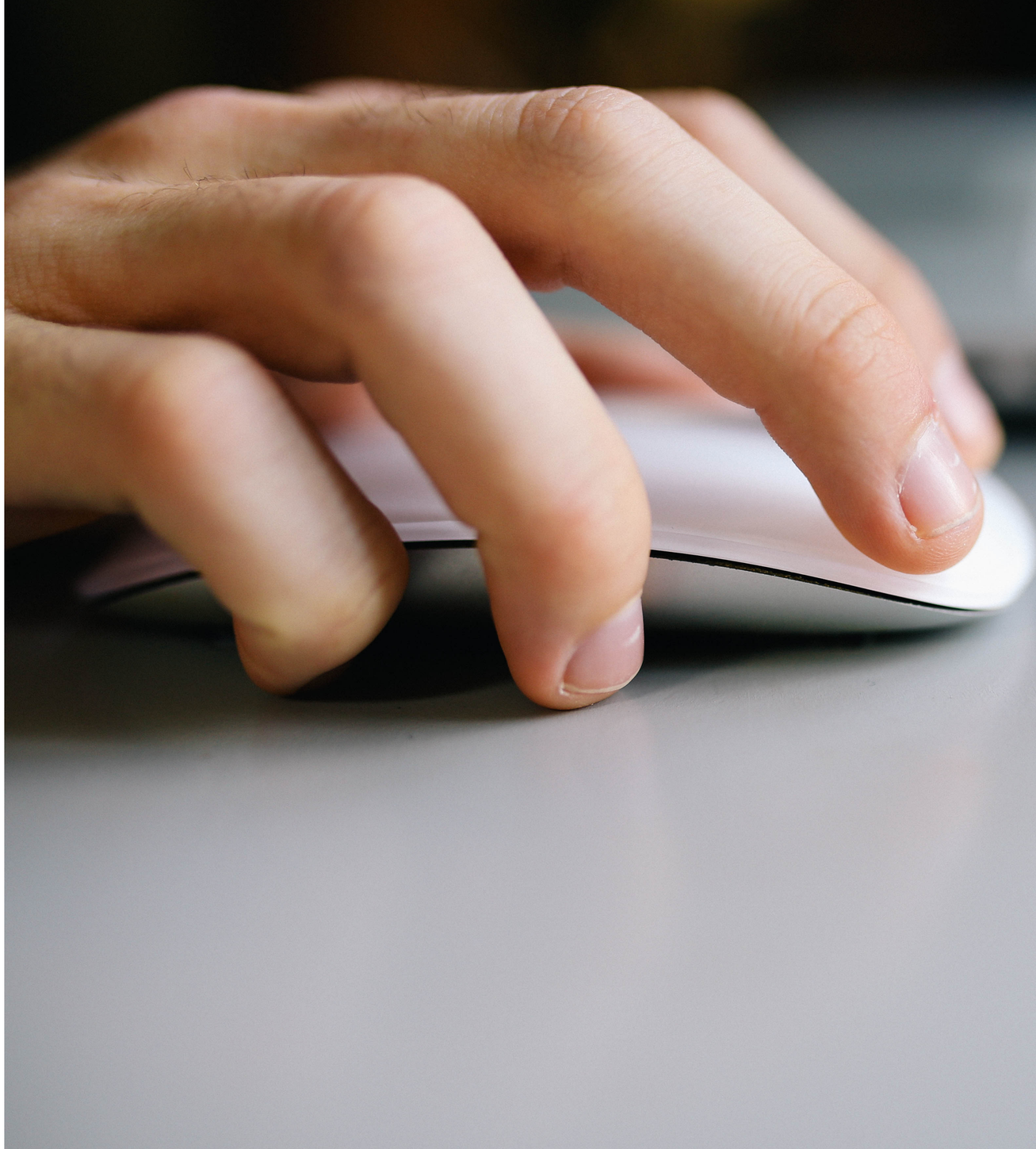
Desarrollo web

Desarrollo web front-end

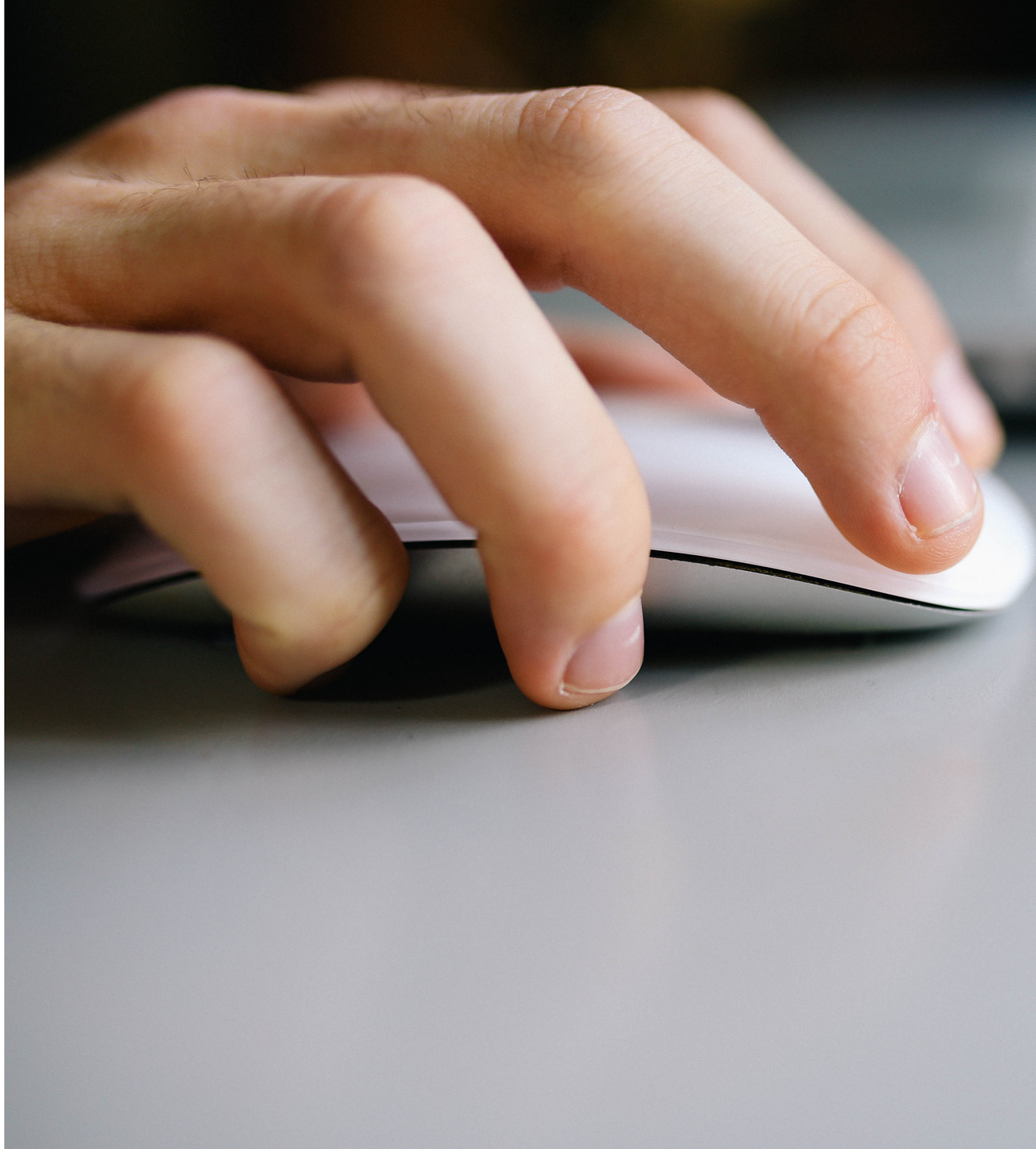
Dirigido a aquellos que quieren aprender a diseñar y construir sitios web desde cero.

Las única herramienta es una computadora con un navegador web y un editor de texto.



