# HTML与CSS入门

众所周知，Web前端几乎只有HTML、CSS、JavaScript这三样东西。其他别的东西大多都是这三种东西特别是JS这种东西的发展。

上次活动我们稍微介绍了HTML *(Hyper-Text Markup Language)*，其实演示中也操作了CSS *（Cascading Style Sheets）*。那么这次我们继续了解一下他们。

## HTML

HTML是非常古老的一门语言

- **1989 HTML被首次提出**
- 1994 关于CSS的建议被首次展示
- 1996 JavaScript的首个版本被应用于浏览器

这二十多年来HTML发展至今已经到HTML5了，从目前的情况来看，HTML是Web前端三件套中争论最少的部分。

而且他也非常简单，也几乎不需要什么学习成本。我们只需要打开一个文本编辑器。然后尽力发挥想象力

> 所以首先，我再推荐几个文本编辑器吧，尽管我已经推荐过很多次了
>
> - Sublime Text
> - Atom
> - Visual Studio Code
>
> 个人观点：
>
> - Sublime Text比较简单和便捷
> - Atom对插件支持最好，最好看，但插件会导致比较明显的性能下降
> - VS Code性能好，插件还可以，但是相较之下比较大型

准备好文本编辑器之后，我们看看，最简单的一段HTML5代码应该长成这样

```
<!DOCTYPE html>
<html lang="zh_CN">
  <head>
    <meta charset="utf-8">
    <title>Ubtitled</title>
  </head>
  <body>
    Hello World!
  </body>
</html>
```

> 我将它写在了 `try1.html` 中

将他拷贝进文本编辑器，然后保存成 `.html` 文件（文件名前面部分随意），然后用浏览器打开它。

> 那么我们再推荐几款优秀的并且适合开发人员的浏览器
>
> - Mozilla Firefox

> - Google Chrome
>
> 还有更多吗，没有了。现在你就可以把那些什么 *360安全浏览器* 、 *QQ浏览器* 卸载了。
>
> 个人拙见：
>
> - Firefox最快、流畅，而且安全、可靠，开源，最重要是优雅（最新的Firefox非常酷）
> - Chrome对于上面说的各方面也都优秀，但是它更像是Google生态的一部分，所以我并不认为他非常专注于做一个优雅的浏览器

当你用浏览器打开，你应该会看见这样的一个页面


看完热闹之后我们来看看这段代码的一些细节。如果你足够仔细，你会发现这里面有很多 `<;xxx>;` ，这些尖括号包裹着一些东西，而且他们一般成对出现 `<;xxx>;...<;/xxx>;` ，中间包裹着一些内容，当然他们有时也只有孤零零一个。这些东西我们称为HTML的 **标签**。再继续深入之前，我们先看看这些标签在这里的含义

- `<;!DOCTYPE html>;` 指示文本属性为HTML5
- `<;html>;...<;/html>;` 内部包含HTML代码
- `<;head>;...<;/head>;` 包含该HTML页面的一些配置信息
- `<;body>;...<;/body>;` 包含该页面的内容
- `<;title>;...<;/title>;` 显示在页面标题栏的内容（注意看上图左上角）
- `<;meta charset=";utf-8";>;` 指示代码文字编码为 `UTF-8` ，这将保证页面中中文的正常显示。

我们暂时不打算对上述标签再做更多的讲解，我们现在关注一下 `<;body>;...<;/body>;` 中的内容，单纯写文字肯定很无趣。但我们可以给文字加很多附加含义。看下面这段稍复杂的代码 `try2.html`

