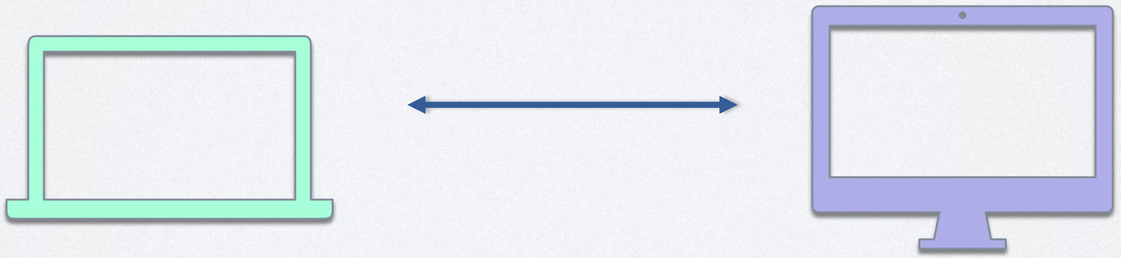
REVISIÓN DE:
FUNDAMENTOS DE DESARROLLO WEB

INTRODUCCIÓN A LA WEB

- La web se basa en el protocolo HTTP (HyperText Transfer Protocol) para transferir información
  - Existen muchos más protocolos como FTP (File Transfer Protocol), usado para transmitir archivos entre máquinas o IMAP, POP y SMTP, asociados al correo electrónico
- La comunicación se basa en una arquitectura cliente-servidor en la que el cliente le envía una petición al servidor y éste le responde con una respuesta
- El cliente tiene que conocer el lugar al que debe de enviar esa petición
  - Todos entendemos la utilidad de los números de teléfono
  - Las comunicaciones de red se hacen con algo parecido, llamado dirección IP
    - Conociendo la dirección IP de una máquina, podemos intentar comunicarnos con ella http://74.125.127.93
  - Las direcciones IP son complicadas de recordar… ¿no habrá una forma mejor?
    - Por suerte existe un concepto llamado DNS (Domain Name System) una especie de directorio gigante que nos dice la dirección IP asociada a un dominio como www.google.com

INTRODUCCIÓN A LA WEB

- Por lo tanto, hay dos roles en HTTP: cliente y servidor (frontend y backend)
- El cliente establece una conexión TCP con el servidor
- Después, ambos se comunican se comunican mediante intercambio de mensajes HTTP



PETICIONES

- Por ejemplo, vamos a hacer una petición con nuestra máquina para acceder a Google España
- En primer lugar, necesitaríamos conocer la dirección IP de la máquina a la que queremos acceder
  - Esto se sale del temario de la asignatura, pero como sabéis, se hace a través de DNS
- Ejemplo de petición que se hace para acceder a http://www.google.es

```
GET  HTTP/1.1 www.google.es
User-Agent: PostmanRuntime/7.16.3
Accept: */*
Cache-Control: no-cache
Host: www.google.es
Accept-Encoding: gzip, deflate
Connection: keep-alive
cache-control: no-cache
```

PETICIONES

- Esta petición contiene:

Versión del protocolo

Tipo de petición GET HTTP/1.1 www.google.es URL del recurso

User-Agent: PostmanRuntime/7.16.3
Accept: */*
Cache-Control: no-cache
Host: www.google.es
Accept-Encoding: gzip, deflate
Connection: keep-alive
cache-control: no-cache

Headers

- Los headers incluyen una serie de metadatos para facilitar la comunicación
- El tipo de petición (o nombre del método) puede tomar una serie de valores concretos

MÉTODOS HTTP

- Algunos de los métodos definidos por HTTP son:
  - GET: solicita acceso de lectura al recurso especificado
  - HEAD: solicita únicamente las cabeceras del GET
  - DELETE: elimina el recurso especificado
  - PUT: sube un recurso al servidor
  - POST: envía datos al servidor para que éste los procese

RESPUESTA DEL SERVIDOR

- Cuando el servidor nos responde a nuestra petición GET, nos envía la siguiente información

Request URL: https://www.google.es/
Request Method: GET
Status Code: 200
…

- Lo más importante es el código de respuesta, que nos dará información acerca de cómo ha ido la petición