Agenda

- Introducción
- HTML
- CSS
- JavaScript
- Librerías JavaScript
- Herramientas de depuración y pruebas

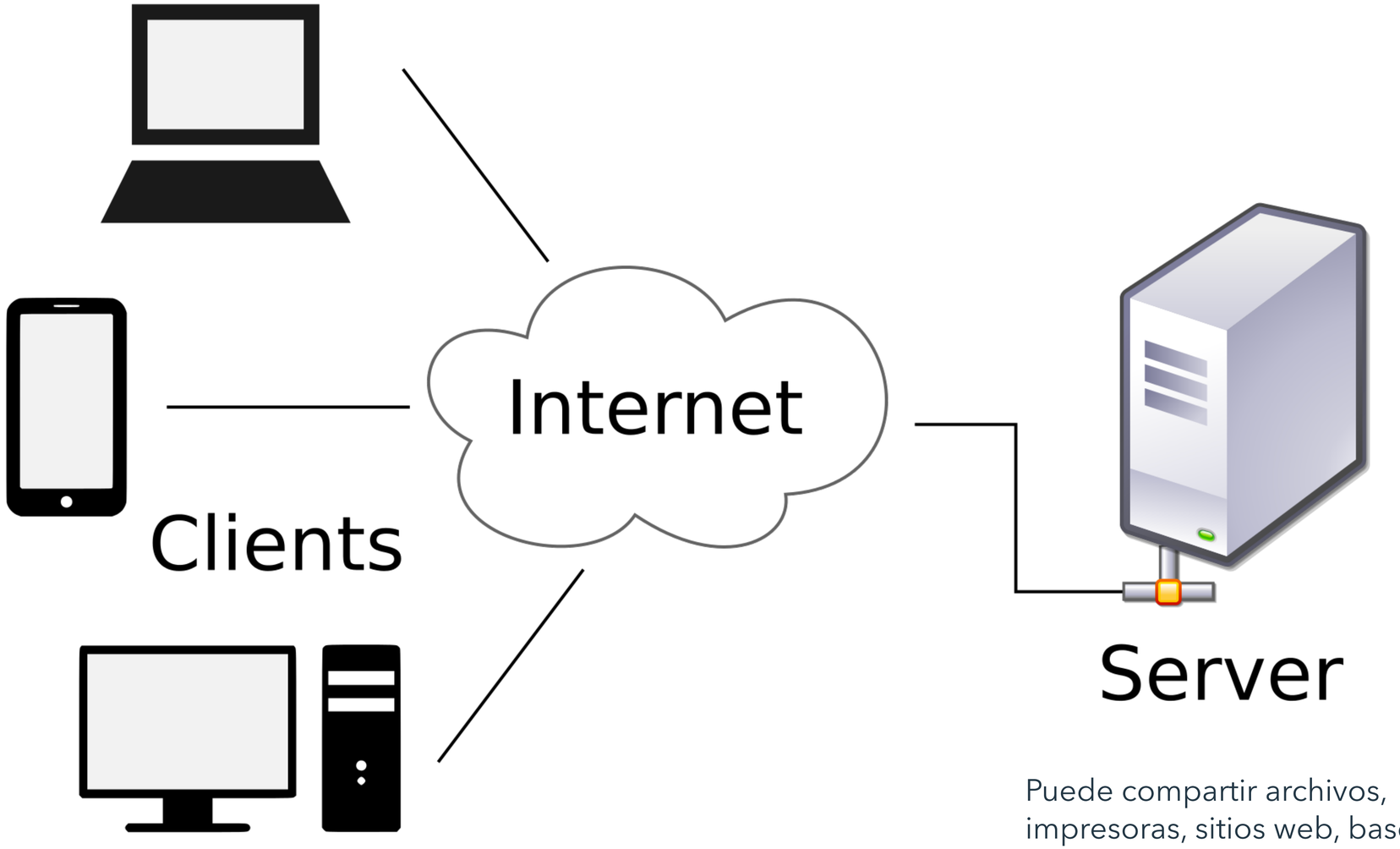


Introducción

Arquitectura cliente - servidor

Es un **modelo de diseño de software** en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes.

Computadoras, tabletas teléfonos u otros servidores pueden fungir como cliente.



Puede compartir archivos, impresoras, sitios web, bases de datos, correo electrónico, etc.

Client–server model
By: Wikipedia

Cómo accesamos a la web

NAVEGADOR WEB

Las personas acceden a los sitios web utilizando un software llamado navegador web.

SERVIDOR WEB

Cuando navegas en internet, las páginas web se solicitan a una computadora especial, el servidor web.

CLIENTES

Accedemos a la web en una gama cada vez mayor de dispositivos que incluyen computadoras, tabletas y teléfonos móviles.

Cómo se crea un sitio web

LO QUE VES

Al visitar un sitio web, el navegador web recibe **HTML** y **CSS** para crear lo que ves; **JavaScript**, para añadir mejoras en la interfaz de usuario; además de **audio** y **vídeo**.

CÓMO SE CREA

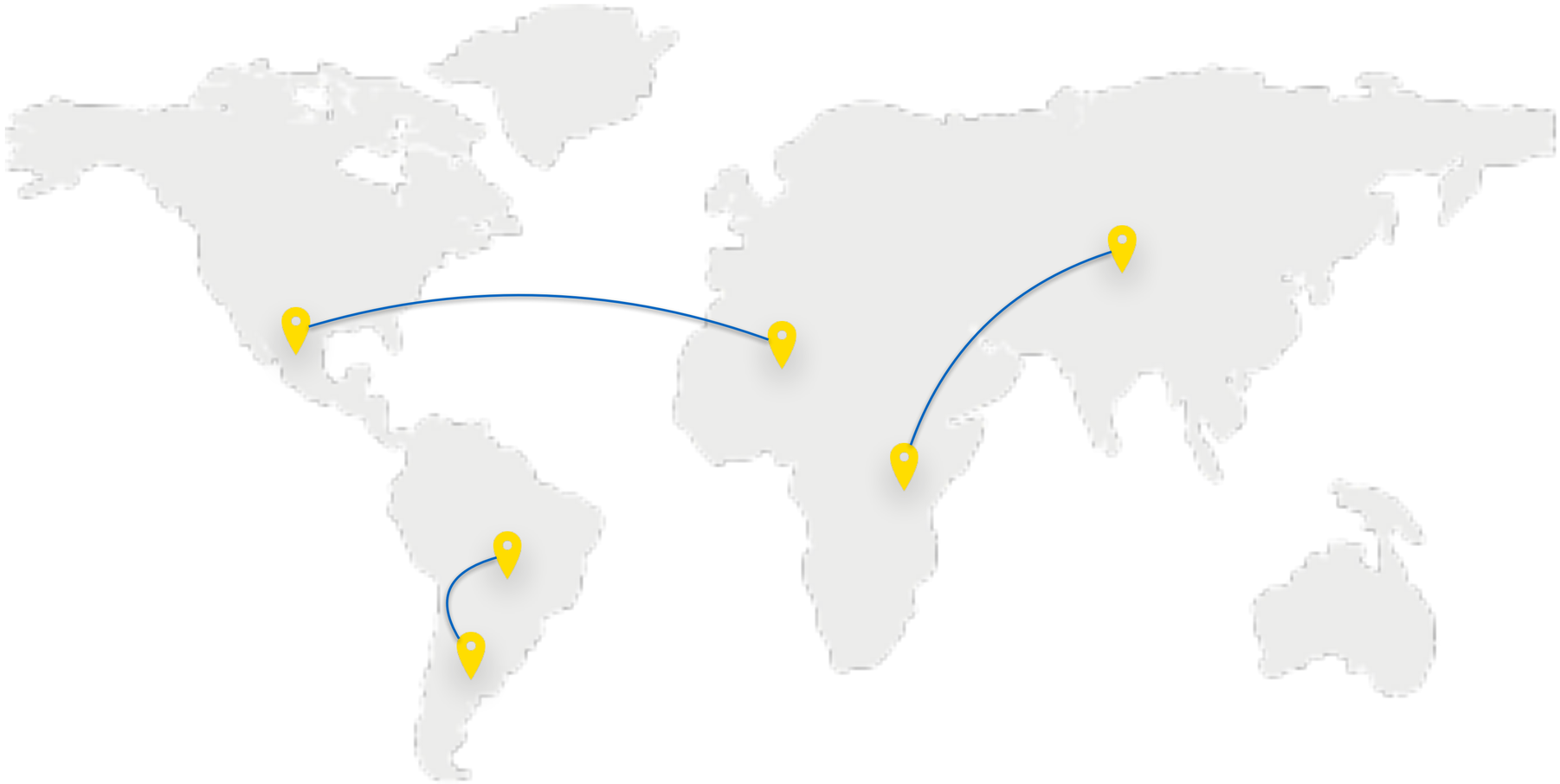
Todos los sitios web usan HTML y CSS, pero los sitios web de redes sociales y las plataformas de comercio electrónico agregan más tecnologías como **PHP** o **Java**.