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>The Zen of Python</title>
  </head>
  <body>
    <h1>The Zen of <a title="Python - Official Site" href="https://www.python.org/">Python</a>,
by Tim Peters</h1>
    <p>Beautiful is better than ugly.</p>
    <p>Explicit is better than implicit.</p>
    <p>Simple is better than complex.</p>
    <p>Complex is better than complicated.</p>
    <p>Flat is better than nested.</p>
    <p>Sparse is better than dense.</p>
    <p>Readability counts.</p>
    <p>Special cases aren't special enough to break the rules.</p>
    <p>Although practicality beats purity.</p>
    <p>Errors should never pass silently.</p>
    <p>Unless explicitly silenced.</p>
    <p>In the face of ambiguity, refuse the temptation to guess.</p>
    <p>There should be one-- and preferably only one --obvious way to do it.</p>
    <p>Although that way may not be obvious at first unless you're Dutch.</p>
    <p>Now is better than never.</p>
    <p>Although never is often better than <em>right</em> now.</p>
    <p>If the implementation is hard to explain, it's a bad idea.</p>
    <p>If the implementation is easy to explain, it may be a good idea.</p>
    <p>Namespaces are one honking great idea -- let's do more of those!</p>
  </body>
</html>
```


# The Zen of Python, by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than *right* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

> 注意到，我们基本上只在 `try1.html` 的基础上修改了 `<;body>;<;/body>;` 内的内容。
>
> 除此之外我修改了 `lang=";en";` ，这个 **属性** 其实是标识页面主要语言的，既然没有一个汉字，我还是改为英文吧
>
> 还有就是修改了 `<;title>;<;/title>;` ，Title是随便你定的

你可能已经发现了，对文字应用不同的标签会使文字呈现出不一样的效果。HTML本质上就是，用大量结构性[1]的、语义[2]化的标签和标签中的文本内容描述一个文档。常用的标签如下

> 他们没有非常多的技术含量，你们可以很快比较熟练地使用它
>
> 介绍一个适合的学习网站W3School

在这里我们列出一些常用的标签，你们自己通过上面的网站学习并试着使用他们，你会发现这非常容易。

## 块级元素

- `<;h1>;` ~ `<;h6>;` ，一至六级标题
- `<;p>;` ，段落
- `<;div>;` ，无特殊含义的块级框（单纯用于结构的标签）
- `<;ol>;` 、 `<;ul>;` 、 `<;li>;` ，列表
- `<;form>;` 、 `<;input>;` 等，表单
- ...

## 行内元素

- `<;a>;` ，锚（超链接）
- `<;b>;` 、 `<;em>;` ，强调
- `<;span>;` ，无特殊含义的行内框
- ...

更深入的内容就暂时不说了，我们就稍微解释一下什么是行内元素，什么是块级元素。因为如果不加区分，在使用CSS修饰他们时可能会产生非常多的混乱。以下是对他们最通俗的理解

- **块级元素**，他对外的表现和一个大方块是类似的。如果不做额外设置，他们的宽度将为整个可视窗口大小，这意味这这个块的左右一般是没有别的内容的。
- **行内元素**，他对外的表现就和别的文本一样。他可以随文本流动，嵌在一般的文本中，他的左右都可以有别的内容。

更深入的理解可以自行Google，先接着讲解一下HTML的救星 -- CSS

# CSS

CSS是如何产生的呢，以及对Web应用有什么意义？在CSS之前，HTML并不是完全乏味的，他有一个很神奇的标签 `<;font>;` ，他给HTML带来了丰富多彩的样式（文字的字体和颜色），比如这样

