Introducción

En este apartado aprenderemos los conceptos básicos de HTML5. Comenzaremos brindando un instructivo para usarlo desde STS. Luego se abordarán los conceptos básicos de HTML5, como la estructura válida de un documento HTML, qué elementos se pueden incluir dentro de otros elementos y cuáles no- Se discutirán el significado y la utilidad de las etiquetas semánticas HTML5, y se trabajarán ejemplos con etiquetas HTML5 esenciales.

Configurando el IDE STS para desarrollar páginas HTML5

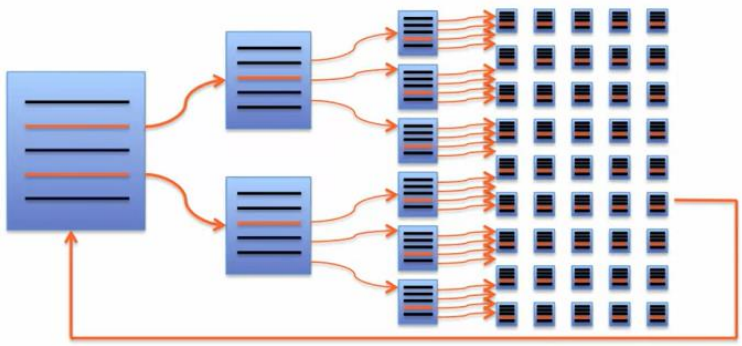
En esta sección veremos cómo configurar el IDE. ¿Qué herramientas vas a necesitar para realizar los ejemplos propuestos en este apunte?:

1) Google Chrome será browser oficial, por lo que se recomienda descargarlo e instalarlo. El navegador Google Chrome ya viene con lo que se llama CDT (Chrome Developer Tools) que se utilizará en muchas ocasiones.
2) Editor de código: Se utilizará el que ofrece STS para mantenernos familiarizados con el IDE. Por lo tanto, no se recomienda que utilicen otros editores tales como Sublime Text o Visual Studio Code (ambos editores muy buenos).

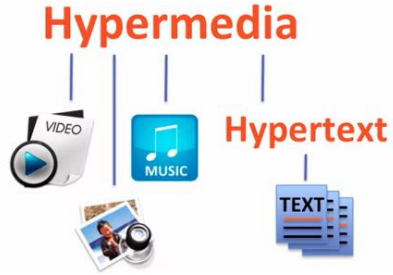
Que es HTML

Entonces, ¿qué es HTML? ¿Qué significa el término HTML y cómo se relaciona con la web? ¿Qué papel juega HTML en el desarrollo web? A medida que avancemos, obtendremos las respuestas a estas simples preguntas. Esto ayudará a tomar decisiones correctas de codificación en el futuro.

En primer lugar, ¿qué significa HTML? Es el acrónimo de **Hyper Text Markup Language**. Entonces ahora la primera pregunta sería ¿Qué es un hipertexto? Originalmente el término hace referencia al texto de un documento que contiene enlaces a otros textos dentro o fuera del documento. Un documento puede apuntar a otro documento que, a su vez, apunta a un montón de otros documentos y a veces estos últimos se vinculan de nuevo al documento original:



Obviamente, en la actualidad estos documentos no solo contienen textos. Gracias a la hipermedia estos documentos pueden contener videos, archivos de sonido, etc. La hipermedia es tan solo una extensión del hipertexto:



El siguiente término que vamos a analizar es **Markup** (marcado, en español). Hace referencia a la forma en la cual se identifica un contenido en particular dentro del documento de hyper texto. Esta identificación se realiza mediante unas marcas denominadas tags (etiquetas en español).

Observe el siguiente ejemplo del título de una página web, un documento HTML:

Si abrimos el documento HTML que contiene ese título mediante un editor de texto obtendríamos algo similar a lo siguiente:

```
<!doctype html>
<html>
<head>
  <title>Why I Love This Course</title>
</head>
<body> [...]
</body>
</html>
```

Como puedes ver aquí, el texto *Why I Love This Course* se encuentra en el interior de "una marca" denominada tag (etiqueta). Ese tag se denomina `<title>` por lo cual podemos deducir que el contenido del texto corresponde al título del documento. Así como nosotros podemos interpretar que ese contenido hace referencia, los programas como el Browser utilizan los tags para interpretar el contenido del documento y generar la salida en la pantalla del Browser. Esta característica de los `tags` de HTML hace posible interpretar un documento HTML, tanto por humanos como por diversos programas. Por tanto, los `tags` parecen instrucciones para estructurar el contenido del documento.

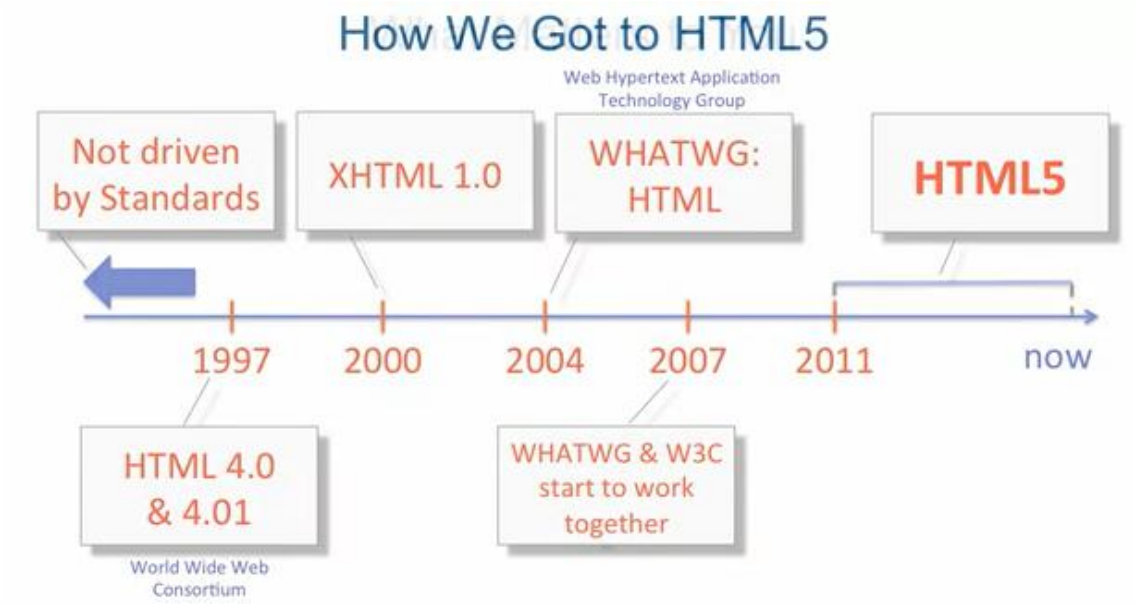
La última palabra del acrónimo HTML es **language** (lenguaje). Se hace referencia al lenguaje de utilizado para definir la sintaxis del uso de los `tags`. Esto significa que hay una forma correcta e incorrecta de codificarlos. Por ejemplo, observe el siguiente ejemplo:

Se han descriptos dos formas para etiquetar el mismo contenido. En la forma indicada a la izquierda, el cierre de tag (indicado con) aparece después del cierre de la etiqueta <h1> (indicado con </h1). Esta forma provoca potenciales errores de representación, debido a que el lenguaje HTML establece un orden para el cierre de las etiquetas, el cual se refleja en la forma de la derecha. Este lenguaje también tiene su propia semántica, lo cual significa que los nombres de etiquetas significan algo para las máquinas o para los humanos.

Con HTML se estructura el contenido de las páginas web sin entrar en los detalles de cómo se visualizan los componentes del documento. Un ejemplo análogo sería el siguiente: suponga que

tiene una casa con tres habitaciones y una cocina. Esta es la estructura de la casa. Con esta información no podemos saber de qué color es la cocina o las paredes. Lo único que se sabe es qué componentes componen la casa.

Historia de HTML



Hablemos brevemente sobre la historia de HTML. Como en la mayoría de las tecnologías, no es particularmente interesante o emocionante. Pero hay ciertas partes de esta historia que no solo brindan una comprensión de cómo se desarrolló esta tecnología, sino que también brindan una apreciación de ciertos aspectos de HTML que aún son relevantes y aplicables en la actualidad.

Antes de 1997, no había estándares, por lo que los browsers básicamente hicieron lo que quisieron. Por ejemplo, cada browser poseía sus propios `tags`, incluso implementaban las mismas etiquetas de manera diferente. Era como el salvaje oeste de la web. Como consecuencia era común entrar en un sitio web y que aparezca un mensaje indicando que el browser no era compatible con este sitio web, por lo que debía utilizar un browser diferente para poder ver el sitio web.

Alrededor de 1997, el World Wide Web Consortium, conocido como W3C, creó los primeros browsers estándar por leer HTML4 y lo actualizaron muy rápidamente a HTML4.01. Ese estándar era bastante flojo, y, el browser tenía, todavía tenía demasiado margen de libertades dentro de ese estándar para representar las páginas.

Entonces, alrededor del año 2000, el W3C presentó otra especificación denominada XHTML 1.0. Esta especificación se basó en XML. XML es un lenguaje de marcado muy rígido pero muy claro. El W3C quería recoger estas características y aplicarlas en las páginas HTML. Se suponía que la estabilización del estándar llegaría con XHTML 2.0 pero surgió un problema. Los proveedores de browsers (que en este punto ya tenían un historial de no aplicar realmente un 100% de ningún estándar), decidieron la nueva propuesta de W3C se movía demasiado lento y en la dirección equivocada.

Entonces, los proveedores de browsers se unieron y crearon otro grupo con el objetivo de generar nuevas especificaciones. Este grupo se denominó WHATWG (abreviatura de Web

Hypertext Application Technology Group). Este grupo fue mucho menos democrático que el W3C. Designaron un editor central, que toma las decisiones finales considerando o no, el resultado de las discusiones del grupo. Este grupo fue el impulsor de HTML 5. Entonces, durante mucho tiempo, las dos organizaciones realmente trabajaron separadas, por lo tanto, convivían 3 estándares diferentes (HTML 4.1, XHTML 1.0 y HTML 5). En al año 2007 la W3C decidió cooperar con WHATWG, después de todo es impulsada por los desarrolladores de los browsers. El fruto de esta cooperación es HTML5. Al estar a cargo de dos grupos, ciertamente sería posible que un browser cumpla con una especificación que aún no está formalizada por el W3C oficial, con todo lo que esto puede generar.

Entonces, la división de responsabilidades que establecieron fue la siguiente:

- W3C estará a cargo del estándar HTML5.
- WHATWG ni siquiera va a dar una versión a su estándar HTML, solo van a decir que es HTML y que está en constante evolución.

Los browsers implementan el estándar de WHATWG. Por tanto, la actividad de W3C consiste en elegir algunas de las cosas más exitosas que los browsers implementan y las incorporan lentamente al estándar oficial.

Lamentablemente esta situación hace que los desarrolladores deban realizar un seguimiento minucioso de las capacidades disponibles del browser del cliente. Y es particularmente importante hoy en día porque todos los principales browsers modernos se actualizan silenciosamente y de forma automática, con el objetivo de incorporar mejores características al HTML y mejorar la seguridad.

El desarrollador se ve en la necesidad de realizar el seguimiento de los estándares para cada browser. El sitio www.escaniuse.com realiza un seguimiento de los estándares HTML5, los estándares SVG, los estándares CSS, las API de JavaScript en los browsers. Entonces, por ejemplo, si busca Srcset, que es un atributo bastante nuevo, el sitio le indicará cual es objetivo del atributo y señalará qué browsers son compatibles e incompatibles en este momento.