DISEÑO DE SOFTWARE

Es necesario razonar el propósito del sitio web para definir los **elementos de software** que lo conforman y las **relaciones** entre ellos.

CÓMO FUNCIONA

1. Escribes el **nombre de dominio**.
2. El navegador web contacta una red de servidores (**DNS**) para obtener la IP asociada.
3. La **IP** es entregada al navegador web para contactar al servidor web que la aloja.
4. El servidor web regresa la **página web** solicitada.



Programación por capas

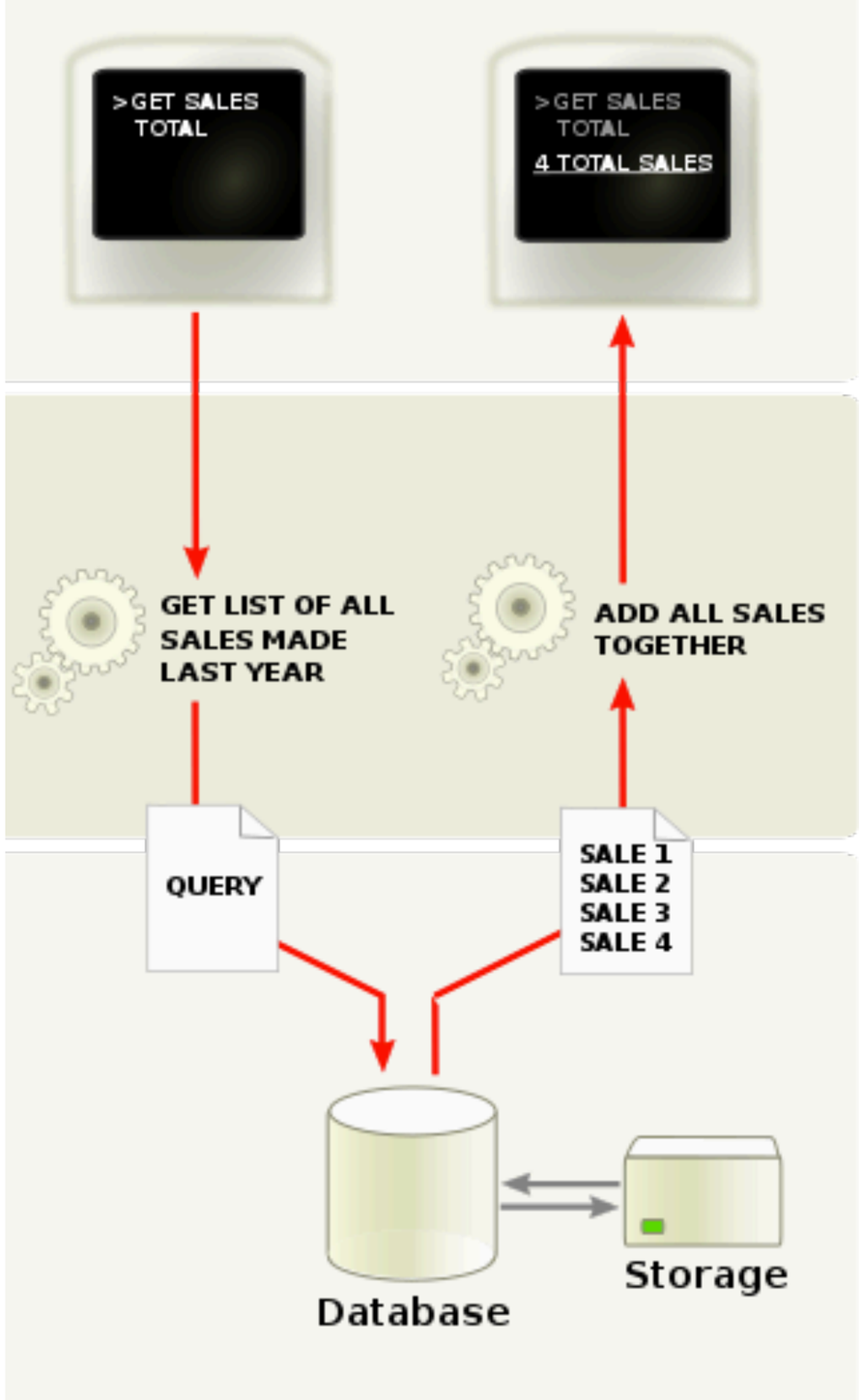
Es una arquitectura cliente - servidor en la que las funciones de presentación, procesamiento de aplicaciones y administración de datos están físicamente separadas. El uso más extendido de esta arquitectura es la arquitectura de tres niveles **Modelo - Vista - Controlador** (MVC).

Arquitectura MVC

CAPA DE PRESENTACIÓN
Proporciona una representación del modelo.

CAPA LÓGICA
Cambia el estado del modelo y coordina la aplicación según la petición del usuario.

CAPA DE DATOS
Mantiene la información, estado y lógica de la aplicación.



Multitier architecture
By: Wikipedia

Front-end y back-end

El **front-end** de una aplicación (capa de presentación) es lo que un humano ve y con lo que éste puede interactuar, encargado de traducir la interacción del usuario a las especificaciones del back-end.

El **back-end** de una aplicación (capas lógica y de datos) trabaja tras bambalinas haciendo útil el front-end. Recibe y otorga respuestas a las peticiones del usuario.

¿Qué se espera de un desarrollador?

FRONT-END

Representar mediante HTML y CSS el diseño de una página web.

Mejorar la experiencia de usuario con JavaScript.

Seguir estándares W3C.

BACK-END

Coordinación de una página web con lenguajes de programación de servidor y bases de datos.

FULL STACK

Conocimiento entre las dos facetas previas.

Capacidad de asumir tareas de diferente índole.

Hacer la vida más fácil para el usuario.

