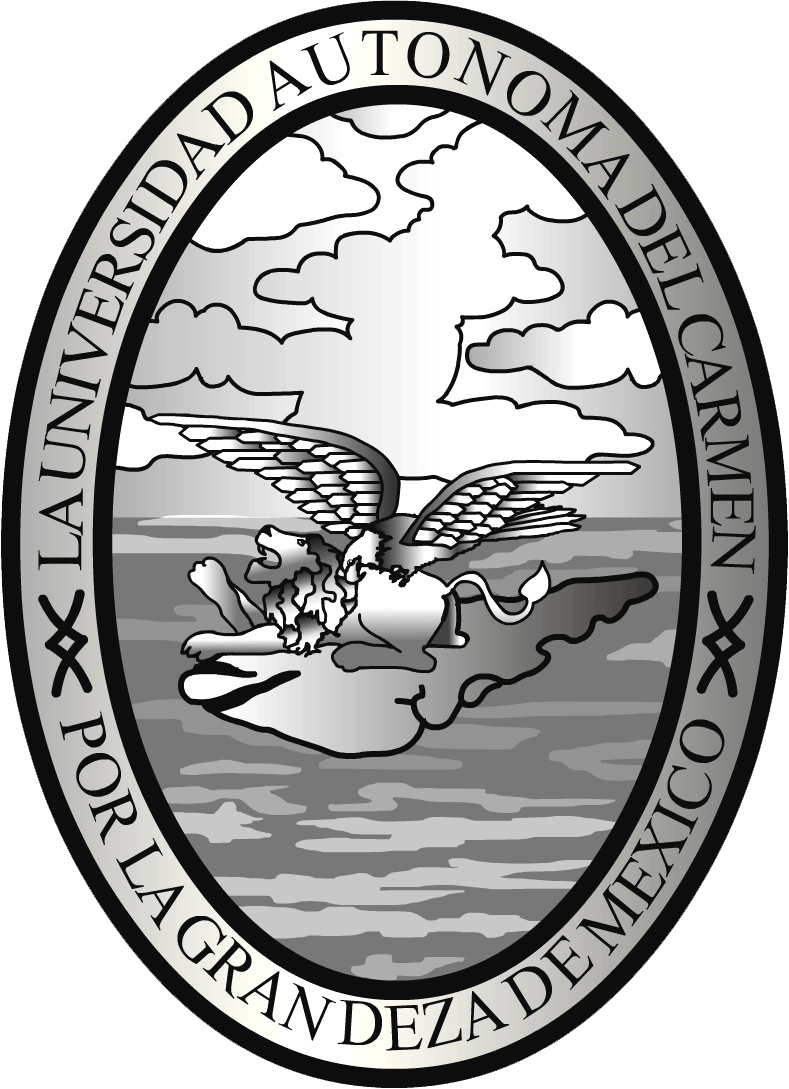
**UNIVERSIDAD AUTONOMA DEL CARMEN**

Facultad de Ciencias de la información

Academia de programación

**MANUAL DE PRÁCTICAS**

**“Aplicaciones con JS”**

**DESARROLLADO Y APROBADO POR LA ACADEMIA DE PROGRAMACIÓN**

**septiembre 2020**

Versión 1.0

**ELABORADO POR**

**MI. Jesús Alejandro Flores Hernández**

**MCC. Damaris Pérez Cruz**

**MCC. José Ángel Pérez Rejón**

**MCC. Juan Carlos Canto Rodríguez**

# INDICE DE TEMAS Y PRÁCTICAS

|  |  |  |
| --- | --- | --- |
| **Tema del curso** | **Práctica** | **Pag.** |
| **Plataforma de trabajo** | Práctica I | 4 |
| **HTML** | Práctica II | 10 |
| **CSS** | Práctica III | 21 |
| **BootStrap** | Práctica IV | 35 |
| **JS** | Práctica V | 43 |
| **MySql** | Practica VI | 73 |
| **PHP** | Practica VII | 80 |

**Contenido**

[INDICE DE TEMAS Y PRÁCTICAS 2](#_Toc50286534)

[INTRODUCCIÓN 5](#_Toc50286535)

[PRACTICA I. Plataforma de trabajo 5](#_Toc50286536)

[I.1. Objetivo 5](#_Toc50286537)

[I.2. Equipo 5](#_Toc50286538)

[I.3. Materiales 5](#_Toc50286539)

[I.4. Descripción 5](#_Toc50286540)

[I.5. Procedimiento 10](#_Toc50286541)

[I.6. Prácticas 10](#_Toc50286542)

[PRACTICA II. HTML 10](#_Toc50286543)

[II.1. Objetivo 10](#_Toc50286544)

[II.2. Equipo 10](#_Toc50286545)

[II.3. Materiales 10](#_Toc50286546)

[II.4. Descripción 10](#_Toc50286547)

[1 Tags HTML 11](#_Toc50286548)

[2 Encabezado 11](#_Toc50286549)

[3 Cuerpo <body> 13](#_Toc50286550)

[4 Atributos 13](#_Toc50286551)

[4.1 Atributos id, name y class 14](#_Toc50286552)

[5 Agrupadores 15](#_Toc50286553)

[5.1 <div> 15](#_Toc50286554)

[5.2 <span> 15](#_Toc50286555)

[5.3 <form> 16](#_Toc50286556)

[5.3.3 Input 20](#_Toc50286557)

[II.5. Procedimiento 21](#_Toc50286558)

[II.6. Prácticas 21](#_Toc50286559)

[PRACTICA III. CSS 21](#_Toc50286560)

[III.1. Objetivo 21](#_Toc50286561)

[III.2. Equipo 21](#_Toc50286562)

[III.3. Materiales 22](#_Toc50286563)

[III.4. Descripción 22](#_Toc50286564)

[1 Sintaxis CSS 22](#_Toc50286565)

[2 Almacenamiento de reglas 24](#_Toc50286566)

[3 Definiendo reglas 24](#_Toc50286567)

[4 Selectores 25](#_Toc50286568)

[III.5. Procedimiento 35](#_Toc50286569)

[III.6. Prácticas 35](#_Toc50286570)

[PRACTICA IV. BootStrap 35](#_Toc50286571)

[IV.1. Objetivo 35](#_Toc50286572)

[IV.2. Equipo 35](#_Toc50286573)

[IV.3. Materiales 36](#_Toc50286574)

[IV.4. Descripción 36](#_Toc50286575)

[IV.5. Procedimiento 42](#_Toc50286576)

[IV.6. Prácticas 42](#_Toc50286577)

[PRACTICA V. JS 43](#_Toc50286578)

[V.1. Objetivo 43](#_Toc50286579)

[V.2. Equipo 43](#_Toc50286580)

[V.3. Materiales 43](#_Toc50286581)

[V.4. Descripción 43](#_Toc50286582)

[Introducción: 43](#_Toc50286583)

[1 Java Script y HTML. 43](#_Toc50286584)

[2 Donde codificar el JS. 45](#_Toc50286585)

[3 Entrada Salida 46](#_Toc50286586)

[3 Entrada JS 48](#_Toc50286587)

[4 Variables, tipos y expresiones. 48](#_Toc50286588)

[5 Funciones. 50](#_Toc50286589)

[6 Eventos. 53](#_Toc50286590)