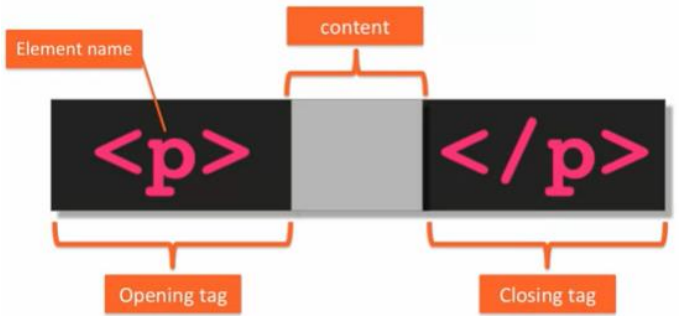
Otro gran recurso para averiguar si su HTML realmente funcionará o no en los browsers es usar un validador y hay un gran validador proporcionado por el sitio web W3.org. Y básicamente puede cortar y pegar todo su sitio web, o puede cargar un archivo HTML en particular. El validador verificará si es válido. Si es válido, es muy probable que funcione muy bien en el browser.

Anatomía de un tag HTML

El núcleo de HTML es el `tag` HTML. Por lo tanto, es bastante importante comprender en qué consiste un `tag` HTML y cómo codificarlo sintácticamente de manera adecuada.

Por lo general, los `tags` HTML presentan una apertura de etiqueta y un cierre de etiqueta que se utilizan para envolver algún contenido.

Observe el siguiente ejemplo:



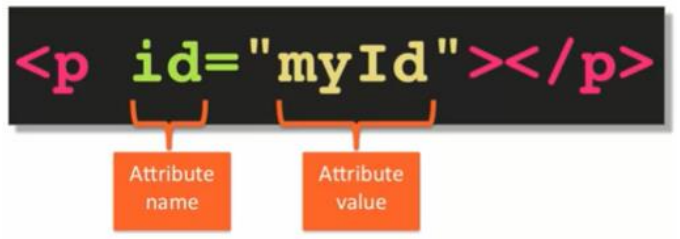
En este caso, el `tag <p>`, indica que el contenido es un párrafo, el cual está delimitado a la zona gris debido a la apertura y cierre de etiqueta.

Un `tag` HTML está compuesto por dos elementos:

- El elemento: describe el tipo de etiqueta, es decir el tipo de contenido de la etiqueta.
- Corchetes angulares: que rodean al elemento.

Ahora bien, la mayoría de los `tag` HTML presentan un cierre de etiqueta de cierre para su correspondiente apertura de etiqueta, pero existen algunos que no las requieren. Por ejemplo, los `tags <br> y <hr>` que significan salto de línea y regla horizontal respectivamente, solo tienen una apertura de etiqueta de apertura y poseen un cierre de etiqueta.

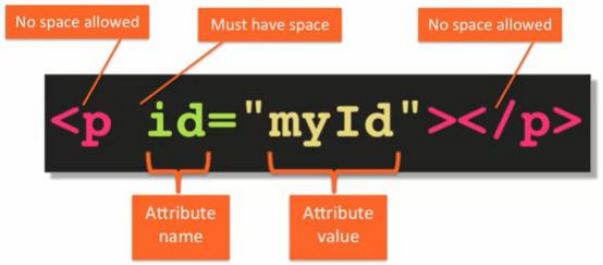
Cada elemento HTML puede poseer atributos predefinidos. Un atributo está formado por el par (nombre = valor). El atributo es entonces, una especie de metadato que actúa sobre el elemento. Entonces, suponga el siguiente ejemplo:



Al `tag <p>` se le ha definido el atributo *`id`* con el valor "**`myId`**".

Cada atributo tiene sus propias reglas para delimitar los valores que pueden adoptar. Por ejemplo, el atributo *`id`*, asignado en el ejemplo, tiene que ser único dentro del alcance de todo el documento HTML. En otras palabras, ningún otro elemento de ningún tipo en la página web puede tener su atributo *`id`* igual a la cadena "**`myId`**". Si hay otro elemento con el mismo valor para id, eso significaría que la página web contiene HTML no válido que potencialmente puede romper alguna jerga de estilo, incluso la funcionalidad de la página.

Ahora vamos a indicar algunas de las reglas básicas sobre el espaciado en el mismo ejemplo:



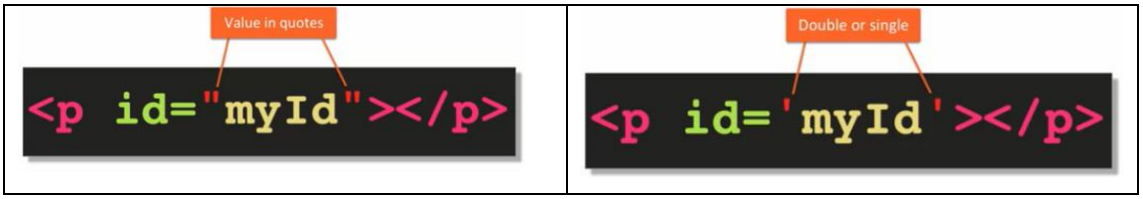
No se permite que exista espacio entre el corchete angular de inicio de la apertura de etiqueta y el nombre del elemento.

Del mismo modo, no se permite espacio entre el corchete angular de inicio del cierre de la etiqueta y la barra diagonal del cierre de etiqueta.

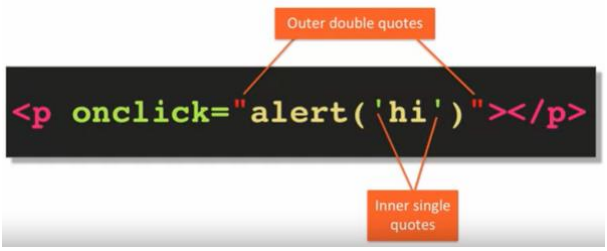
Sin embargo, debe tener al menos un espacio entre el propio elemento y cualquiera de sus atributos, y el espacio está permitido en todas partes, pero simplemente se ignora. Entonces, si tiene espacio adicional después del elemento p en la apertura de etiqueta, o si tiene espacio adicional entre el signo igual del nombre del atributo y el valor del atributo; o si tiene otros espacios, incluso otros caracteres, todo eso es completamente ignorado por el browser.

Una regla más, los atributos solo se pueden especificar en la apertura de etiqueta, por lo que no puede especificar un atributo en el cierre de etiqueta.

Respecto del valor de los atributos, en HTML5, técnicamente es válido encerrar el valor del atributo entre comillas, pero en realidad no es obligatorio en todas las circunstancias. Sin embargo, es una buena práctica encerrar siempre el valor del atributo entre comillas simples o dobles. No importa si usa comillas simples o dobles. Realmente son equivalentes en HTML.



Un caso más interesante surge cuando el valor del atributo en sí contiene comillas, es decir, el valor real tiene comillas como parte de su valor. En este tipo de situación, lo único que debe tener en cuenta es asegurarse de cerrar la cotización en el orden opuesto al de abrirlas, como se puede observar en el siguiente ejemplo:



El valor del atributo puede comenzar con comillas dobles, como lo hemos hecho en este ejemplo, o podría comenzar con comillas simples. También puede anidarlos tantas veces como desee siempre que los cierre, obviamente en el orden correcto, pero en la práctica es muy raro que deba tener más de dos niveles de comillas. De hecho, si los utilizara de esa manera, el código se tornaría desordenado y difícil de leer.

Si ha trabajado con versiones anteriores de HTML, especialmente XHTML, puede estar familiarizado con la idea de cierre automático de una etiqueta. Un cierre automático de etiqueta es básicamente un tipo XML de notación abreviada para una etiqueta que no contiene ningún contenido, Por ejemplo, para el `tag` `<p>` se vería como lo indica la parte izquierda de la siguiente imagen:

Estos tipos de tags generalmente se utilizan como marcadores de posición dentro del documento HTML para indicar que en el algún momento se insertará de forma dinámica un contenido en ese lugar.

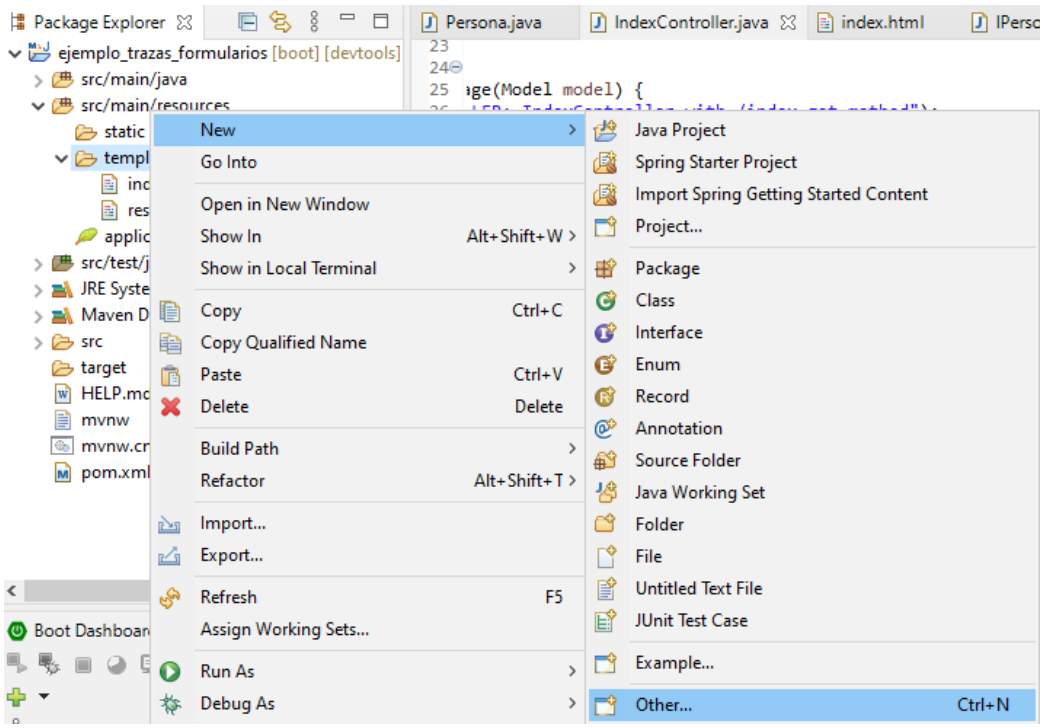
Sin embargo, en HTML5, las etiquetas son más semánticas. Entonces, si la etiqueta HTML5 es capaz de considerar que una etiqueta debe poseer algún contenido, entonces será ilegal que esa etiqueta posea un cierre automático, incluso si no posee contenido en ese momento.

Por lo tanto, debe proporcionar las aperturas y cierres de etiquetas sin ningún espacio entre ellas para indicar un marcador de posición al cual le falta el contenido que será obtenido generalmente de forma dinámica.

