¿Qué son los estándares web?

Cuando hablamos de estándares web, a lo que nos referimos son especificaciones formales a las que Internet y todo lo que debe cumplir. Por lo tanto, con frecuencia se trata menos de cómo aparece la interfaz de un sitio web y más sobre cómo está estructurado el back-end del mismo.

Los estándares web no se centran solo en el desarrollo web tampoco. Tocan navegadores, HTTP, software de diseño y desarrollo, así como dispositivos de consumo. Esencialmente, los estándares web se desarrollan y formalizan para dar fuerza y consistencia al núcleo de la web. Cuanto más nos adherimos a estos estándares, más accesible será la web para todos.

Incluso si no está involucrado en la codificación de sus sitios web, es probable que esté familiarizado con los estándares web actuales:

HTML, CSS y JavaScript válidos

El código mal escrito puede causar muchos problemas para el rendimiento de un sitio web, sin mencionar los errores que puede introducir. Entonces, esta fue una de las primeras cosas que necesitábamos para manejar.

Dado que HTML, CSS y JavaScript forman la columna vertebral de la web, existen estándares estrictos relativos a cómo se escriben y cuándo se usan. Además, a medida que las variaciones de estos idiomas entran en el léxico de la web, como HTML5 y CSS3, también se crean estándares para ellos.

Al estandarizar la codificación, hacemos posible que todos los desarrolladores y diseñadores hablen el mismo idioma, y que cada navegador web o software los comprenda.

Gráficos

Este es uno importante para los diseñadores web, aunque no es tanto un estándar estricto como un conjunto de mejores prácticas para usar gráficos en la web. Por ejemplo, esto es lo que recomienda el W3C:

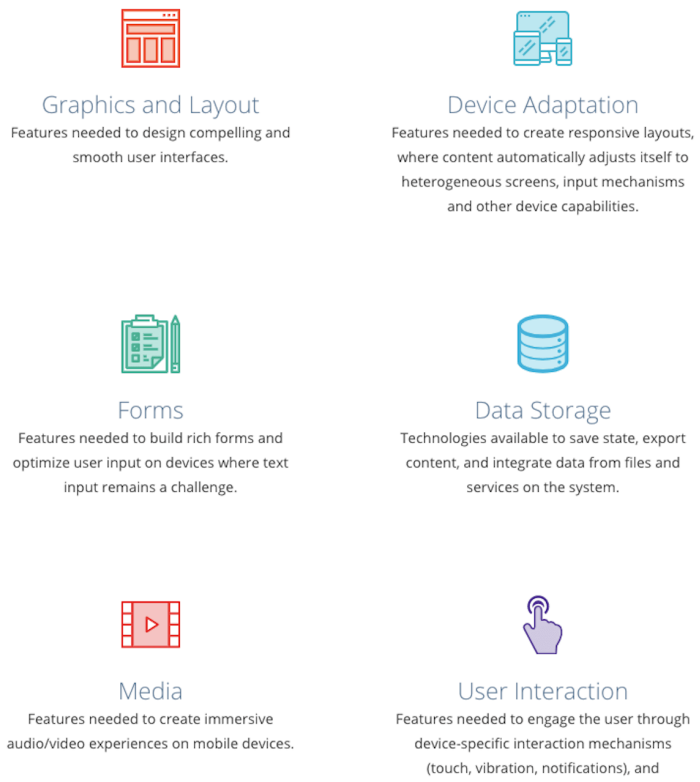
- PNG para fotos;
- SVG para visualización de datos;
- CSS para mejorar HTML básico;
- API de Canvas para crear degradados, formas y otros efectos de diseño;
- WebCGM para gráficos vectoriales.

Si desea que su sitio web funcione de la manera más eficiente posible, es importante tomar en serio recomendaciones como estas.
Capacidad de respuesta móvil

Con la proliferación de dispositivos inteligentes y la inmensa variación en los tipos de dispositivos disponibles, se ha vuelto crítico tener estándares para la web móvil.

Dicho esto, los organismos de normalización no solo han estandarizado el diseño receptivo. También han creado un conjunto de mejores prácticas para la web móvil.