[7 Objetos (opcional). 59](#_Toc50286591)

[8 prototipos de objetos (opcional). 60](#_Toc50286592)

[8.1 Creando Objetos 60](#_Toc50286593)

[8.2 Agregar propiedades y métodos a objetos. 61](#_Toc50286594)

[8.3 Agregando propiedades y métodos al prototipo 62](#_Toc50286595)

[9 Formato JSON 63](#_Toc50286596)

[10 Recorre JSON 65](#_Toc50286597)

[11 Acceso al DOM 69](#_Toc50286598)

[V.5. Procedimiento 73](#_Toc50286599)

[V.6. Prácticas 73](#_Toc50286600)

[PRACTICA VI. Mysql 73](#_Toc50286601)

[VI.1. Objetivo 73](#_Toc50286602)

[VI.2. Equipo 73](#_Toc50286603)

[VI.3. Materiales 74](#_Toc50286604)

[VI.4. Descripción 74](#_Toc50286605)

[VI.5. Procedimiento 80](#_Toc50286606)

[VI.6. Prácticas 80](#_Toc50286607)

[PRACTICA VII. PHP 80](#_Toc50286608)

[VII.1. Objetivo 80](#_Toc50286609)

[VII.2. Equipo 80](#_Toc50286610)

[VII.3. Materiales 81](#_Toc50286611)

[VII.4. Descripción 81](#_Toc50286612)

[1 Introducción. 81](#_Toc50286613)

[2 Bases de datos 81](#_Toc50286614)

[3 Conexión a la base de datos. 85](#_Toc50286615)

[4 JS-fetch 89](#_Toc50286616)

[5 Insertar registro con fetch 92](#_Toc50286617)

[6 Recibir datos con fetch desde php 96](#_Toc50286618)

[7 Clases en PHP (tema opcional) 98](#_Toc50286619)

[VII.4. Prácticas 101](#_Toc50286620)

[FUENTES CONSULTADAS 103](#_Toc50286621)

# INTRODUCCIÓN

En este manual de prácticas se realizarán programas que usan técnicas de programación y plataformas web, en la solución de diferentes problemas que permiten al alumno desarrollar sus habilidades como programador WEB.

Este manual esta pensado para el uso de estudiantes de programación avanzada.

El paradigama de programación que se usa en este material esa el de programación estructurada.

Para este material el alumno, deberá tener conocimientos de los lenguajes como los de:

* Programación estructurada
* Programación orientada a objetos
* Diseño de bases de datos
* MySql

# PRACTICA I. Plataforma de trabajo

## I.1. Objetivo

El alumno será capaz de instalar la plataforma de trabajo sobre la que construirán su aplicación.

## I.2. Equipo

Computadora personal, navegador web (Chrome, Mozilla, Opera).

## I.3. Materiales

Hojas de papel bond o cuaderno, lápiz o bolígrafo.

## I.4. Descripción

WAMP es un acrónimo que significa Windows, Apache, MySQL y PHP. Es un stack o conjunto de soluciones de software que significa que cuando instalas WAMP, estás instalando Apache, MySQL y PHP en tu sistema operativo (Windows en el caso de WAMP). Aunque puede instalarlos por separado, por lo general son empaquetados, y también por una buena razón.

Lo que es bueno saber es que WAMP deriva de LAMP (la L significa Linux). La única diferencia entre estos dos es que WAMP se usa para Windows, mientras que LAMP para sistemas operativos basados ​​en Linux.

Repasemos rápidamente lo que representa cada letra:

«W» significa Windows; también hay LAMP (para Linux) y MAMP (para Mac).

«A» significa Apache. Apache es el software de servidor que se encarga de servir las páginas web. Cuando solicitas ver una página, Apache cumple tu solicitud a través de HTTP y te muestra el sitio.

«M» significa MySQL. El trabajo de MySQL es ser el sistema de gestión de base de datos para tu servidor. Almacena toda la información relevante, como el contenido de tu sitio, los perfiles de usuario, etc.

«P» significa PHP. Es el lenguaje de programación en el cual está escrito WordPress y actúa como aglutinante para toda este stack de soluciones. PHP se ejecuta junto con Apache y se comunica con MySQL.

En lugar de instalar y probar WordPress en tu cuenta de hosting, puedes hacerlo en tu computadora personal (localhost).

WAMP actúa como un servidor virtual en tu computadora (WampServer). Te permite probar todas las funciones de WordPress sin ninguna consecuencia, ya que está ubicado en tu máquina y no está conectado a la web.

En primer lugar, esto significa que no necesitas esperar hasta que los archivos se carguen en tu sitio, y en segundo lugar, esto facilita mucho la creación de copias de seguridad.

Un WampServer acelera el proceso de trabajo tanto para los desarrolladores como para los diseñadores de temas. Además, también obtienes el beneficio de poder jugar con su sitio y tener algo de diversión.

Para implementar sus bases de datos necesitaremos instalar el WAMP.

Por lo general, si creamos una página web solo con lenguaje frontend, por ejemplo, en HTML y CSS, el propio navegador es capaz de interpretarla correctamente, por lo que no necesitamos disponer de ningún servidor adicional instalado en el sistema. Sin embargo, si vamos a hacer uso de lenguajes backend, por ejemplo, en PHP, debemos instalar los servidores necesarios en nuestro ordenador (o servidor) de manera que estos sean capaces de interpretar el lenguaje y nos permiten ver el resultado final a través del navegador que, sin dicho servidor, el navegador no interpretaría.

Wamp es un paquete de servidores web para Windows totalmente gratuito y de código abierto que nos va a permitir instalar desde un único paquete un servidor Apache con PHP y un servidor MySQL para poder ejecutar en local cualquier página web que desarrollemos por nuestra cuenta sin tener que subir los archivos a un servidor web dedicado conectado a la nube.

Wamp Server para Windows

Descargar e instalar Wamp en un equipo Windows.

En esta liga puede encontrar un video que le ayuda en su instalación: <https://www.youtube.com/watch?v=5X7symb1QdE&list=PLExyZ7hD-J5b8ftq57xTwBceLXP6ErJzc&index=1>

A continuación una explicación:

Lo primero que debemos hacer es descargar el programa desde su página web principal. El programa es totalmente gratuito y está libre de software publicitario y no deseado.

En su página web vamos a poder elegir si queremos descargar la versión de 32 bits o la de 64 bits. En nuestro caso vamos a trabajar con la versión de 64 bits, ya que tanto nuestro procesador como nuestro sistema operativo lo permiten. Pulsamos sobre la versión deseada de Wamp Server y, en la ventana que nos aparece, pulsamos sobre el enlace «download directly«.

Descargar Wamp Server

El paquete, de poco más de 40 megas, empezará a descargarse a nuestro ordenador. Una vez lo tengamos ya podemos comenzar la instalación.

Ejecutamos el instalador que hemos descargado en el paso anterior y podremos ver la primera ventana del sencillo asistente, el cual no tiene demasiada complicación.

Asistente instalación Wamp

Seguimos con este asistente y a continuación debemos leer y aceptar los términos de licencia. Wamp es software libre, por lo que la licencia que utiliza el GNU.

Licencia de Wamp

A continuación, elegimos la ruta de instalación. Por defecto se instalará en raíz de c:\, una buena ruta y de fácil acceso, por lo que si no tenemos ninguna necesidad específica podemos dejarlo así y continuar.