CÓDIGOS DE RESPUESTA HTTP

- Son universales y están estructurados, veamos algunos ejemplos:
  - 1XX: Respuesta informativa
  - 2XX: Éxito
    - 200 OK: la petición ha tenido éxito y se envía la respuesta. Es la más habitual
    - 201 Created: la petición ha tenido éxito y se ha creado un nuevo recurso
  - 3XX: Redirección
    - 301 Moved Permanently: el recurso al que se ha accedido se ha movido a otra dirección. Lo correcto sería acceder a él a través de su nueva dirección
  - 4XX: Error de cliente
    - 400 Bad Request: el servidor no ha podido entender la petición
    - 404 Not Found: el recurso solicitado no existe en el servidor
  - 5XX: Error del servidor
    - 500 Internal Server Error: el servidor ha experimentado un error intentando satisfacer nuestra petición

RESPUESTA DEL SERVIDOR

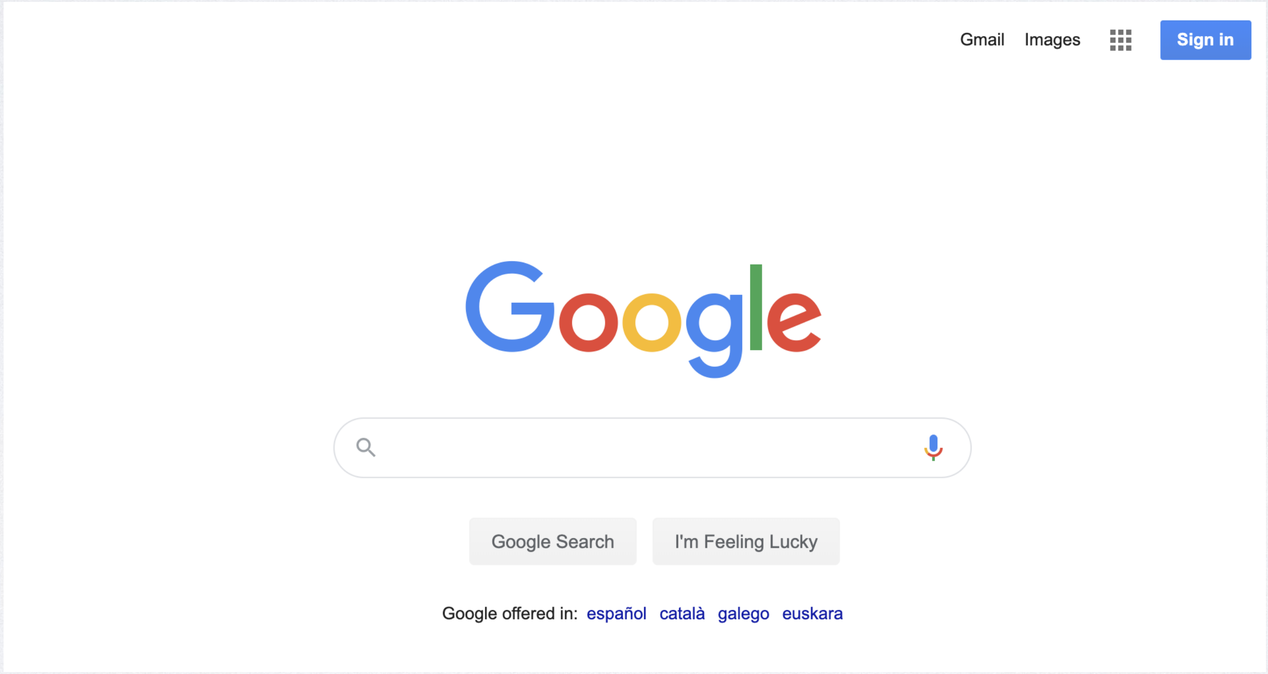
- Además, en este caso, la respuesta tiene además el siguiente cuerpo

```
<!doctype html>
<html itemscope="" itemtype="http://schema.org/WebPage" lang="es">

<head>
    <meta
        content="Google.es permite acceder a la información mundial en castellano, catalán, gallego, euskara e inglés."
        name="description">
    <meta content="noodp" name="robots">
    <meta content="text/html; charset=UTF-8" http-equiv="Content-Type">
    <meta content="/images/branding/googleg/1x/googleg_standard_color_128dp.png" itemprop="image">
    <title>Google</title>
    <script nonce="pfXcO1WHzwI1x55g2zJfJA==">…</script>
    <style>…</style>
```

