



HTML5

INTRODUCCIÓN A LA WEB SEMÁNTICA

HTML5 (*HyperText Markup Language*, versión 5) es la quinta revisión importante del lenguaje básico de HTML.

HTML5 también es un término de marketing para agrupar las nuevas tecnologías de desarrollo de aplicaciones web: HTML5, CSS3 y nuevas capacidades de Javascript.

Es importante mencionar que, HTML4 y HTML5 son 100% compatibles entre sí, por lo tanto todo el código creado en HTML básico seguirá funcionando sin problemas en HTML5.

HTML5 especifica dos variantes de sintaxis para HTML:

- «clásico» HTML (text/html), la variante conocida como *HTML5*.
- XHTML conocida como sintaxis *XHTML5* que deberá ser servida como XML (XHTML) (application/xhtml+xml). Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.

Nota: Al no ser reconocido en viejas versiones de navegadores por sus nuevas etiquetas, se le recomienda al usuario común actualizar a la versión más nueva, para poder disfrutar de todo el potencial que provee HTML5.

Principios Fundamentales

En el año 2007 el W3C publica un documento llamado W3C: HTML Design Principles, Principios de diseño HTML. Este documento describe un conjunto de principios básicos que guiarán al grupo de trabajo que desarrollará HTML5:

- Compatibilidad
- Utilidad
- Interoperabilidad
- Acceso universal

En la introducción diferencia entre un **lenguaje conforme** a las especificaciones y un **lenguaje soportado**. La evolución de HTML deberá tener en cuenta ambos conjuntos, pues requerirá que siga soportando características de especificaciones anteriores así como otras que, sin estar en ellas, han sido y son ampliamente utilizadas.

Compatibilidad

HTML-5 debe ser compatible con versiones anteriores y posteriores, pero lo más importante es que HTML debe ser sobre todo una **evolución** más que una **revolución**:

- **Soportar contenido existente**

Los navegadores deben soportar cualquier especificación anterior de HTML, incluso en aquellas características que ya no formaban parte de las últimas especificaciones. De ninguna forma la evolución de HTML podrá dejar de lado todo el extenso contenido ya existente que fue escrito basado en especificaciones anteriores. Por ejemplo, el elemento `<u>` para subrayado ya fue declarado como obsoleto en HTML-4.01, pero es un elemento que existe en muchas páginas web, quizás antiguas, pero no por eso podemos ignorarlas. Así HTML-5 lo rescata de nuevo.

- **Degradado elegante.**

El término *degrade gracefully* se refiere en este caso a cómo deben afrontar nuevas características que no se adaptan a los navegadores y viceversa. Por ejemplo, un navegador que no soporte el elemento `<canvas>` *Elemento no soportado* `</canvas>` simplemente mostrará el contenido de texto interior. De la misma forma podemos usar CSS o Javascript para emular nuevas características en navegadores antiguos o actuales que aún no las soporten.

- **Reutilización**

Si ya existe algo que se usa extensamente y funciona no hay razones cambiarlo, este es un principio ampliamente utilizado en el desarrollo de aplicaciones. Un ejemplo es el atributo `contenteditable=""` y lo relacionado con la edición HTML de elementos en línea, algo que ya era ampliamente usado y que no formaba parte de las especificaciones. De esta forma la evolución de HTML debe recoger lo que ya existe en uso tal cual, sin tratar de "reinventar la rueda".

- **Construyendo sobre lo que ya existe.**

"Pave the Cowpaths" (pavimentar el camino de vacas) en el sentido de que cuando una práctica está extendida, es mejor adoptarla que prohibirla o sustituirla por algo nuevo: es mejor mejorar un viejo sendero de tierra que hacer una calle nueva al lado. El ejemplo de la barra para cerrar un elemento vacío en XHTML como `
` en lugar de `
` resulta muy ilustrativo. Si ya infinidad de personas cierran con una barra los elementos vacíos, entonces la evolución de HTML debería adoptarlo como otra forma de presentar elementos vacíos.

- **Evolución y no revolución.**

Finaliza esta parte con esta idea que yo considero importante. Las revoluciones cambian el mundo muchas veces para mejor. Pero no siempre lo hacen. A veces es mejor una evolución **incremental** que una revolución seca y cortante, tal como se intentó con XHTML. Ahora el concepto es ir cambiando las cosas sobre lo ya construido y *"no incendiar la ciudad para construir otra nueva"*.

Utilidad

Son principios para que el diseño de la especificación HTML consiga de forma eficaz los fines que persigue.