Ruta instalación Wamp

A continuación, podremos elegir si queremos crear o no iconos en el escritorio o en el menú inicio.

Accesos directos a Wamp

Seguimos con el asistente y llegamos caso al final. Podremos ver un resumen de todas estas opciones.

Resumen instalación Wamp

Si todo está correcto podemos comenzar con la instalación. Pulsamos sobre «Install» y comenzará la copia de archivos.

Instalando Wamp

Este proceso puede tardar algunos minutos, por lo que debemos esperar a que finalice correctamente. Durante la copia de archivos recibiremos dos avisos:

El primero de ellos nos preguntará por nuestro navegador por defecto. Si queremos que el navegador por defecto sea el que utilice Wamp para abrir las webs que alojemos en él, aceptamos dicha ventana.

Navegador por defecto Wamp

El segundo de ellos nos pide rellenar una información básica de cara al servidor de correo SMTP que habilita Wamp. Podemos dejar perfectamente los valores por defecto ya que, como vamos a trabajar en localhost, no tendremos problemas.

Listo. Wamp está instalado en nuestro ordenador y ya podemos empezar a utilizarlo.

Instalación Wamp finalizada.

## I.5. Procedimiento

1. En la sección de ejercicios se describen una serie de problemas en los que es necesario aplicar WAMP.
2. Realice los ejercicios replicando la explicaciópn de la descripción.

## I.6. Prácticas

1. Instale WAP en su computadora.
2. Instale Heidi Sql en su computadora.
3. Implemente una base de datos con 2 tablas.
4. Inserte 2 registros en una de las tablas de base de datos.
5. Realice una consulta select de la tabla.

# PRACTICA II. HTML

## II.1. Objetivo

El alumno será capaz de elaborar inetrfaces de programas que den solución a problemas de gestión de datos, en la parte de interfaz, usando html.

## II.2. Equipo

Computadora personal, dispositivo de almacenamiento secundario (disquete, memoria USB, etc.), navegador web.

## II.3. Materiales

Hojas de papel bond o cuaderno, lápiz o boligrafo.

## II.4. Descripción

HTML5 Elementos

HTML, que significa Lenguaje de Marcado de Hipertextos (HyperText Markup Language), es la pieza más básica para la construcción de la web y se usa para definir el sentido y estructura del contenido en una página web. Otras tecnologías además de HTML son usadas generalmente para describir la apariencia/presentación de una página web (CSS) o su funcionalidad (JavaScript).

"Hipertexto" se refiere a los enlaces que conectan las páginas web entre sí, ya sea dentro de un mismo sitio web o entre diferentes sitios web. los vínculos son un aspecto fundamental de la web. Al subir contenido a Internet y vincularlo a páginas creadas por otras personas, te haces participante activo en la red mundial (World Wide Web).

HTML usa "marcado" (markup en inglés) para anotar textos, imágenes y otro contenido para ser mostrado en un navegador web. El marcado en HTML incluye "elementos" especiales tales como <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, <div>, <span>, <img>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video>, <ul>, <ol>, <li>, y muchos otros más.

Un elemento HTML se separa de otro texto en un documento por medio de "etiquetas", las cuales consisten en elementos rodeados por "<" y ">". El nombre de un elemento dentro de una etiqueta no es sensible a mayúsculas. Esto es, puede ser escrito en mayúsculas, minúsculas o una combinación. Por ejemplo, la etiqueta <title> puede ser escrita como <Title>, <TITLE> o de cualquier otra forma.

## 1 Tags HTML

El principio esencial del lenguaje HTML es el uso de las etiquetas (tags). Funcionan de la siguiente manera:

<XXX> Este es el inicio de una etiqueta.

</XXX> Este es el cierre de una etiqueta.

Las letras de la etiqueta pueden estar en mayúsculas o minúsculas, indiferentemente. Lo que haya entre ambas etiquetas estará influenciada por ellas. Por ejemplo, todo el documento HTML debe estar entre las etiquetas <HTML> y </HTML>:

<HTML> [Todo el documento] </HTML>

Un documento HTML en sí está dividido en dos zonas principales:

* El encabezado, comprendido entre las etiquetas <HEAD> y </HEAD>
* El cuerpo, comprendido entre las etiquetas <BODY> y </BODY>
* El lenguaje de marcas de hipertexto (HTML) funciona por medio de marcas o tags y casi todos tienen inicio y final.

## 2 Encabezado

En el encabezado podemos poner el título del documento, comprendido entre las etiquetas <TITLE> y </TITLE>

Ejemplo:

|  |
| --- |
| Archivo HTML: titulo.html |
| <!DOCTYPE html>  <html>  <head>  <title>mi pagina web</title>  <meta charset="UTF-8"/>  </head>  <body>  <p>cuerpo de la página</p>  <p>Este es un ejemplo de acentos á é í ó ú ñññ ÑÑÑ</p>  </body>  </html> |

Tabla 1:

Ejecútelo así:

1. Cree una carpeta de trabajo, recomiendo que la carpeta no esté en ***mis documentos***, créela en el disco c así: ***c:trabajo/html***
2. El contenido de la tabla 1 lo puede encontrar en su carpeta “html” con el nombre “titulo.html” ábralo en su editor de código Visual Studio Code (VSC), o en Notepad++ o en Programers Notepad. Se debe ver como en la tabla 1. Guárdelo en: ***c:trabajo/html.***
3. En su carpeta ***trabajo/html*** de clic derecho en el archivo “titulo.html” y elija abrir con. Ahí selección su navegador (alguno de estos: EDGE, Mozilla, Google Chrome, Opera) Chrome por ejemplo, como resultado verá el contenido de su página en la ventana en su navegador.

Esta marca: <!DOCTYPE html> indica que estamos usando HTML 5

<head> </head> definen el encabezado de la página

<body> </body> define el cuerpo de la página.

Dentro de <head> </head> también podemos ver la marca

<title> </title> que definen el título de la página.

otros elementos que pueden ponerse en el <head> son:

<meta> para describir características de los datos, por ejemplo, el conjunto de caracteres a usar en la página:

<meta charset="UTF-8"/>

## 3 Cuerpo <body>

Aquí se ponen todos los elementos que definen la página, como cuadros de texto, combos, formularios, contenedores etcétera.

P. ej. Cuando queremos poner un texto sin ninguna característica especial, lo ponemos directamente poniendo el texto entre los tag

***<p> mi parrafo </p>***

Ejemplo:

<https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_paragraphs1>

Otra etiqueta o tag son las h1, h2, hasta h6 que sirven para definir encabezados de diferentes tamaños, ejemplo:

<https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_headers>

Si queremos separar los párrafos, o cualquier otra cosa, pero sin dejar una línea en blanco, usamos una etiqueta parecida <BR> (break, o romper). Este tag no tiene etiqueta de cierre.

Para destacar alguna parte del texto se pueden usar:

<B> y </B> para poner algo en negrita (bold).

<I> y </I> para poner algo en cursiva (italic).