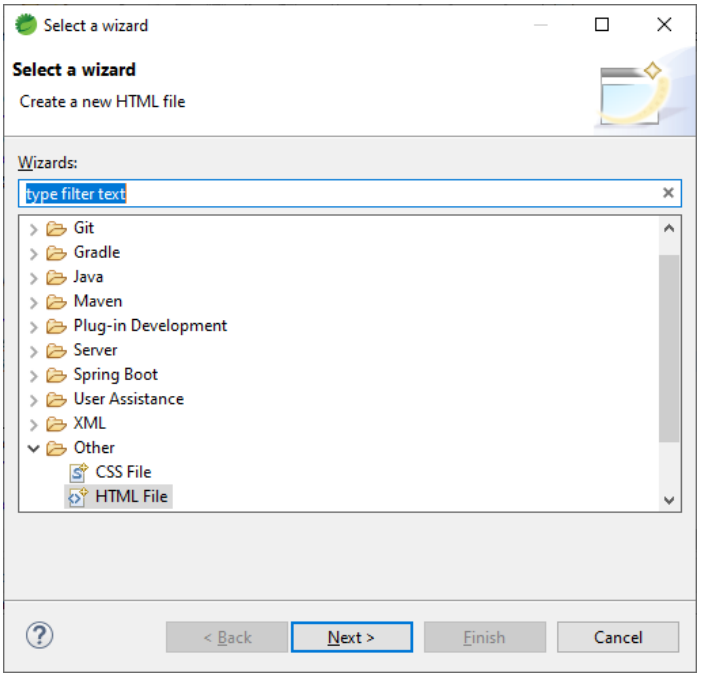
Crear un documento HTML en STS

Para crear un documento se procede de la siguiente manera (Suponemos que estás dentro de un proyecto Spring Boot creado con Maven y las dependencias Spring web, Spring dev tools y thymeleaf como mínimo. Además damos por hecho que para visualizarlos usamos controllers):

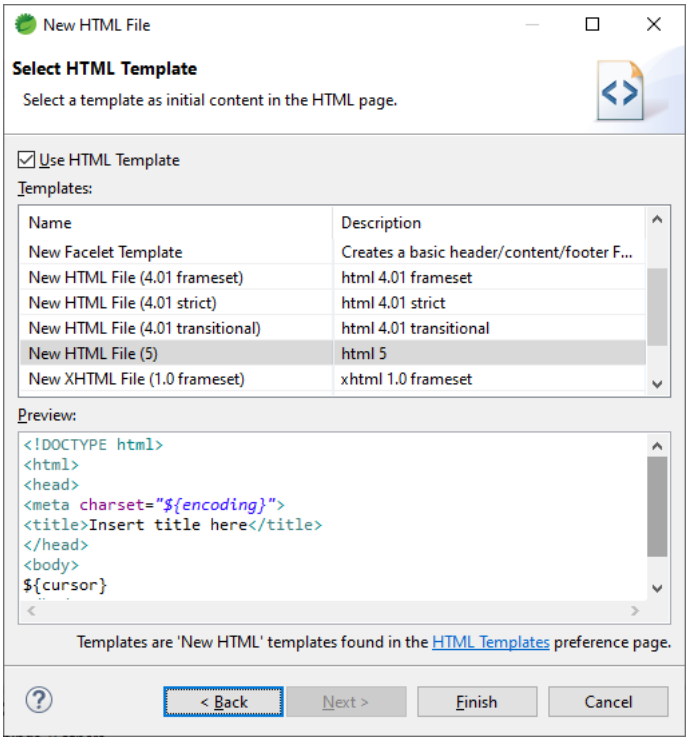
1) Posicionar el puntero sobre la carpeta src/main/resources/templates y hacer botón derecho del mouse y elegir *New →HTML File*



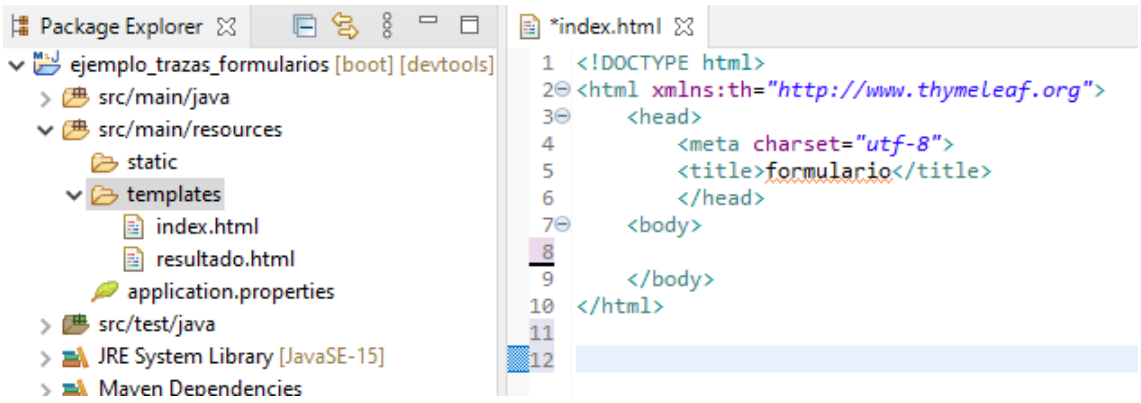
2) Aparecerá una ventana donde se pedirá la ubicación y nombre del documento. Para este ejemplo coloque el nombre index.html (es el nombre por defecto de la página inicial de la aplicación/sitio web) y luego presione el botón *Next>*



3) Se verá otra ventana en la cual se elige el tipo de documento HTML. En este caso seleccionaremos New HTML File (5) y presionaremos el botón *Finish*

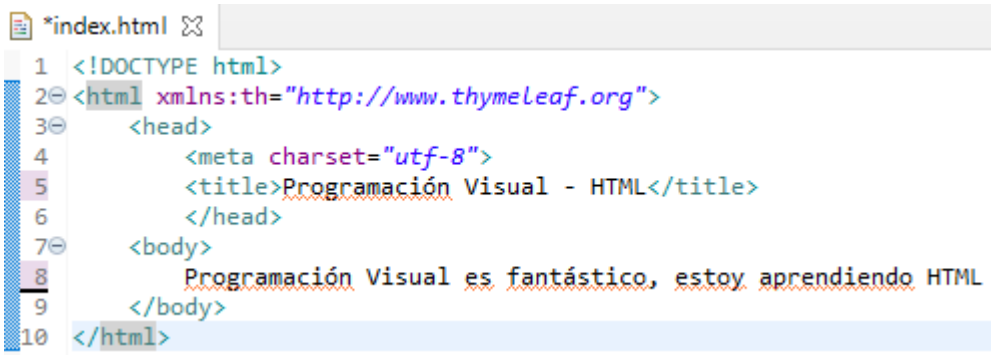


Ahora se puede observar el documento en la estructura del proyecto y su contenido en el editor de código del IDE.



Estructura Básica de un documento HTML

En esta sección editaremos la página creada y luego lo validaremos en el sitio web de validación de w3c. Bien, entonces editaremos en el editor de STS la página `index.html` de la siguiente manera



1) Cada página HTML debe comenzar declarando el tipo de documento. Los elementos de esta etiqueta pueden escribirse en minúsculas o mayúsculas, pero no debe haber espacio entre el signo de exclamación y la palabra `DOCTYPE`. En el pasado, esta declaración tenía un aspecto bastante complicado, ciertamente no mucha gente podría escribirlas sin copiar y pegar. En HTML 5, sin embargo, cambió todo eso. Ahora la declaración es tan simple como puede se puede observar. Esta declaración le indica al browser que debe prepararse para renderizar HTML. El término renderizar significa dibujar las etiquetas. ¿Puede una página HTML poseer contenido que no sea HTML? La respuesta es NO. Entonces ¿para qué usar una etiqueta que indique lo obvio? No parece haber ningún propósito práctico para esta declaración. Esta declaración es realmente en gran parte histórica. Cuando los estándares HTML se volvieron populares, la web estaba llena de páginas que no cumplían con los estándares. Para ayudar a los browsers a representar las páginas correctamente usaban esta etiqueta para distinguir entre páginas que cumplen y no cumplen los estándares. Las páginas no conformes al estándar se representan usando el denominado "peculiaridades", mientras que las páginas conformes al estándar se representaron en el modo estándar. Entonces en la actualidad, esta declaración le indicará al navegador que debe tratar sus páginas como una que sigue el estándar HTML.
2) A continuación, se declara el `tag <html>`, la cual básicamente, contiene todo el documento HTML. Observe que esta etiqueta inicia en la línea 2 y finaliza en la línea 10. Adicionalmente para nuestros tipos de proyectos agregamos el namespace de thymeleaf. Esto permite que dentro de nuestro proyecto podamos usar etiquetas que no pertenecen a HMTL sino que corresponden a la tecnología de Thymeleaf. Algo

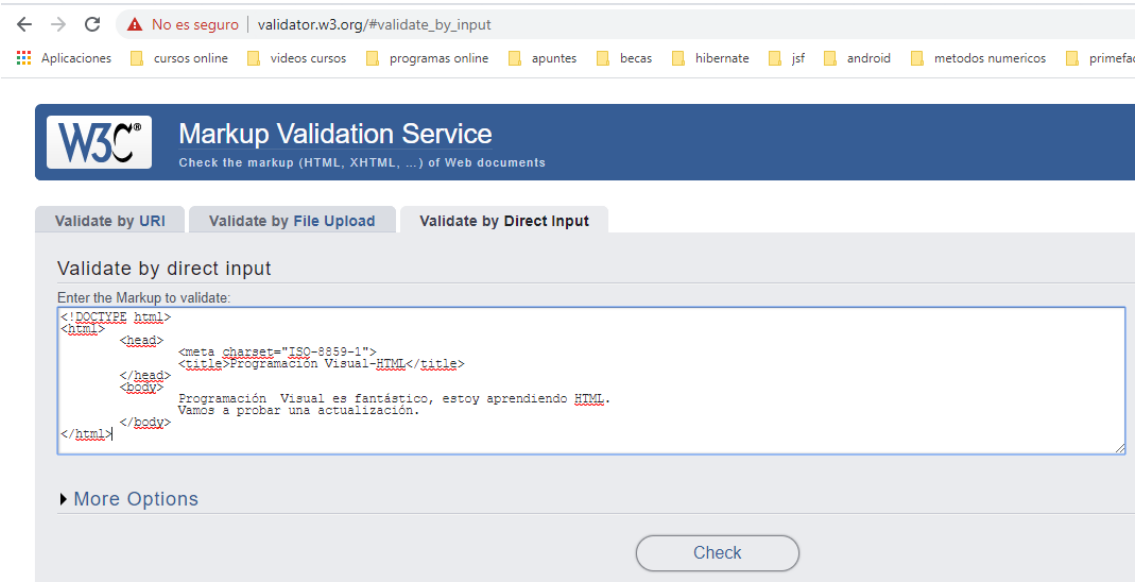
interesante es que esto nos sirve en tiempo de diseño, es decir cuando creamos la página, ya que al momento de renderizar si observamos el contenido de la misma, este namespace no aparecerá, y las etiquetas de thymeleaf se verán como etiquetas HTML. En definitiva, el browser solo interpreta etiquetas HTML.

3) Después del `tag <html>`, se declara el `tag <head>`. Contiene elementos que describen el contenido principal de la página, tales como la codificación de caracteres que debe usar el navegador para el contenido principal (línea 4). También puede contener la descripción de los autores de la página, el título de la página (línea 5) y cualquier otro recurso externo que se necesite para representar la página correctamente, entre otras cosas. El punto central para considerar es que esta etiqueta contiene algunos metadatos sobre el contenido principal.
   Observe que el `tag <meta>` no posee cierre de etiqueta. Este tipo de etiquetas se denominan etiquetas independientes. Su información es utilizada por los browsers.
   La etiqueta para indicar el título de la página es obligatoria. Sin él, el HTML será inválido.
4) Después del `tag <head>` va el `tag <body>`. Dentro de esta etiqueta se vierte todo el contenido que es visible para el usuario. Observe que inicia en la línea 7 y finaliza en la línea 9. Para nuestro ejemplo solo hemos colocado una línea de contenido.