What is a Full Stack developer?
By: Laurence Gellert

HTML

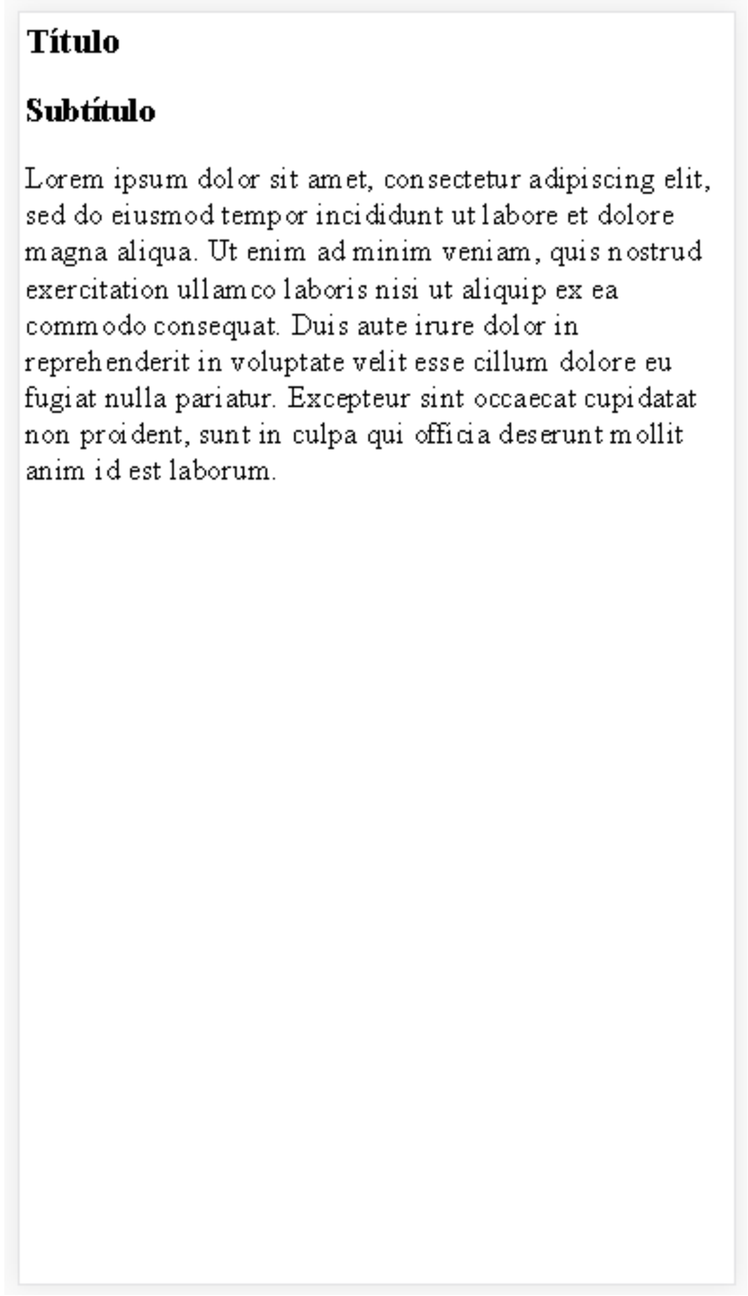
Estructura

Para cualquier tipo de documento, la estructura es muy importante para ayudar a los lectores a entender el mensaje.

Como en un periódico, las páginas web suelen tener un título, algún texto, y posiblemente imágenes, audio o vídeo.

Estructura HTML

```
<html>
  <body>
    <h1>Título</h1>
    <h2>Subtítulo</h2>
    <p>
      Lorem ipsum...
    </p>
  </body>
</html>
```



Etiquetas

APERTURA

Los caracteres entre paréntesis indican el propósito de la etiqueta.

CIERRE

Se caracteriza por una diagonal.

Una elemento HTML comprende la etiqueta de apertura y de cierre.

ATRIBUTOS

Proveen información adicional.

Siempre en la etiqueta de apertura.

Se componen de nombre y valor.

Etiquetas relevantes

- <html>
- <head>
- <title>
- <body>

Texto

Cuando se crea una página web, se utilizan etiquetas (marcas) adicionales para agregar contenido a las páginas.

Para facilitar la lectura del código, se suelen agregar saltos de línea y espacios adicionales. Sin embargo, el navegador web suele hacer un colapso del espacio en blanco.

Marcado

ESTRUCTURAL

Son los elementos que se puede usar para describir tanto los títulos como los párrafos

SEMÁNTICO

Proporciona información adicional; como cuando se pone énfasis en una oración, en una cita, etc.

Etiquetas relevantes

- <h1> - <h6>
- <p>
-
- <hr />

Listas

ORDENADAS

Listas donde cada elemento de la lista está numerado.

DESORDENADAS

Listas que utilizan una viñeta en lugar de un número.

DEFINICIÓN

<dl>

Se componen de un conjunto de términos junto con sus definiciones.

Etiquetas relevantes

-
- <dt>
- <dd>

Enlaces

Los enlaces son una característica importante de la web porque permiten pasar de una página web a otra, lo que permite la idea de navegar.

Nos permiten navegar a otra página web del mismo sitio u otro, incluso a otra parte de la misma página. Pueden ser configurados para abrir el enlace en otra pestaña, o abrir un gestor de correo electrónico y escribirle a alguien.

Escribiendo enlaces

ETIQUETA

<a>

Los enlaces son creados utilizando la etiqueta **<a>**.

CONTENIDO

<a>Clic aquí

El usuario podrá hacer clic en el contenido de la etiqueta.

ATRIBUTO HREF

<a href="...">

Se especifica el destino del enlace utilizando el atributo **href**.

Uniform Resource Locator (URL)

ABSOLUTA

Empieza con el nombre de dominio para el sitio, y puede continuar con la ruta hacia una página específica.

RELATIVA

Son utilizadas para enlazar a una página del mismo sitio. No requieren el nombre de dominio para hacer el enlace.

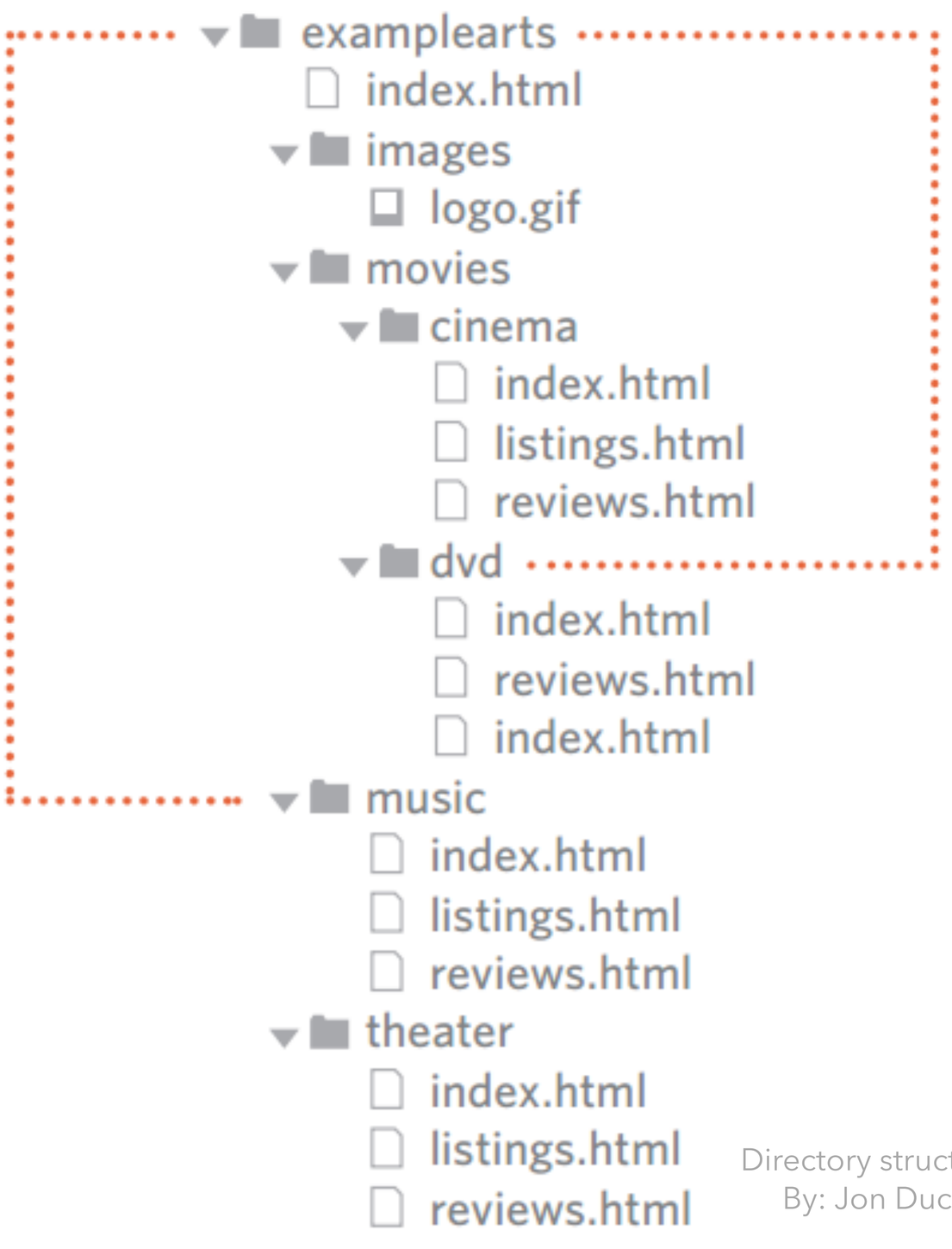
URL Relativa

MISMA CARPETA
Sólo utilizar el nombre del recurso.

