不会进行隐式类型转换)
逻辑运算符	<code>&&</code> , <code> </code> , <code>!</code>
位运算符	<ul style="list-style-type: none"><code>&</code> 按位与<code> </code> 按位或<code>~</code> 按位取反<code>^</code> 按位异或
移位运算符	<ul style="list-style-type: none"><code><<</code> 左移<code>>></code> 有符号右移(算术右移)<code>>>></code> 无符号右移(逻辑右移)
三元运算符	条件表达式 <code>?</code> true_value <code>:</code> false_value

代码示例:

```
1 <script>
2   var age = 20;
3   var age1 = "20";
4   var age2 = 20;
5
6   console.log(age == age1); //true, 比较值
7   console.log(age === age1); //false, 类型不一样
8   console.log(age == age2); //true, 值和类型都一样
9 </script>
```

3.3 JavaScript对象

3.3.1 数组

3.3.1.1 数组定义

创建数组有两种方式

1. 使用 new 关键字创建

```
1 // Array 的 A 要大写
2 var arr = new Array();
```

2. 使用字面量方式创建 [常用]

```
1 var arr = [];  
2 var arr2 = [1, 2, 'haha', false]; // 数组中保存的内容称为 "元素"
```

注意: JS 的数组不要求元素是相同类型。

这一点和 C, C++, Java 等静态类型的语言差别很大. 但是 Python, PHP 等动态类型语言也是如此。

3.3.1.2 数组操作

1. 读: 使用下标的方式访问数组元素(从 0 开始)
2. 增: 通过下标新增, 或者使用 push 进行追加元素
3. 改: 通过下标修改
4. 删: 使用 splice 方法删除元素

代码示例:

```
1 <script>
2   var arr = [1, 2, 'haha', false];
3   //读取数组
4   console.log(arr[0]); //1
5   //添加数组元素
6   arr[4] = "add"
7   console.log(arr[4]); //add
8   console.log(arr.length); //5, 获取数组的长度
9   //修改数组元素
10  arr[4] = "update"
11  console.log(arr[4]); //update
12  //删除数组元素
13  arr.splice(4,1); // 第一个参数表示从下标为4的位置开始删除。第二个参数表示要删除的元
    素个数是 1 个
14  console.log(arr[4]); //undefined 元素已经删除, 如果元素不存在, 结果为undefined
15  console.log(arr.length); //4, 获取数组的长度
```

注意:

1. 如果下标超出范围读取元素, 则结果为 undefined
2. 不要给数组名直接赋值, 此时数组中的所有元素都没了.

相当于本来 arr 是一个数组, 重新赋值后变成字符串了.

```
1 var arr = ['小猪佩奇', '小猪乔治', '小羊苏西'];
2 arr = '小猫凯迪';
```

3.3.2 函数

3.3.2.1 语法格式

```
1 // 创建函数/函数声明/函数定义
2 function 函数名(形参列表) {
3     函数体
4     return 返回值;
5 }
6
7 // 函数调用
8 函数名(实参列表)           // 不考虑返回值
9 返回值 = 函数名(实参列表) // 考虑返回值
```

- 函数定义并不会执行函数体内容, 必须要调用才会执行. 调用几次就会执行几次.

```
1 function hello() {
2     console.log("hello");
3 }
4 // 如果不调用函数, 则没有执行打印语句
5 hello();
```

- 调用函数的时候进入函数内部执行, 函数结束时回到调用位置继续执行. 可以借助调试器来观察.
- 函数的定义和调用的先后顺序没有要求. (这一点和变量不同, 变量必须先定义再使用)

```
1 // 调用函数
2 hello();
```

```
3 // 定义函数
4 function hello() {
5     console.log("hello");
6 }
```

3.3.2.2 关于参数个数

实参和形参之间的个数可以不匹配. 但是实际开发一般要求形参和实参个数要匹配

1. 如果实参个数比形参个数多, 则多出的参数不参与函数运算

```
1 sum(10, 20, 30);    // 30
```

1. 如果实参个数比形参个数少, 则此时多出来的形参值为 undefined

```
1 sum(10);    // NaN, 相当于 num2 为 undefined.
```

JS 的函数传参比较灵活, 这一点和其他语言差别较大. 事实上这种灵活性往往不是好事.