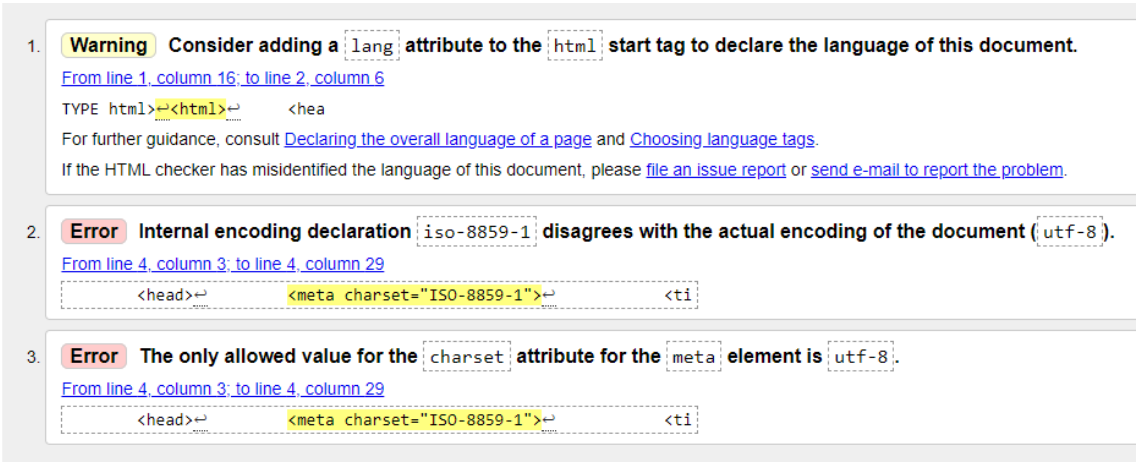
Validar una página web

Debido a que los browsers actualizan su implementación de HTML de forma independiente al estándar W3C, resulta conveniente validar el contenido de las páginas web para asegurar que el mismo se ejecutará en gran medida de la misma forma en los browsers más utilizados. Una buena opción para realizar la validación es utilizar el validador de W3C.

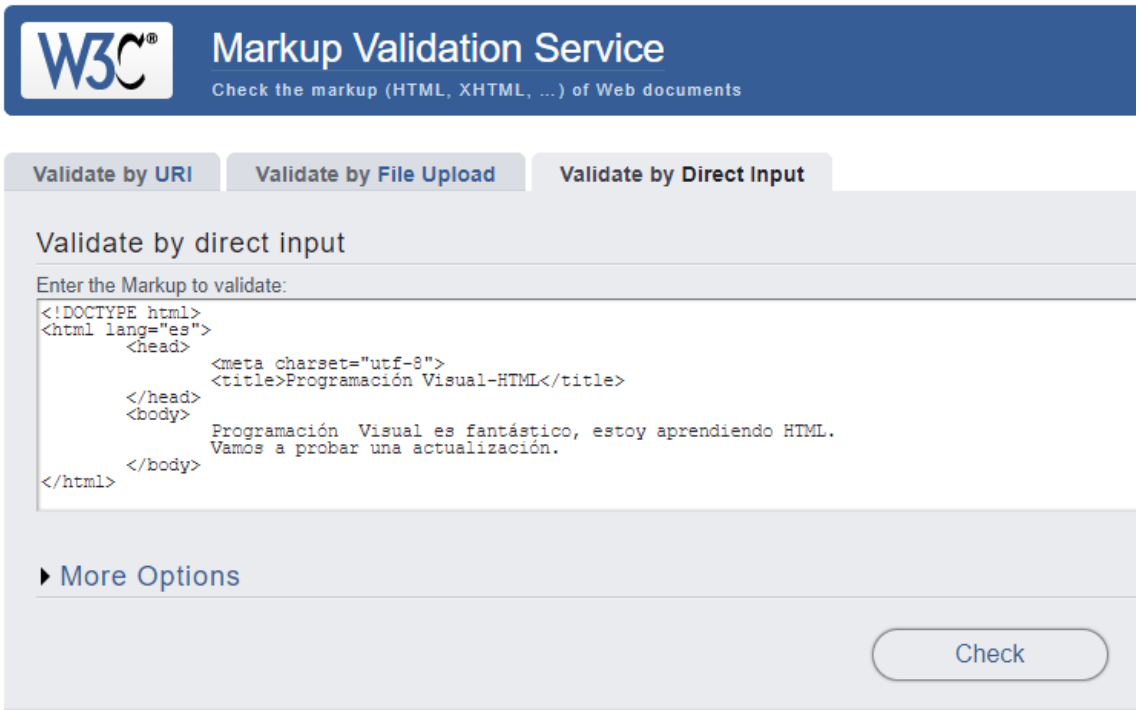
Simplemente se copia todo el contenido de la página web a evaluar, se accede a la pagina web del validador y luego se pega el contenido a evaluar. Luego se debe presionar el botón *CHECK*



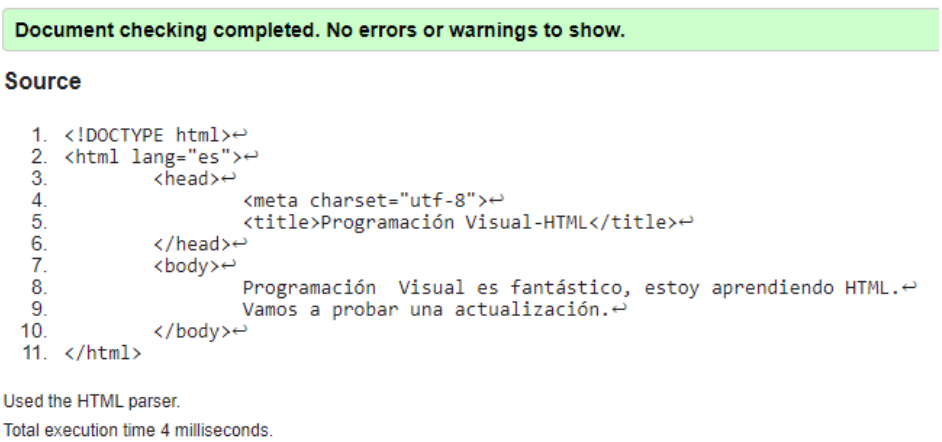
Observe que para un ejemplo tan simple, el validador devuelve mensajes de advertencia (warnings) y de error:



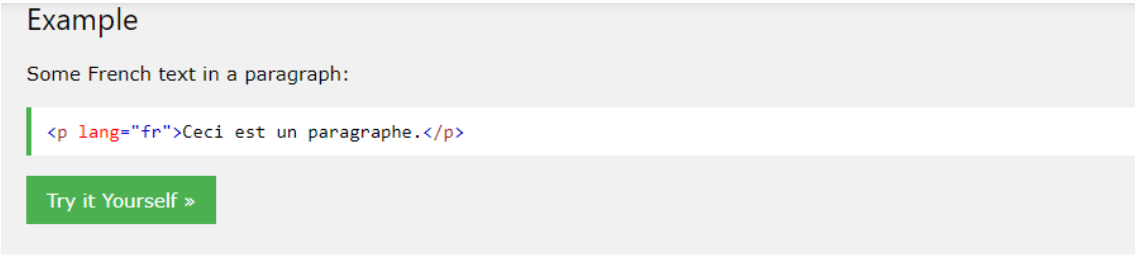
Por lo cual es pertinente realizar los cambios sugeridos:

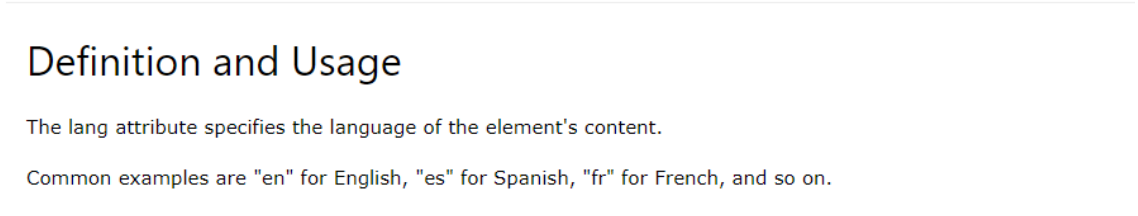


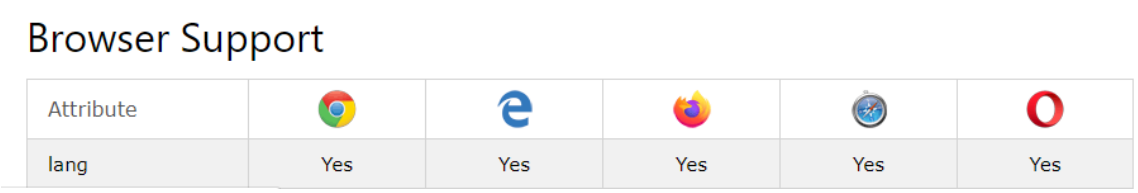
Lo cual generará la siguiente salida:



El warning que indica la página hacía referencia al atributo lang. Una breve búsqueda de información sobre este atributo en la página del W3C arroja el siguiente resultado:







Observe que es un atributo global, es decir puede ser utilizado en todas las etiquetas. Especifica el lenguaje del contenido del documento y el estándar recomienda definirlo. En este caso se realizó el mismo en el `tag <html>` para que alcance a todo el contenido del documento. Observe que adicionalmente se brinda información de los browsers que soportan este atributo.

Respecto del mensaje de error sobre el conjunto de caracteres que por defecto utiliza las páginas que genera STS, resulta que "ISO-5928-1" no es el juego de caracteres aceptado por el estándar. El juego de caracteres "utf-8" es actualmente el estándar aceptado por los browsers.

Nota 1: Reglas adicionales para el uso de etiquetas

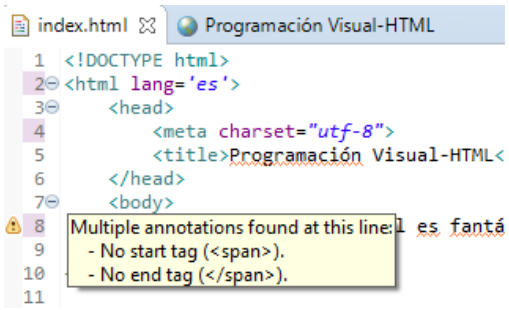
Una regla importante al anidar `tags` html es que se cierran en el orden inverso en que se abren. Si no lo hace, el documento HTML no es válido.

Suponga el siguiente ejemplo:

index.html | Programación Visual-HTML

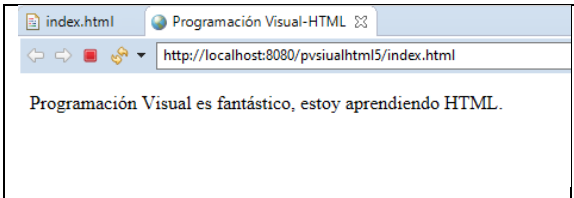
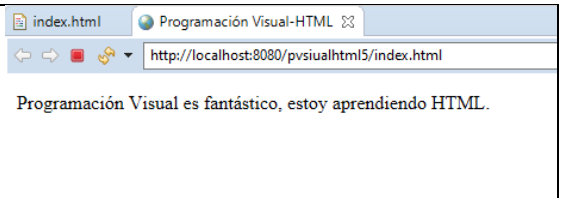
```
1  <!DOCTYPE html>
2  <html lang='es'>
3      <head>
4          <meta charset="utf-8">
5          <title>Programación Visual-HTML</title>
6      </head>
7      <body>
8          <p>Programación  Visual es fantástico, <span>estoy aprendiendo HTML.</p></span>
9      </body>
10 </html>
```

El IDE está indicando con mensajes de advertencia que algo parece no ser válido. De hecho, si usamos el IDE, este generalmente coloca de forma automática los cierres de etiqueta, minimizando la posibilidad de escribir el documento de forma inválida. Si además posamos el puntero del mouse sobre el signo de advertencia, obtendremos un mensaje indicando los inconvenientes que detecta el IDE:



Este mensaje aparece porque el IDE detecta un cierre de etiqueta que no coincide con el orden en el que debería ejecutarse.

Si ejecutamos la página el browser no saltará estos errores generando una salida, por lo cual no se puede confiar siempre en la vista de resultados:

	
Ejecución con errores	Ejecución sin errores

Sin embargo, si copiamos el documento a la página del validador obtendremos un detalle de lo que está sucediendo:

Si considera que entiende esta regla indique cual documento html es inválido

```
<div>Hey there!

I am just a lonely quiz answer. Will you click me? I am bored!

</div>
```

```
<section>
<p>The sale numbers are in...
</section> You ARE a closer, Johnson!
</p>
```

```
<article>
<h2>Wow!
</h2>
<p>You're ignoring all the other answers just to
look at me?! I AM special!
...
Wait! Where are you going?
What if I AM the answer you've been searching for???!
</p>
</article>
```

```
<div><span>Will you just tag along with me?</span></div>
```

Nota 2: sobre la renderización de páginas

Cuando el browser abre una página HTML siempre renderiza o interpreta el código HTML secuencialmente de arriba abajo empezando por la declaración `doctype` y así sucesivamente hasta que llega a el último cierre de etiqueta. Será importante recordar esto a medida que avancemos en el curso.