Otra etiqueta interesante es <PRE> y </PRE>. El texto que se encuentre entre ella estará preformateado, es decir que aparecerá como si hubiera sido escrito con una máquina de escribir, con una fuente de espaciado fijo (tipo Courier). Además, se respetarán los espacios en blanco y retornos del carro, tal como estaban en nuestro documento HTML. Es muy apropiada para confeccionar tablas y otros documentos similares. Ejemplo:

<https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_pre>

En las fórmulas matemáticas puede interesar poder escribir índices y subíndices, que se consiguen con las etiquetas <SUP> </SUP> y <SUB> </SUB> respectivamente. Ejemplo:

<https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_sup>

## 4 Atributos

Los elementos de HTML pueden tener atributos, por ejemplo:

***<a href="http://www.w3schools.com">Esta es mi liga</a>***

El elemento <a></a> permite al usuario seguir una liga a INTERNET, este elemento tiene el atributo href, que indica la dirección de la liga a seguir.

Un ejemplo práctico:

|  |
| --- |
| a.html |
| <!DOCTYPE html>  <html>  <body>  <a href="https://www.w3schools.com/html/">Esta es mi liga</a>  </body>  </html> |

Tabla 2

Pruebe el código de la tabla 2 y de clic en el enlace, al hacer clic debe llevarlo a la nueva página.

Otros elementos pueden tener atributos:

<meta charset="UTF-8">

El elemento <meta> cuando se usa con el atributo: charset=”UTF-8” define el conjunto de caracteres a usar en la página.

## 4.1 Atributos id, name y class

Los tags pueden tener varios atributos entre ellos el atributo id que identifica de manera única al elemento en la página. El atributo id no debe repetirse en la página.

<input type="text" id="fname" >

Otro atributo es el atributo name que identifica de manera única a un tag dentro de un formulario:

<input type="text" id="fname" name="nombre">

El atributo name no debe repetirse en el formulario.

En estos ejemplos también se está usando el atributo type=”” que define el tipo de input que se desea, más adelante se explicará el input.

Otro atributo es el atributo class que identifica a un grupo de tags, este atributo puede repetirse y permite hacer referencia a un conjunto de controles o tags.

<input type="text" placeholder="celular" class="form-control" />

<button class=" form-control">nuevo</button>

Aquí además se muestra el atributo placeholder, se trata de un texto que aparece dentro de la caja de entrada de texto y que se usa para describir brevemente el propósito de la caja de texto.

ver ejemplo en: <https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_placeholder>

Para mayor referencia de id y class vea:

1. <https://developer.mozilla.org/es/docs/Web/HTML/Atributos_Globales/class#:~:text=El%20atributo%20global%20class%20es,getElementsByClassName%20del%20DOM.>
2. <https://www.w3schools.com/html/html_id.asp>

Otro atributo es:

style="color:green"

Que permite poner color al texto.

## 5 Agrupadores

Contenedor de bloque es un contenedor que agrupa un conjunto de controles y pone una línea en blanco de separación al final.

Un contenedor de línea agrupa uno o más controles sin moverlos del lugar donde se encuentran.

Los agrupadores sirven para agrupar a un conjunto de controles, para ponerles formato o enviar sus datos a aun servidor, por ejemplo.

### 5.1 <div>

Es un contenedor de bloque, permite agrupar un conjunto de controles para por ejemplo aplicarles un estilo en particular, o hacer referencia a ellos en forma grupal. Ejemplo:

<https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_div_test>

### 5.2 <span>

Es un contenedor de línea, permite agrupar un conjunto de controles para por ejemplo aplicarles un estilo en particular, o hacer referencia a ellos en forma grupal. Ejemplo:

<https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_span>

### 5.3 <form>

En un contenedor de bloque que permite agrupar otros elementos y se usa para crear formularios en html, por ejemplo:

<https://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_submit>

En los formularios usaremos tags o controles como:

Cuadros de texto que son los <input type=”text”>

Botones que son los <input type=”button”>

Check box que son los <input type="checkbox">

Radio button que son los <input type="radio">

Y otros.

#### 5.3.1 listas

Podemos escoger entre tres tipos distintos:

* Listas desordenadas (no numeradas)
* Listas ordenadas (numeradas)
* Listas de definición.

Las listas desordenadas (Unordered Lists) sirven para presentar cosas que, por no tener un orden determinado, no necesitan ir precedidas por un número. Su estructura es la siguiente:

<UL>

<LI> Un elemento

<LI> Otro elemento

<LI> Otro más

<LI> etc.

</UL>

Es decir, toda la lista está dentro de la etiqueta <UL> y </UL>, y luego cada elemento va precedido de la etiqueta <LI> (list ítem). El resultado de lo anterior es el siguiente:

Se puede anidar una lista dentro de otra. Por ejemplo:

<UL>

<LI> Mamíferos

<LI> Peces

  <UL>

  <LI> Sardina

  <LI> Bacalao

  </UL>

<LI> Aves

</UL>

Las listas ordenadas (Ordered Lists) sirven para presentar elementos en un orden determinado. Su estructura es muy similar a la anterior. La diferencia estriba en que en el resultado aparecerá automáticamente un número correlativo para cada elemento.

<OL>

<LI> Primer Elemento

<LI> Segundo Elemento

<LI> Tercer Elemento

<LI> etc.

</OL>

Al igual que las listas desordenadas, también se pueden anidar las listas ordenadas.

El tercer tipo lo forman las listas de definición. Como su nombre indica, son apropiadas para glosarios (o definiciones de términos). Toda la lista debe ir englobada entre las etiquetas <DL> y </DL>. Y a diferencia de las dos que hemos visto, cada renglón de la lista tiene dos partes:

* El nombre de la cosa a definir, que se consigue con la etiqueta <DT> (definition term).
* La definición de dicha cosa, que se consigue con la etiqueta <DD> (definition definition).

<DL>

<DT> Una cosa a definir

<DD> La definición de esta cosa

<DT> Otra cosa a definir

<DD> La definición de esta otra cosa

</DL>

Ejemplo listas: <https://www.w3schools.com/html/tryit.asp?filename=tryhtml_lists_intro>

#### 5.3.2 tablas

Las tablas son un modo sencillo de disponer el texto en columnas, añadir un título a una ilustración, pero hay modos de sacar un gran partido de una característica aparentemente sencilla. La etiqueta <TABLE> puede ser una poderosa herramienta de formato. Se puede hacer, por ejemplo, no mostrar el borde de una tabla en absoluto. También se puede hacer uso de la etiqueta <TABLE> para ubicar texto e imágenes con precisión, en prácticamente casi cualquier lugar de una página.