RESPUESTA DEL SERVIDOR

- Nuestro navegador (cliente) procesa la respuesta y nos muestra el resultado



WEB ESTÁTICA

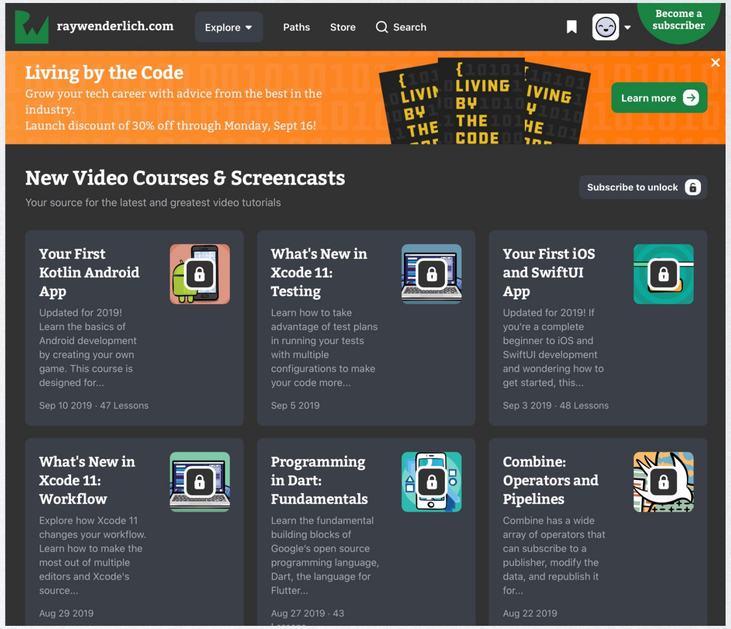
- En los orígenes de internet, las webs eran estáticas
- Esto quiere decir que, acceder a una página web no era más que descargar un fichero del disco duro del servidor y mostrarlo
  - La web se mostrará exactamente igual a todos los usuarios que consigan acceder a ella
  - Modificar el contenido de la web conlleva tocar el documento html

WEB ESTÁTICA

- Hoy en día sigue habiendo un gran número de webs estáticas, ya que tienen sus ventajas:
  - Al no ejecutarse código, suelen ser más seguras
  - Al no ejecutarse código, ofrecen un mejor rendimiento en el cliente
  - Fáciles de desplegar
- Por lo tanto, para determinados casos, la web estática puede ser la mejor opción
- En el curso de fundamentos, vimos principalmente este tipo de web

WEB DINÁMICA

- Se habla de webs dinámicas en dos casos:
  - Cuando el servidor construye, a partir de una plantilla “en tiempo real” la página que enviará al cliente y éste mostrará
  - Cuando la página web contiene scripts que se ejecutarán en la máquina del cliente
- Este es el web en el que nos centraremos



WEB DINÁMICA

- En el primer caso, el uso de plantillas hace que podamos añadir contenido a nuestra web sin tener que realizar el tedioso trabajo de tocar el documento html
  - Puede verse en prácticamente cualquier web de noticias
- El segundo caso, permite realizar webs más interactivas y en muchos casos más ligeras
  - Utilizando scripts en JavaScript u otros lenguajes de scripting, podemos por ejemplo pedirle al servidor los datos que nos interesan si hacemos click en un enlace en lugar de tener que descargar toda una URL al completo

FUNDAMENTOS: LENGUAJES DE MARCAS

¿QUÉ SON LOS LENGUAJES DE MARCAS?

- Los lenguajes de marcas surgen en los años 60 y codifican un documento de forma que combinan el propio texto del documento con unas etiquetas (o marcas) que anotan:
  - La estructura del texto (pueden tener jerarquía)
  - Forma de presentación
  - Forma de procesado

¿QUÉ SON LOS LENGUAJES DE MARCAS?