Modelos de contenido

El término modelo de contenido se refiere al comportamiento completo que el browser aplica sobre los elementos de ese modelo de contenido, y a las reglas de anidamiento de esos elementos, esto es, indica qué elementos pueden anidarse dentro de qué otros elementos. Antes de la especificación HTML5, los elementos HTML eran elementos de dos categorías: nivel de bloque o nivel de línea. HTML5 divide estos dos modelos de contenido en siete modelos.

- Los modelos tradicionales: todos los elementos se dividen básicamente en dos categorías. Elementos de nivel de bloque o elementos de nivel de línea. Los elementos de nivel de bloque se renderizan en una nueva línea y no comparten esa línea con ninguna otra etiqueta de forma predeterminada. Podría cambiar esto con CSS. Cada vez que especifique un elemento de nivel de bloque en HTML, el browser lo colocará automáticamente en una nueva línea siguiendo el flujo del documento. Los elementos de nivel de bloque pueden contener elementos en línea u otros elementos de nivel de bloque dentro de ellos. Los elementos en nivel de línea se renderizan en la misma línea de forma predeterminada. Una vez más, puede cambiar eso mediante CSS. Entonces si se colocan varios elementos de nivel en línea uno al lado del otro, todos se renderizarán en la misma línea, como si fueran un único elemento de línea. Los elementos de nivel de línea también tienen una restricción: solo pueden contener otros elementos de nivel de línea. En otras palabras, un elemento en línea no puede tener como parte de su contenido un elemento de nivel de bloque. Esta distinción entre elementos de nivel de bloque y elementos en línea es bastante práctica porque permite trabajar de forma integrada con las reglas CSS existentes
- Los nuevos modelos de contenido: HTML5 realmente reemplaza las definiciones tradicionales por un conjunto más complejo de categorías de contenido. A pesar de los nuevos nombres de modelos de contenido y las nuevas subcategorías; al final aún se podría ver categorizar su codificación, como elementos de nivel de bloque y elementos en línea. En HTML5 se agregan el contenido de flujo próximo (roughly Flow content), el cual se puede considerar un tipo de categoría de bloque; y l contenido de fraseo próximo (roughly phrasing content) que puede considerarse un tipo de categoría de nivel de línea.

Así que veamos un ejemplo que demuestre estos conceptos. Se define un documento HTML que contiene `tags <div>` y `<span>`.

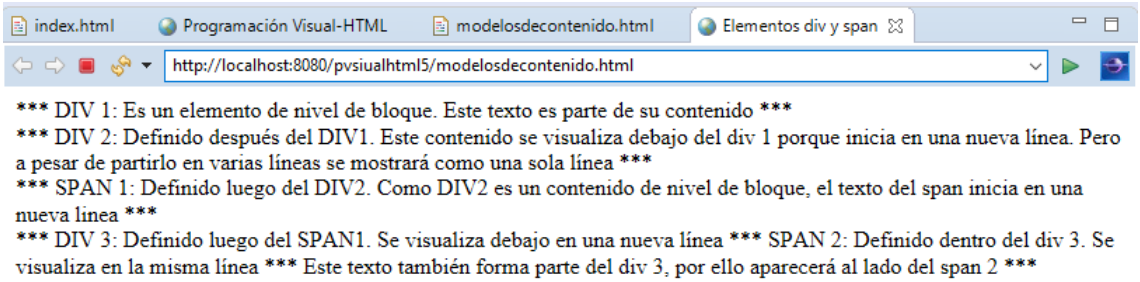
- `<div>`permite dividir en documento en diferentes secciones y es la etiqueta de nivel de bloque más genérico.
- `<span>` permite seleccionar una parte de texto o una parte del documento. Por sí mismo no genera ningún cambio visual sobre el texto o parte del documento seleccionado, pero mediante el uso de CSS o Javascript puede formatearse o manipularse respectivamente. Es la etiqueta de contenido de nivel de línea más genérica.

La definición del documento permite visualizar texto explicativo de la forma en que el browser renderizará las etiquetas en base al concepto de modelo de contenido de bloque y de línea:

index.html | Programación Visual-HTML | modelosdecontenido.html | Elementos div y span

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Elementos div y span</title>
</head>
<body>
  <div>*** DIV 1: Es un elemento de nivel de bloque. Este texto es parte de su contenido ***</div>
  <div>*** DIV 2: Definido después del DIV1.
          Este contenido se visualiza debajo del  div 1
          porque inicia en una nueva línea. Pero a pesar de partirlo en varias líneas
          se mostrará como una sola línea ***</div>
  <span>*** SPAN 1: Definido luego del DIV2. Como DIV2 es un contenido de nivel de bloque,
            el texto del span inicia en una nueva linea ***</span>
  <div>
    *** DIV 3: Definido luego del SPAN1. Se visualiza debajo en una nueva línea
    <span>*** SPAN 2: Definido dentro del div 3. Se visualiza en la misma línea ***</span>
    Este texto también forma parte del  div 3, por ello aparecerá al lado del span 2 ***
  </div>
</body>
</html>
```

Ahora observemos el resultado de la renderización por parte del browser:



Entonces puede ver que el elemento DIV 1 está solo en su propia línea. Y también lo está el elemento DIV 2. Ambos se renderizan en su propia línea nueva. Ahora el SPAN 1 sigue directamente después de DIV 2. Y aunque el span es un elemento en línea, ya que DIV 2 requiere estar en su propia línea, entonces se renderizará en una nueva línea. También empuja el siguiente elemento en línea a su propia línea. Y esto es exactamente lo que sucede con DIV 3. Luego, aunque `<span>` es un elemento en línea, y por lo tanto cualquier otra etiqueta debería renderizarse al lado de ella, DIV 3 se renderiza en una nueva línea porque es una etiqueta de nivel de bloque y como tal, requiere su propia línea.

SPAN2 se define dentro del DIV3. Esto es válido porque las etiquetas de nivel de bloque pueden contener etiquetas de nivel de línea. Y como tal, simplemente se renderiza al lado del texto que previamente se ha escrito. El último texto se renderiza en la misma línea por la misma razón.

Por lo tanto, los saltos de línea dentro del contenido de una etiqueta de nivel de línea son ignorados al momento de renderizar.

Nota: No es el objetivo de esta asignatura cubrir los detalles de los modelos de contenidos que permite gestionar HTML5. Si es de su interés puede acceder a:

https://html.spec.whatwg.org/multipage/dom.html#kinds-of-content (anteriormente en:

www.w3.org/TR/html5/dom.html#kinds-of-content)

donde se enumeran los siete tipos de contenido que HTML5 define y brinda algunos detalles de los elementos sobre los que se aplican.

¿Entendió los conceptos de este apartado? Entonces anímese a responder las siguientes preguntas:

Pregunta1: ¿Cuantas líneas de texto renderizará un browser respecto del siguiente código HTML?

```
<div>Dear all,
<span>I took this really cool course
</span></div>
<span>on Coursera.org. I think it's
my favorite course I've EVER taken!
Here is the URL for it:
</span>
<a href="...">HTML, CSS and JS for Web Developers</a>
<div>
Does anyone know how I can give this course 6
out of 5 stars?
</div>
<div>
Thank you,
-Yaakov.... I mean a random student! Definitely not Yaakov.
</div>
```

Pregunta 2: ¿Cuál de las siguientes es una definición inválida de código HTML?

```
<div>I love <span>gardening!</span></div>
```

```
<span>I love <div>gardening</div></span>
```

```
<a>I love <div>gardening<span>!</span></div></a>
```

Tags semánticos de HTML para el encabezado

En primer lugar, ¿qué significa la palabra semántica? Bueno, en el lenguaje o en la lógica la definición de semántica se relaciona con el significado. Por ejemplo, en el lenguaje, las palabras tienen un significado inherente, los nombres tienen un significado inherente.

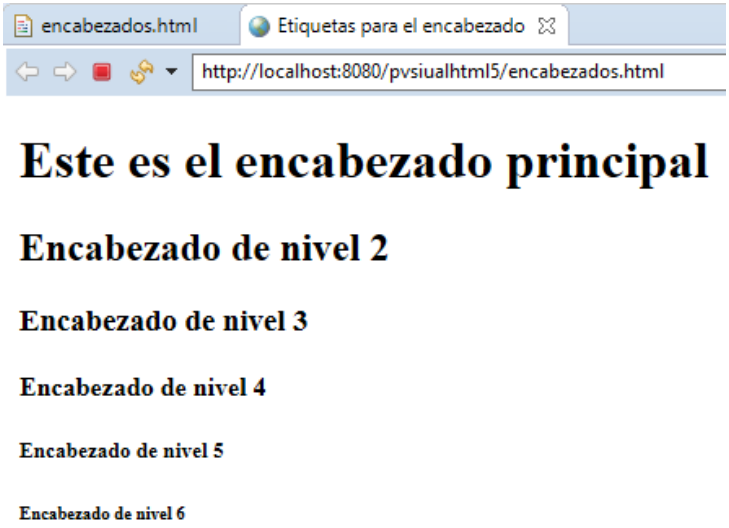
Cuando se aplica este concepto a HTML, ¿qué significa que un tag semántico? Significa que esa etiqueta implica algún significado para el contenido. En otras palabras, la etiqueta dice algo sobre el contenido que alberga, por ejemplo, su importancia o su descripción, sugieren ese significado. La importancia de estas etiquetas radica en que brindan la posibilidad a los humanos y/o máquinas de entender el significado de su contenido. También, pueden ayudar a que los motores de búsqueda puedan realizar mejores clasificaciones, es decir permiten realizar SEO (Search Engine Optimization, Optimización de motores de búsqueda en español).

Ahora realizaremos una revisión sobre etiquetas semánticas para los encabezados, mediante un ejemplo. Se define un documento HTML muy simple cuyo nombre es `encabezados.html`. Posee los elementos h1, h1, h2, h3, hasta el elemento h6. Estos elementos representan los encabezados disponibles en HTML. Además, brindan una información de manera implícita: el contenido del encabezado h1 es el más importante del documento. El contenido del elemento h6 también sería un encabezado del documento, pero es el menos importante de todos.

Elementos div y span | encabezados.html