3.3.2.3 函数表达式

另外一种函数的定义方式

```
1 var add = function() {
2     var sum = 0;
3     for (var i = 0; i < arguments.length; i++) {
4         sum += arguments[i];
5     }
6     return sum;
7 }
8 console.log(add(10, 20));    // 30
9 console.log(add(1, 2, 3, 4));    // 10
10
11 console.log(typeof add);    // function
```

此时形如 `function() { }` 这样的写法定义了一个匿名函数, 然后将这个匿名函数用一个变量来表示.

后面就可以通过这个 `add` 变量来调用函数了.

JS 中函数是一等公民, 可以用变量保存, 也可以作为其他函数的参数或者返回值.

3.3.3 对象

在 JS 中, 字符串, 数值, 数组, 函数都是对象.

每个对象中包含若干的属性和方法.

- 属性: 事物的特征.
- 方法: 事物的行为.

JavaScript 的对象 和 Java 的对象概念上基本一致. 只是具体的语法表项形式差别较大.

1. 使用 字面量 创建对象 [常用]

使用 {} 创建对象

```
1 var a = {}; // 创建了一个空的对象
2
3 var student = {
4     name: '蔡徐坤',
5     height: 175,
6     weight: 170,
7     sayHello: function() {
8         console.log("hello");
9     }
10 };
```

- 使用 {} 创建对象
- 属性和方法使用键值对的形式来组织.
- 键值对之间使用 , 分割. 最后一个属性后面的 , 可有可无
- 键和值之间使用 : 分割.
- 方法的值是一个匿名函数.

使用对象的属性和方法:

```
1 // 1. 使用 . 成员访问运算符来访问属性 `.` 可以理解成 "的"
2 console.log(student.name);
3 // 2. 使用 [ ] 访问属性, 此时属性需要加上引号
4 console.log(student['height']);
5 // 3. 调用方法, 别忘记加上 ()
6 student.sayHello();
```

2. 使用 new Object 创建对象


```

1 var student = new Object(); // 和创建数组类似
2 student.name = "蔡徐坤";
3 student.height = 175;
4 student['weight'] = 170;
5 student.sayHello = function () {
6     console.log("hello");
7 }
8
9 console.log(student.name);
10 console.log(student['weight']);
11 student.sayHello();

```

注意, 使用 {} 创建的对象也可以随时使用 `student.name = "蔡徐坤";` 这样的方式来新增属性.

3. 使用 构造函数 创建对象

```

1 function 构造函数名(形参) {
2     this.属性 = 值;
3     this.方法 = function...
4 }
5
6 var obj = new 构造函数名(实参);

```

注意:

- 在构造函数内部使用 `this` 关键字来表示当前正在构建的对象.
- 构造函数的函数名首字母一般是大写的.
- 构造函数的函数名可以是名词.
- 构造函数不需要 `return`
- 创建对象的时候必须使用 `new` 关键字.

| `this` 相当于 "我"

使用构造函数重新创建猫咪对象

```

1 function Cat(name, type, sound) {
2     this.name = name;
3     this.type = type;
4     this.miao = function () {
5         console.log(sound); // 别忘了作用域的链式访问规则
6     }
7 }

```

```
8
9 var mimi = new Cat('咪咪', '中华田园猫', '喵');
10 var xiaohei = new Cat('小黑', '波斯猫', '猫呜');
11 var ciqu = new Cat('刺球', '金渐层', '咕噜噜');
12
13 console.log(mimi);
14 mimi.miao();
```

3.4 JQuery

W3C 标准给我们提供了一系列的函数, 让我们可以操作:

- 网页内容
- 网页结构
- 网页样式

但是原生的JavaScript提供的API操作DOM元素时, 代码比较繁琐, 冗长. 我们学习使用JQuery来操作页面对象.