SUBCARPETA
Utilizar el nombre de la carpeta,
seguida por una diagonal y, finalmente,
el nombre del archivo.

CARPETA CONTENEDORA
Utilizar **../** para indicar la carpeta
superior de la actual, después el
nombre del archivo.

CARPETA PRINCIPAL
Utilizar **/** para apuntar a raíz.



Directory structure
By: Jon Duckett

Funciones relevantes

- href=“mailto:ex@3ct.mx"
- target="_blank"
- href="#top"
- href=“http://3ct.mx#top“

Imágenes

Hay muchas razones por las que una imagen deba ser añadida a una página web: un logo, una fotografía, un diagrama o una gráfica.

Tres reglas para imágenes:

- Formato adecuado
  (JPEG, + colores; GIF, - colores; PNG, transparencias; SVG, logotipos)
- Correcto tamaño
- Correcta resolución (72 ppi máximo)

Añadiendo imágenes

ETIQUETA

Para añadir una imagen se requiere de la etiqueta **<img>**.

CONTENIDO

El atributo **src** le dice al navegador web dónde encontrar la imagen. Generalmente, en el propio sitio.

Tablas

Algunos tipos de información requieren ser presentados en una tabla. Por ejemplo, los reportes que pueden presentar una gran cantidad de datos complejos.

Formularios

Tradicionalmente, se comprende como un formulario un documento que contiene espacios para proporcionar información.

HTML ofrece este concepto de formulario para recolectar información de los usuarios del sitio.

Protocolo HTTP

HTTP

El protocolo **Hypertext Transfer Protocol** (HTTP) permite las transferencias de información en la World Wide Web (WWW).

MÉTODO GET

- Datos anexados a la URL.
- Predeterminado.
- Hasta 2 000 caracteres.

MÉTODO POST

- Datos anexados a la solicitud HTTP.
- No hay límites de tamaño.
- Permite carga de archivos.

Métodos de petición

```
/test/demo_form.asp?name1=value1&name2=value2
```

```
POST /test/demo_form.asp HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

HTTP Methods: GET vs. POST
By: W3Schools

Estructura de formularios

DÓNDE

<form action="server.do">

El atributo **action** se utiliza para definir a dónde enviar la información.

CÓMO

<form method="get">

Mediante el atributo **method** se indica cómo se enviará la información.

Cómo funciona

INFORMACIÓN

username=“John”

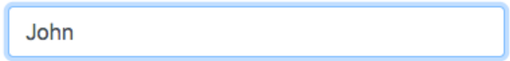
La información viaja en parejas **clave - valor**.

CLAVE

<input name=“username” />

Cada elemento de formulario debe tener al atributo **name** que representa la clave.

VALOR



El valor será otorgado por el usuario, o por valores definidos por el desarrollador (atributo **value**).

Controles de texto

TEXTO

<input type="text" />

Se usa para una sola línea de texto, como direcciones de correo electrónico y nombres.

CONTRASEÑA

<input type="password" />

Recolecta información del usuario, pero no la muestra.

ÁREA DE TEXTO

<textarea>

Para grandes áreas de texto multilínea.

Controles de selección

BOTONES DE RADIO

<input type="radio" />

Para cuando un usuario debe seleccionar una de muchas opciones.

CASILLAS DE VERIFICACIÓN

<input type="checkbox" />

Para cuando un usuario debe seleccionar y deseleccionar una o más opciones.

LISTAS DESPLEGABLES

<select>

Para cuando un usuario debe seleccionar una de muchas opciones de una lista.

Otros controles

BOTÓN DE ENVIAR

```
<input type="submit">
```

Para cuando se quiere enviar la información recolectada al servidor.

ARCHIVOS

```
<input type="file" />
```

Permite a los usuarios enviar archivos al servidor.

CSS

Selectores CSS

Cascading Style Sheets (CSS) tiene como propósito crear páginas web más atractivas controlando el diseño de éstas.

CSS permite crear reglas para especificar cómo el contenido de un elemento debe ser mostrado.

Reglas de estilo

SELECTOR

p { … }

El **selector** indica a qué elemento aplica la regla.

REGLAS

p { color: “red” }

La **regla** indica cómo el elemento referido debe ser mostrado.

PROPIEDADES

color: “red”

La **propiedad** indica qué aspecto de los elementos se van a cambiar.

El **valor** es la configuración usada para la propiedad.

CSS Externo

ETIQUETA

<link />

La etiqueta **<link>** puede ser usada dentro de un documento HTML para importar un archivo CSS.

ATRIBUTO *rel*

rel="stylesheet"

El atributo **rel** especifica la relación entre la página HTML y el archivo enlazado.

ATRIBUTO href

href="…"

href especifica la ruta al archivo CSS.

CSS Interno

ETIQUETA

<style>

También se puede especificar reglas CSS colocándolas dentro de la etiqueta **<style>**.

RECOMENDACIÓN

Mantener el código CSS en el mismo archivo HTML es una mala práctica.

Se recomienda utilizar un archivo CSS externo.

Selectores CSS

UNIVERSAL

```
* { ... }
```

Aplica a **todos** los elementos en el documento.

TIPO

```
p { ... }
```

Aplica a todos los **elementos** de la etiqueta indicada.

CLASE