```
1  <!doctype html>
2  <html>
3  <head>
4    <meta charset="utf-8">
5    <title>Etiquetas para el encabezado</title>
6  </head>
7  <body>
8    <h1>Este es el encabezado principal</h1>
9    <h2>Encabezado de nivel 2</h2>
10   <h3>Encabezado de nivel 3</h3>
11   <h4>Encabezado de nivel 4</h4>
12   <h5>Encabezado de nivel 5</h5>
13   <h6>Encabezado de nivel 6</h6>
14 </body>
15 </html>
```

Entonces, echemos un vistazo a cómo se ve este documento en el browser



El browser utiliza el estilo predeterminado para las etiquetas, así, visualmente el encabezado principal es el más grande. El encabezado número 2 es un poco menos importante y el resto de los encabezados disminuye en tamaño.

Un par de puntos importantes para entender sobre estos elementos:

Primero, a pesar de que la renderización predeterminada por el browser parece dar una distinción visual a los encabezados más relevantes, no deben usarse para diseñar. Estos elementos solo están destinados a describir la estructura de la página HTML. El desarrollador debe entonces usar CSS para diseñar la apariencia de las etiquetas de encabezado. Entonces, ¿por qué no usar un `tag <div>` para establecer el estilo de los encabezados? La respuesta es simple: si lo hiciéramos, perderíamos el significado de lo que es un encabezado.

En segundo lugar, algo que está marcado como h1 es obviamente la descripción más importante y generalizada del contenido de esa página. Mediante el SEO se pueden utilizar las etiquetas semánticas para ayudar a su motor de búsqueda a clasificar una búsqueda. Si los encabezados contienen un texto que transmite el tema central del resto del contenido, se podría clasificar la importancia del texto buscado dentro de los documentos en base a la jerarquía de los encabezados.

Ahora observaremos otro documento HTML. Este documento presenta nuevas etiquetas semánticas HTML5. En concreto presentan las etiquetas `<header>`, `<nav>`, `<section>` y `<article>`:

encabezadossemanticoshtml5.html

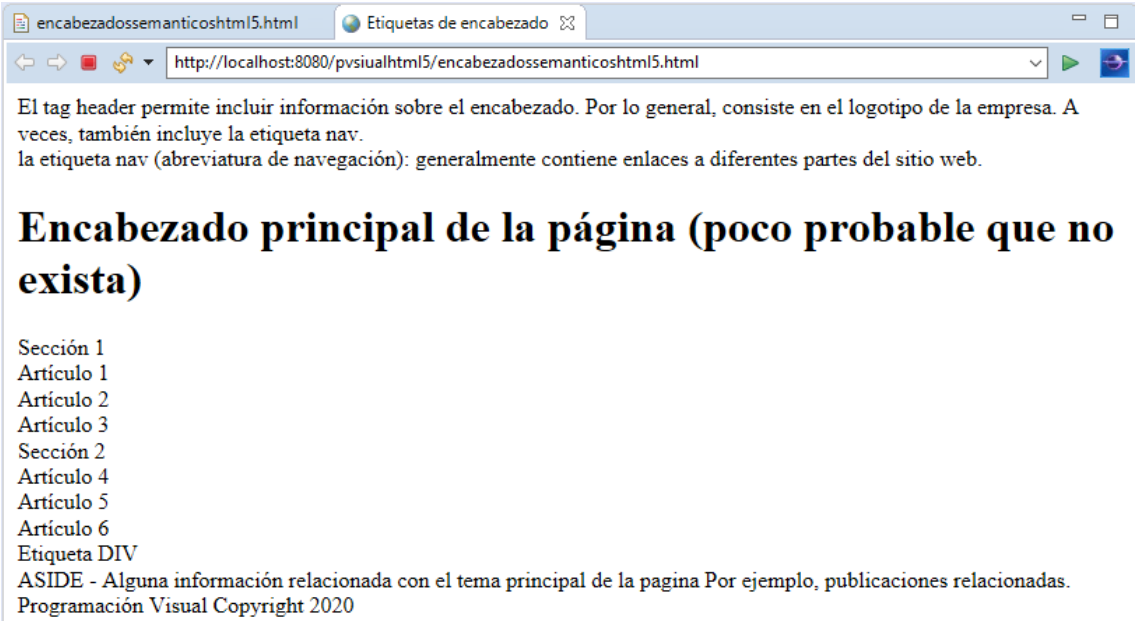
```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Etiquetas de encabezado</title>
</head>
<body>
  <header>
    El tag header permite incluir información sobre el encabezado.
    Por lo general, consiste en el logotipo de la empresa.
    A veces, también incluye la etiqueta nav.
    <nav>
        la etiqueta nav (abreviatura de navegación): generalmente contiene enlaces a
        diferentes partes del sitio web.
    </nav>
  </header>
  <h1>Encabezado principal de la página (poco probable que no exista)</h1>
  <section>
    Sección 1
    <article>Artículo 1</article>
    <article>Artículo 2</article>
    <article>Artículo 3</article>
  </section>
  <section>
    Sección 2
    <article>Artículo 4</article>
    <article>Artículo 5</article>
    <article>Artículo 6</article>
    <div>Etiqueta DIV</div>
  </section>
  <aside>
    ASIDE - Alguna información relacionada con el tema principal de la pagina
    Por ejemplo, publicaciones relacionadas.
  </aside>

  <footer>
    Programación Visual Copyright 2020
  </footer>
</body>
</html>
```

- Tag `<header>`: básicamente contiene información sobre el encabezado de la página. Por lo general, consiste en el logotipo de la empresa, un eslogan, a veces, etiquetas `<nav>`.
- Tag `<nav>`: presenta algún contenido que se utiliza para la navegación dentro de nuestro sitio web.
- Tag `<section>`: por lo general contiene varias etiquetas <article> aunque no existe una regla que indique que deba ser de esa manera, de hecho, es posible anidarlos de forma inversa.
- Tag `<aside>`: es básicamente un elemento que comunica que hay algo dentro de este elemento que está relacionado con el contenido principal de la página, pero no establece una relación tan directa como el contenido principal.
- Tag `<footer>`: contiene la información de pie de página.

Tenga en cuenta que todas estas etiquetas son elementos de nivel de bloque. Entonces, en lo que a nosotros respecta, visualmente podríamos haber usado las etiquetas `<div>` en lugar de ellas. Sin embargo, si observa el código, es obvio lo fácil que es leer y comprender lo que está pasando estructuralmente en esta página HTML. Por ejemplo, puede ver fácilmente que los artículos 1, 2 y 3 están relacionados de alguna manera y que son diferentes a los artículos 4, 5 y

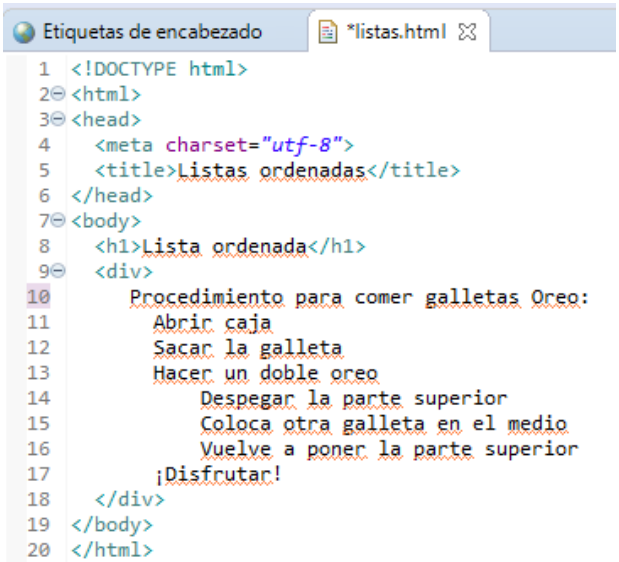
6, ya que están contenidos en dos secciones diferentes. Ese es el poder y la ventaja de los elementos semánticos. Inherentemente transmiten algún significado.

Visualicemos esta página en el browser:



Las listas

Son unas estructuras HTML increíblemente útiles que permiten agrupar contenido relacionado. Para explicar su funcionamiento y codificación nos valdremos de otro ejemplo. Observe el siguiente documento:



Seguramente ya te debes haber dado cuenta que si intentamos mostrar los pasos para comer una galleta oreo como corresponde de esta manera no lo lograremos. Recuerde que los dívs son componentes de nivel de bloque y el texto que contiene sigue un modelo de contenido de nivel de línea. Por lo tanto, se deben recurrir a etiquetas específicas.

Las listas ordenadas

El siguiente documento que altera el código HTML anterior mediante los tags `<ol>` y `<li>`.

El objetivo es crear una lista ordenada debido a que el procedimiento para comer galletas requiere que se sigan los pasos sin mezclarlos. Para lograr esto se utilizan los tags:

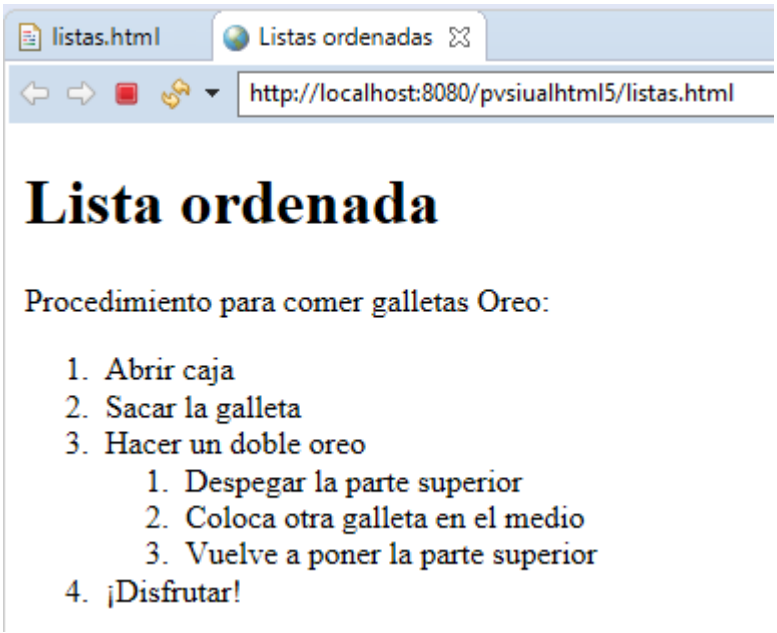
- `<ol>`: indica que el contenido se estructura como una lista ordenada. El término `ol` significa **ordered list**.
- `<li>`: Indica que el contenido es un elemento de la lista. El término `li` significa **list item**.

Con esta información se estructura el documento de la siguiente manera:

Etiquetas de encabezado | listas.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="utf-8">
5    <title>Listas ordenadas</title>
6  </head>
7  <body>
8    <h1>Lista ordenada</h1>
9    <div>
10      Procedimiento para comer galletas Oreo:
11      <ol>
12        <li>Abrir caja</li>
13        <li>Sacar la galleta</li>
14        <li>Hacer un doble oreo
15          <ol>
16            <li> Despegar la parte superior</li>
17            <li> Coloca otra galleta en el medio</li>
18            <li> Vuelve a poner la parte superior</li>
19          </ol>
20        </li>
21        <li> ¡Disfrutar!</li>
22      </ol>
23   </div>
24 </body>
25 </html>
```

Si se renderiza esta página mediante el browser se obtendrá lo siguiente:



Listas desordenadas

Muchas veces se requiere estructurar listas desordenadas. Por ejemplo, suponga que debe hacer referencia a una lista de compras. En este caso no es importante el orden de los elementos. La forma de estructurar listas desordenadas requiere el uso de dos etiquetas:

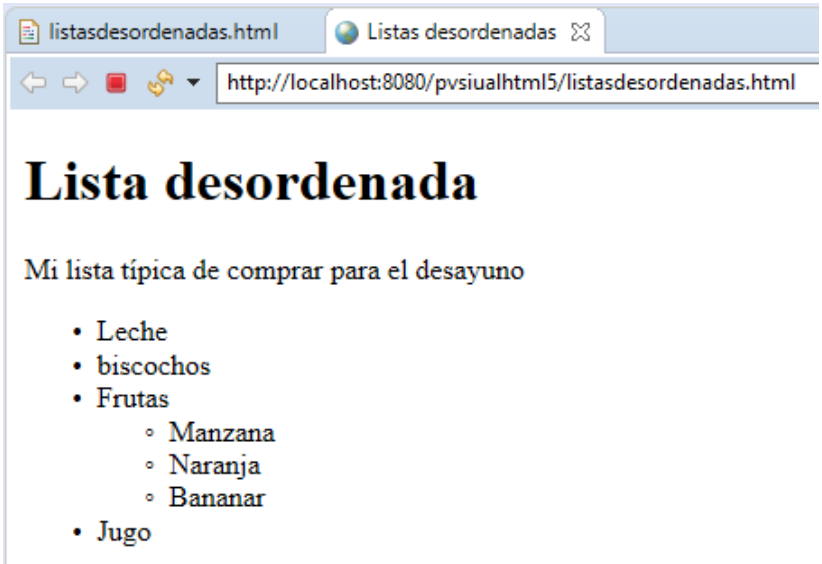
- `<ul>`: indica que el contenido se estructura como una lista desordenada. El término `ul` significa **unordered list**.
- `<li>`: Indica que el contenido es un elemento de la lista. El término `li` significa **list item**.