- **Resolver problemas reales.**

Cualquier modificación de la especificación debería resolver problemas del mundo real. Yo diría que lo que interesa de las posibles modificaciones es que realmente resuelvan los problemas, más que estén basadas en una bella teoría pero cuya aplicación práctica apenas sea demandada.

- **Prioridad de los usuarios en caso de conflicto.**

"Priority of constituencies". Pero *constituency* se debe interpretar como el conjunto de personas que les une una identidad y objetivo común. Así en el diseño de una especificación HTML encontramos a los usuarios de la web, los autores que escriben las páginas, los implementadores (navegadores), los que

redactan las especificaciones y por último los puramente teóricos. Y es precisamente en este orden como dice este principio que debería resolverse un conflicto en la evolución de HTML. Yo me quedo satisfecho al ver que los usuarios son los primeros de la lista, algo que no siempre ha sido así. Espero que lo cumplan.

- **La seguridad será tomada en cuenta**

Titulado como *Secure by design* algo así como "seguro por diseño", se refiere a que se deben contemplar los aspectos del modelo de seguridad de la web.

- **Separación de contenido y presentación**

Para el título *Separation of concerns* se debe tener en cuenta que *concern* en informática es un particular conjunto de comportamientos necesarios o requeridos de un programa o sistema. Y *Separation of concerns* se refiere al proceso de separar un programa en sus distintas características de tal forma que su solapamiento funcional sea el mínimo posible. Aquí está haciendo clara referencia a separar el contenido (HTML) de la presentación (CSS), algo que la especificación anterior HTML-4.01 intentó en algunos aspectos. Por ejemplo, el elemento `` era y es empleado como un simple resaltador de letra en negrita (de ahí la "b" de *bold* en inglés). Estaba destinado a ser suprimido pues eso podía hacerse con estilo CSS. Aunque se considera mejor darle otro cometido que prohibirlo, esto no quita la necesidad de promover el concepto de separar contenido de presentación. Por ejemplo, el nuevo elemento `<article>` se define para contener algo diferenciado como un artículo, pero no se dice nada de como debe ser presentado: en toda la página, a dos columnas, etc.

- **Consistencia del DOM**

Los navegadores leen el documento HTML y cargan los elementos en el árbol de nodos, usualmente denominado **DOM** (*Document Object Model*). Este principio se refiere a que sea cual sea la forma de construir ese DOM, todos deberían tener la misma respuesta cuando un script accede a ellos. En este aspecto

HTML-5 supone un cambio significativo especificando también las interfaces comunes del DOM.

Interoperabilidad

Estos principios mejoran la interoperabilidad de las implementaciones de HTML. La interoperabilidad es la capacidad de dos o más sistemas informáticos para trabajar juntos, intercambiando información. Para ello puede ser necesario que ambos entiendan sus respectivas interfaces y que no restrinjan accesos e implementaciones.

- **Funcionamiento bien definido.**

Se buscará definir con claridad el funcionamiento de cada característica de la especificación, con el objeto de que los autores de contenido sepan a que atenerse. Así ya no tendremos que preocuparnos de cómo se verá este elemento en tal o cual navegador. Y sí todos los navegadores hacen lo mismo ¿cuál será la diferencia entre ellos?. Pues ese principio acaba diciendo que aún los implementadores tendrán campo por delante para mejorar áreas tales como las interfaces de usuario y la calidad en el renderizado o presentación. Suena como a disculpa ante la tendencia que se seguía, donde un navegador iba por su cuenta creando sus propias características para así diferenciarse del resto y obtener mayor cuota de mercado. Es un principio muy importante, si los implementadores lo respetan.

- **Evitar la complejidad innecesaria.**

Son preferibles las soluciones sencillas que las complejas, siempre que sea posible. Las características sencillas son más fáciles de implementar, y de entender por los autores. Pero esto no debe ser excusa para intentar afrontar otros principios que pudieran requerir soluciones más complejas.

- **Control de errores**

El control de errores deberá ser definido para lograr implementaciones interoperables. Serán preferibles las recuperaciones elegantes ante un error y no los fallos que corten de forma brusca una ejecución, de tal forma que los

usuarios no queden expuestos gravemente a los errores de edición de los autores.

Aquí no pone ningún ejemplo, pero supongo que se refiere a definir cómo debe recuperarse el navegador, por ejemplo, ante un error del autor en el anidamiento de elementos. Se mantiene entonces la idea de que el navegador tratará de corregir estos errores de la mejor forma posible para que el usuario aún pueda ver la página. Pero ahora se definirá cómo se corrigen esos errores para que todos los navegadores se comporten de forma similar.

Acceso universal

Esta categoría cubre varios principios relacionados.