jQuery是一个快速、简洁且功能丰富的JavaScript框架, 于2006年发布. 它封装JavaScript常用的功能代码, 提供了简洁而强大的选择器和DOM操作方法. 使用jQuery可以轻松地选择和操作HTML元素, 从而减少了开发人员编写的代码量, 提高了开发效率, 它提供的 API 易于使用且兼容众多浏览器, 这让诸如 HTML 文档遍历和操作、事件处理、动画和 Ajax 操作更加简单. JQuery对于事件的处理也进行了简化, 提供了一个简单的API来绑定、触发和处理事件, 使开发人员能够更方便地处理各种交互行为.

3.4.1 引入依赖

使用jQuery需要先引入对应的库

在使用jQuery CDN时, 只需要在HTML文档中加入如下代码

```
1 <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
```

参考地址: <https://releases.jquery.com/>

其中, `src` 属性指明了jQuery库所在的URL. 这个URL是CDN(内容分发网络)服务提供商为jQuery库提供的一个统一资源定位符.

如果需要使用其他版本的jQuery, 可以在官网进行下载



write less, do more.

jQuery CorejQuery UIjQuery MobilejQuery ColorQUnitPEP

jQuery CDN – Latest Stable Versions

Powered by **fastly**

jQuery Core

Showing the latest stable release in each major branch [See all versions of jQuery Core.](#) [选择其他版本的jQuery](#)

jQuery 3.x

- jQuery Core 3.7.1 - [uncompressed](#), [minified](#), [slim](#), [slim minified](#)

jQuery 2.x

- jQuery Core 2.2.4 - [uncompressed](#), [minified](#)

jQuery 1.x

- jQuery Core 1.12.4 - [uncompressed](#), [minified](#)

jQuery Core – All Versions

Powered by **fastly**

jQuery Core & Migrate - Git Builds

UNSTABLE, NOT FOR PRODUCTION

- jQuery git build - [uncompressed](#), [minified](#), [slim](#), [slim minified](#)
- jQuery 3.x git build - [uncompressed](#), [minified](#), [slim](#), [slim minified](#)
- jQuery Migrate git build - [uncompressed](#), [minified](#)

jQuery Core - All 3.x Versions

- jQuery Core 3.7.1 - [uncompressed](#), [minified](#), [slim](#), [slim minified](#)
- jQuery Core 3.7.0 - [uncompressed](#), [minified](#), [slim](#), [slim minified](#)
- jQuery Core 3.6.4 - [uncompressed](#), [minified](#), [slim](#), [slim minified](#)
- jQuery Core 3.6.3 - [uncompressed](#), [minified](#), [slim](#), [slim minified](#)
- jQuery Core 3.6.2 - [uncompressed](#), [minified](#), [slim](#), [slim minified](#)
- jQuery Core 3.6.1 - [uncompressed](#), [minified](#), [slim](#), [slim minified](#)
- jQuery Core 3.6.0 - [uncompressed](#), [minified](#), [slim](#), [slim minified](#)

jQuery官方共提供了4种类型的jQuery库

uncompressed : 非压缩版本(易读, 但是文件较大, 传输速度慢)

minified: 压缩版(不易读, 文件小, 性能高, 开发中推荐)

slim: 精简瘦身版, 没有Ajax和一些特效

slim minified : slim 的压缩版

开发时, 建议把jQuery库下载到本地, 放在当前项目中. 引入外部地址, 会有外部地址不能访问的风险.

下载方式:

1. 通过浏览器访问上述连接
2. 右键 -> 另存为.... 保存到本地, 放在项目中即可.

也可以从其他CDN上下载引用jQuery

比如:

```
1 <script src="https://cdn.bootcdn.net/ajax/libs/jquery/3.7.1/jquery.min.js">
  </script>
```

3.4.2 JQuery 语法

jQuery 语法是通过选取 HTML 元素, 并对选取的元素执行某些操作

基础语法

```
1 $(selector).action()
```

- `$()` 是一个函数, 它是 jQuery 提供的一个全局函数, 用于选择和操作 HTML 元素.
- Selector 选择器, 用来"查询"和"查找" HTML 元素
- action 操作, 执行对元素的操作

jQuery 的代码通常都写在 `document ready` 函数中.

| document: 整个文档对象, 一个页面就是一个文档对象, 使用document表示.