Con esta información se estructura el documento de la siguiente manera:

listasdesordenadas.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Listas desordenadas</title>
</head>
<body>
  <h1>Lista desordenada</h1>
  <div>

    Mi lista típica de comprar para el desayuno
    <ul>
      <li>Leche</li>
      <li>biscochos</li>
      <li>Frutas
        <ul>
          <li>Manzana</li>
          <li>Naranja</li>
          <li>Bananar</li>
        </ul>
      </li>
      <li>Jugo</li>
    </ul>


  </div>
</body>
</html>
```

Si se renderiza esta página mediante el browser se obtendrá lo siguiente:



Observe que tanto para la lista ordenada como para la lista desordenada es posible generar sublistas. Para ello, se puede anidar las etiquetas de lista. También se puede deducir que la etiqueta de item de lista debe explicitarse (mediante la apertura y cierre de esta etiqueta) de lo contrario la página no será válida.

Referencias de entidades de caracteres HTML

Hay muchas aplicaciones para referencias de entidades de caracteres HTML, sin embargo, nos concentramos en cómo podemos usarlas para resolver un problema muy común. Dado que HTML usa ciertos caracteres para su sintaxis, necesitamos una forma de diferenciar a esos caracteres como HTML y como contenido. Si queremos que el browser interprete caracteres HTML especiales como contenido regular, necesitamos brindar lo que se denominan caracteres de escape. Estos caracteres de escape permiten que el browser no los interprete como HTML.

Específicamente hay tres caracteres que necesitan de caracteres de escape para que HTML no los interprete como código HTML:

Por lo que a cada uno de ellos se les asigna los siguientes caracteres de escape:

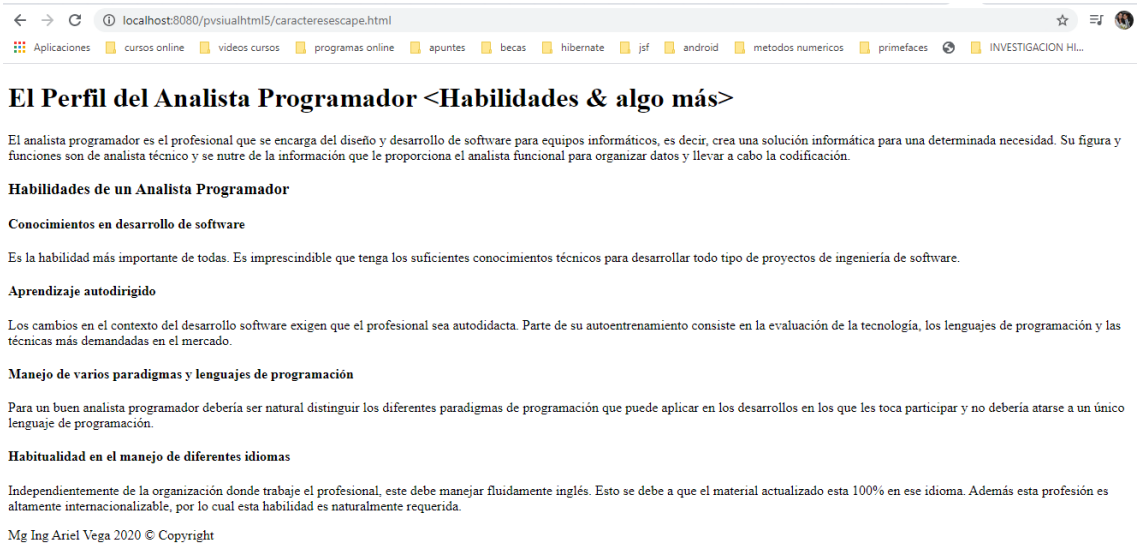
Observe el siguiente código en el cual se aplican estos caracteres:

caracteresescape.html — Entidades HTML

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="utf-8">
5    <title>Entidades HTML</title>
6  </head>
7  <body>
8  <h1>El Perfil del Analista Programador &lt;Habilidades &amp; algo más&gt;</h1>
9  <p>
10   El analista programador es el profesional que se encarga del diseño y desarrollo de software
11   para equipos informáticos, es decir, crea una solución informática para una determinada necesidad.
12   Su figura y funciones son de analista técnico y se nutre de la información que le proporciona
13   el analista funcional para organizar datos y llevar a cabo la codificación.
14 </p>
15 <h3>Habilidades de un Analista Programador</h3>
16 <h4>Conocimientos en desarrollo de software</h4>
17 <p>
18   Es la habilidad más importante de todas. Es imprescindible que tenga los suficientes conocimientos
19   técnicos para desarrollar todo tipo de proyectos de ingeniería de software.
20 </p>
21 <h4>Aprendizaje autodirigido</h4>
22 <p>
23   Los cambios en el contexto del desarrollo software exigen que el profesional sea autodidacta. Parte
24   de su autoentrenamiento consiste en la evaluación de la tecnología, los lenguajes de programación y
25   las técnicas más demandadas en el mercado.
26 </p>
27 <h4>Manejo de varios paradigmas y lenguajes de programación</h4>
```

```
28 <p>
29   Para un buen analista programador debería ser natural distinguir los diferentes paradigmas de
30   programación que puede aplicar en los desarrollos en los que les toca participar y no debería
31   atarse a un único lenguaje de programación.
32 </p>
33 <h4>Habitualidad en el manejo de diferentes idiomas</h4>
34 <p>
35   Independientemente de la organización donde trabaje el profesional, este debe manejar fluidamente
36   inglés. Esto se debe a que el material actualizado esta 100% en ese idioma. Además esta profesión
37   es altamente internacionalizable, por lo cual esta habilidad es naturalmente requerida.
38 </p>
39
40 <p>Mg Ing Ariel Vega 2020 &copy; Copyright</p>
41 </body>
42 </html>
```

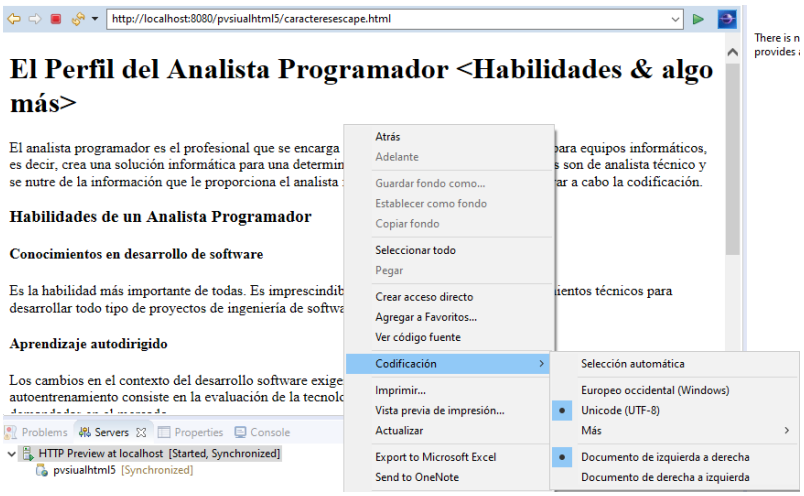
Observe que en la línea 8 se utilizaron estos caracteres de escape. Si se renderiza esta página se obtendrá lo siguiente:



Otros caracteres especiales

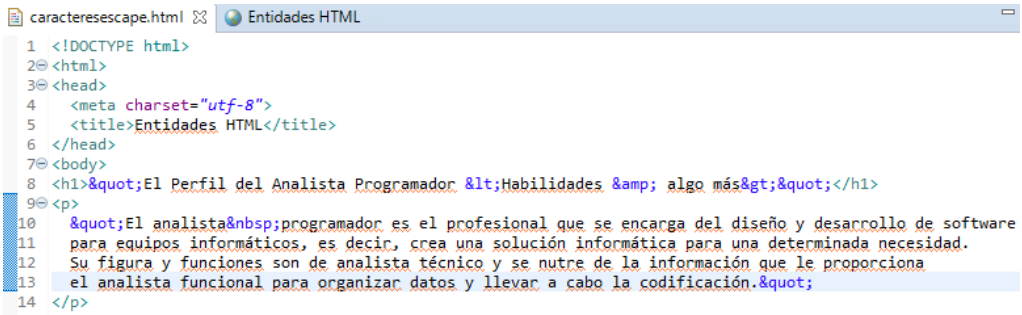
HTML contiene varias referencias de entidades HTML diferentes que sirven para expresar caracteres especiales. A continuación, se describirán algunas de ellas:

- Símbolo de copyright: es particularmente común y no está disponible fácilmente en los teclados. Observe la línea 40 del ejemplo anterior. Podrá observar que este carácter se obtiene a través de la siguiente referencia `&copy;`
- No romper espacios entre palabras: Hay palabras que en conjunto tienen un significado por lo cual no deberían separarse. Por ejemplo, la profesión Analista Programador tiene un significado propio mientras se los lea en conjunto, ya que no es lo mismo que un Analista o que un Programador. Para lograr esto se recurre a la referencia ` `. Esta referencia se coloca en el lugar que le correspondería al espacio en blanco. Esto es particularmente útil cuando se redimensiona el tamaño de la ventana del browser. Al momento de redimensionar el texto se ajusta para lograr mantenerse visible. En esta situación las palabras podrían cortarse, para evitarlo se utiliza esta referencia.
- &quot: Se usa con mucha frecuencia y es especialmente útil cuando alguien intenta escribir un correo electrónico basado en HTML. Dado que los clientes de correo electrónico son conocidos por usar un conjunto de caracteres mucho más limitado que UTF-8, algunos de los caracteres a veces se confunden. Para ilustrar las consecuencias de esto se ejecutará el ejemplo anterior en el browser del STS y luego se ingresará al menú vista para cambiar la codificación de caracteres. Para ello sobre la ventana del browser presione el botón derecho del mouse e ingrese a codificación:



Luego seleccione la opción más y elegir por ejemplo Europeo Occidental (Windows), entonces la apariencia se verá de la siguiente manera

Como se puede observar algunos caracteres se ven extraños, ¿cómo resolvemos eso?: Mediante el uso de la referencia `&quot`. A continuación, se aplica esta referencia para evitar que se cambie el juego de caracteres en el título y el primer párrafo del documento:



Si se refresca la página se observará que se mantiene el título y el primer párrafo sin cambiar la codificación definida en la página, es probable que aún así el formato sea diferente, pero al menos los caracteres no se verán raros.

Crear enlaces

Estudiaremos los diferentes tipos de enlaces y la forma de usarlos en HTML. El primer tipo de enlaces que vamos a ver son los enlaces internos. Observe el contenido de un documento donde se han definido enlaces internos.

