

**网络安全社团网站项目**

**小**

**组**

**代**

**码**

**规**

**范**

G14

目录

目录

[一、HTML 代码规范 3](#_Toc20509)

[1.文档声明与编码 3](#_Toc25074)

[2.标签使用原则 3](#_Toc2708)

[3.缩进与格式： 4](#_Toc16254)

[4.属性规范 4](#_Toc30915)

[5.注释清晰： 4](#_Toc22421)

[二、CSS 代码规范 4](#_Toc9005)

[1.选择器命名规则 4](#_Toc12398)

[2.样式声明顺序 5](#_Toc32764)

[3.代码缩进与换行 5](#_Toc5398)

[4.注释与空格使用 5](#_Toc19818)

[5.避免内联样式 5](#_Toc29874)

[三、 JavaScript 代码规范 5](#_Toc10368)

[1.变量命名规范 5](#_Toc216)

[2.函数命名与定义 5](#_Toc2680)

[3.代码缩进与括号使用： 6](#_Toc11217)

[4.注释与代码复用 6](#_Toc27274)

[5.严格模式与错误处理 6](#_Toc28093)

[四、通用规范与团队协作 6](#_Toc28412)

[1.文件组织与命名： 6](#_Toc12264)

[2.版本控制与代码合并： 6](#_Toc12183)

[3.代码审查与团队沟通： 6](#_Toc30687)

# 一、HTML 代码规范

## 1.文档声明与编码

始终以<!DOCTYPE html>作为 HTML 文档开头声明其为 HTML5 文档，使用<meta charset="UTF-8">确保字符编码统一为 UTF-8，支持多语言字符显示，避免乱码问题。

## 2.标签使用原则

* 1. 标签应语义化清晰，如标题用<h1> - <h6>标签层级明确，列表用<ul>、<ol>配合<li>标签准确表示无序列表与有序列表，段落用<p>标签，避免滥用<div>和<span>。
  2. 自闭合标签格式规范，如<img>、<input>、<br>等需正确闭合，<img src="image.jpg" alt="描述" />，空格和斜杠位置准确，保证 HTML 结构完整性与解析正确性。

## 3.缩进与格式：

采用统一缩进风格，通常为 2 或 4 个空格，增强代码可读性与层次结构，便于团队成员理解 HTML 文档布局和元素嵌套关系。

## 4.属性规范

* 1. 属性值用双引号包裹，如<a href="https://example.com">链接</a>，保持一致性，避免单引号与双引号混用引发错误。
  2. 布尔属性简化写法，如<input type="checkbox" checked>表示选中状态，去除冗余属性值，遵循 HTML 标准语法简洁性原则。

5.注释清晰：

关键代码段和复杂结构添加注释说明用途与功能逻辑，便于后续维护与他人理解代码意图。

# 二、CSS 代码规范

## 1.选择器命名规则

* 1. 采用有意义、语义化的命名方式，遵循 BEM（Block Element Modifier）或类似命名规范变体，增强样式与 HTML 结构关联性，提升代码可维护性。例如，.header-nav\_\_item--active表示头部导航栏中当前激活的菜单项，\_\_区分块元素与子元素，--标识修饰符状态。
  2. 避免过度简写或使用无意义字符组合命名选择器，确保名称清晰传达样式作用对象与功能，降低理解成本。

## 2.样式声明顺序

遵循逻辑顺序排列样式声明，如先布局（display、position、float、width、height等），再盒模型（margin、padding、border等），接着文本样式（font-size、color、text-align等），最后其他样式（background、opacity、z-index等）。

## 3.代码缩进与换行

与 HTML 缩进风格统一，用 2 或 4 个空格缩进选择器内样式声明，保持代码层次清晰；长样式声明换行书写，提高可读性

## 4.注释与空格使用

重要样式声明或样式块添加注释解释用途与作用范围；属性声明间适当空格分隔，提升视觉清晰度与可读性，如：margin: 0 10px 15px 5px;而非margin:0 10px 15px 5px;。

## 5.避免内联样式

尽量将样式集中于 CSS 文件管理，减少 HTML 内联样式使用，利于样式复用、维护和整体样式一致性控制；特殊情况少量内联样式使用需谨慎权衡，并添加注释说明必要性。

# JavaScript 代码规范

## 1.变量命名规范

采用小驼峰命名法（camelCase）命名变量，首字母小写，后续单词首字母大写，变量名表意明确，反映存储数据含义，如userName、productList、isLoggedIn等。

## 2.函数命名与定义

* 1. 函数名遵循小驼峰命名法，语义清晰准确传达函数功能，如calculateTotalPrice()、validateFormData()等；函数声明提前，置于使用前，保证代码执行顺序逻辑清晰、可预测，便于理解和调试。
  2. 函数参数命名遵循变量命名规则，参数数量合理控制，过多参数可考虑封装对象传递，增强函数可读性与易用性。

## 3.代码缩进与括号使用：

统一用 4 个空格缩进代码块；函数声明、控制结构（if、else、for、while等）括号使用规范，括号独占一行，增强代码块辨识度与层次感。

## 4.注释与代码复用

关键代码逻辑、复杂算法及函数用途添加详细注释，解释功能、参数意义、返回值等关键信息；注重代码复用，提取公共函数或模块，避免重复代码编写，提高开发效率与代码可维护性。

## 5.严格模式与错误处理

使用严格模式（'use strict';）提升代码质量与错误捕获能力，在函数或脚本顶部声明启用；合理使用try...catch语句处理异步操作或可能抛错代码段，优雅处理错误，避免程序崩溃，确保用户体验流畅性与稳定性。

# 四、通用规范与团队协作

## 1.文件组织与命名

项目文件依功能模块或页面结构合理组织目录，如styles目录存放 CSS 文件、scripts目录放置 JavaScript 文件、views或pages目录存 HTML 文件；文件命名语义清晰反映内容，统一用小写字母、单词间用连字符分隔，如main.css、index.js、contact-page.html。

## 2.版本控制与代码合并

使用版本控制系统（如 Git）管理代码，团队成员频繁提交代码变化并推送远程仓库；分支开发协作，合并代码前解决冲突，遵循团队制定合并流程与规范，保证代码集成稳定性与可靠性；提交注释准确描述修改内容、目的与影响范围，方便回溯与审查代码历史。

## 3.代码审查与团队沟通

定期开展代码审查活动，成员交叉审查代码质量、规范遵循与潜在问题；审查注重代码功能正确性、性能优化、安全漏洞及规范一致性等方面；团队成员间保持密切沟通交流，及时讨论技术难题、分享最佳实践经验，促进共同进步与项目顺利推进。