```
<font size="8" color="grey" face="consolas">Hello World!</font>
```

Hello World!

> 上面只是一个页面的HTML代码的一个片段，他并不应该被单独写在一个文件中，像上面的其他代码一样，这段东西一般来说应该放在 `<;body>;<;/body>;` 中

`<;font>;` 标签给页面带来了更棒的效果，这种东西在过去被广泛且大量地使用。而且各主流浏览器大厂还为 `<;font>;` 标签增加了更多的功能（因为毕竟HTML标准最终是由浏览器落实的）以实现更复杂和酷炫的效果。开发人员也很乐于写他们，以愉悦老板。

故事讲到这里似乎非常美好，Web前端的事业将会在这条路上越走越远，最终发展成今天这样，拥有赋予页面无比华丽特效的能力。 **事实当然不是这样！**

我们想象一些这样一个场景，你的老板让你写一个页面，那里面有很多，比如100个，特殊的字词，为了表示某种强调效果，你需要让他们都是红色的。很好，于是你需要写100个 `<;font>;` 标签，并将他们的 `color` 属性设置成合适的值。这看起来问题不大，因为我们都会复制粘贴嘛，100个也没什么大不了的。

好的，你的老板看了之后很高兴，并且告诉你“我们应该低调一点，红色太扎眼了，还是把他们都换成蓝色吧。”还能接受吗？其实问题不大，只有100个。因为真实情况一般是，有很多页面，各种特殊样式的文字分布在各个角落，你需要把他们一个一个找出来并且做修改。

而且 `<;font>;` 标签一般都是修饰行内元素的，他对块级元素无能为力。而且 `<;font>;` 标签不带有语义，尽管你把某些字设置为某种颜色或者某种字体、大小，但是你不能从代码中看出为什么要这么做。最重要的是，HTML设计的理念是用标签表示语义和结构，而不应该用他们表示样式。由此带来的混乱将会极大地限制Web前端的发展。所以，CSS是作为HTML的救星而出现的

---

那我们看看CSS到底长什么样吧。