- **Independencia del medio.**

Las características deberán trabajar en diferentes plataformas, dispositivos y medios (visual, impreso, auditivo, etc). Una característica no debería ser omitida sólo porque no puede ser representada en un medio. Por ejemplo, un hipervínculo no puede ser accionado en un documento impreso, pero por eso no vamos a omitir el elemento `<a>` en la especificación para el medio impreso.

- **Soporte a los lenguajes mundiales**

Posibilitar la publicación en todos los lenguajes del mundo. El existente soporte de UNICODE que permite texto en la mayor parte de los lenguajes es un éxito ya implementado. Pero la especificación también tiene en cuenta algunos detalles de mejora para los lenguajes no occidentales.

- **Accesibilidad.**

Se han de diseñar características para mejorar la accesibilidad de personas con discapacidades. El texto alternativo para un elemento `` es un ejemplo ya en uso que facilita entender una imagen a las personas invidentes.

Web Semántica

La Web Semántica es una Web extendida, dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida. Al dotar a la Web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla. Esta Web extendida y basada en el significado, se apoya en lenguajes universales que resuelven los problemas ocasionados por una Web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante.

Uno de los ejemplos más conocidos de aplicación de Web Semántica es [FOAF](#).

- FOAF es un proyecto de Web Semántica, que permite crear páginas Web para describir personas, vínculos entre ellos, y cosas que hacen y crean. Se trata de un vocabulario RDF, que permite tener disponible información personal de forma sencilla y simplificada para que pueda ser procesada, compartida y reutilizada. Dentro de FOAF podemos destacar FOAF-a-Matic, que se trata de una aplicación Javascript que permite crear una descripción FOAF de uno mismo. Con esta descripción, los datos personales serán compartidos en la Web pasando a formar parte de un motor de búsqueda donde será posible descubrir información a cerca de una persona en concreto y de las comunidades de las que es miembro de una forma sencilla y rápida.
- Ejemplo de extracción de datos usando RDFa, GRDDL y SPARQL:
 - Se desea establecer una reunión entre tres personas, que tienen publicados en sus sitios Web los calendarios de sus citas y eventos. Estos datos están expuestos en páginas XHTML de forma gráfica, pero además se incluye información en RDFa.

- Una herramienta nos permite extraer, mediante GRDDL, los datos de sus calendarios en un formato homogéneo y fácil de tratar (RDF), para poder procesarlo posteriormente.
- Se realiza una consulta sobre la disponibilidad de las personas para un cierto día a una hora concreta. Los datos consultados están en formato RDF y la consulta se podría realizar mediante SPARQL.
- La herramienta procesa y analiza el resultado obtenido, concluyendo si las personas están disponibles en el instante que se había elegido previamente.

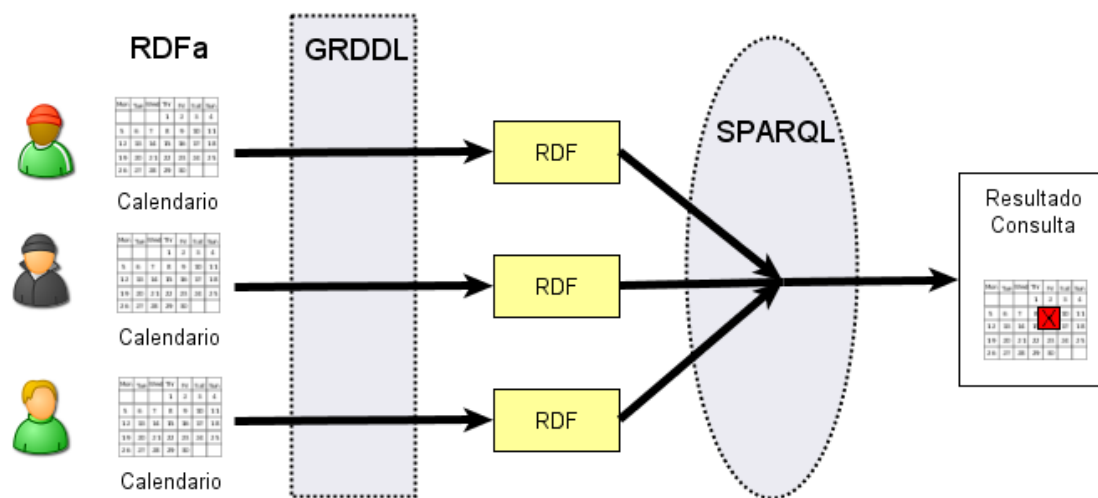


Figura 3 - Ilustración del ejemplo de consulta de eventos de calendario

Los buscadores semánticos son un ejemplo más de aplicaciones basadas en Web Semántica. El objetivo es satisfacer las expectativas de búsqueda de usuarios que requieren respuestas precisas. Otros ejemplos de aplicaciones basadas en Web Semántica pueden encontrarse en [SWAD](#)-Europe: Aplicaciones de Web Semántica - análisis y selección.