Aquí hay un ejemplo de algunos de los estándares web móviles que tenemos:



Las pautas no solo se proporcionan para el diseño o las herramientas utilizadas tampoco. También se centran en cosas como el procesamiento de pagos, la seguridad del sitio web y el rendimiento.

El modelo de estándares web: HTML, CSS y JavaScript

En el último apartado hemos visto brevemente las herramientas de construcción básicas de la web: HTML (o XHTML), CSS y JavaScript. Ahora es el momento de ampliar un poco más y observarlos individualmente: qué hacen y cómo interactúan interaccionan entre ellos para crear una página web.

1. ¿Por qué separar?

Ésta es, generalmente, la primera pregunta que se formula sobre los estándares web. Se puede crear el contenido, el estilo y el formato sólo utilizando HTML: elementos de tipo de letra para el estilo y tablas HTML para el formato, de manera que, ¿por qué preocuparse de este asunto de XHTML/CSS? El uso de tablas para el formato, etc., es como se solía hacer en los malos tiempos del diseño web, y mucha gente todavía lo hace de esta manera (aunque no se debería de hacer) y, de hecho, es uno de los motivos por los que hemos creado este curso. Aquí no trataremos estos métodos. Éstas son las razones más importantes para utilizar CSS y HTML en vez de métodos obsoletos:

1. **Eficiencia del código**: cuantos mayores sean los archivos, más tardarán en descargarse y más dinero les costará a algunas personas (algunas personas todavía pagan por megabyte descargado y las tarifas móviles acostumbran a tener límites estrictos). Por lo tanto, no se debe malgastar el ancho de banda con páginas grandes abarrotadas de información de estilo y de formato en cada archivo HTML. Una alternativa mucho mejor es que los archivos HTML estén desglosados y limpios, e incluir la información de estilo y de formato sólo una vez en un archivo CSS independiente o en varios.

   Eficiencia del código

   Para ver un caso real de este hecho en acción, podéis ver el artículo sobre la reescritura de Slashdot, en A List Apart Slashdot, donde el autor tomó una página web muy popular y la reescribió en XHTML/CSS.

2. **Facilidad de mantenimiento**: en relación con el último punto, si la información de estilo y formato sólo se especifica en un sitio, quiere decir que sólo habrá que hacer actualizaciones en un lugar si se quiere cambiar el aspecto de la página web. ¿Preferiríais tener que actualizar esta información en cada página de vuestra web? No lo creo.

3. **Accesibilidad**: los usuarios de la web con problemas visuales pueden utilizar un tipo de software conocido como "lector de pantalla" para acceder a la información mediante el sonido en vez de la vista; literalmente, les lee la página. Además, el software de introducción de datos mediante la voz que utilizan las personas con problemas de movilidad también se beneficia de las páginas web con una semántica bien construida. De manera muy parecida al lector de pantalla que utiliza las instrucciones del teclado para navegar por los encabezamientos, enlaces y formularios, un usuario que interaccione interaccione mediante voz utilizará instrucciones con su voz. Los documentos web marcados semánticamente, en vez de presentacionalmente, pueden resultar más fáciles de navegar y la información que contienen es más fácil de encontrar por parte del usuario. En otras palabras, cuanto más rápidamente "entre en materia" (el contenido), mejor. Los lectores de pantalla no pueden acceder al texto dentro de imágenes y encuentran confusos algunos usos de JavaScript. Aseguraos de que el contenido más importante está disponible para todo el mundo.

4. **Compatibilidad de dispositivos**: ya que vuestra página HTML es simplemente etiquetado normal, sin información de estilo, se puede reformatear para diferentes dispositivos con atributos ampliamente variados (por ejemplo, el tamaño de pantalla), simplemente aplicando una hoja de estilos alternativa; podéis hacerlo de varias maneras. CSS también permite especificar hojas de estilo diferentes a nivel nativo para distintos métodos de presentación o tipos de soporte (por ejemplo, visualización en pantalla, impresión, visualización en un dispositivo móvil, etc.).

