
黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
Web 开发
GO

目录 Contents

1. Web 开发基础

1.1 概述

下棋游戏（XXOO）是通过 Web（网页）开发实现的，因此，我们要学习 Web 开发的相关知识。

Web 开发涵盖以下内容：HTML、CSS、JavaScript （HTML、CSS不是本课程的重点，仅简单讲解）。

三者之间的关系：

- （结构）HTML 负责创建页面结构。
- （样式）CSS 负责美化页面结构（相当于化妆）。
- （行为）JavaScript 负责让页面“动”起来，解锁更多动效。

1.2 HTML

HTML（Hyper Text Markup Language，即：超文本标记语言）负责创建页面结构。

创建第一个 HTML 步骤：

1. 创建 a.html 文件。
2. 快速生成 HTML 基本骨架：在文件中输入英文叹号（!），然后，按 tab 键。
3. 创建按钮标签：在 body 标签中，输入 button，按 tab 键。
4. 打开 HTML 页面：在文件夹中找到页面文件，双击打开。

注意：页面中可见的内容，写在 body 标签中。

```
<div></div>     布局标签（独占一行）
<p></p>         段落标签（存放文字）
<h1></h1>       标题标签
```

```
<button></button>     按钮标签
<img src="" />        图片标签
...
```

1.3 CSS

CSS（Cascading Style Sheets，即：层叠样式表）负责美化页面结构。

使用 CSS 的三种方式：

1. style 属性：在 HTML 标签中，通过 style 属性来添加样式。

```
<p style="color: red; font-size: 50px;">天青色等烟雨 ...</p>
```

2. style 标签：在 head 标签中，创建 style 标签。

```
<style>
  /* 标签选择器 */
  p { color: red; }
</style>
```

技巧： 先通过选择器获取标签，再设置样式。

1.3 CSS

常用的 CSS 选择器：

```
/* 标签选择器 */
p { color: red; }
/* id 选择器 */
#txt { font-size: 50px; }
/* 类（名）选择器  -- 推荐 */
.cls { backgroud-color: pink; }
```

推荐：使用 类选择器 来给标签添加样式！

3. CSS 文件：创建 .css 文件，将样式放在该文件中，然后在 head 中通过 link 标签引入该文件。

```
<link ref="stylesheet" href="./index.css" />
```

1. Web 开发基础

总结：

作用：美化页面结构。

使用方式：

1. HTML 标签的 style 属性。

2. style 标签（在 head 标签中创建）。

3. CSS 文件（在 head 中通过 link 标签引入）。

常用的 CSS 选择器：

```
p { color: red; } /* 标签选择器 */
#txt { font-size: 50px; } /* id 选择器 */
.cls { backgroud-color: pink; } /* 类（名）选择器  -- 推荐 */
```

1.4 浏览器中使用 JavaScript

JavaScript（简称：JS），负责让页面“动”起来，为页面添加动效。

使用 JS 的两种方式：

1. script 标签：在 body 标签的最后面，创建 script 标签。

注意：console.log 方法打印的内容，需要在浏览器控制台中查看。

打开控制台（console）的方式：在页面中点击鼠标右键，选择“检查”，切换到 Console 面板。

2. 独立 js文件：创建 index.js 文件，在 body 标签的最后面，通过 script 标签引入。

```
<script src="./index.js"></script>
```

1.5 自动刷新浏览器

问题：每次修改页面内容后，都要手动刷新浏览器，才能看到修改后的内容。

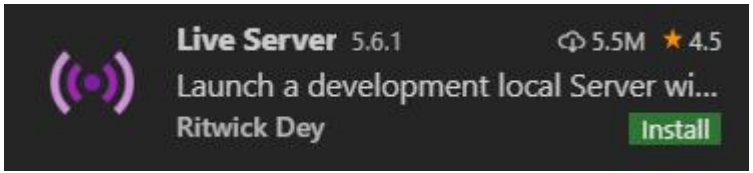
解决方式：使用 VSCode 的 Live Server 插件，实现自动刷新浏览器。