- A finales de los 80, nace HTML, la base de las páginas web
  - Legible y fácil de aprender
  - Extensible
  - Flexible (excesivamente)

¿QUÉ SON LOS LENGUAJES DE MARCAS?

- Ejemplo:

```
<producto>
    <nombre>Chocolatina</nombre>
    <precio>2</precio>
    <categoria>Alimentación</categoria>
</producto>
```

- Permiten separar datos de metadatos

ETIQUETAS (TAGS)

- Normalmente, las etiquetas vendrán por pares:
  - Etiqueta de inicio: <nombre>
  - Etiqueta de fin: </nombre> (con esto cerraremos la última etiqueta “nombre” que hayamos abierto)
- No son sensibles a mayúsculas/minúsculas, pero se recomienda utilizar siempre minúsculas
- Una buena nomenclatura de etiquetas, aumenta su legibilidad para los humanos: intentemos utilizar nombres descriptivos y concisos
- Se desaconseja utilizar caracteres raros, tildes, caracteres de puntuación…

ELEMENTOS

- Un elemento es el conjunto etiqueta de inicio + contenido + etiqueta de fin
- Existen los denominados “elementos vacíos”
  - Únicamente contienen etiqueta de inicio
- Los elementos pueden contener texto (simples) o incluso otros elementos (compuestos)

ELEMENTOS

- Ejemplo:

```
<producto>
    <nombre>Chocolatina</nombre>
    <precio>2</precio>
    <categoria>Alimentación</categoria>
</producto>
```

- El nombre, precio y categoría serían elementos simples
- El producto, sería un elemento compuesto
- Ninguno de ellos es un elemento vacío

ATRIBUTOS

- Parejas nombre-valor que indican propiedades de los elementos
- Van dentro de la etiqueta de inicio
- Normalmente, el valor irá entrecomillado

```
<producto>
    <nombre>Chocolatina</nombre>
    <precio moneda=“euro">2</precio>
    <categoria>Alimentación</categoria>
</producto>
```

FUNDAMENTOS: HTML

INTRODUCCIÓN A HTML

- HTML (HyperText Markup Language) es el lenguaje de marcas utilizado para definir páginas web

- A grandes rasgos se compone de tres grandes bloques:

  - Directivas: mediante DOCTYPE, identifica el tipo de documento

  - Cabecera (head): no se dibuja en el navegador, contiene algunos metadatos como el título de la web

  - Cuerpo (body): dentro del body introducimos los elementos visibles de nuestra web