```
<p style="font-family: Consolas; font-size: 40px; color: grey;">Hello World</p>
```

> 以上写法效果和上面的 `<;font>;` 写的效果类似，可以自行尝试

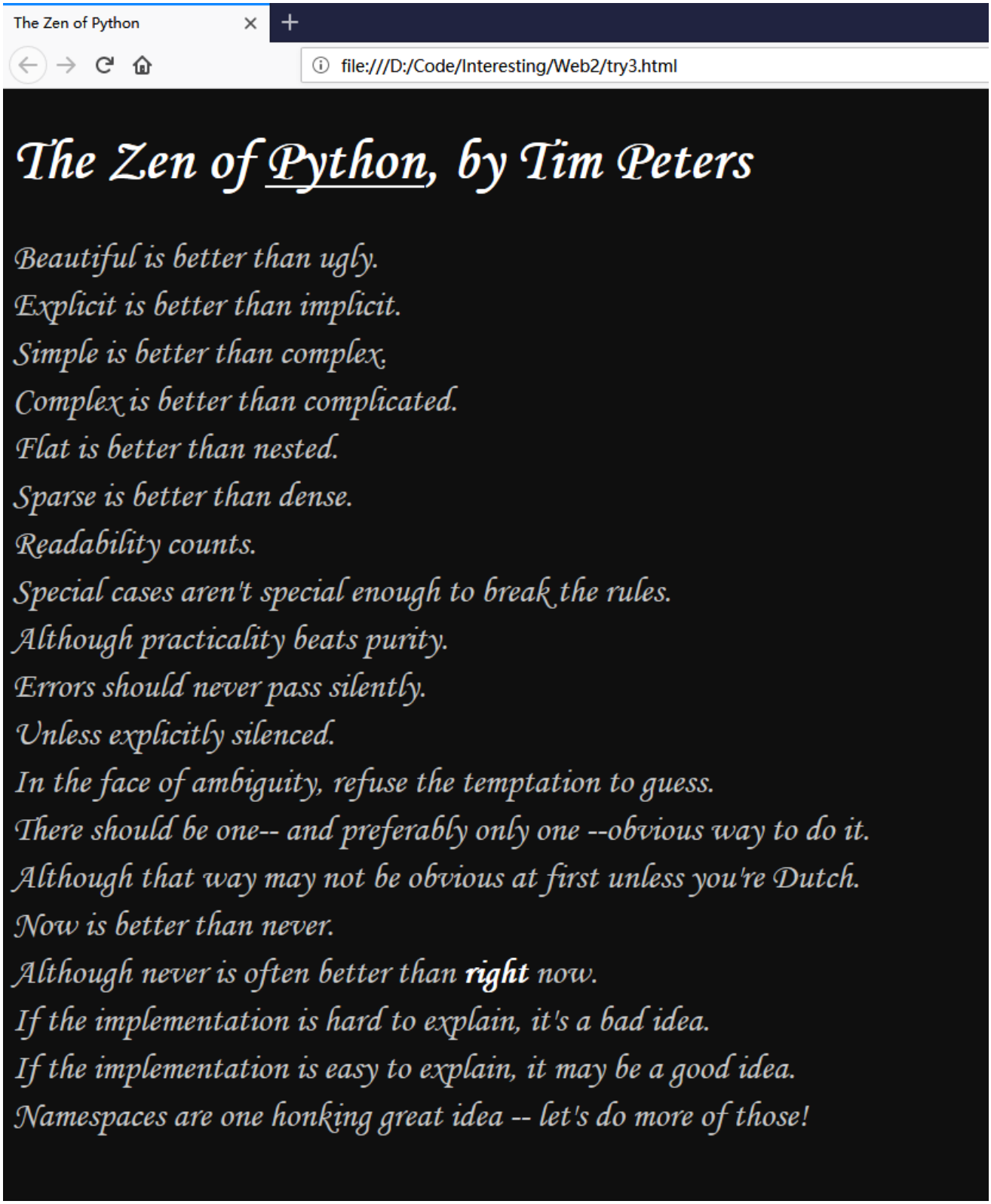
CSS可以写在绝大多数元素标签中，作为 `style` 属性的值。这样一看，貌似CSS也没有比 `<;font>;` 高到哪里去呀，不就是和一般标签结合了嘛，还是那么复杂（甚至字数还多了）。其实即使是这样CSS也有很明显的优势，他可以修饰块级元素，他可以控制元素位置、大小、背景、边框、边距、...反正他能控制的比 `<;font>;` 多得多；他和一般标签结合，这意味着他并不会破坏文档的结构，我们并不需要为了某些效果，增加一堆没有语义的标签。最重要的是，CSS还有另外两种更有用的存在形式。

上面的例子中，CSS写在 `style` 属性中，这种方式叫做 **内联的样式表** ，CSS还有另外两种更常用的形式

## 内部的样式表

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>The Zen of Python</title>
    <style>
      body {
        background-color: #0f0f0f;
        font-family: "Monotype Corsiva";
      }
      a, em, b {
        color: #ffffff;
        font-weight: bolder;
      }
      h1 {
        color: #ffffff;
        font-size: 50px;
      }
      p {
        margin: 5px 0;
        color: #c8c8c8;
        font-size: 30px;
      }
    </style>
  </head>
  <body>
    <h1>The Zen of <a title="Python - Official Site" href="https://www.python.org/">Python</a>,
by Tim Peters</h1>
    <p>Beautiful is better than ugly.</p>
    <p>Explicit is better than implicit.</p>
    <p>Simple is better than complex.</p>
    <p>Complex is better than complicated.</p>
    <p>Flat is better than nested.</p>
    <p>Sparse is better than dense.</p>
    <p>Readability counts.</p>
    <p>Special cases aren't special enough to break the rules.</p>
    <p>Although practicality beats purity.</p>
    <p>Errors should never pass silently.</p>
    <p>Unless explicitly silenced.</p>
    <p>In the face of ambiguity, refuse the temptation to guess.</p>
    <p>There should be one-- and preferably only one --obvious way to do it.</p>
    <p>Although that way may not be obvious at first unless you're Dutch.</p>
    <p>Now is better than never.</p>
    <p>Although never is often better than <em>right</em> now.</p>
    <p>If the implementation is hard to explain, it's a bad idea.</p>
    <p>If the implementation is easy to explain, it may be a good idea.</p>
    <p>Namespaces are one honking great idea -- let's do more of those!</p>
  </body>
</html>
```

