你不可不知的 HTML 优化技巧

2017-07-14 前端大全

> 作者：伯乐在线/葡萄城控件
> http://web.jobbole.com/87908/
> 如有好文章投稿，请点击 → 这里了解详情

如何提升Web页面的性能，很多开发人员从多个方面来下手如JavaScript、图像优化、服务器配置，文件压缩或是调整CSS。

很显然HTML 已经达到了一个瓶颈，尽管它是开发Web 界面必备的核心语言。HTML页面的负载也是越来越重。大多数页面平均需要40K的空间,像一些大型网站会包含数以千计的HTML 元素，页面Size会更大。

如何有效的降低HTML 代码的复杂度和页面元素的数量，本文主要解决了这个问题，从多个方面介绍了如何编写简练，清晰的HTML 代码，能够使得页面加载更为迅速，且能在多种设备中运行良好。

在设计和开发过程中需要遵循以下原则：

- 结构分离：使用HTML 增加结构，而不是样式内容；
- 保持整洁：为工作流添加代码验证工具；使用工具或样式向导维护代码结构和格式
- 学习新语言：获取元素结构和语义标记。
- 确保可访问： 使用ARIA 属性和Fallback 属性等
- 测试： 使网站在多种设备中能够良好运行，可使用emulators和性能工具。

HTML5

Sensible meta elements:
events.google.com/io2015 has more

Use external CSS files, inline with build
Put links in the <head>, no type needed
Void elements don't need closing slash

Put headings in heading elements

Remembered closing tags

Make use of HTML5 semantic elements

Begin article elements with <h1>:
see text for explanation

alt attribute for accessibility
sizes and srcset for responsiveness

<img> is self-closing: no closing slash

Maintain heading hierarchy:
heading level adds meaning to content

Set class on parent, not all children

 closing tag is optional...

...but is definitely not

The right kind of list

Boolean attributes don't have a value:
if the attribute is there, it's true

Provide alternative file types

Provide alternative content

Fallback content

Label element for accessibility

Use input type and placeholder

Script elements at the bottom
No type attribute required...
...but closing tag IS required

```
<!DOCTYPE html>
<html>
<head>

  <meta charset="utf-8">
  <meta name="description" content="Best pesto recipes">
  <meta name="author" content="//samdutton.com">
  <meta name="viewport" content="width=device-width, minimum-scale=1.0, initial-scale=1">
  <meta itemprop="name" content="Pesto recipes">
  <meta itemprop="image" content="/images/icon_192x192.png">
  <meta name="mobile-web-app-capable" content="yes">
  <meta id="theme-color" name="theme-color" content="#fff">
  <meta name="application-name" content="Pesto recipes">

  <title>Pesto: classic</title>

  <link rel="shortcut icon" href="/images/icon_192x192.png">

  <link rel="stylesheet" href="/css/global.css">
  <link rel="stylesheet" href="css/local.css">

</head>

<body>

  <h1>Best pesto recipes</h1>

  <p>Pesto is good to eat...</p>
  <p>...and pesto is really easy to make!</p>

  <article>

    <h1>Classic pesto</h1>

    <p>This is the simplest and best of all pesto recipes.</p>

    <img src="classic-small.jpg" sizes="50vw"
      srcset="classic-small.jpg 500w, classic-medium.jpg 1000w, classic-large.jpg 1500w"
      alt="Making pesto with a mortar and pestle">

    <h2>Ingredients</h2>

    <ul class="ingredients">
      <li>Basil
      <li>Pine nuts
      <li>Garlic
      <li>Olive oil
      <li>Salt
    </ul>

    <h3>Method</h3>

    <ol>
      <li>Chop ingredients.
      <li>Mash ingredients into a paste using a mortar and pestle.
      <li>Eat on spaghetti boiled with some green beans and sliced potato.
    </ol>

    <video autoplay controls>
      <source src="video/classic-en.webm" type="video/webm">
      <source src="video/classic-en.mp4" type="video/mp4">
      <track label="English subtitles" kind="subtitles" srclang="en"
        src="tracks/classic-subtitles-en.vtt" default>
      <p>This browser does not support the video element.</p>
    </video>

  </article>

  <footer>
    <label for="newsletter-email">Subscribe to the newsletter:</label>
    <input id="newsletter-email" placeholder="email address" type="email">
    <button id="newsletter-button">Subscribe</button>
  </footer>

  <script src="/js/global.js"></script>
  <script src="js/local.js"></script>
  <script src="/js/lib/ga.js"></script>

</body>
</html>
```

HTML、CSS 和JavaScript三者的关系

HTML 是用于调整页面结构和内容的标记语言。HTML 不能用于修饰样式内容，也不能在头标签中输入文本内容，使代码变得冗长和复杂，相反使用CSS 来修饰布局元素和外观比较合适。HTML元素默认的外观是由浏览器默认的样式表定义的，如在Chrome中h1标签元素会渲染成32px的Times 粗体。