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Test page</title>
</head>
<body>
    Hello world!
</body>
</html>
```

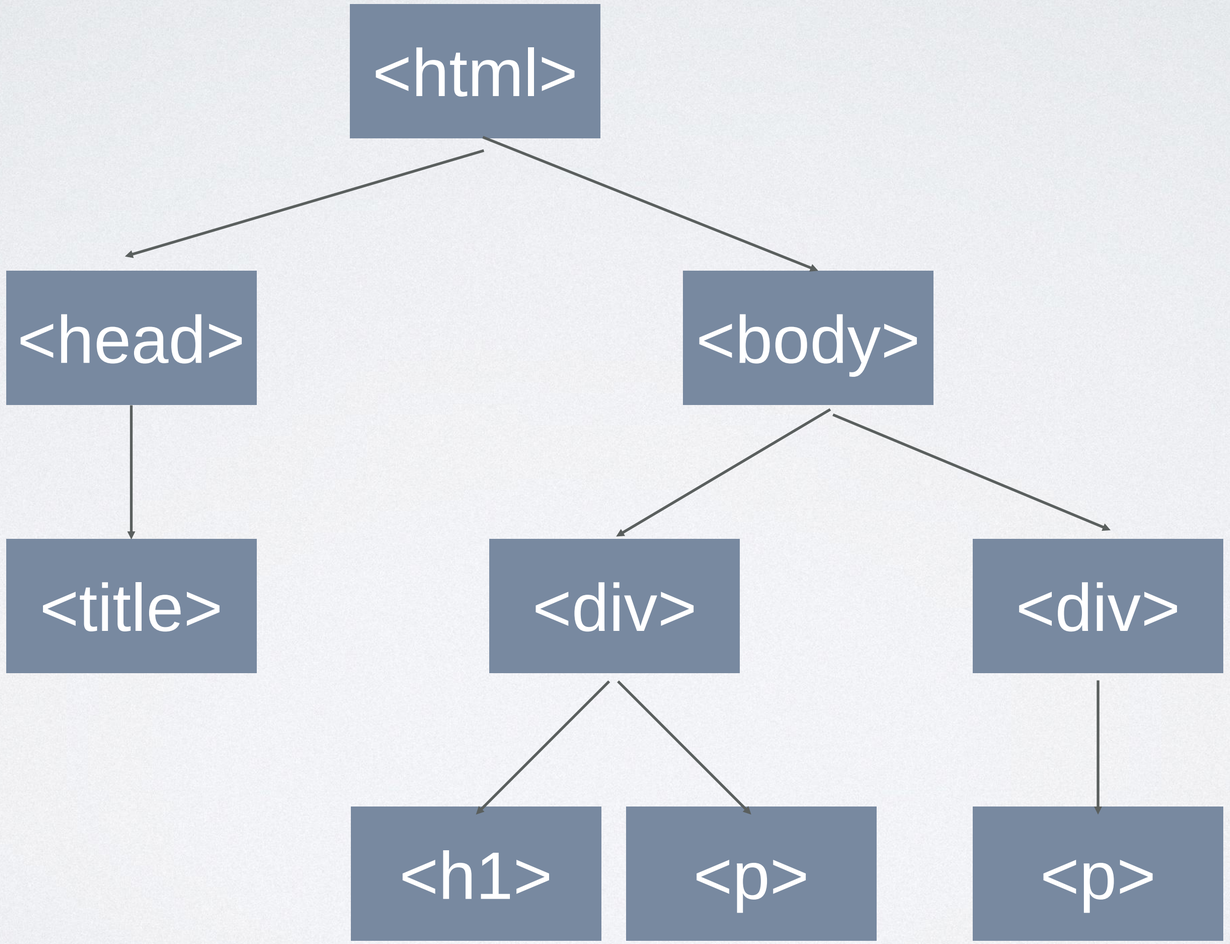
INTRODUCCIÓN A HTML

- Una de las “ventajas” de html es lo laxo que es:
  - ¿Cuándo utilizamos atributos y cuándo utilizamos elementos?
  - Los elementos deberían utilizarse para encapsular entidades
  - Los atributos, para dar información adicional
- Al ser un documento de texto, puede editarse con cualquier editor de texto plano como el bloc de notas o aplicaciones más especializadas como Atom, Sublime Text, Brackets, Visual Studio Code, Notepad++, WebStorm…
  - También existen editores online como https://codepen.io/pen/o https://jsfiddle.net

INTRODUCCIÓN A HTML

- Para que un HTML sea sintácticamente correcto:
  - Los elementos deben de estar correctamente cerrados y anidados
  - En caso de haber atributos, estarán únicamente en etiquetas de inicio
  - Todo el documento estará envuelto en un tag <html>
    - Los HTML tienen estructura de árbol, con un elemento raíz del que van colgando otros elementos
- Que un HTML sea sintácticamente correcto no significa que esté bien

HTML - ESTRUCTURA



HTML - HEAD

- Contiene datos que, aunque no se consideren parte del documento, aportan información del mismo
- Por ejemplo, el título (<title>) se mostrará como el nombre de la pestaña en la que tengamos abierta nuestra web
- Puede tener otros elementos como:
  - <style> para añadir hojas de estilo (lo veremos en la clase de css)
  - <script> para incluir código Javascript (lo veremos en la clase de Javascript)

HTML - BODY

- En él estarán los datos que el navegador “pintará”: el contenido del documento
- Es el elemento html más importante de todos
- Es descendiente directo de <html>
- Si el documento tiene <head>, el <body> irá después

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Prueba body</title>
</head>
<body>
    Aquí va nuestro contenido
</body>
</html>
```