5. **Motores de búsqueda en la web**: probablemente os interesará que vuestras páginas web sean fáciles de encontrar al buscar en Google o en otros motores de busca. Un motor de búsqueda utiliza un *crawler* o araña web, que es una pieza de software especializada para leer vuestras páginas. Si esta araña tuviera problemas para encontrar el contenido de vuestras páginas o interpretara mal lo que es importante porque no habéis definido los encabezamientos como encabezamientos, etc., entonces seguro que vuestra posición en los resultados de busca se verá afectada.
6. **Es simplemente una buena práctica**: se trata de un motivo un poco del tipo "porque lo digo yo", pero hablad con cualquier desarrollador o diseñador web profesional consciente de los estándares y seguro que os dice que separar el contenido, el estilo y el comportamiento es la mejor manera de desarrollar una aplicación.

2. Etiquetado, la base de cada página

HTML y XHTML son lenguajes de etiquetado formados por elementos que contienen atributos (algunos opcionales y otros obligatorios). Estos elementos se utilizan para etiquetar los diferentes tipos de contenidos en los documentos, especificando que cada trozo de contenido se debe entregar supuestamente como en los navegadores web (por ejemplo, encabezamientos, párrafos, tablas, listas con viñetas, etc.).

Como sería de esperar, los **elementos** definen el tipo de contenido actual, mientras que los **atributos** definen información adicional sobre estos elementos, como una ID para identificar el elemento o una ubicación para que le señale un enlace. Deberíais tener en cuenta que se supone que el etiquetado es el más semántico posible, es decir, se supone que debe describir la función del contenido de la manera más cuidada posible. La figura 1 muestra la anatomía de un elemento (X)HTML característico.

Figura 1. Anatomía de un elemento (X)HTML.

Teniendo esto en cuenta, ¿cuál es la diferencia entre HTML y XHTML?

2.1. ¿Qué es el XHTML?

La *X* de XHTML quiere decir "extensible", es decir, ampliable. Una de las preguntas más habituales para los que empiezan es: "¿he de utilizar HTML o XHTML, y qué diferencia hay?". Son casi lo mismo; la principal diferencia radica en la estructura. Podéis consultar la tabla 1 para ver las diferencias.

HTML	XHTML

Tabla 1. Diferencias entre HTML y XHTML.

Los elementos y atributos no distinguen entre mayúsculas y minúsculas, `<h1>` es lo mismo que `<H1>`.	Los elementos y atributos distinguen entre mayúsculas y minúsculas; todos están en minúsculas.
Algunos elementos no necesitan una etiqueta de cierre (por ejemplo, párrafos, `<p>`), mientras que otros (denominados *elementos vacíos*) prohíben la marca de cierre (por ejemplo, las imágenes, `<img>`).	Todos los elementos se deben cerrar claramente (por ejemplo, `<p>A paragraph</p>`). Los elementos sin contenido se pueden cerrar utilizando una barra inclinada en la marca inicial (por ejemplo, `<hr></hr>` y `<hr/>` significan lo mismo). Si servís el texto XHTML como `text/html`, deberéis utilizar la sintaxis abreviada en todos los elementos que están definidos como "vacíos" y colocar un espacio antes de la barra inclinada. Deberíais utilizar la forma larga (con marcas inicial y finales independientes) en cualquier elemento que no esté definido como vacío, aunque no tenga ningún contenido.
Algunos valores de atributos pueden estar escritos sin estar entre comillas.	Los valores de los atributos se han de incluir entre comillas.
Algunos atributos se pueden abreviar (por ejemplo, `<option selected>`).	Se debe utilizar la forma de atributo entera para todos los atributos (por ejemplo, `<option selected="selected">`).
Los servidores habrían de servir el HTML al cliente con un tipo de medio `text/html`.	El XHTML debería utilizar el tipo de medio `application/xhtml+xml` pero se puede utilizar `application/xml`, `text/xml` o `text/html`. Si se utiliza `text/html`, se deberían seguir las directrices de compatibilidad de HTML porque los navegadores lo tratarán como HTML (y utilizar la recuperación de errores para representar las diferencias entre idiomas).

Por ahora, os recomendamos que no os preocupéis mucho sobre si estáis escribiendo HTML o XHTML. Seguid los consejos que se dan a lo largo de este curso y utilizad el tipo de documento HTML e iréis por el buen camino.