***Estructura de una tabla***

Vamos a ver ordenadamente (de fuera hacia dentro) las etiquetas necesarias para confeccionar las tablas.

|  |  |
| --- | --- |
| <TABLE>  [resto de las etiquetas]  </TABLE> | **Es la etiqueta general, que engloba a todas las demás.** |
| <TABLE BORDER=n>  [resto de las etiquetas]  </TABLE> | **Presenta los datos tabulados con un borde, haciendo las tablas más atractivas, y el grosor es de n pixeles.** |
| <TR>  [etiquetas de las distintas celdas de la primera fila]  </TR> | **Permite formar cada fila de la tabla. Hay que repetirla tantas veces como filas queremos que tenga la tabla.** |
| <TD>  [contenido de cada celda (imágenes, texto, etc.)]  </TD> | **Permite formar las distintas celdas que contendrá cada fila de la tabla. Hay que repetirlas tantas veces como celdas queramos que tenga la fila.** |
| <TH>  [encabezamiento de tabla]  </TH> | **Es utilizada para colocar encabezamientos en negrita sobre las columnas** |

Tabla 1.2-3

Ejemplo:

|  |
| --- |
| Tabla en HTML |
| **<!DOCTYPE html>**  **<html>**  **<body>**  **<table>**  **<thead>**  **<tr>**  **<th>Month</th>**  **<th>Savings</th>**  **</tr>**  **</thead>**  **<tfoot>**  **<tr>**  **<td>Sum</td>**  **<td>$180</td>**  **</tr>**  **</tfoot>**  **<tbody>**  **<tr>**  **<td>January</td>**  **<td>$100</td>**  **</tr>**  **<tr>**  **<td>February</td>**  **<td>$80</td>**  **</tr>**  **</tbody>**  **</table>**  **</body>**  **</html>** |

Otro ejemplo de tabla en:

<https://www.w3schools.com/tags/tryit.asp?filename=tryhtml_table_test>

### 5.3.3 Input

Los campos de texto <input> son los controles de formulario básicos. Son un modo muy cómodo de permitir al usuario introducir cualquier tipo de datos.

El primero es la entrada de texto vea el ejemplo:

<https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_text>

Tiene otros tipos como radio del que puede ver un ejemplo aquí:

<https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_radio>

el tipo check:

<https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_checkbox>

el tipo date:

<https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_date>

tipo number:

<https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_number>

el tipo button:

<https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_button>

Vea una referencia completa de las características del input en:

<https://www.w3schools.com/html/html_form_input_types.asp>

Realice las siguientes práctiocas, no se preocupe ahorita por el formato.

## II.5. Procedimiento

1. En la sección de ejercicios se describen una serie de problemas en los que es necesario usar elementos html para hacer el programa que de solución al problema descrito.
2. Abra su editor de código Microsoft Visual Studio Code.
3. Codificque el programa en lenguaje html.
4. Guardar el programa y probarlo con 3 diferentes navegadores.

## II.6. Prácticas

1. Construya una página web sin formatos, que muestre su curriculum.
2. Construir un menú basado en una lista, cada elemento de la lista debe enviarlos a otra página en blanco páginas: altas, mostrar, buscar, bajas.
3. En la página de altas agregue el código html para capturar los datos de un alumno, datos: matricula, nombre, carrera, sexo, semestre, lenguajes que maneja (opciones c, vb, java, js, python), el alumno puede manejar mas de un lenguaje.
4. En la página de buscar agregue los controles para capturar la matricula del alumno y un botón para buscar.
5. En la página mostrar construya una tabla con datos de ejemplo.
6. En su carpeta de ejercicios “ejemplos\html\interfaz” pruebe el ejemplo ejmnu0.html. Compare con el ejercicio 2.
7. En su carpeta de ejercicios “ejemplos\html\interfaz” pruebe el ejemplo ejmnu0.html. Compare con el altaAlumno.html.
8. En su carpeta de ejercicios “ejemplos\html\interfaz” pruebe el ejemplo ejmnu0.html. Compare con el mostrarAlumnos.html.

**Nota**. En estos ejercicios solo debe mostrarse la interfaz html, no debrá agregar funcionalidad ni estilos.

# PRACTICA III. CSS

## III.1. Objetivo

El alumno será capaz de elaborar programas que den solución a problemas en los que se requiere el uso de sentencias de bifurcación de código.

## III.2. Equipo

Computadora personal, dispositivo de almacenamiento secundario (disquete, memoria USB, etc.), lenguaje de programación pascal, c, java, etc.

## III.3. Materiales

Hojas de papel bond o cuaderno, lápiz o boligrafo.

## III.4. Descripción

1 Sintaxis CSS

Para dar formato a una página web se utilizan las hojas de estilo CSS. Antes de empezar repase los conceptos de id, class, y los agrupadores, div, span, form, que se explicaron en la parte de html.

Aunque no es la mejor práctica, se puede usar reglas de html en línea, esto consiste en aplicar estilos agregando atributos a sus elementos html.

ejemplo:

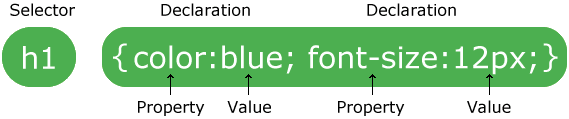
|  |
| --- |
| css-inline-01.html |
| <!DOCTYPE HTML>  <html>  <body>  <p style="color:blue;font-size:46px;">  letras grandes color azul  </p>  <p style="background-color: lightblue;">este es un texto con estilo</p>  </body>  </html> |

Tabla 1 ejemplo uso de css en línea.

En este ejemplo los estilos se aplican directamente en los tag. Pruebe el ejemplo de la tabla 1.

Una segunda y mejor opción, es agrupar todos los estilos entre los tag <style></style> en el head del documento. En este caso habrá que especificar a qué elementos se aplica el estilo.

En este caso una regla o conjunto de reglas CSS consiste en un selector y un bloque de declaraciones:



El selector indica a que elementos se aplica el estilo. La regla anterior indica que los elementos HTML del tipo h1 tienen su propiedad color en azul y su tamaño de fuente en 12. El selector indica el elemento HTML al que se desea dar estilo. El bloque de declaraciones contiene una o más declaraciones de estilo separadas por punto y coma.

|  |
| --- |
| ejcss01.html |
| <!DOCTYPE html> <html>     <head>         <style>             p {                 color: red;                 text-align: center;             }         </style>     </head>     <body>          <p>Hola mundo!</p>         <p>Este parrafo tiene un estilo con CSS.</p>      </body> </html> |

Tabla 2

La regla de la tabla 2:

**p {**  
**color: red;**  
**text-align: center;**  
**}**

indica que los elementos HTML del tipo <p> (párrafo), deberán ir centrados y en color rojo. Como resultado deberá ver en su navegador dos líneas de texto centradas (text-align: center) y en color rojo:

|  |
| --- |
| Hola mundo  Este parrafo tiene un estilo con CSS. |

2 Almacenamiento de reglas

Las reglas en CSS, se pueden almacenar de manera interna en la página HTML como en el ejemplo de la tabla 2, veamos ahora el almacenamiento de la misma regla ahora en un archivo externo con extensión .css:

|  |  |
| --- | --- |
| Archivo: ejcss03.html | Archivo: miEstilo.css |
| **<!DOCTYPE html>** **<html>** **<head>** **<link rel="stylesheet" href="miEstilo.css">** **</head>** **<body>** **<p>Hola mundo!</p>** **<p>Este parrafo tiene un estilo con CSS.</p>** **</body>** **</html>** | **p {** **color: red;** **text-align: center;** **}** |

Tabla 3 Estilos en un archivo css

En su editor de textos escriba los dos archivos de la tabla 2 y guárdelos con los nombres indicados y en la misma carpeta, pruebe el archivo: ejcss03.html en su navegador.

3 Definiendo reglas

Las reglas se definen para tipos de elementos HTML, ejemplo:

|  |
| --- |
| Ejemplo |
| **<!DOCTYPE html>** **<html>** **<head>** **<style>** **body {** **background-color: linen;** **}** **h1 {** **color: maroon;** **margin-left: 40px;** **}** **</style>** **</head>** **<body>**  **<h1>This is a heading</h1>** **<p>This is a paragraph.</p>**  **</body>** **</html>** |

Tabla 4 Reglas para body y p

Explicación:

La regla:

            body {  
                background-color: linen;  
            }

Define que al elemento body se le aplicara el color de fondo linen.