CUERPO HTML: HEADINGS

- Es muy habitual utilizar títulos (headings) en los documentos html
- Ayudan a estructurar el documento
- Se crean mediante las etiquetas <h1>, <h2>, <h3>, <h4>, <h5> y <h6>
  - Definen seis niveles ordenados de mayor a menor importancia (<h1> es el más importante, <h6> el menos importante)
- Es importante utilizarlos bien, ya que muchas veces los motores de búsqueda los utilizan para indexar el documento
- Más adelante aprenderéis a utilizar css para alterar el tamaño y estilo de los diferentes elementos
- Normalmente, los títulos de categorías 1 a 3 serán de un tamaño mayor al texto estándar y los 5 y 6 serán menores

```
<!DOCTYPE html>
<head>
</head>
<body>
    <h1>Heading 1</h1>
    <h2>Heading 2</h2>
    <h3>Heading 3</h3>
    <h4>Heading 4</h4>
    <h5>Heading 5</h5>
    <h6>Heading 6</h6>
</body>
</html>
```

CUERPO HTML: PARAGRAPHS

- Los párrafos se marcan con <p>
- Ofrecen otra herramienta para estructurar el texto del documento
- Normalmente, el navegador insertará un salto de línea entre párrafos para separarlos

```
<!DOCTYPE html>
<head>
</head>
<body>
  <p>Aquí tenemos un párrafo.</p>
  <p>Aquí otro.</p>
  <p>Y aquí un tercero.</p>
</body>
</html>
```

CUERPO HTML: LINE BREAKS

- Los saltos de línea se insertan con
  - No podemos confiar en los saltos de línea del teclado para dar formato a un html
- Al insertarlos, no estamos creando un nuevo párrafo
- Es uno de los elementos vacíos, es decir, que no tiene versión de cierre (</br>)
- Pueden parecer lo mismo que los párrafos, pero los saltos de línea no describen la estructura del documento

```
<!DOCTYPE html>
<head>
</head>
<body>
  Aquí tenemos una línea.<br>
  Aquí otra.<br>
  Otra más.<br><br>
  Y una última.<br>
</body>
</html>
```

CUERPO HTML: ESTILO - B

- Todo el texto dentro entre dos elementos irá en negrita
  - Debe utilizarse como último recurso si no existe un tag más apropiado para nuestro contenido
    - Los headings deberían de ir con <h*>
    - El texto enfatizado, con
    - El especialmente importante, con
    - También podemos destacar con <mark>

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Prueba article</title>
</head>
<body>
   Vamos a <b>resaltar este texto</b>.
</body>
</html>
```

CUERPO HTML: ESTILO - I

- Suele utilizarse para representar texto en cursiva, aunque esto no quiere decir que el navegador vaya a utilizar la versión cursiva de la fuente

- De igual modo que con <b>, debemos utilizar <i> únicamente si no hay otro recurso más apropiado.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Prueba article</title>
</head>
<body>
Vamos a <i>poner esto con otra estética</i>.
</body>
</html>
```

CUERPO HTML: ESTILO - SUB Y SUP

- Utilizamos <sub> para poner texto como subíndice y <sup> para marcarlo como superíndice
- Útiles para fórmulas químicas o matemáticas

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Prueba article</title>
</head>
<body>
Vamos a poner una expresión un tanto inútil: x<sub>n</sub> = x<sup>n+1</sup> + 5
</body>
</html>
```

CUERPO HTML: AGRUPACIONES - DIV

- HTML5 nos proporciona varias herramientas para agrupar contenido, las más importante es el
- Es un bloque genérico para estructurar los documentos
  - Es muy habitual combinarlo con un estilo css
  - Normalmente, el navegador mete un salto de línea antes y después del

```
<!DOCTYPE html>
<head>
</head>
<body>
  <div style="background-color:lightsalmon">
      <h1>Título</h1>
      <p>Texto normal (párrafo).</p>
  </div>
</body>
</html>
```

CUERPO HTML: AGRUPACIONES - HEADER

- Durante el desarrollo de html, se observaron patrones comunes y se introdujeron elementos para facilitar su implementación
  - <header> es uno de esos elementos
- Define un encabezado
  - Es habitual que contenga ayudas de navegación, un logo etc
- Puede haber varios en un mismo documento