```
.title { ... }
```

Aplica a cualquier elemento cuyo atributo **class** tiene un valor que coincide con el especificado.

Selectores CSS

IDENTIFICADOR

#name { ... }

Aplica a un elemento cuyo atributo **id** coincide con el especificado.

DESCENDIENTE

div a { ... }

Aplica a un elemento que es **descendiente** de otro.

MULTISELECTOR

h1, .h2, #h3 { ... }

Aplica a todos los elementos listados.

Por qué Hojas de Estilo en Cascada

ÚLTIMA REGLA

Si dos selectores son idénticos, el último de los dos prevalecerá.

ESPECIFICIDAD

Si un selector es más específico que otro, éste tendrá mayor jerarquía.

IMPORTANCIA

Añadir **!important** al final de una regla indicará que ésta es más importante que otras.

HERENCIA

Algunas propiedades aplicadas en elementos padres serán heredadas por sus hijos y otras no.

Se puede inducir este comportamiento con **inherit**.

Color

color

color = “red”

La propiedad **color** permite especificar el color de texto de un elemento.

background-color

background-color = “red”

Puede indicar el color de fondo del elemento mediante **background-color**.

Color

VALORES RGB

rgb(100, 100, 90)

Expresa el color en términos de qué tanto rojo, verde y azul.

El rango aceptable es 0 - 255.

CÓDIGO HEXA

#EE3E80

Es un código de seis dígitos que representan colores RGB cada dos dígitos.

El rango es 0 - 9 y A - F.

NOMBRES

red

Aproximadamente 150 colores diferentes de los cuales elegir.

Texto

FAMILIA

font-family

Especifica las posibles fuentes de un texto.

Si el navegador no soporta primera, elegirá una segunda, etc.

PESO

font-weight

Permite añadir énfasis a un texto.

Valores como **light** y **bold** son válidos.

ESTILO

font-style

Especifica el estilo de un texto.

Valores como **normal** e **italic** son válidos.

Texto

TRANSFORMACIÓN

text-transform

Se usa para cambiar a mayúsculas o minúsculas un texto.

Valores como **uppercase**, **lowercase** y **capitalize** son válidos.

DECORACIÓN

text-decoration

Permite añadir un estilo adicional al texto.

Valores como **none**, **underline** y **line-through** son válidos.

ALTURA

line-height

Indica el espacio vertical entre las líneas de texto. Éste se especifica en unidades como **pixeles**, **porcentajes** o **ems**.

Texto

ALINEACIÓN

text-align

Permite controlar la alineación horizontal de un texto.

Valores como **left**, **right**, **center** y **justify** son válidos.

ALINEACIÓN

vertical-align

Permite controlar la alineación vertical de un texto.

Valores como **bottom**, **middle** y **top** son válidos.

Selectores relevantes

- :first-letter
- :first-line
- :link
- :visited
- :hover
- :active
- :focus

Cajas

ANCHO

width

La propiedad **width** permite especificar el ancho de un elemento.

Éste se especifica en unidades como **pixeles**, **porcentajes** o **ems**.

ALTO

height

La propiedad **height** permite especificar el ancho de un elemento.

Cajas

BORDE

border

Cada caja tiene un borde, incluso si éste no se ve.

Éste se especifica en unidades como **pixeles**, **porcentajes** o **ems**.

MARGEN

margin

Se refiere al espacio ente un elemento y otro a partir del borde.

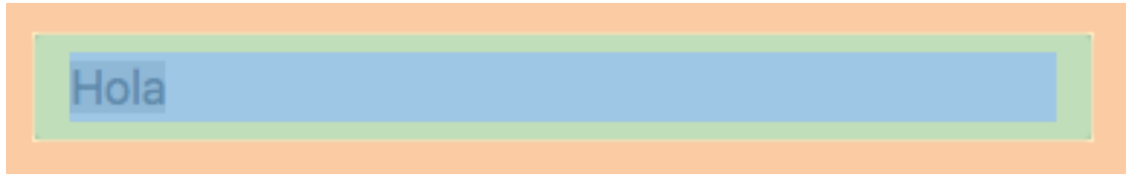
MARGEN INTERNO

padding

Se refiere al espacio entre el contenido de la caja y el borde.

Cajas

- content
- background-color
- padding
- border
- margin



Visibilidad

DESPLEGAR

display

Controla el comportamiento de la caja.

Valores como **inline** y **block** son válidos.

ALINEACIÓN

vertical-align

Permite controlar la alineación vertical de un texto.

Valores como **bottom**, **middle** y **top** son válidos.

Visibilidad

EN LÍNEA

display: inline

Permite que un elemento no tenga margen y comparta la línea donde se encuentra con otros.

EN BLOQUE

display: block

Obliga a que el margen de un elemento crezca tanto como el elemento padre. **Impide** a que más un elemento se muestre en la misma línea.

NINGUNA

display: none

El elemento no se mostrará.

Propiedades relevantes

- border-width
- border-style
- border-color
- box-shadow
- Border-radius

JavaScript & jQuery

JavaScript

JavaScript puede ser utilizado en los navegadores web para hacer a los sitios web más interactivos, interesantes y amigables.

JavaScript accede y modifica el contenido usado en una página web sin necesidad de editar el código HTML original.

Es el **único lenguaje de programación** que trabaja en el front-end.

¿Cómo funciona?

ACCESAR

Con JavaScript puede **seleccionar** un elemento, atributo o texto HTML.

MODIFICAR

Puede **añadir** elementos, atributos y texto a la página, o **remover** otros ya existentes.

PROGRAMAR

Especificar una **serie de pasos** para que el navegador web los siga (como una receta) y así accesar y modificar el contenido.

REACCIONAR

Puede indicar la ejecución de un script que debería ejecutarse tras un **evento** específico.

Ejemplos JavaScript

- Galería de imágenes
- Validación de formularios
- Recargar parte de una página
- Filtrar información

jQuery

Al contrario de JavaScript, no es un lenguaje de programación, sino que se trata de una **biblioteca** JavaScript.

Características principales:

- Selectores simples
- Tareas comunes con menos código
- Compatibilidad entre navegadores web

HTML, CSS y JavaScript

CAPA DE CONTENIDO

< html >

El HTML proporciona la estructura de la página web y añade la semántica.

Archivos **.html**.

CAPA DE PRESENTACIÓN

{ css }

CSS mejora la presentación del HTML con reglas que lo modifican.

Archivos **.css**.

CAPA DE COMPORTAMIENTO

javascript ()

JavaScript es quien cambia el cómo la página se comporta añadiendo interactividad.

Archivos **.js**.

JS Interno y Externo

ETIQUETA

<script>

La etiqueta **<script>** es usada dentro de un documento HTML para importar un archivo o definir código.

Se ejecuta donde ésta se encuentre en el HTML.

ATRIBUTO *src*

src="..."

El atributo **src** especifica la ubicación del archivo enlazado.

Sólo se requiere para JS externo.

CONTENIDO

<script></script>

El contenido de la etiqueta puede estar o no dependiendo de si se usa JS interno o externo.

La etiqueta de cierre no es opcional.

Objetos y métodos

OBJETO

document

El objeto **document** representa a la página web entera.

MIEMBROS

.write()

Se accede a **funciones** y **propiedades** con el operador **.** (punto) .

PARÁMETROS

.write("Hola")

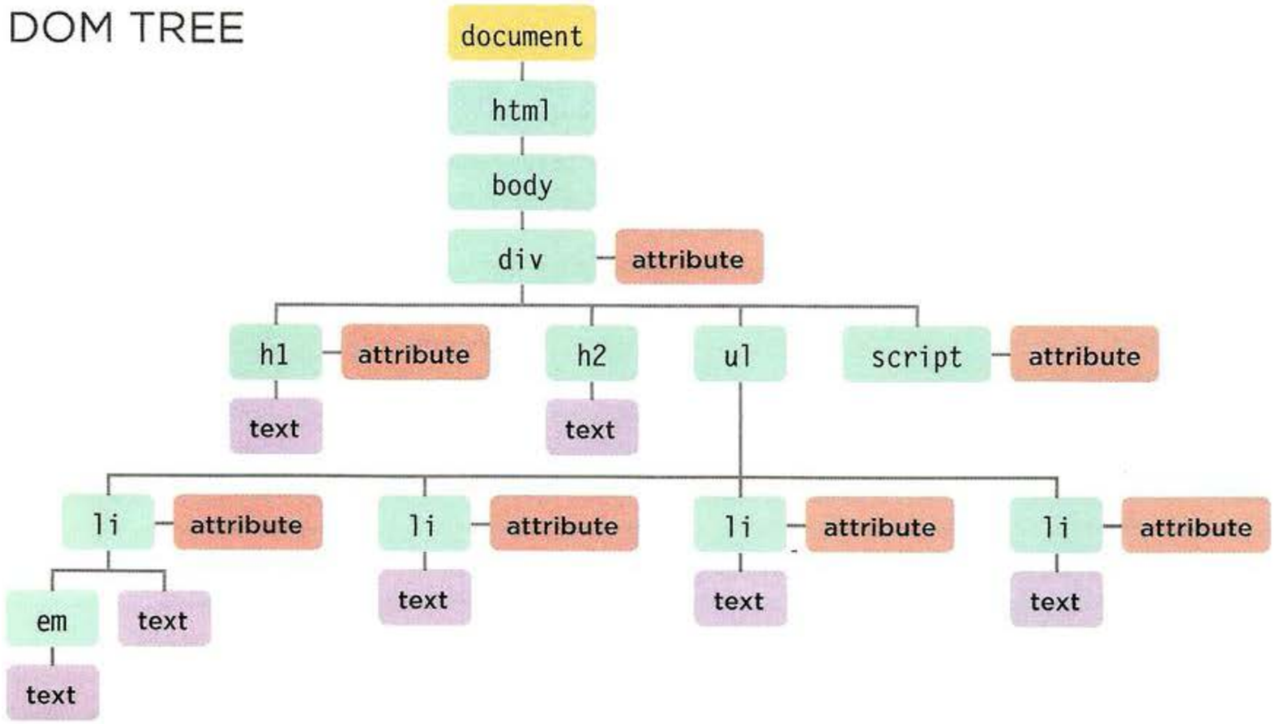
Algunos métodos requieren información para trabajar correctamente. Tal información se proporciona dentro de los paréntesis.

Document Object Model (DOM)

El DOM especifica cómo los navegadores web deberían crear un **modelo** (árbol DOM) de la página HTML, y cómo JavaScript puede acceder y modificar el contenido de ésta mientras está en la ventana del navegador.

El DOM es un objeto con métodos y propiedades (**Application Programming Interface**, **API**) que permiten realizar las funciones de JavaScript.