La regla:

            h1 {  
                color: maroon;  
                margin-left: 40px;  
            }

Indica que a los elementos h1 se les aplicará un color marrón y margen izquierdo de 40 pixels.

4 Selectores

Los selectores son empleados para encontrar o seleccionar elementos HTML por medio de su nombre de elemento, clase, id, atributo y más.

4.1 Selector de elemento por su tipo

Este selecciona los elementos basados en su tipo de elemento.

ejemplo:

|  |
| --- |
| p {     text-align: center;     color: red; } |

Tabla 5

Selecciona todos los elementos <p> en la página, con esta regla todos los elementos <p> quedaran centrados y en color rojo.

Ejemplo de uso:

|  |
| --- |
| <!DOCTYPE html> <html> <head>     <style>         p {             text-align: center;             color: red;            }     </style> </head> <body>      <p>Every paragraph will be affected by the style.</p>     <p id="para1">Me too!</p>     <p>And me!</p>  </body> </html> |

Tabla 6

**4.2 Selector id**

Este usa el atributo id de los elementos HTML para aplicar la regla, el id de un elemento HTML debe ser único dentro de la página. La siguiente regla se aplica al elemento con id="para1":

|  |
| --- |
| #para1 {     text-align: center;     color: red; } |

Tabla 7

Ejemplo de uso:

|  |
| --- |
| <!DOCTYPE html> <html> <head>     <style>         #para1 {           text-align: center;           color: red;         }     </style> </head> <body>      <p id="para1">Hola mundo!</p>     <p>Este párrafo no fes afectado por el estilo.</p>  </body> </html> |

Tabla 8

**4.3 Selector de clase**

Este selector, selecciona elementos con un específico nombre de clase, varios elementos pueden tener la misma clase, incluso si no son del mismo tipo, el siguiente selector, solo los elementos <p> con clase "center" serán centrados:

|  |
| --- |
| .center {     text-align: center;     color: red; } |

Ejemplo de uso:

|  |
| --- |
| <!DOCTYPE html> <html> <head>     <style>         .center {            text-align: center;            color: red;         }     </style> </head> <body>      <h1 class="center">Red and center-aligned heading</h1>     <p class="center">Red and center-aligned paragraph.</p>  </body> </html> |

Tabla 9

Usted puede indicar que solo un tipo específico de elementos HTML sea afectado por la clase:

|  |
| --- |
| p.center {     text-align: center;     color: red; } |

Esta regla selecciona los elementos p con clase center.

Ejemplo de uso:

|  |
| --- |
| <!DOCTYPE html> <html> <head>     <style>         p.center {            text-align: center;            color: red;         } </style> </head> <body>      <h1 class="center">This heading will not be affected</h1>     <p class="center">This paragraph will be red and center-aligned.</p>  </body> </html> |

Tabla 10

**Mas sobre selector por tipo de elemento**

Puede seleccionar todos los párrafos, o todas las tablas o todos los encabezados del tipo h1.

|  |
| --- |
| h1 {     text-align: center;     color: red; }  h2 {     text-align: center;     color: red; }  p {     text-align: center;     color: red; } |

Tabla 11

El ejemplo de selección anterior se puede también escribir como sigue:

|  |
| --- |
| h1, h2, p {     text-align: center;     color: red; } |

Tabla 12

Este último aplica el estilo a los elementos del tipo h1, h2 y p.

Ejemplo de uso:

|  |
| --- |
| <!DOCTYPE html> <html> <head>     <style>         h1, h2, p {            text-align: center;            color: red;         } </style> </head> <body>      <h1>Hello World!</h1>     <h2>Smaller heading!</h2>     <p>This is a paragraph.</p>  </body> </html> |