三条通用设计规则：

1. 使用HTML 来构造页面结构，CSS修饰页面呈现，JavaScript实现页面功能。CSS ZenGarden 很好地展示了行为分离。
2. 如果能用CSS或JavaScript实现就少用HTML代码。
3. 将CSS和JavaScript文件与HTML 分开存放。这可有助于缓存和调试。

文档结构方面也可以做优化，如下：

使用HTML5 文档类型，以下是空文件：

```
<!DOCTYPE html>
<html>

<head>
<title>Recipes: pesto</title>
</head>

<body>

 <h1>Pesto</h1>

 <p>Pesto is good!</p>

</body>
</html>
```

在文档起始位置引用CSS文件，如下：

```
<head>
 <title>My pesto recipe</title>

 <link rel="stylesheet" href="/css/global.css">
 <link rel="stylesheet" href="css/local.css">

</head>
```

使用这两种方法，浏览器会在解析HTML代码之前将CSS信息准备好。因此有助于提升页面加载性能。

在页面底部body结束标签之前输入JavaScript代码，这样有助于提升页面加载的速度，因为浏览器在解析JavaScript代码之前将页面加载完成，使用JavaScript会对页面元素产生积极的影响。

```
<body>

 ...

 <script src="/js/global.js">
 <script src="js/local.js">

</body>
```

使用Defer和async属性，脚本元素具有async 属性无法保证会按顺序执行。

可在JavaScript代码中添加Handlers。千万别加到HTML内联代码中，比如下面的代码则容易导致错误且不易于维护：

index.html:

```
<head>

 ...

 <script src="js/local.js">

</head>

<body onload="init()">

 ...

 <button onclick="handleFoo()">Foo</button>

 ...

</body>
```

下面的写法比较好：

index.html:

```
<head>
 ...
</head>
<body>
 ...
 <button id="foo">Foo</button>
 ...
 <script src="js/local.js">
</body>
```

js/local.js:

```
init();
var fooButton =
   document.querySelector('#foo');
fooButton.onclick = handleFoo();
```

验证

优化网页的一种方法就是浏览器可处理非法的HTML 代码。合法的HTML代码很容易调试，且占内存少，耗费资源少，易于解析和渲染运行起来更快。非法的HTML代码让实现响应式设计变得异常艰难。

当使用模板时，合法的HTML代码显得异常重要，经常会发生模板单独运行良好，当与其他模块集成时就报各种各样的错误，因此一定要保证HTML代码的质量，可采取以下措施：

- 在工作流中添加验证功能：使用验证插件如HTMLHint或SublineLinter帮助你检测代码错误。
- 使用HTML5文档类型

- 确保HTML的层次结构易于维护，要避免元素嵌套处于左开状态。
- 保证添加各元素的结束标签。
- 删除不必要的代码；没有必要为自关闭的元素添加结束标签；Boolean 属性不需要赋值，如果存在则为True;

```
<video src="foo.webm" autoplay="" controls=""/>
```

代码格式

格式一致性使得HTML代码易于阅读，理解，优化，调试。

语义标记

语义指意义相关的事物，HTML 可从页面内容中看出语义：元素和属性的命名一定程度上表达了内容的角色和功能。HTML5 引入了新的语义元素，如<header>，<footer>及<nav>。

选择合适的元素来编写代码可保证代码的易读性：

- 使用<h1>(<h2>,<h3>...)表示标题，<ul>或<ol>实现列表
- 注意使用<article> 标签之前应添加<h1>标签；
- 选择合适的HTML5语义元素如<header>，<footer>,<nav>,<aside>;
- 使用<p>描述Body 文本，HTML5 语义元素可以形成内容，反之不成立。
- 使用和标签替代<i>和<b>标签。
- 使用<label>元素，输入类型，占位符及其他属性来强制验证。
- 将文本和元素混合，并作为另一元素的子元素，会导致布局错误，

例如：

```
<div>Name: <input type="text" id="name"></div>
```

换种写法会更好

```
<div>
  <label for="name">Name:</label><input type="text" id="name">
</div>
```

布局

要提高HTML代码的性能，要遵循HTML 代码以实现功能和为目标，而不是样式。

- 使用<p>元素修饰文本，而不是布局；默认<p>是自动提供边缘，而且其他样式也是浏览器默认提供的。
- 避免使用
分行，可以使用block元素或CSS显示属性来代替。
- 避免使用<hr>来添加水平线，可使用CSS的border-bottom 来代替。
- 不到关键时刻不要使用div标签。
- 尽量少用Tables来布局。
- 可以多使用Flex Box
- 使用CSS 来调整边距等。

CSS

虽然本文讲解的是如何优化HTML，下面介绍了一些使用css的基本技能：

- 避免内联css
- 最多使用ID类 一次
- 当涉及多个元素时，可使用Class来实现。

以上就是本文介绍的优化HTML代码的技巧，一个高质量高性能的网站，往往取决于对细节的处理，因此我们在日常开发中，能够考虑到用户体验，后期维护等方面，则会产生更高效的开发。