```
<html>
  <body>
    <div id="page">
      <h1 id="header">List</h1>
      <h2>Buy groceries</h2>
      <ul>
        <li id="one" class="hot"><em>fresh</em> figs</li>
        <li id="two" class="hot">pine nuts</li>
        <li id="three" class="hot">honey</li>
        <li id="four">balsamic vinegar</li>
      </ul>
      <script src="js/list.js"></script>
    </div>
  </body>
</html>
```



Búsqueda de los elementos

INDIVIDUAL

$("#id")

Accede al valor de in elemento a través de su atributo **id**.

MÚLTIPLE

$(".class")

Selecciona a todos los elementos que tengan la **clase** especificada.

BÚSQUEDA

$(...).find(...)

Puede movilizarse a través del árbol DOM de un elemento (**nodo**) a otro.

- find(), closest()
- children(), siblings()

Manipulación de los elementos

TEXTO

<a>Enlace

El texto entre las etiquetas es almacenado en el **nodo texto**.

- text()
- val()

CONTENIDO HTML

...

Si el contenido entre las etiquetas es código HTML, éstos son **elementos nodo** en el árbol DOM y no texto.

- html()
- prepend(), append()
- remove(), empty()

ATRIBUTO

<a href="...">

El contenido de un atributo es almacenado en el **nodo atributo**.

- attr(), removeAttr()
- addClass(), removeClass()
- css()

Eventos

Al navegar en internet, el navegador web detecta diferentes tipos de eventos y el código puede responder a éstos.

JavaScript representa un evento mediante el objeto Event; con **target** y **type** como sus principales propiedades, y **preventDefault()** como su método más importante.

Tipos de eventos

INTERFAZ GRÁFICA

Ocurren cuando el usuario interactúa con la interfaz gráfica (User Interface, UI).

- load
- resize
- scroll

TECLADO

Ocurren cuando el usuario interactúa con el teclado.

- keydown
- keyup
- keypress

RATÓN

Ocurren cuando se interactúa con un ratón o una pantalla táctil.

- click
- dblclick
- mousedown
- mouseup
- mousemove
- mouseover
- mouseout

Tipos de eventos

FOCO

Ocurren cuando un elemento gana o pierde el foco.

- focus
- blur

FORMULARIO

Ocurren cuando el usuario interactúa con un elemento de formulario.

- change
- submit
- cut, copy, paste
- select

Escuchar eventos

SELECCIONAR NODO

```
$( "#id" )
```

Seleccionar el **elemento** al que se pretende asociar un escuchador de eventos.

ESPECIFICAR EVENTO

```
$( ... ).on( "click" )
```

Indicar qué **evento** se estará escuchando el nodo seleccionado.

INVOCAR CÓDIGO