作用：监视 html 或引入的css、js的变化，在保存文件时，该插件就会帮我们自动刷新浏览器。

如何使用？　　注意：不再双击打开 html 页面！

使用方式：在 html 文件中，点击鼠标右键，再点击 Open with Live Server 按钮即可。

注意：html 文件所在的文件夹名称**不要包含中文**，否则，打开页面失败，插件功能无法生效！



目录 Contents

2. 浏览器中运行TS

注意：浏览器中只能运行 JS，**无法直接运行 TS**，因此，需要将 TS 转化为 JS 然后再运行。

浏览器中运行 TS 的步骤：

1. 使用命令 `tsc index.ts` 将 ts 文件转化为 js 文件。
2. 在页面中，使用 script 标签引入生成的 js 文件（注意是 js 文件）。

```
<script src="./index.js"></script>
```

问题：每次修改 ts 文件后，都要重新运行 tsc 命令将 ts 转化为 js 。

解决方式：使用 tsc 命令的监视模式。

```
tsc --watch index.ts
```

解释： --watch 表示启用监视模式，只要重新保存了 ts 文件，就会自动调用 tsc 将 ts 转化为 js。

目录 Contents

3. DOM 操作

3.1 概述

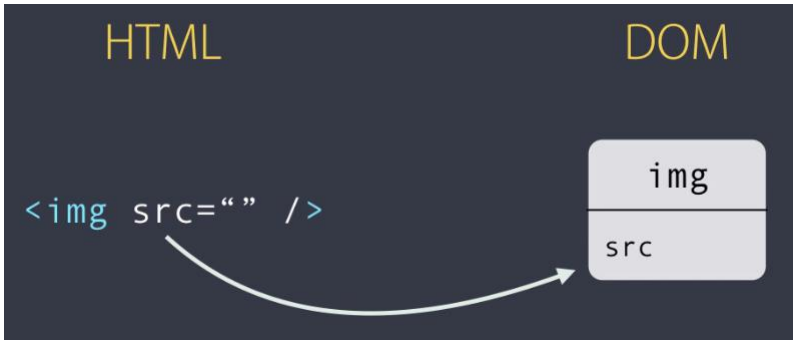
DOM（Document Object Modal）：文档对象模型。

DOM 是浏览器提供的（**浏览器特有**），专门用来**操作网页内容**的一些 JS 对象。

目的：让我们可以使用 JS/TS 代码来操作页面（HTML）内容，让页面“动”起来，从而实现 Web 开发。

HTML：超文本标记语言，用来创建网页结构。

两者的关系：浏览器根据 HTML 内容创建相应的 DOM 对象，也就是：每个 HTML 标签都有对应的 DOM 对象。



3. DOM 操作

3.1 概述

学习四个常用 DOM 操作：1 获取 DOM 元素（DOM 对象） 2 设置样式 3 设置内容 4 绑定（解绑）事件。

DOM 操作的套路（技巧）：先找人 后做事。

- 找人：获取 DOM 元素。
- 做事：设置样式、内容、绑定（解绑）事件。

```
document.title = '等你下课'
```

比如： 将 p 标签中的内容修改为：天青色等烟雨而我在等你。

步骤： 1 获取 p 元素 2 设置内容。

3. DOM 操作

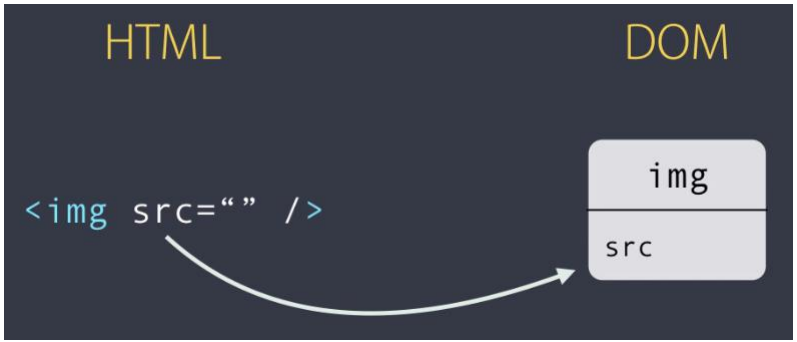
总结：

DOM 是浏览器提供的（**浏览器特有**），专门用来**操作网页内容**的一些 JS 对象（API）。

通过 DOM 操作，可以让 JS/TS 控制页面（HTML）内容，让页面“动”起来，从而实现 Web 开发。

HTML 标签和 DOM 的关系：每个 HTML 标签都有对应的 DOM 对象。

DOM 操作的套路（技巧）：先找人 后做事。



3. DOM 操作

3.2 获取元素

常用方法有两个：

- `querySelector(selector)` 作用：获取某一个 DOM 元素。
- `querySelectorAll(selector)` 作用：同时获取多个 DOM 元素。

3. DOM 操作

3.2 获取元素

1. 获取一个 DOM 元素：

```
document.querySelector(selector)
```

document 对象：文档对象（整个页面），是操作页面内容的入口对象。

selector 参数：是一个 CSS 选择器（标签、类、id 选择器等）。

作用：查询（获取）与选择器参数匹配的 DOM 元素，但是，只能获取到第一个!