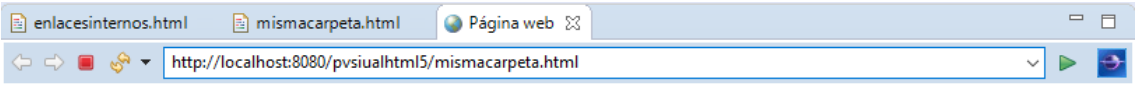
*enlacesinternos.html | mismacarpeta.html | Enlaces internos

```
1  <!doctype html>
2  <html>
3  <head>
4    <meta charset="utf-8">
5    <title>Enlaces internos</title>
6  </head>
7  <body>
8    <h1>Enlaces internos</h1>
9    <section>
10     Podemos crear un enlace a un archivo ubicado en la misma carpeta del archivo actual, por ejemplo
11     otra pagina HTML. Para ello se recurre al elemento href del tag &lt;a&gt;:
12     <a href="mismacarpeta.html" title="Enlace dentro de la misma carpeta">
13         Enlace a un archivo en la misma carpeta
14     </a>
15
16     <a href="mismacarpeta.html" title="Enlace dentro de la misma carpeta">
17       <div> DIV como enlace a un archivo en la misma carpeta</div>
18     </a>
19   </section>
20 </body>
21 </html>
```

La forma de crear enlaces es especificando un tag que incluya el atributo `href`. Este atributo significa referencia de hipertexto. El valor de `href` puede ser una URL relativa o absoluta. En nuestro caso aquí, dado que estamos discutiendo enlaces internos (aquellos que apuntan a páginas web internas de la aplicación), son todos enlaces URL relativos. Dado que no proporcionamos información de directorio, el browser asumirá que *mismacarpeta.html* es un archivo que se ubica en el mismo directorio que *enlacesinternos.html*. Es recomendable especificar siempre el atributo `title` cuando crea enlaces dentro del tag `<a>`, ya que por ejemplo, los lectores de pantalla utilizan este atributo para ayudar a las personas con discapacidad visual a navegar a través de la página web. ¿Qué papel juega el contenido dentro de los enlaces? El contenido de la etiqueta actuará como enlace, cuando el usuario lo seleccione irá a la página indicada en `href`.

Observe el segundo tag `<a>`. En él se define un enlace y su contenido es un tag `<div>`. Esto provoca que todo el contenido de la etiqueta sea un enlace.

Ambos enlaces apuntan entonces a la misma pagina *mismacarpeta.html*, el cual muestra el siguiente contenido:



Este documento se ubica en el mismo directorio de enlacesinternos.html

Volver a la pagina enlacesinternos.html

Donde se puede deducir que simplemente contiene otro enlace a la página *enlacesinternos.html.* Volviendo de nuevo al código html de esta página podemos observar que en la línea 17 el IDE nos está indicando un mensaje de advertencia. Resulta interesante observar aquí es que el primer enlace es claramente una etiqueta de nivel de línea, mientras que el segundo enlace al utilizar un `<div>`: ¿Estamos usando el elemento de nivel de bloque ( ), dentro de un elemento en línea? Bueno, resulta que las cosas son un poco más interesantes. En HTML5 el tag `<a>` es tanto un contenido de flujo como un contenido de fraseo. En otras palabras, en términos de HTML4.x esta es tanto un elemento en línea como un elemento de nivel de bloque. Y esto es lo que nos permite tomar esta etiqueta y utilizar dentro de ella un `<div>`. Los autores de la especificación HTML5 se dieron cuenta de que muchas veces se podría tener la necesidad de poder hacer clic en toda una región que actúe como enlace (por ejemplo,

el logotipo o algún tipo de nombre de empresa en la esquina superior izquierda: se esperaría poder hacer clic para ir a la página principal de la empresa. Antes de HTML5, los desarrolladores tenían que usar todo tipo de trucos para lograr ese efecto porque la etiqueta `<a>` solo era una etiqueta de nivel de línea. Por lo tanto, este mensaje de advertencia no debería mostrarse.

A continuación, observemos un ejemplo de uso de enlaces externos:

enlacesinternos.html | mismacarpeta.html | Página web | *enlacesexternos.html

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Enlaces externos</title>
</head>
<body>
  <h1 id="top">Enlaces externos</h1>
  <section>
    <p>
      Este enlace apunta al campus virtual donde podrá ver una descripción
      de la asignatura programación visual
      <!-- Enlace del aula virtual con TARGET -->
      <a href="https://campus.fi.unju.edu.ar/course/index.php?categoryid=21"
      target="_blank" title="Es nuestra aula virtual!">campus fi UNJu</a>
    </p>
  </section>
</body>
</html>
```

Este documento contiene básicamente un solo enlace. Realmente no hay nada especial en los enlaces externos. Normalmente en el atributo `href` se deberá colocar la referencia absoluta de la url, especialmente si están alojados en un nombre de dominio diferente al de su sitio web. Sin embargo, hay una característica que se desea abordad, y es el atributo `target`. Cuando se establece el valor _blank, obliga al browser a abrir esta página en una nueva pestaña o en una nueva ventana. En términos de navegabilidad esta opción es una de las más adecuadas para que el sitio origen se mantenga disponible, de lo contrario el usuario probablemente al utilizar un enlace continuará navegando y en algún momento perderá la página desde la cual inicio la navegación por otros sitios.

Otro tipo de enlace que es extremadamente importante conocer es el identificador de fragmento. Observe el siguiente ejemplo:

enlacesinter... | mismacarpeta.... | enlacesexter... | enlacesenmis... | caracteresesc... | Enlaces

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Enlaces</title>
</head>
<body>
  <h1 id="top">Enlaces a secciones dentro de la misma página</h1>
  <h3 id="top">Ética del estudiante universitario</h3>
  <section>
    <ul>
      <!-- Link to every section in the page -->
      <li><a href="#section1">#Plagio Académico</a></li>
      <li><a href="#section2">#Cuando se presenta</a></li>
      <li><a href="#section3">#Otras circunstancias de plagio académico</a></li>
      <li><a href="#section4">#¿Que se puede hacer?</a></li>
    </ul>

  </section>

  <section id="section1">
    <h3>(#Sección 1) Plagio Académico</h3>
    <p>Huerta (2006) define el plagio académico como la acción de hacer pasar como propios,
    ideas o textos que pensaron otros y que nos fueron transmitidos por ellos, bien por escrito,
    oralmente o con algún otro mecanismo de comunicación.</p>
  </section>

```

```
28  <section id="section2">
29    <h3>(#section2) Cuando se presenta</h3>
30    <p>Plantea el autor que el plagio se consuma en dos circunstancias:</p>
31    <p>1-Cuando se usan ideas textuales de otro sin respetar las comillas indicativas de literalidad.</p>
32    <p>2-Cuando no se presentan al lector los indicios necesarios para identificar de qué autor, libro o
33        documento o circunstancia fue tomada la idea ajena</p>
34  </section>
35  <section id="section3">
36    <h3>(#section3) Otras circunstancias de plagio académico</h3>
37    <p>1-Comprar los servicios profesionales de otros para la elaboración de un escrito
38       (artículo, ensayo, ejercicio, trabajo especial de grado, tesis) que luego aparecerá
39       ante la comunidad académica como de elaboración intelectual propia</p>
40    <p>2-Cobrar al futuro autor por guiarlo en el proceso de construcción de un escrito
41       (trabajo especial de grado, tesis); ello cuando quien cobra es un docente universitario y
42       quien paga es un homólogo de su misma universidad. En esencia, la labor de tutoría está dentro
43       de las funciones que un profesor universitario debe proporcionar a la académica como
44       contraprestación de servicios y por compromiso con la propia academia de donde procede y
45       quien le formó.</p>
46  </section>
47  <section id="section4">
48    <h2>(#section4) ¿Que se puede hacer?</h2>
49    <p>1-Conocer los valores y principios éticos que rigen a la universidad, incorporarlos al conjunto
50       de valores personales y actuar en sintonía con ellos.</p>
51    <p>2-Promover la reflexión acerca de los valores personales y de la ética respecto a los productos
52       intelectuales que se generan en el quehacer universitario.</p>
53    <p>3-Establecer diálogos permanentes a fin de distinguir entre lo que es una producción académica
54       propia y otra ajena; revisar las consecuencias, trascendencias e implicaciones del plagio tanto
55       para vida como estudiante y como profesional</p>
56  </section>
57  <div>
58    <p>Fuente: http://educapuntes.blogspot.com/2016/06/etica-y-plagio.html</p>
59    <p><a href="#top">Volver al índice</a></p>
60  </div>
61 </body>
62 </html>
```

Como puede ver, todos los enlaces que se han configurado aquí tienen un formato muy específico en el atributo `href`, se utiliza un # seguido de un nombre como section1, section2, etc. Ahora a lo que apuntan estos enlaces es a una sección de nuestra página. Observe la manera en la cual los objetivos de los enlaces son identificados, las etiquetas objetivo poseen un atributo `id` con el nombre asignado en los atributos `href` del enlace. Observe que el nombre de la sección no contiene el signo #. Solo el enlace a esa sección contiene el signo #. Esa es una forma de identificar una sección dentro de la página.

Otra forma de generar los identificadores de fragmento se realiza mediante los puntos de anclaje. Observe en el ejemplo la línea 59. Se crea una etiqueta con el atributo href, cuyo valor es el identificador asignado al título de la página (línea 8). La forma en que se refiere a estas secciones es exactamente la misma, coloca un # delante del nombre de la sección y pega ese valor en el atributo href de una etiqueta de anclaje.

Por último, observe que se ha identificado con el mismo valor, tanto al encabezado de nivel 1, como al encabezado de nivel 3. El IDE no indica una advertencia ni un error, pero esto podría generar inconvenientes al navegar nuestra página se vuelve más compleja, por lo tanto, hay que tener cuidado en el manejo de los identificadores: esto es no deben existen componentes dentro de la página con el mismo identificador.

Si ejecutamos esta página en el browser tendrá la siguiente representación:

(#Sección 1) Plagio Académico

Huerta (2006) define el plagio académico como la acción de hacer pasar como propios, ideas o textos que pensaron otros y que nos fueron transmitidos por ellos, bien por escrito, oralmente o con algún otro mecanismo de comunicación.

(#section2) Cuando se presenta

Plantea el autor que el plagio se consuma en dos circunstancias:

1-Cuando se usan ideas textuales de otro sin respetar las comillas indicativas de literalidad.

2-Cuando no se presentan al lector los indicios necesarios para identificar de qué autor, libro o documento o circunstancia fue tomada la idea ajena

(#section3) Otras circunstancias de plagio académico

1-Comprar los servicios profesionales de otros para la elaboración de un escrito (artículo, ensayo, ejercicio, trabajo especial de grado, tesis) que luego aparecerá ante la comunidad académica como de elaboración intelectual propia

2-Cobrar al futuro autor por guiarlo en el proceso de construcción de un escrito (trabajo especial de grado, tesis); ello cuando quien cobra es un docente universitario y quien paga es un homólogo de su misma universidad. En esencia, la labor de tutoría está dentro de las funciones que un profesor universitario debe proporcionar a la académica como contraprestación de servicios y por compromiso con la propia academia de donde procede y quien le formó.

(#section4) ¿Que se puede hacer?

1-Conocer los valores y principios éticos que rigen a la universidad, incorporarlos al conjunto de valores personales y actuar en sintonía con ellos.

2-Promover la reflexión acerca de los valores personales y de la ética respecto a los productos intelectuales que se generan en el quehacer universitario.

3-Establecer diálogos permanentes a fin de distinguir entre lo que es una producción académica propia y otra ajena; revisar las consecuencias, trascendencias e implicaciones del plagio tanto para vida como estudiante y como profesional

Fuente: http://educapuntes.blogspot.com/2016/06/etica-y-plagio.html

Lo realmente interesante de los identificadores de fragmentos es que los browsers al interpretar un url pueden incluir la información del identificador, entonces si uno posee la siguiente url

```
http://localhost:8080/pvsiualhtml5/enlacesenmismapagina.html#section4
```

no solamente ingresará a la página solicitada, sino que se posicionará dentro de la página en la ubicación indiada por el identificador de fragmento. Si bien los identificadores de fragmentos son muy útiles para saltar a diferentes partes de la misma página, recientemente se han vuelto aún más importantes, ya que se utilizan para la navegación dentro de las aplicaciones SPA (Simple Page Application). Las SPA se han vuelto extremadamente populares. Por lo tanto, saber cómo funcionan los identificadores de fragmentos es un paso importante para poder codificar aplicaciones de una sola página.

Referencia teórica: Curso MOOC Html, CSS and Javascript for web developers de Yaakov Chaikin, profesor de la Jhons Hopkins University y Coursera.

Traducción: Mg Ariel Alejandro Vega

Ejemplos usados en el entorno STS: Mg Ariel Alejandro Vega