2.2. Validación, ¿qué es eso?

Como HTML y XHTML son estándares establecidos (y CSS también, en realidad), el World Wide Web Consortium (W3C) ha creado una gran herramienta denominada **validador** que comprueba automáticamente las páginas que queráis y señala cualquier problema-error que pueda tener vuestro código, como la falta de etiquetas de cierre o la falta de comillas alrededor de los atributos.

- El validador de HTML está disponible en línea en http://validator.w3.org/. Detectará automáticamente si estáis utilizando HTML o XHTML y qué tipo de documento estáis usando.
- Si queréis comprobar el CSS, el validador está disponible en http://jigsaw.w3.org/css-validator/

3. CSS: añadimos un poco de estilo

Las **hojas de estilo en cascada** permiten un control preciso sobre el formato y la disposición del documento. Podéis cambiar o añadir colores, fondo, tipo de letra, tamaños y estilos de tipo de letra, e incluso la posición de elementos de vuestra página web en diferentes lugares.

Hay tres maneras básicas de aplicar estilos mediante CSS: redefinición de un elemento, aplicación de un estilo a una ID o aplicación de un estilo a una clase. Echemos un vistazo a cada uno de ellos:

1. **Redefinición de un elemento**. Podéis cambiar la forma en la que cualquier elemento de (X)HTML se muestra mediante la definición de una regla de estilo.

   Ejemplo de redefinición de un elemento

   Si queréis que todos los párrafos sean a doble espacio y en verde, podéis añadir esta declaración en CSS:

   ```
   p {
      line-height: 2;
      color: green;
   }
   ```

   Ahora, todo el contenido incluido entre las etiquetas `<p></p>` tendrá una altura de línea doble y será de color verde.

2. **Definición de una ID**. Podéis darle a un elemento un atributo `id` para identificarlo de manera única en una página (cada ID se puede utilizar sólo una vez por página), por ejemplo, `id="navigation_menu"`. Esto os permite un control más preciso sobre el formato de una página.

   Ejemplo de definición de una ID

   Si sólo queréis que un párrafo determinado sea a doble espacio y destacado con texto verde, asignad una ID.

   ```
   <p id="highlight">Contenido del párrafo</p>
   ```

   Y entonces aplicadle una regla CSS, tal como se indica a continuación:

   ```
   #highlight {
   ```

```
   line-height: 2;
   color: green;
}
```

Esto sólo aplicará la regla CSS al párrafo de la página con un atributo `id` del tipo `highlight` (el símbolo de almohadilla es sólo una convención de CSS para indicar que se trata de una ID).

3. **Definición de una clase**. Las clases son como las ID, excepto que se puede tener más de un elemento de la misma clase en cada página.

Ejemplo de definición de una clase

Siguiendo con nuestro ejemplo de doble espacio, si queréis que los dos primeros párrafos de una página sean a doble espacio y estén destacados, les habríais de añadir clases como:

```
<p class="highlight">Contenido del párrafo</p>
<p class="highlight">El contenido del segundo párrafo</p>
```

Y entonces aplicadles una regla CSS como se indica a continuación:

```
.highlight {
   line-height: 2;
   color: green;
}
```

En este caso, `highlight` es una clase, no una ID: el punto sólo es una convención de CSS para indicar que se trata de una clase.

El ejemplo siguiente os dará una idea mejor de cómo CSS aplica estilos a HTML.

4. JavaScript: adición de comportamiento a las páginas web

Por último, JavaScript es el lenguaje de *script* que se utiliza para añadir comportamiento a sus páginas web.

JavaScript se puede utilizar para validar los datos que se introducen en un formulario (discriminar si están en el formato correcto o no), para ofreceros la funcionalidad de arrastrar y soltar, para cambiar estilos sobre la marcha, para animar elementos de las páginas como los menús, para tratar las funciones de los botones y un millón de cosas más.

La mayoría del JavaScript moderno funciona localizando un elemento HTML de destino y después haciéndole algo, igual que el CSS, pero la manera de funcionamiento, la sintaxis, etc., es bastante diferente.

El JavaScript es un tema más complicado y extenso que el HTML y el CSS, de modo que para poner las cosas simples y evitar confusiones tan pronto no lo incluimos en el ejemplo siguiente.