推荐：使用 id 选择器（唯一）。

```
let title = document.querySelector('#title')
```

解释：获取页面中 id 为 title 的 DOM 元素。

3. DOM 操作

类型断言

问题：调用 querySelector() 通过 id 选择器获取 DOM 元素时，拿到的元素类型都是 Element。

因为无法根据 id 来确定元素的类型，所以，该方法就返回了一个**宽泛**的类型：元素（Element）类型。

不管是 h1 还是 img 都是元素。

导致新问题：无法访问 img 元素的 src 属性了。

因为：Element 类型只包含所有元素共有的属性和方法（比如：id 属性）。

类型断言

解决方式：使用类型断言，来手动指定**更加具体**的类型（比如，此处应该比 `Element` 类型更加具体）。

语法：

```
值 as 更具体的类型
```

比如：

```
let img = document.querySelector('#image') as HTMLImageElement
```

解释：我们确定 `id="image"` 的元素是图片元素，所以，我们将类型指定为 `HTMLImageElement`。

技巧：通过 `console.dir()` 打印 DOM 元素，在属性的最后面，即可看到该元素的类型。

类型断言

总结：

类型断言：手动指定**更加具体（精确）**的类型。

使用场景：当你比 TS 更了解某个值的类型，并且需要指定更具体的类型时。

```
// document.querySelector() 方法的返回值类型为：  Element

// 如果是 h1 标签：
let title = document.querySelector('#title') as HTMLHeadingElement
// 如果是 img 标签：
let image = document.querySelector('#image') as HTMLImageElement
```

技巧：通过 `console.dir()` 打印 DOM 对象，来查看该元素的类型。

3.2 获取元素

2. 获取多个 DOM 元素：

```
document.querySelectorAll(selector)
```

作用：获取所有与选择器参数匹配的 DOM 元素，返回值是一个列表。

推荐：使用 class 选择器。

示例：

```
let list = document.querySelectorAll('.a')
```

解释：获取页面中所有 class 属性包含 a 的元素。

3.3 操作文本内容

读取：

```
dom.innerText
```

设置：

```
dom.innerText = '等你下课'
```

注意：需要通过类型断言来指定 DOM 元素的具体类型，才可以使用 innerText 属性。

注意：设置内容时，会**覆盖原来的内容**。如何实现追加内容（比如，青花瓷 → 青花瓷 – 周杰伦）？

```
dom.innerText = dom.innerText + ' - 周杰伦'
// 简化
dome.innerText += ' - 周杰伦'
```

3. DOM 操作

3.4 操作样式

两种方式：

- dom.style 属性：行内样式操作，可以设置每一个样式属性（比如，字体大小、文字颜色等）。
- dom.classList 属性：类样式操作，也就是操作类名，比如，添加类名、移除类名等。

3.4 操作样式

1. style 属性（行内样式）

读取：

```
dom.style.样式名称
```

设置：

```
dom.style.样式名称 = 样式值
```

说明：所有的样式名称都与 CSS 相通，但命名规则为驼峰命名法。

```
dom.style.fontSize = '30px'
dom.style.display = 'none'
```

3.4 操作样式

2. classList 属性（类样式）

包含三个常用方法：添加、移除、判断是否存在。

添加：

```
dom.classList.add(类名1, 类名2, ...)
```

参数表示：要添加的类名，可以同时添加多个。

比如：

```
<p class="a"></p>
dom.classList.add('b', 'c')        // 添加 class 样式 ==> class="a b c"
```

3.4 操作样式

2. classList 属性（类样式）

移除：

```
dom.classList.remove(类名1, 类名2, ...)
```

参数表示：要移除的类名，可以同时移除多个。

比如：

```
<p class="a b c"></p>
dom.classList.remove('a', 'c')        // 移除 class 样式 ==> class="b"
```

3.4 操作样式

2. classList 属性（类样式）

判断类名是否存在：

```
let has = dom.classList.contains(类名)
```

参数表示：要判断存在的类名。

比如：

```
<p class="b"></p>
dom.classList.contains('a')        // false
dom.classList.contains('b')        // true
```