这是为了防止文档在完全加载 (就绪) 之前运行 jQuery 代码, 即在 文档加载完成后才可以对 页面进行操作。

如果在文档没有完全加载之前就运行函数, 操作可能失败

```
1 $(document).ready(function(){
2
3   // jQuery functions go here
4
5 });
```

示例:

```
1 <button type="button">点我消失</button>
2 <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
3 <script>
4     $(document).ready(function(){
5         $('button').click(function(){
6             $(this).hide();
7         });
8     });
9
10 </script>
```

给按钮添加了click事件, 点击后元素消失.

简洁写法:

```
1 $(function(){
2
3     // jQuery functions go here
4
5 });
```

3.4.3 JQuery 选择器

我们通过JQuery选择器来选择一些HTML元素. 然后对元素进行操作.

JQuery选择器 基于已经存在的CSS选择器, 除此之外, 还有一些自定义的选择器.

jQuery 中所有选择器都以 `$` 开头: `$()`.

语法	描述
<code>\$("*")</code>	选取所有元素
<code>\$(this)</code>	选取当前 HTML 元素
<code>\$("p")</code>	所有 <code><p></code> 元素
<code>\$("p:first")</code>	选取第一个 <code><p></code> 元素
<code>\$("p:last")</code>	最后一个 <code><p></code> 元素
<code>\$(".box")</code>	所有 <code>class="box"</code> 的元素

\$("#box")	id="box" 的元素
\$(".intro .demo")	所有 class="intro" 且 class="demo" 的元素
\$(p.intro)	选取 class 为 intro 的 <p> 元素
\$(ul li:first)	选取第一个 元素的第一个 元素
\$(":input")	所有 <input> 元素
\$(":text")	所有 type="text" 的 <input> 元素
\$(":checkbox")	所有 type="checkbox" 的 <input> 元素

3.4.4 JQuery事件

JS 要构建动态页面, 就需要感知到用户的行为.

用户对于页面的一些操作(点击, 选择, 修改等) 都会在浏览器中产生一个个事件, 被 JS 获取到, 从而进行更复杂的交互操作.

浏览器就是一个哨兵, 在侦查敌情(用户行为). 一旦用户有反应(触发具体动作), 哨兵就会点燃烽火台的狼烟(事件), 后方就可以根据狼烟来决定下一步的对敌策略.

事件由三部分组成:

- 1. 事件源: 哪个元素触发的
- 2. 事件类型: 是点击, 选中, 还是修改?
- 3. 事件处理程序: 进一步如何处理. 往往是一个回调函数.

例如: 某个元素的点击事件:

```
1 $("p").click(function(){
2     //动作发生后执行的代码
3 });
```

常见的事件有:

事件	代码
文档就绪事件(完成加载)	\$(document).ready(function)
点击事件	\$(selector).click(function)
双击事件	\$(selector).dblclick(function)

元素的值发生改变	<code>\$(selector).change(function)</code>
鼠标悬停事件	<code>\$(selector).mouseover(function)</code>

3.4.5 操作元素

3.4.5.1 获取/设置元素内容

三个简单的获取元素内容的JQuery方法

JQuery方法	说明
<code>text()</code>	设置或返回所选元素的文本内容
<code>html()</code>	设置或返回所选元素的内容（包括 HTML 标签）
<code>val()</code>	设置或返回表单字段的值

这三个方法即可以获取元素的内容, 又可以设置元素的内容.
有参数时, 就进行元素的值设置, 没有参数时, 就进行元素内容的获取.