```
<!DOCTYPE html>
<head>
</head>
<body>
 <header>
    <div style="background-color:lavenderblush">
       <h1>Título</h1>
       <p>Texto normal (párrafo).</p>
    </div>
 </header>
</body>
</html>
```

CUERPO HTML: AGRUPACIONES - TABLE

- El tag <table> sirve para definir tablas
- Proporcionan una forma de organizar información
- Con <tr> definimos filas
- Con <td> creamos celdas, en las que puedo meter “cualquier cosa”
- Se nos genera la tabla correspondiente de forma automática
- ¿Cómo podemos hacer que la tabla muestre bordes?
  - Probemos con el atributo border

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Tablas</title>
  </head>
  <body>
  <table>
    <tr>
      <td>Alfonso</td>
      <td>Izquierdo</td>
      <td>42</td>
    </tr>
    <tr>
      <td>Juan</td>
      <td>Pérez</td>
      <td>32</td>
    </tr>
  </table>
  </body>
</html>
```

CUERPO HTML: AGRUPACIONES - TABLE

- Podemos definir celdas “cabecera” con el tag <th> (en lugar de <td>)
  - Vemos que el estilo por defecto de los <th> es diferente al de los <td> (centrado y en negrita)
- Podemos incluso fusionar celdas con rowspan y colspan
- Si queremos darles estilos personalizados de forma muy básica (y “deprecada”):
  - Podemos establecer su tamaño de forma manual con width y height
  - Con cellpadding y cellspacing podemos jugar con los espaciados de la tabla

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8">
  <title>Tablas</title>
 </head>
 <body>
 <table border="1" bgcolor="aqua" cellpadding = "5" cellspacing = "5">
  <tr>
   <th>Nombre</th>
   <th>Apellidos</th>
   <th>Edad</th>
  </tr>
  <tr>
   <td>Alfonso</td>
   <td>Izquierdo</td>
   <td>42</td>
  </tr>
  <tr>
   <td bgcolor="#cd853f">Juan</td>
   <td>Pérez</td>
   <td>32</td>
  </tr>
 </table>
 </body>
</html>
```

CUERPO HTML: AGRUPACIONES - TABLE

- Otra posibilidad es definir secciones en las tablas:
  - <thead> encapsula una/s fila/s y las marca como cabecera de la tabla
  - <tbody> define cuerpo de la tabla
  - <tfoot> lo mismo que la cabecera, pero con pie de la tabla
- ¿Qué diferencia hay entre thead y th?
  - th marca una celda específica para que tenga estilo de cabecera, pero no aporta significado jerárquico

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8">
  <title>Tablas</title>
 </head>
 <body>
 <table border="1" bgcolor="aqua">
  <thead>
   <tr>
    <th>Nombre</th>
    <th>Apellidos</th>
    <th>Edad</th>
   </tr>
  </thead>
  <tbody>
   <tr>
    <td rowspan="2">Alfonso</td>
    <td>Izquierdo</td>
    <td>42</td>
   </tr>
   <tr>
    <td bgcolor="#cd853f">Juan</td>
    <td>Pérez</td>
    <td>32</td>
   </tr>
  </tbody>
 </table>
 </body>
</html>
```

CUERPO HTML: LISTA NO ORDENADA

- Definimos una lista no ordenada con el tag
- Cada elemento de la lista será un
- Por defecto, los elementos irán enumerados con pequeños círculos negros

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Tablas</title>
  </head>
  <body>
  <p>Ingredientes:</p>
  <ul>
    <li>1 Yogur natural</li>
    <li>2 cucharadas de aceite</li>
    <li>375gr de harina</li>
  </ul>
  </body>
</html>
```

CUERPO HTML: LISTA ORDENADA

- Definimos una lista ordenada con el tag
- Cada elemento de la lista será un
- Por defecto, los elementos irán enumerados con números

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Tablas</title>
  </head>
  <body>
  <p>Pasos a seguir:</p>
  <ol>
    <li>Formamos una bola de masa con la levadura, la harina y el aceite</li>
    <li>Dejamos reposar la masa durante una hora</li>
    <li>Dividimos la masa en seis porciones y las cocinamos en una sartén</li>
  </ol>
  </body>
</html>
```