3.4 操作样式

总结：

类样式（classList）的操作有三种：

```
// 添加
dom.classList.add('a', 'b')
// 移除
dom.classList.remove('b', 'c')
// 判断是否存在
let has = dom.classList.contains('a')
```

3.5 操作事件

在浏览网页时，我们经常会通过移入鼠标、点击鼠标、敲击键盘等操作，来使用网站提供的功能。

如果要让我们自己实现这样的功能，就需要通过操作事件来实现了。

实际上，移入鼠标、点击鼠标、敲击键盘等，都是常见的 DOM 事件。

操作事件的两个方法：

- `addEventListener` 添加（绑定）事件。
- `removeEventListener` 移除（解绑）事件。

3.5 操作事件

1. addEventListener 添加事件

作用：给 DOM 元素添加事件。

```
dom.addEventListener(事件名称, 事件处理程序)
```

事件名称：字符串，比如：'click'（鼠标点击事件）、'mouseenter'（鼠标进入事件）。

事件处理程序：回调函数，指定要实现的功能，该函数会在触发事件时调用。

示例：鼠标点击按钮，打印内容。

```
btn.addEventListener('click', function () {
  console.log('鼠标点击事件触发了')
})
```

3.5 操作事件

事件对象（event），是事件处理程序（回调函数）的参数。

表示：与当前事件相关的信息，比如：事件类型（type）、触发事件的 DOM 元素（target）等。

```
btn.addEventListener('click', function (event) {
  console.log(event.type)            // click
  console.log(event.target)          // btn 元素
})
```

3.5 操作事件

2. removeEventListener 移除事件

作用：移除给 DOM 元素添加的事件，移除后，事件就不再触发了。

```
dom.removeEventListener(事件名称, 事件处理程序)
```

事件名称：同添加事件的第一个参数。

事件处理程序：**必须要跟添加事件时的事件处理程序是同一个**，否则无法移除！

3.5 操作事件

2. removeEventListener 移除事件

正确方式：

```
function handleClick() {}
btn.addEventListener('click', handleClick)
btn.removeEventListener('click', handleClick)
```

说明：添加和移除事件时，事件处理程序是同一个，都是函数 handleClick。

错误演示：

```
btn.addEventListener('click', function () {})
btn.removeEventListener('click', function () {})
```

注意：以上两个函数虽然长的一样，却是不同的函数（双胞胎，不是同一个人）。

3.5 操作事件

如果事件只需要触发一次，可以在添加事件时处理。

处理方式：传入第三个参数，将 once 属性设置为 true。

```
btn.addEventListener('click', function () {}, { once: true })
```

once：如果值为 true，会在触发事件后，自动将事件移除，达到只触发一次的目的。

3.5 操作事件

移除事件总结：

当 DOM 元素的事件不再使用时，就可以通过 removeEventListener 方法移除事件。

注意：**添加和移除的事件处理程序必须是同一个**，否则无法移除！

```
function handleClick() {}
btn.addEventListener('click', handleClick)
btn.removeEventListener('click', handleClick)
```

如果事件只需要触发一次，可以在添加事件时，通过 once 属性来实现。

```
btn.addEventListener('click', function () {}, { once: true })
```

函数声明形式的事件处理程序说明

1. 可以先使用函数，再声明函数。

```
btn.addEventListener('click', handleClick)
function handleClick() {}
```

原因：函数声明在当前 ts 文件中的任意位置都有定义。

```
// 1 先调用函数
fn()
// 2 再声明函数
function fn() {}
```

函数声明形式的事件处理程序说明

2. 使用事件对象参数时，应该指定类型注解，否则，访问事件对象的属性时没有任何提示。

```
btn.addEventListener('click', handleClick)
function handleClick(event: MouseEvent) {
  console.log(event.target)
}
```

技巧：使用原始方式（匿名回调函数）查看参数类型。

函数声明形式的事件处理程序说明

总结：

函数声明在当前 ts 文件中的任意位置都有定义。

```
btn.addEventListener('click', handleClick)
function handleClick() {}
```

```
fn()                    // 先调用函数
function fn() {}        // 再声明函数
```

在函数声明形式的事件处理程序中，使用事件对象时，应该指定参数类型。

```
btn.addEventListener('click', handleClick)
function handleClick(event: MouseEvent) {}
```

传智播客旗下高端IT教育品牌