Hasta aquí hemos revisado conceptualmente lo que implica esta versión de HTML y a que apunta. Ahora veremos cuáles son los cambios y mejoras a nivel etiquetas y atributos que trae consigo HTML5.

Elementos semánticos en HTML5

La gran diferencia para el modelado de un documento HTML es el correcto uso de las etiquetas semánticas, HTML 5 introduce una serie de elementos estructurales que facilitarán tanto el desarrollo de las páginas como también el análisis de las mismas por buscadores.

Etiqueta <header>

La etiqueta <header> es una de las nuevas etiquetas semánticas de HTML 5 que permite una mejor estructura del documento HTML, que permite una mejor estructura del documento HTML, la misma se utiliza para crear en encabezados del archivo dentro del cuerpo del mismo, como podemos ver en el ejemplo siguiente:

Ejemplo

```
<body>
  <header>
    <h1>Bienvenidos a mi sitio web.</h1>
    <p>Formación Frontend ADA</p>
  </header>
</body>
```

Dentro de ésta etiqueta podemos incluir información relevante como ser, el título principal del documento.

La etiqueta <header> se puede usar las veces que sea necesaria, aunque muchos autores recomiendan usarla una sola vez por documento.

Soporte de navegadores

La etiqueta <header> es soportada por la mayor parte de los navegadores en sus versiones más actuales (IE desde la versión 9):

**Atributos soportados**

La etiqueta `<header>` soporta atributos globales y atributos de evento.

Etiqueta `<nav>`

La etiqueta `<nav>` se utiliza para crear en el área de la navegación principal que un documento conteniendo los links a las distintas secciones del sitio, se utiliza dentro del cuerpo el mismo.

Ejemplo

```
<body>
  <nav>
    <a href = "home.php" >Home</a>
    <a href = "clients.php">Clientes</a>
    <a href = "contact.php">Contacto</a>
  </nav>
</body>
```

Si bien según la especificación hasta el día de hoy de la W3C indica que se puede usar más de una vez la etiqueta `<nav>`, muchos autores recomiendan utilizar esta etiqueta una única vez por documento para la navegación principal del sitio.

Soporte de navegadores

La etiqueta `<nav>` es soportada por la mayor parte de los navegadores en sus versiones más actuales, a saber:



Nota: IE desde la versión 9.

Atributos soportados

La etiqueta `<nav>` soporta atributos globales y atributos de evento.

Etiqueta <footer>

La etiqueta <footer> se utiliza para crear el pie de página del documento, se utiliza dentro del cuerpo del documento y permite contener la información relacionada a los legales. La etiqueta <footer> se puede usar las veces que sea necesaria, aunque muchos autores recomiendan utilizarla una sola vez por documento. La etiqueta <footer> soporta atributos globales y atributos de evento.

Ejemplo

```
<body>  
  <footer>Este documento fue escrito en el 2011</footer>  
</body>
```

Soporte de navegadores:



Nota: IE (desde la versión 9)

Etiqueta <section>

La etiqueta <section> se utiliza para crear una sección dentro del documento HTML, en muchos casos la utilizamos donde antes sólo podíamos utilizar un div. Esta etiqueta soporta atributos globales y atributos de evento. La etiqueta <section> se puede usar las veces que sea necesario en cada documento.

Ejemplo

```
<body>  
  <section>  
    <h1>WWF</h1>  
    <p>La world wildlife Foundation nació en el año 1961...</p>  
  </section>  
</body>
```

Soporte de navegadores:



Nota: IE (desde la versión 9)

Etiqueta <article>

La etiqueta <article> es una de las nuevas etiquetas semánticas de HTML 5 que permite una mejor estructura del documento HTML, la misma se utiliza para crear artículos dentro de los documentos HTML. La etiqueta <article> soporta atributos globales y atributos de evento.

```
<body>
  <article>
    <a href= "http://industriait.com.ar/blog/3-pasos-para-volver-tu-perfil-de-
    linkedin-mas-atrayante-los-reclutadores-rrhh">
      lindas notas
    </a>
  </article>
</body>
```

Modo de uso de ésta etiqueta:

La etiqueta <article> se puede usar las veces que sea necesario en cada documento. Si bien su uso sigue siendo un tanto confuso, habitualmente se emplea para incorporar contenidos independientes como:

- post de foros
- artículos de diarios
- entradas de blogs
- comentarios de usuarios
- rss

También se utiliza para introducir contenido estático, por ejemplo, la típica información de quienes somos.