5. Una página de ejemplo

Hay muchos detalles que no hemos incluido, pero lo trataremos todo durante este curso de diseño web. Por ahora, os presentaremos una página real de ejemplo, sólo para que veáis una pequeña muestra de aquello con lo que trabajaréis durante el resto de apartados.

El ejemplo que presentamos a continuación es una página de referencias que se puede utilizar para citar referencias al final de, pongamos por caso, un ensayo psicológico sobre la dinámica de grupo de un equipo de desarrollo web, o un informe para trabajar sobre el uso de Internet de banda ancha en Estados Unidos. Tened en cuenta que si sois muy estrictos con la redacción académica canónica, este ejemplo muestra el formato APA (tenía que elegir uno).

5.1. `index.html`

```
<!DOCTYPE html>

<html>
<head>
  <meta charset="utf-8" />
  <title>References</title>
  <style type="text/css">
    @import url("styles.css");
  </style>
</head>
<body>
  <div id="bggraphic"></div>
  <div id="header">
    <h1>References</h1>
  </div>
  <div id="references">
    <cite class="article">Adams, J. R. (2008). The Benefits of Valid
Markup:
     A Post-Modernistic Approach to Developing Web Sites. <em>The Journal
     of Awesome Web Standards, 15:7,</em> 57-62.</cite>
    <cite class="book">Baker, S. (2006). <em>Validate Your Pages....
     Or Else!.</em> Detroit, MI: Are you out of your mind
publishers.</cite>
    <cite class="article">Lane, J. C. (2007). Dude, HTML 4, that's like
     so 2000. <em>The Journal that Publishes Genius, 1:2, </em> 12-
34.</cite>
    <cite class="website">Smith, J. Q. (2005). <em>Web Standards and
You.</em>
     Retrieved May 3, 2007 from <a
href="http://www.webstandardsandyou.com/">
     Web standards and you</a>.</cite>
  </div>
  <div id="footer">
    <p>The content of this page is copyright © 2007
      <a href="mailto:jonathanlane@gmail.com">J. Lane</a></p>
  </div>
</body>
</html>
```

No haremos una disección de este archivo línea por línea, ya que veréis muchos ejemplos en futuros apartados; sin embargo, a continuación, se indican algunos elementos importantes que cabe tener en cuenta.

La línea 1 es lo que se denomina *declaración de tipo de documento* o *doctype*. En este caso, la página es "HTML5" (aunque no explota todas las características de HTML5). El tipo de documento especifica una serie de reglas que debe seguir el etiquetado y gracias a las que se puede validar.

Las líneas 5 a 7 importan un archivo CSS a la página: los estilos contenidos en este archivo se aplicarán a los diferentes elementos de la página. En el siguiente subapartado veréis el contenido del archivo CSS que trata todo el formato de la página.

Hemos asignado una clase diferente a cada uno de los diferentes tipos de referencia. Hacerlo de esta manera nos permite aplicar un estilo diferente a cada tipo de referencia, así, en nuestro ejemplo hemos puesto un color diferente a la derecha de cada referencia para que sea más fácil analizar la lista.

Ahora demos un vistazo al CSS que aplica estilo al HTML.

5.2. `styles.css`

```
body{
  background: #fff url('images/gradbg.jpg') top left repeat-x;
  color: #000;
  margin: 0;
  padding:0;
  border: 0;
  font-family: Verdana, Arial, sans-serif; font-size: 1em;
}
div {
  width: 800px;
  margin: 0 auto;
}
#bggraphic {
  background: url('images/pen.png') top left no-repeat;
  height: 278px;
  width: 362px;
  position: absolute;
  left: 50%;
  z-index: 100;
}
h1 {
  text-align: center;
  text-transform: uppercase;
  font-size: 1.5em;
  margin-bottom: 30px;
  background: url('images/headbg.png') top left repeat;
  padding: 10px 0;
}
#references cite {
  margin: 1em 0 0 3em;
  text-indent: -3em;
  display: block;
  font-style: normal;
  padding-right: 3px;
}
.website {
  border-right: 5px solid blue;
}
.book {
  border-right: 5px solid red;
}
.article {
  border-right: 5px solid green;
}
#footer {
  font-size: 0.5em;
  border-top: 1px solid #000;
  margin-top: 20px;
}
#footer a {
  color: #000;
  text-decoration: none;
}
#footer a:hover{
  text-decoration: underline;
}
```

Hemos exagerado un poco con el estilo de esta página y hemos añadido algunos bonitos efectos de fondo para mostraros algunas de las cosas que se pueden conseguir con el CSS.