Tabla 13

**4.4 Resumen selectores**

|  |  |  |
| --- | --- | --- |
| **Selector** | **Ejemplo** | **Descripción** |
| [.*class*](https://www.w3schools.com/cssref/tryit.asp?filename=trycss_sel_class) | .intro | Selecciona todos los elementos con class="intro" |
| [#*id*](https://www.w3schools.com/cssref/sel_id.asp) | #firstname | Selecciona los elementos con id="firstname" |
| [\*](https://www.w3schools.com/cssref/sel_all.asp) | \* | Selecciona todos los elementos |
| [*element*](https://www.w3schools.com/cssref/sel_element.asp) | p | Selecciona todos los elementos <p> |
| [*element,element*](https://www.w3schools.com/cssref/sel_element_comma.asp) | div, p | Selecciona todos los elementos <div> y todos los elementos <p> |
| [*element* *element*](https://www.w3schools.com/cssref/sel_element_element.asp) | div p | Selecciona todos los elementos <p> dentro de los elementos <div> |
| [*element*>*element*](https://www.w3schools.com/cssref/sel_element_gt.asp) | div > p | Selecciona todos los elementos <p> que tienen como padre a un elemento <div> |
| [*element*+*element*](https://www.w3schools.com/cssref/sel_element_pluss.asp) | div + p | Selecciona todos los <p> que se encuentran inmediatamente después de un elemento <div> |
| [*element1*~*element2*](https://www.w3schools.com/cssref/sel_gen_sibling.asp) | p ~ ul | Selecciona cada elemento <ul> que esta precedido por un elemento <p> |
| [[*attribute*]](https://www.w3schools.com/cssref/sel_attribute.asp) | [target] | Selecciona todos los elementos con atributo target |
| [[*attribute*=*value*]](https://www.w3schools.com/cssref/sel_attribute_value.asp) | [target=\_blank] | Selecciona todos los elementos con target="\_blank" |
| [[*attribute*~=*value*]](https://www.w3schools.com/cssref/sel_attribute_value_contains.asp) | [title~=flower] | Selecciona todos los elementos con atributo title que contenga la palabra "flower" |
| [[*attribute*|=*value*]](https://www.w3schools.com/cssref/sel_attribute_value_lang.asp) | [lang|=en] | Selecciona todos los elementos con atributo lang que comienzan con "en" |
| [[*attribute*^=*value*]](https://www.w3schools.com/cssref/sel_attr_begin.asp) | a[href^="https"] | Selecciona todos los elementos <a> cuyo valor de atributo href inicia con "https" |
| [[*attribute*$=*value*]](https://www.w3schools.com/cssref/sel_attr_end.asp) | a[href$=".pdf"] | Selecciona todos los elementos <a> cuyo valor de atributo href termina con ".pdf" |
| [[*attribute*\*=*value*]](https://www.w3schools.com/cssref/sel_attr_contain.asp) | a[href\*="w3schools"] | Selecciona todos los elementos <a> cuyo valor de atributo href contiene la subcadena "w3schools" |
| [:active](https://www.w3schools.com/cssref/sel_active.asp) | a:active | Selecciona los link activos |
| [::after](https://www.w3schools.com/cssref/sel_after.asp) | p::after | Inserta algo después del contenido de cada <p> |
| [::before](https://www.w3schools.com/cssref/sel_before.asp) | p::before | Inserta algo antes del contenido de <p> |
| [:checked](https://www.w3schools.com/cssref/sel_checked.asp) | input:checked | Selecciona cada elemento <input> seleccionado |
| [:disabled](https://www.w3schools.com/cssref/sel_disabled.asp) | input:disabled | Selecciona todos los elementos <input> no seleccionados |
| [:empty](https://www.w3schools.com/cssref/sel_empty.asp) | p:empty | Selecciona todos los elementos <p> sin hijos (incluyendo nodos de texto) |
| [:enabled](https://www.w3schools.com/cssref/sel_enabled.asp) | input:enabled | Selecciona todos los elementos <input> activos |
| [:first-child](https://www.w3schools.com/cssref/sel_firstchild.asp) | p:first-child | Selecciona cada elemento <p> que es el primer hijo de su padre |
| [::first-letter](https://www.w3schools.com/cssref/sel_firstletter.asp) | p::first-letter | Selecciona la primera letra de cada elemento <p> |
| [::first-line](https://www.w3schools.com/cssref/sel_firstline.asp) | p::first-line | Selecciona la primera línea de cada elemento <p> |
| [:first-of-type](https://www.w3schools.com/cssref/sel_first-of-type.asp) | p:first-of-type | Selecciona cada elemento<p> que es el primer elemento <p> de su padre |
| [:focus](https://www.w3schools.com/cssref/sel_focus.asp) | input:focus | Selecciona el elemento input que tiene el foco |
| [:hover](https://www.w3schools.com/cssref/sel_hover.asp) | a:hover | Selecciona los link con el ratón sobre ellos |
| [:in-range](https://www.w3schools.com/cssref/sel_in-range.asp) | input:in-range | Selecciona los elementos input con un valor dentro de un rango específico |
| [:invalid](https://www.w3schools.com/cssref/sel_invalid.asp) | input:invalid | Selecciona a todos los elementos input con un valor invalido |
| [:lang(*language*)](https://www.w3schools.com/cssref/sel_lang.asp) | p:lang(it) | Selecciona cada elemento <p> con un atributo lang igual a "it" (Italian) |
| [:last-child](https://www.w3schools.com/cssref/sel_last-child.asp) | p:last-child | Selecciona cada elemento <p> que es el último hijo de su padre |
| [:last-of-type](https://www.w3schools.com/cssref/sel_last-of-type.asp) | p:last-of-type | Selecciona cada elemento <p> que es el último elemento <p> de su padre |
| [:link](https://www.w3schools.com/cssref/sel_link.asp) | a:link | Selecciona todos los links no visitados |
| [:not(*selector*)](https://www.w3schools.com/cssref/sel_not.asp) | :not(p) | Selecciona cada elemento que no es un elemento <p> |
| [:nth-child(*n*)](https://www.w3schools.com/cssref/sel_nth-child.asp) | p:nth-child(2) | Selecciona cada elemento <p> que es el segundo hijo de su padre |
| [:nth-last-child(*n*)](https://www.w3schools.com/cssref/sel_nth-last-child.asp) | p:nth-last-child(2) | Selecciona cada elemento <p> que es el segundo hijo de su padre contado desde el último |
| [:nth-last-of-type(*n*)](https://www.w3schools.com/cssref/sel_nth-last-of-type.asp) | p:nth-last-of-type(2) | Selecciona cada elemento <p> que es el segundo elemento <p> de su padre, contado desde el último |
| [:nth-of-type(*n*)](https://www.w3schools.com/cssref/sel_nth-of-type.asp) | p:nth-of-type(2) | Selecciona cada elemento <p> que es el segundo elemento <p> de su padre |
| [:only-of-type](https://www.w3schools.com/cssref/sel_only-of-type.asp) | p:only-of-type | Selecciona cada elemento <p> que es el único elemento <p> de su padre |
| [:only-child](https://www.w3schools.com/cssref/sel_only-child.asp) | p:only-child | Selecciona cada elemento <p> que es el único hijo de su padre |
| [:optional](https://www.w3schools.com/cssref/sel_optional.asp) | input:optional | Selecciona todos los elementos input sin atributo "required" |
| [:out-of-range](https://www.w3schools.com/cssref/sel_out-of-range.asp) | input:out-of-range | Selecciona los elementos  input con un valor fuera del rango especificado |
| [:read-only](https://www.w3schools.com/cssref/sel_read-only.asp) | input:read-only | Selecciona los elementos input con atributo "readonly" especificado |
| [:read-write](https://www.w3schools.com/cssref/sel_read-write.asp) | input:read-write | Selecciona los elementos input con atributo "readonly" NO especificado |
| [:required](https://www.w3schools.com/cssref/sel_required.asp) | input:required | Selecciona los elementos input con atributo "required" especificado |
| [:root](https://www.w3schools.com/cssref/sel_root.asp) | :root | Selecciona el elemento raiz del documento |
| [::selection](https://www.w3schools.com/cssref/sel_selection.asp) | ::selection | Selecciona la porción de un elemento que esta seleccionado or el usuario |
| [:target](https://www.w3schools.com/cssref/tryit.asp?filename=trycss3_target) | #news:target | Selecciona elemento corriente activo #news (cuando se hace clic en el URL  que contiene su ancla) |
| [:valid](https://www.w3schools.com/cssref/sel_valid.asp) | input:valid | Selecciona todos los elementos  input con un valor valido |
| [:visited](https://www.w3schools.com/cssref/sel_visited.asp) | a:visited | Selecciona todos loe elementos link visitados |

Ejercicios:

Usar directivas css para dar formato a un menú

|  |
| --- |
| Archivo: menu01css.html  En la carpeta css |
| <!DOCTYPE html>  <html>  <head>  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>  <style>  ul {  list-style-type: none; /\*sin la marca de la lista (el punto)\*/  margin: 0;  padding: 0;  overflow: hidden; /\* no mostrar barra de desplazamiento\*/  border: 1px solid #e7e7e7;  background-color: #f3f3f3;  }  li {  float: left;  }  li a {  display: block;  color: #666;  text-align: center;  padding: 14px 16px;  text-decoration: none;  }  li a:hover:not(.active) {  background-color: #ddd;  }  li a.active {  color: white;  background-color: #4CAF50;  }  </style>  </head>  <body>  <ul id="mi\_menu">  <li><a id="inicio" class="active" href="#">inicio</a></li>  <li><a id="pagina1" href="#">altas</a></li>  <li><a id="pagina2" href="#">bajas</a></li>  <li><a id="pagina3" href="#">mostrar</a></li>  </ul>  <div id="area\_trabajo"></div>  </body>  </html> |

Otros ejemplos de menú:

1. <https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_vertical_menu>
2. <https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_menu_hor_scroll>

Ejemplos de formularios con CSS:

1. <https://www.w3schools.com/css/tryit.asp?filename=trycss_forms>

Referencias css:

1. <https://www.w3schools.com/css/css_float.asp>

## III.5. Procedimiento

1. En la sección de ejercicios se describen una serie de problemas en los que es necesario aplicar formato.
2. Elaborar los códigos css para dar formato al html.
3. Pruebe en 3 diferentes navegadores.