```
$( ... ).on( ..., f() )
```

Indicar mediante una **función** el código que se ejecutará cuando el evento se presente.

AJAX & JSON

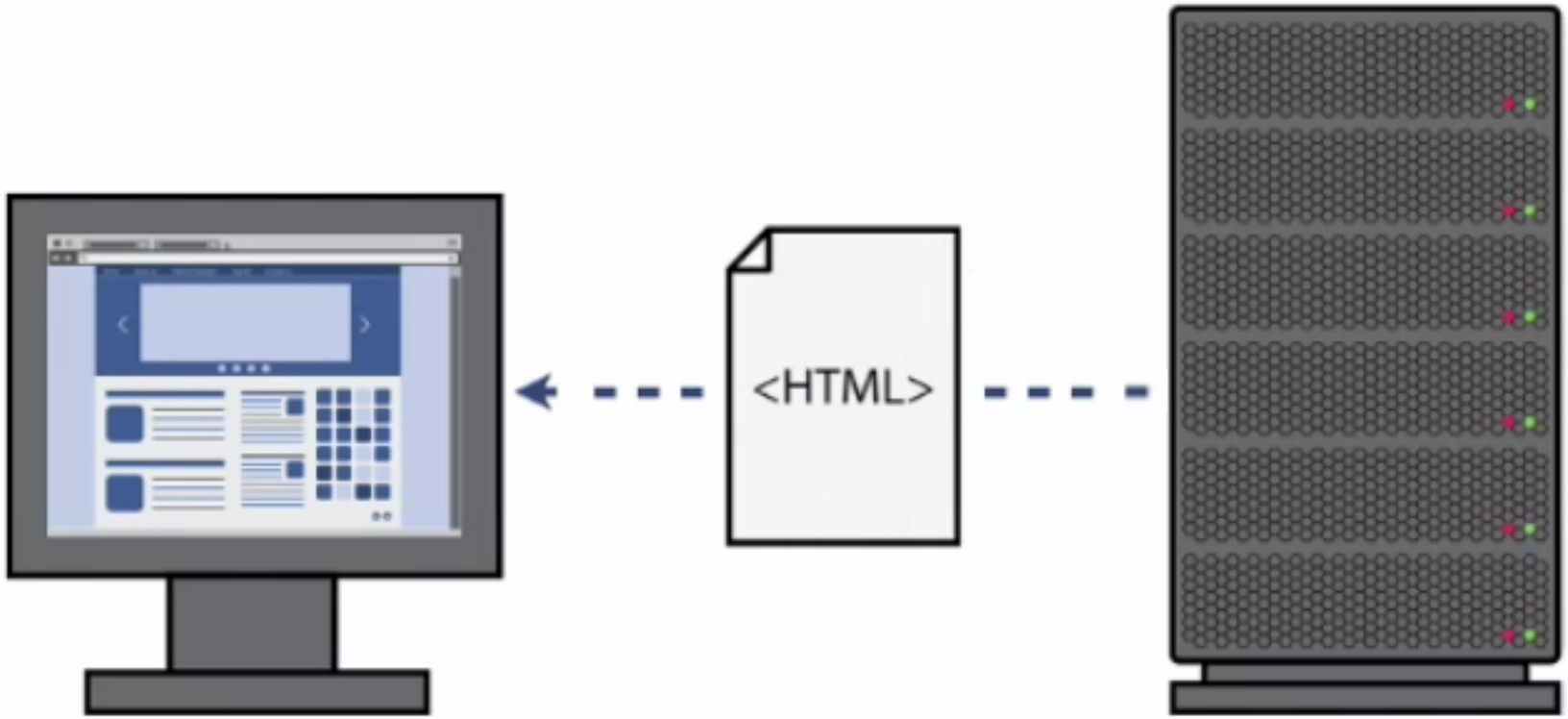
AJAX

Asynchronous **J**avaScript **A**nd **X**ML (AJAX) permite actualizar a las páginas web con información proveniente del servidor sin tener que volver a ser cargadas.

AJAX no es un lenguaje de programación, sino una combinación entre el objeto XMLHttpRequest, JavaScript y el DOM HTML.

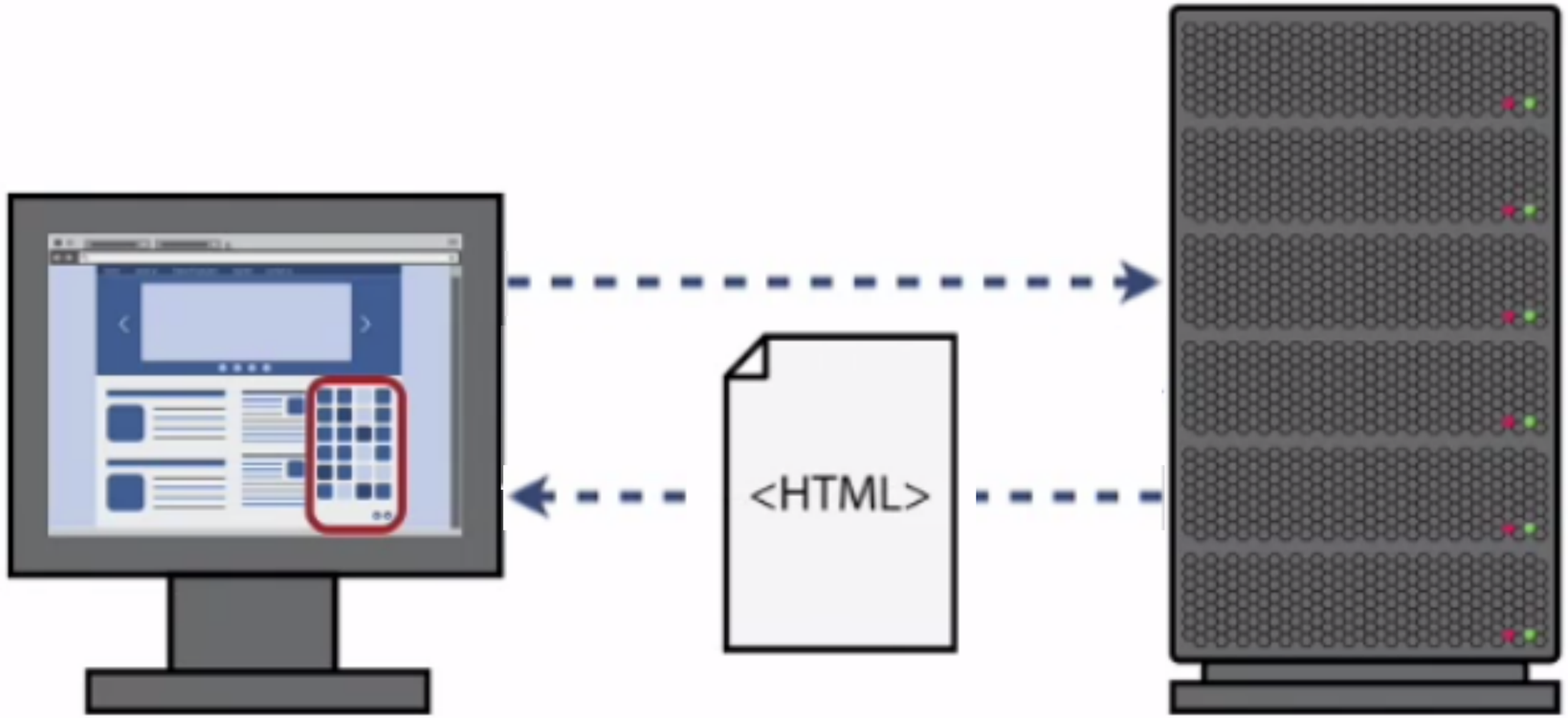
¿Cómo funciona?

- Un navegador web solicita una página
- El servidor responde



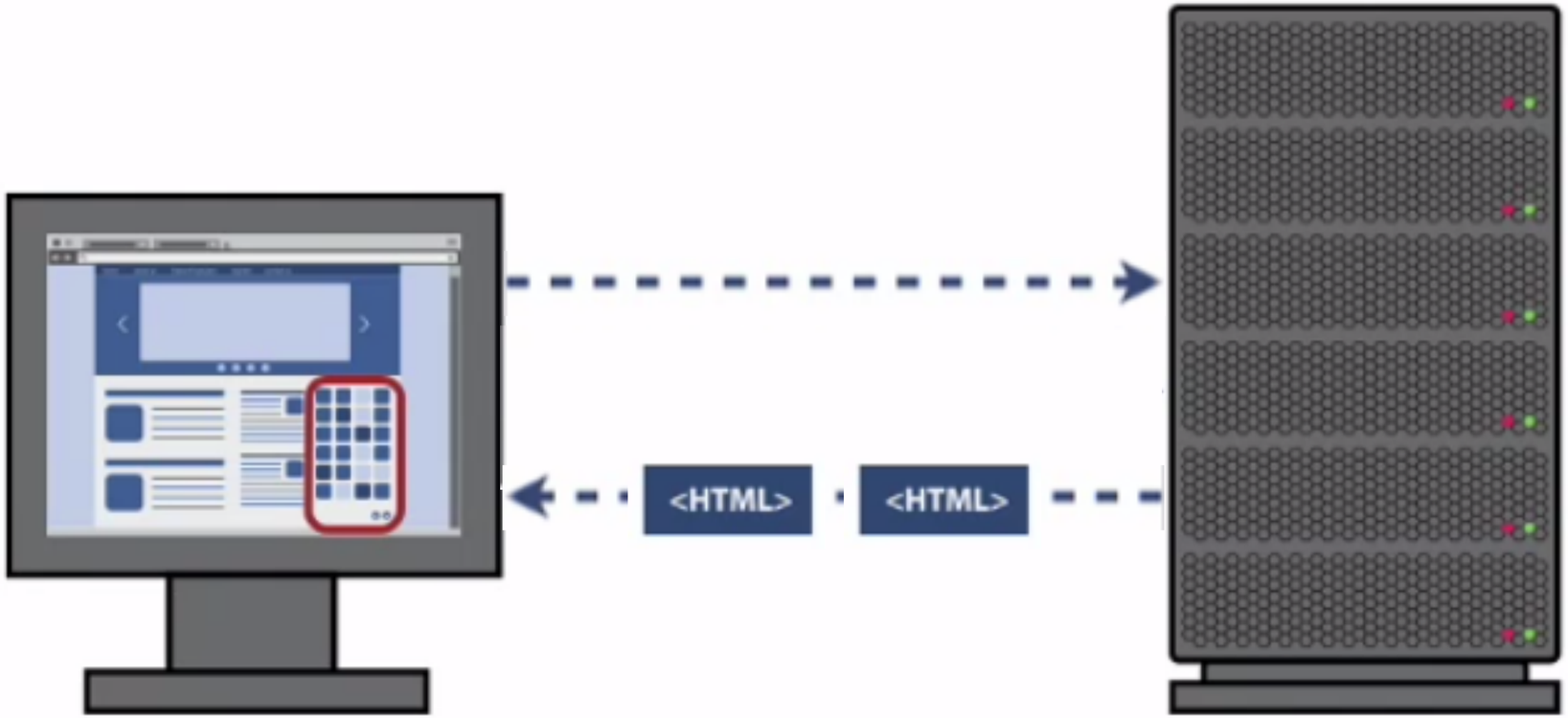
Sin AJAX

- Un navegador web hace una solicitud al servidor
- El servidor responde con una nueva página completa
- Recarga información innecesaria



Con AJAX

- El servidor responde sólo con la información necesaria



Con AJAX

- Comunicación a través de XMLHttpRequest

AJAX

- No es XML
- Formato de la información:
  - HTML
  - XML
  - JSON

JSON

Java**S**cript **O**bject **N**otation (JSON) es texto escrito con la notación de objetos JavaScript para almacenar e intercambiar información.

Beneficios respecto a XML y HTML:

- Puede ser invocado desde cualquier dominio (JSON Padding, JSONP)
- Conciso

JSON

- Parejas clave - valor
- Claves entre comillas dobles
- Valores: string, number, boolean, array, object, null

```
{
  “nombre”: "Juan Pérez”,
  “edad” : 25,
  “mexicano” : true
}
```

```
{
  “personas" : [
    {
      “nombre" : "Juan Pérez",
      “edad” : 25,
      “mexicano" : true
    },
    {
      "nombre" : "Pedro López",
      “edad” : 28,
      “mexicano" : false
    }
  ]
}
```

AJAX con jQuery

jQuery provee de métodos para realizar peticiones AJAX que constan de dos pasos: realizar la petición y manejar la respuesta.

- .load()
- $.get()
- $.post()
- $.getJSON()
- $.getScript()
- $.ajax()

Otros métodos

- .serialize()
- .serializeArray()