La línea 1 establece algunos valores predeterminados para el documento, como el color del texto y del fondo, el ancho del borde que se debe añadir en torno al texto, etc. Algunas personas no se preocuparán de indicar explícitamente valores predeterminados como éstos, y la mayoría de los navegadores modernos aplicarán sus propios valores predeterminados, pero es una buena idea, ya que le permite un control mayor sobre cómo se verá vuestra página web en diferentes navegadores.

En la línea siguiente hemos establecido la anchura de página en 800 px (aunque en este caso podríamos haber especificado un porcentaje para que la página se expanda/contraiga en función del tamaño de la ventana del navegador). La configuración de márgenes que hemos utilizado garantizará que el contenido de la página siempre se mantenga centrado en la ventana.

Las imágenes PNG semitransparentes no funcionan en Internet Explorer 6 o versiones anteriores, pero funcionan prácticamente en todos los navegadores modernos; podéis consultar la corrección de JavaScript para el IE de Dean Edwards para una solución para IE 6 a este problema, que también resuelve algunos de los problemas de compatibilidad con CSS de IE 6.

Pasemos ahora a fijarnos en las imágenes de fondo utilizadas en la página (se aplican utilizando las declaraciones de fondo: url). En esta página tenemos 3 elementos de fondos diferentes. El primero es una gradación de azul a blanco que empieza desde la parte superior de la página. En segundo lugar, hemos utilizado un PNG semitransparente para el gráfico de la pluma para crear un poco de contraste con el texto que tiene encima y respetar la gradación. Por último, hemos utilizado otro PNG para el fondo del encabezamiento de página. Le da al título un poco más de contraste y ofrece un efecto de calidad.

El ejemplo queda como se ve en la figura 2.

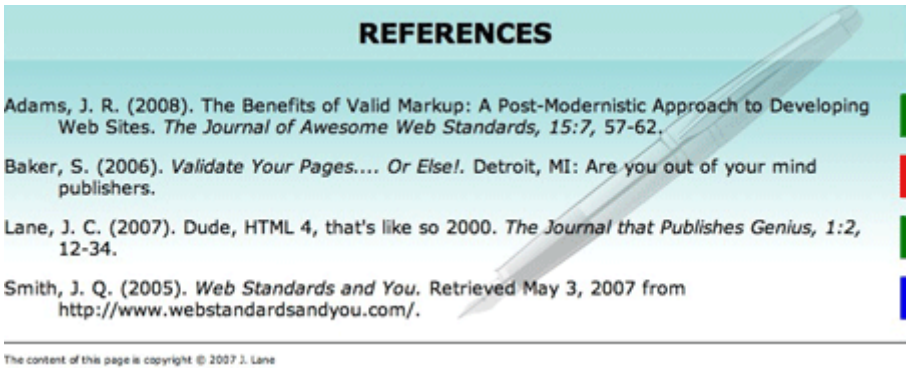


Figura 2. El ejemplo acabado con los estilos aplicados.

Resumen

(X)HTML, CSS y JavaScript no tienen nada de complicado. Son simplemente una evolución de la web. Si ya habéis tenido un poco de contacto con HTML, lo podéis aprovechar perfectamente. Todo lo que sabéis sigue siendo válido, sólo hay que tratar

algunos conceptos de manera diferente (y tener un poco más de cuidado con el etiquetado).

Aparte de tener la satisfacción de un trabajo bien hecho, el desarrollo de los estándares web es totalmente lógico. Los argumentos en contra del desarrollo con estándares son que se debe invertir mucho tiempo y es muy pesado hacer que un diseño funcione en distintos navegadores. Se podría utilizar el argumento contrario para hacer que un formato no basado en estándares funcione en diferentes dispositivos y con futuras versiones de navegadores. ¿Cómo se puede tener la certeza de que un etiquetado que se sostiene con pinzas se verá bien en Opera 87, Firefox 100 e IE 14? No se puede, a no ser que se sigan algunas reglas.