> 这段代码写在 `try3.html` 中




明显，这漂亮了很多，而且，我们并没有在每个元素标签中都加上 `style` 属性。取而代之的是在文件的 `<;head>;` `<;/head>;` 中加了一个 `<;style>;<;/style>;` 标签。而且你会发现CSS语句变得很不一样，如果你仔细看他们的格式，你会发现，在这里，除了描述样式的语句，还有一部分指示样式应用于什么元素的语句，比如 `p {...}` 表示大括号内的语句是描述页面中所有的 `<;p>;<;/p>;` 标签的。由此整个页面的样式可以集中写在文件头部一个地方，你再也不用为了修改一个颜色，找遍整个页面。

## 外部的样式表

到这里CSS已经很完美了，但是并不够。我们经常看到，有些网站有很多页面，每个页面虽然内容不同，但都有非常相似的风格。难道我们需要在每个页面中都写上很大一段一模一样的CSS代码吗。并不用，我们可以把CSS写在一个单独的文件中，然后仅通过一条语句即可引入到HTML页面中。

首先写一个 `.css` 文件，比如我用的是 `main.css`

```
body {
  background-color: #0f0f0f;
  font-family: "Monotype Corsiva";
}
a, em, b {
  color: #ffffff;
  font-weight: bolder;
}
h1 {
  color: #ffffff;
  font-size: 50px;
}
p {
  margin: 5px 0;
  color: #c8c8c8;
  font-size: 30px;
}
```

将这个文件保存在 `try3.html` 的同一个目录中，然后我们可以把 `try3.html` 中那一大段 `<;style>;...<;/style>;` 换成这样一句话

```
<link href="./main.css" rel="stylesheet">
```

> `<;link>;` 标签用于为页面引入一些静态文件，比如图标、样式表； `href` 属性指示静态文件所在的位置，这是一个URL，可以使用绝对目录，也可以使用相对目录； `rel` 属性指示引入文件的类型是样式表

你把它们保存好，然后用浏览器打开 `try3.html` ，会发现，效果还是一样的，这个 `main.css` 会被浏览器自动加载（还记得上次活动讲的那堆自动访问的URL、自动发送的HTTP请求吗，这就是其中之一）。而且这个 `main.css` 还可以被你用在其他页面中以获得和 `try3.html` 相似的风格。

## CSS语法基础

CSS的学习比HTML要难一些，而且，CSS总是要结合HTML来设计的，这同样在W3School中有合适的教程，大家可以跟着边学边试。我们只阐述几个学习的要点

- 注意区分应用于行内元素和块级元素的样式
- 熟练使用各种选择器可以让你的样式设计比较有条理，但是注意一点，一般来说 **id选择器** 尽量不用， `id` 一般留给JS使用，CSS主要使用 **class选择器**
- 样式规则不用死记硬背，要用时查查就是了，背再多也不会用，用多了就记住了
- 多思考CSS的组织方式，尽量使他们保持整洁且易于维护，这非常重要，这可以避免你被自己写的代码气死

---

好的，那我们至此已经讲完了HTML和CSS的一些基础，这些主要靠自己去学。因为其实很简单，这两个东西也不是前端的难点，从开始学习他们到可以写出一个像样的页面应该可以只花不到一个月（只要你愿意花时间），前端的重难点和争论最聚集的地方其实是JavaScript，这个我们一段时间后再讲。

学习HTML/CSS最好的方法是，边学边试，但不要只试教程的例子，应该尽量尝试完成一个一般你们能见到的页面，比如SS::STA - Official Site，看着一个别人的页面，尽量尝试自己去复现。试着试着遇到不行的部分就查，这样提高非常快。

推荐一些学习资料

> W3School作为HTML/CSS的学习教程或手册
>
> 《CSS权威指南》现在英文版据说出到第4版了，中文版只到第3版，不急着买，作为CSS的进阶读物
>
> 《JavaScript高级程序设计》学习JS的合适读物，不急着买，双十二之前你可能都不会需要涉及到JS

---

1. 结构一般是指各种父子、兄弟关系，就是包围与被包围和并列关系↩

2. 语义是指标签包含关于文本在文档中的功能的附加含义↩