Soporte de navegadores:

Nota: IE (desde la versión 9)

Etiqueta <aside>

La etiqueta <aside> es una de las nuevas etiquetas semánticas de HTML 5 que permite una mejor estructura del documento HTML, la misma se utiliza para agregar contenidos complementarios al contenido principal del documento, son contenidos relacionados pero de menos importancia, que podrían en todo caso ser omitidos, la misma se utiliza dentro del cuerpo del documento. La etiqueta <aside> soporta atributos globales y atributos de documento.

Ejemplo

```
<body>
<p>Me vou de vacaciones a Miami este verano</p>
<aside>
<h4>Miami</h4>
Es una ciudad en el sur de EEUU
</aside>
</body>
```

La etiqueta <aside> es comúnmente usada en la barra lateral de la derecha y dentro de una etiqueta <article>

Modo de uso de ésta etiqueta:

La etiqueta <aside> se puede usar las veces que sea necesario en cada documento HTML.

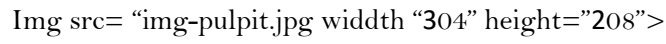
Esta etiqueta posee contenido que no es esencial para el sitio, la misma es ignorada cuando se ingresa mediante un dispositivo móvil o un lector de pantalla.

Soporte de navegadores:

Nota: IE (desde la versión 9)

Etiqueta <figure>

La etiqueta <figure> es una de las nuevas etiquetas semánticas de HTML 5. que permite una mejor estructura del documento HTML, la misma se utiliza para crear una figura, es decir, agrupa una imagen con un contenido de texto relevante, se utiliza dentro del cuerpo del documento, como se puede apreciar en el siguiente ejemplo:

```
<body>
<figure>
<p>vista desde la cúpula de la catedral <p>

</figure>
</body>
```

Modo de uso de ésta etiqueta:

La etiqueta <figure> se puede usar las veces que sea necesario en cada documento HTML.

El contenido de la etiqueta <figure> debe ser relevante para el contenido principal de la página, pero sin el contenido debe poder ser comprendido independientemente, es decir **la** si la etiqueta <figure> es omitida no debe afectar a la comprensión del contenido del sitio.

Soporte de navegadores:

La etiqueta <figure> es soportada por la mayor parte de los navegadores en sus versiones más actuales, a saber:



IE (desde la versión 9)

Atributos que puede utilizar:

Soporta atributos globales y atributos de evento.

Atributos Globales de HTML 5

Como ya hemos visto, los atributos de HTML dan sentido y contexto a los elementos.

A continuación, se listan atributos globales de HTML 5

ATRIBUTO	VALOR	DESCRIPCION
accesskey	caracter	Asigna una tecla como acceso directo
	GOOGLE	
class	nombre de clase	Asigna la clase que usará ese elemento
	<p class="important">Este es un párrafo importante</p>	
contenteditable	true false	Especifica si el usuario permite o no que el contenido sea editable
	<p contenteditable="true">Este es un párrafo editable</p>	
contextmenu	menú_id	Especifica el menú contextual de un elemento
	<p contextmenu="supermenu">Párrafo con menú contextual "supermenu"</p> <menu id="supermenu"> <command label="Step 1: Write Tutorial" onclick="doSomething()"> <command label="Step 2: Edit Tutorial" onclick="doSomethingElse()"> </menu>	
dir	ltr	Especifica la dirección del texto de un elemento

	rtl	
	<element dir="ltr rtl auto">	
draggable	true	Especifica si es posible mover el elemento
	false	
	auto	
	<p draggable="true" ondragstart="drag(event)">Párrafo arrastrable</p>	
dropzone	copy	Especifica que pasa si el elemento es movido y soltado en otro elemento
	move	
	link	
	<element dropzone="copy move link">	
Hidden	hidden	Especifica si el elemento está visible o no
	<p hidden="hidden">Párrafo oculto</p>	
id	id	Especifica el nombre del elemento
	<element id="id">	
lang	language_code	Especifica el lenguaje del elemento
	<element lang="language_code">	
spellcheck	true	Especifica si el elemento necesita ser

	false	comprobado gramaticalmente
	<p contenteditable="true" spellcheck="true">Párrafo para editar con revisión</p>	
tabindex	number	Especifica el orden de tabulación del elemento
	<element tabindex="number">	
title	tex	Especifica el título del elemento
	<p title="mensaje">Este texto tiene un tooltip</p>	
style	style_definition	Permite agregar un estilo en línea
	<p style="color: green">Párrafo verde</p>	