CUERPO HTML: FORMULARIOS

- Mediante la etiqueta <form> creamos un formulario en el que el usuario puede introducir datos
- Para cada campo que el usuario tenga que rellenar, añadiremos un elemento <input>
  - Estableciendo su atributo type, elegimos el tipo de input que creamos
  - Algunos ejemplos de type son: text, radio, submit, checkbox, color, date, datetime-local, file, hidden, password, range, reset, submit, week…
- Todos los input tienen que llevar un atributo name, que se enviará al servidor junto al valor seleccionado para ese input
    - Podemos explorar el uso de otros atributos como value, placeholder, min, max, maxlength, readonly o required o bien la posibilidad de marcar un input como checked o disabled

CUERPO HTML: FORMULARIOS

- Podemos añadir <label> para etiquetar los diferentes inputs del form
  - Los label proporcionan características adicionales como la capacidad de que un sistema de accesibilidad lea al usuario la etiqueta de los campos del formulario para que haga su elección
  - Para conectar un label con su input, utilizamos el atributo for, que lo conectará con el input del ese id
- El atributo id de cualquier elemento de html sirve para darle un identificador único
- El atributo name, para nombrar los campos de un formulario y poder identificarlos cuando los enviemos

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8">
  <title>Forms</title>
 </head>
 <body>
  <form>
   <input type="radio" id="musica" name="hobby" value="musica">
   <label for="musica">Música</label><br>
   <input type="radio" id="cine" name="hobby" value="cine">
   <label for="cine">Cine</label><br>
   <input type="submit" id="submit" value="Enviar">
  </form>
 </body>
</html>
```

CUERPO HTML: FORMULARIOS

- ¿Cómo se envían los datos al servidor?
- Los forms tienen muchísima funcionalidad: estamos arañando la superficie
- Más adelante, en la sesión de Bootstrap, los revisitaremos para conocer sus atributos method, target y action

HTML: COMENTARIOS

- Sirven para introducir texto de utilidad para los humanos, que será ignorado por el navegador (y por tanto, no se visualizará cuando se dibuje la web)
  - Útiles para hacer anotaciones sobre el código

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Tablas</title>
  </head>
  <body>
  <p>Esto es un párrafo, será visible en el navegador</p>
<!--  Esto un comentario de una línea, no será visible-->
  <p>Otro párrafo visible.</p>
  <!--  Esto un comentario de varias líneas,
        que continuará hasta que cierre el comentario-->
  <p>Terminamos con otro párrafo.</p>
  </body>
</html>
```

HTML: ENLACES

- Una de las características de la navegación web es la capacidad de seguir enlaces a partes del mismo documento o incluso otros documentos diferentes
- Esto se hace mediante el tag <a>
  - El atributo href especificará el destino de ese enlace
  - Puede ser una URL relativa, absoluta o el identificador de algún elemento del mismo documento

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Tablas</title>
  </head>
  <body>
  <a href="http://www.google.es">Ir a Google</a>
  <a href="#pie">Pie de página</a>
  <h3 id="pie">Esto es el pie de página</h3>
  </body>
</html>
```

HTML: IMÁGENES

- Mediante el tag <img> podemos insertar imágenes en nuestro documento
  - Este tag requiere un atributo src que especifique de dónde sacar la imagen
    - Puede ser una ruta absoluta o relativa
  - Además, mediante el atributo alt especificamos un texto alternativo que mostrará si no puede cargarse la imagen

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Imágenes</title>
  </head>
  <body>
<img src="ok.png" alt="OK" width="16">
<img src="https://www.nationalgeographic.com.es/medio/2020/07/06/un-colibri-gorjirrubi_5d84c2a5_1280x720.jpg" alt="Colibrí" width="16">

  </body>
</html>
```

ENLACES DE INTERÉS

- Validador online de html: https://validator.w3.org
- w3schools: https://www.w3schools.com
- Mozilla Developer: https://developer.mozilla.org/en-US/docs/Learn
- HTML for Beginners: https://html.com