代码示例:

获取元素内容:

```
1 <div id="test"><span>你好</span></div>
2 <input type="text" value="hello">
3
4 <script>
5     $(document).ready(function () {
6         var html = $("#test").html();
7         console.log("html内容为:"+html);
8
9         var text = $("#test").text();
10        console.log("文本内容为:"+text);
11
12        var inputVal = $("input").val();
13        console.log(inputVal);
14    });
15
16 </script>
```

设置元素内容

```

1 <div id="test"></div>
2 <div id="test2"></div>
3 <input type="text" value="">
4
5 <script>
6     $(document).ready(function () {
7         $("#test").html('<h1>设置html</h1>');
8         $("#test2").text('<h1>设置text</h1>');
9         $("input").val("设置内容");
10    });
11
12 </script>

```

3.4.5.2 获取/设置元素属性

JQuery attr() 方法用于获取属性值。

代码示例

获取元素属性

```

1 <p><a href="https://www.bitejiuyeke.com/index" id="bite">比特就业课</a></p>
2
3 <script>
4     $(function(){
5         var href = $("p a").attr("href")
6         console.log(href);
7     });
8
9 </script>

```

设置元素属性

```

1 <p><a href="https://www.bitejiuyeke.com/index" id="bite">比特就业课</a></p>
2
3 <script>
4     $(function(){
5         $("p a").attr("href","baidu.com")
6         console.log($("p a").attr("href"));
7     });
8
9 </script>

```


3.4.5.3 获取/返回css属性

css() 方法设置或返回被选元素的一个或多个样式属性

代码示例

获取元素属性

```
1 <div style="font-size: 36px;">我是一个文本</div>
2
3 <script>
4     $(function(){
5         var fontSize = $("div").css("font-size");
6         console.log(fontSize);
7     });
8 </script>
```

设置元素属性

```
1 <div style="font-size: 36px;">我是一个文本</div>
2
3 <script>
4     $(function(){
5         $("div").css("font-size", "24px");
6
7     });
8 </script>
```

3.4.5.4 添加元素

添加 HTML 内容

1. append(): 在被选元素的结尾插入内容
2. prepend(): 在被选元素的开头插入内容
3. after(): 在被选元素之后插入内容
4. before(): 在被选元素之前插入内容

代码示例:

```
1 <ol>
2     <li>List item 1</li>
3     <li>List item 2</li>
```

```

4     <li>List item 3</li>
5 </ol>
6 
7 <script>
8     $(function () {
9         $("ol").append("<li>append</li>");
10        $("ol").prepend("<li>prepend</li>");
11
12        $("img").before("图片前插入");
13        $("img").after("图片后插入");
14    });
15 </script>

```

3.4.5.5 删除元素

删除元素和内容，一般使用以下两个 jQuery 方法：

1. remove()：删除被选元素（及其子元素）
2. empty()：删除被选元素的子元素。

代码示例：

```

1 <div id="div1">我是一个div</div>
2 <button>删除 div 元素</button>
3
4 <script>
5     $(function () {
6         $('button').click(function(){
7             $('#div1').remove();
8         });
9     });
10 </script>

```

删除被选元素的子元素

```

1 <ol>
2     <li>List item 1</li>
3     <li>List item 2</li>
4     <li>List item 3</li>
5 </ol>
6 <button>删除列表元素</button>
7 <script>
8     $(function () {
9         $('button').click(function(){

```

```
10      $('ol').empty();
11      });
12  });
13 </script>
```

3.5 综合案例

3.5.1 猜数字

预期效果

重新开始一局游戏

请输入要猜的数字:

已经猜的次数: 0

结果:

猜

代码实现

```
1 <button type="button" id="reset">重新开始一局游戏</button>
2 <br>
3
4 请输入要猜的数字: <input type="text" id="number">
5 <button type="button" id="button">猜</button>
6 <br>
7 已经猜的次数: <span id="count">0</span>
8 <br>
9 结果: <span id="result"></span>
10 <script>
11     $(function () {
12         // 随机生成一个 1-100 的数字
13         var guessNumber = Math.floor(Math.random() * 100) + 1 // 0 - 1 之间的数
14         var count = 0;
15         // click: 点击
16         // 事件驱动 (Event-Drive) : 只要真正发生了点击事件时, 才执行该函数
17         $("#button").click(function () {
18             count++;
19             $("#count").text(count);
20
21             var userGuess = parseInt($("#number").val());
22             if (userGuess == guessNumber) {
23                 $("#result").text("猜对了");
24                 $("#result").css("color", "gray");
25             } else if (userGuess < guessNumber) {
```

```

26         $("#result").text("猜小了");
27         $("#result").css("color","blue");
28     } else {
29         $("#result").text("猜大了");
30         $("#result").css("color","red");
31     }
32 });
33 $("#reset").click(function(){
34     guessNumber = Math.floor(Math.random() * 100) + 1;
35     count = 0;
36     $("#count").text(count);
37     $("#result").text("");
38     $("#number").val("");
39 });
40
41
42 });
43
44 </script>

```

3.5.2 表白墙

预期效果

表白墙

输入后点击提交, 会将信息显示在表格中

谁:

对谁:

说什么:

提交

需求: 按要求在文本框中输入内容, 点击提交按钮, 输入内容显示在页面下方.

代码实现

1. 提前准备如下HTML和CSS代码

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8   <style>
9     * {
10       margin: 0;
11       padding: 0;
12     }
13
14     .container {
15       width: 400px;
16       margin: 0 auto;
17     }
18
19     h1 {
20       text-align: center;
21       padding: 20px 0;
22     }
23
24     p {
25       color: #666;
26       text-align: center;
27       font-size: 14px;
28       padding: 10px 0;
29     }
30
31     .row {
32       height: 40px;
33       display: flex;
34       justify-content: center;
35       align-items: center;
36     }
37
38     span {
39       width: 100px;
40       line-height: 40px;
41     }
42
43     .edit {
44       width: 200px;
45       height: 30px;
46     }
```

```

47
48     .submit {
49         width: 304px;
50         height: 40px;
51         color: white;
52         background-color: orange;
53         border: none;
54     }
55 </style>
56 </head>
57
58 <body>
59     <div class="container">
60         <h1>表白墙</h1>
61         <p>输入后点击提交，会将信息显示在表格中</p>
62         <div class="row">
63             <span>谁: </span>
64             <input class="edit" type="text">
65         </div>
66         <div class="row">
67             <span>对谁: </span>
68             <input class="edit" type="text">
69         </div>
70         <div class="row">
71             <span>说什么: </span>
72             <input class="edit" type="text">
73         </div>
74         <div class="row">
75             <input type="button" value="提交" class="submit">
76         </div>
77     </div>
78
79 </body>
80
81 </html>

```

2. 实现提交

```

1 $(function () {
2     // 给点击按钮注册点击事件
3     $(".submit").click(function () {
4         // 1. 获取到编辑框内容
5         var from = $(''.edit:eq(0)').val();
6         var to = $(''.edit:eq(1)').val();
7         var message = $(''.edit:eq(2)').val();

```

```
8
9     console.log(from + "," + to + "," + message);
10    if (from == '' || to == '' || message == '') {
11        return;
12    }
13    // 2. 构造 html 元素
14    var html = '<div class="row">' + from + '对' + to + '说: ' + message +
15    '</div>';
16    // 3. 把构造好的元素添加进去
17    $(''.container').append(html);
18    // 4. 同时清理之前输入框的内容
19    $(':text').val("");
20 });
21
```

4. 总结

1. HTML是一种超文本标记语言, 主要用于页面内容的显示和排版. 如果需要更漂亮的样式展示和页面效果, 需要搭配CSS和JavaScript来使用
2. 学习HTML主要是学习标签, HTML的标签特别多, 了解基本语法即可
3. CSS重点是学习CSS的选择器使用
4. JavaScript是一个脚本语言, 和Java没有关系. JQuery是一个JavaScript框架, 使用JQuery可以轻松地选择和操作HTML元素, 提高我们的开发效率.
5. 前端开发对于后端开发人员而言不是很重要, 不必花费过多时间在这个上面. 达到借助网络能阅读代码的水平即可.