## III.6. Prácticas

1. A su ejercicio de curriculum, agregue formatos de manera que se vea profesional, agregue colores, imágenes, tablas.
2. Modificar su menú para que se muestre horizontal en la parte superior y tenga colores de fonto y de texto.
3. Elija algún diseño css de los vistos en esta descripción y aplíquelo a su ventana de altas y buscar.
4. Elija un formato de tabla y aplíquelo a su página de mostrar datos
5. Construya un formulario en html que capture los datos de un alumno: matricula, nombre, sexo, estado de origen y edad, apliquele un estilo en css.
6. En su carpeta de ejercicios “ejemplos\css\interfaz” pruebe el ejemplo ejmnu1.html. Pruebe hacer clic en las opciones, Compare con el ejercicio 2.

# PRACTICA IV. BootStrap

## IV.1. Objetivo

El alumno será capaz de utilizar la biblioteca BootStrap para dar formato a sus códigos HTML.

## IV.2. Equipo

Computadora personal, dispositivo de almacenamiento secundario (disquete, memoria USB, etc.), lenguaje de programación pascal, c, java, etc.

## IV.3. Materiales

Hojas de papel bond o cuaderno, lápiz o boligrafo.

## IV.4. Descripción

Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end.

Bootstrap es el segundo proyecto más destacado en GitHub1​ y es usado por la NASA y la MSNBC entre otras organizaciones

Bootstrap es compatible con las versiones estables más recientes de todos los navegadores y sistemas operativos más populares. De hecho, Bootstrap soporta hasta Internet Explorer 10-11 y Microsoft Edge en Windows.

El resto de navegadores alternativos no están soportados oficialmente, pero es muy probable que Bootstrap funcione bien en ellos porque suelen utilizar los mismos motores WebKit, Blink y Gecko que los navegadores populares.

Para usar el framework puede bajar la biblioteca desde: [getbootstrap.com](http://getbootstrap.com/getting-started/) o puede cargar las bibliotecas en línea de la siguiente forma:

En el encabezado de su HTMP debe poner:

|  |
| --- |
| <!-- Latest compiled and minified CSS --> <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">  <!-- jQuery library --> <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>  <!-- Latest compiled JavaScript --> <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script> |

Tabla 4.1

Ejemplo, primera página con BootStrap

|  |
| --- |
| <!DOCTYPE html> <html lang="en"> <head>   <title>Bootstrap Example</title>   <meta charset="utf-8">   <meta name="viewport" content="width=device-width, initial-scale=1">   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script> </head> <body>  <div class="container">   <h1>My First Bootstrap Page</h1>   <p>This is some text.</p>  </div>  </body> </html> |

Tabla 4.2

BootStrap funciona dividiendo la pantalla en un grid.

El sistema de grids en Bootstrap tiene hasta 12 columnas en la página.

Usted no necesita usar las 12 columnas, puede agruparlas como se muestra abajo.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 | span 1 |
| span 4 | | | | span 4 | | | | span 4 | | | |
| span 4 | | | | span 8 | | | | | | | |
| span 6 | | | | | | span 6 | | | | | |
| span 12 | | | | | | | | | | | |

Tabla 4.3

El sistema de grids de Bootsrap es responsivo y las columnas se reorganizan automáticamente dependiendo del tamaño de la ventana.

El sistema de grids de Bootstrap tiene 4 clases:

* xs (para telefonos)
* sm (para tablets)
* md (para computadoras)
* lg (para computadoras grandes)

Para usar las regillas en BS primero debe crear un renglón (<div class="row">), despues debe agregar el número de columnas que necesite, (aquí debe poner un tag con apropiado nombre de clase y número de columnas) <div class="col-*nombreDeClase*-*numeroDeColumnas*"></div> por ejemlo:

|  |
| --- |
| <div class="row">   <div class="col-sm-4">.col-sm-4</div>   <div class="col-sm-4">.col-sm-4</div>   <div class="col-sm-4">.col-sm-4</div> </div> |

Tabla 4.4

Vea que en las columnas de esta usando la clase sm y un número de columnas de 4.

Nota: recuerde que la suma de columnas de un renglón es de máximo 12.

Ejemplo

Pruebe el siguiente ejemplo en su navegador:

|  |
| --- |
| <!DOCTYPE html> <html lang="en">   <head>     <title>Bootstrap Example</title>     <meta charset="utf-8">     <meta name="viewport" content="width=device-width, initial-scale=1">     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>   </head>   <body>      <div class="container-fluid">     <h1>Hello World!</h1>     <p>Resize the browser window to see the effect.</p>     <div class="row">     <div class="col-sm-4" style="background-color:lavender;">.col-sm-4</div>     <div class="col-sm-8" style="background-color:lavenderblush;">.col-sm-8</div>     </div>     </div>    </body> </html> |

Tabla 4.5

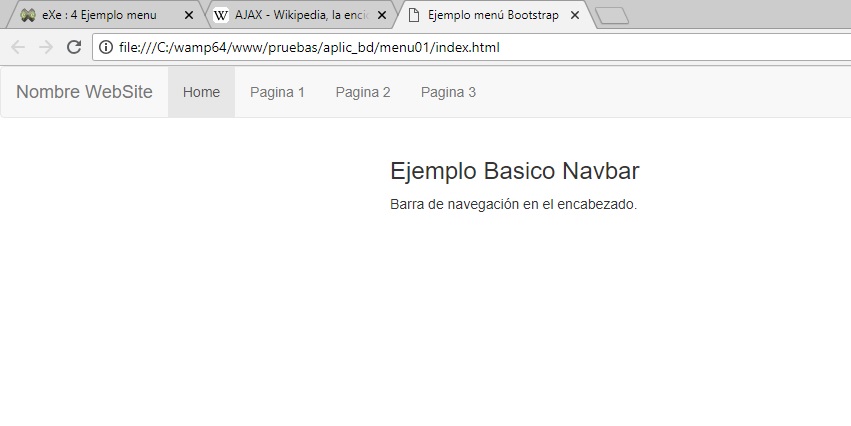
Vea los siguientes ejemplos del uso de BootTrap:

De la página: <http://www.w3schools.com/bootstrap/>  realice lo ejercicios:

* <http://www.w3schools.com/bootstrap/bootstrap_tables.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_images.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_jumbotron_header.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_wells.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_alerts.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_buttons.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_button_groups.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_glyphicons.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_badges_labels.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_progressbars.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_list_groups.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_collapse.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_forms_inputs.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_forms_inputs2.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_forms_sizing.asp>
* <http://www.w3schools.com/bootstrap/bootstrap_modal.asp>

**Ejemplo de un menú con BootStrap**

En este ejemplo construiremos un menú con una barra de navegación, este debe verse así:



Necesitamos el siguiente código:

|  |
| --- |
| index.html |
| <!DOCTYPE html>  <html lang="en">  <head>      <title>Ejemplo menú Bootstrap</title>      <meta charset="utf-8">      <meta name="viewport" content="width=device-width, initial-scale=1">      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>      <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>  </head>  <body>      <nav class="navbar navbar-default">          <div class="container-fluid">              <div class="navbar-header">                  <a class="navbar-brand" href="#">Nombre WebSite</a>              </div>              <ul class="nav navbar-nav">                  <li class="active"><a href="#">Home</a></li>                  <li><a href="#">Pagina 1</a></li>                  <li><a href="#">Pagina 2</a></li>                  <li><a href="#">Pagina 3</a></li>              </ul>          </div>      </nav>      <div class="container">          <h3>Ejemplo Basico Navbar</h3>          <p>Barra de navegación en el encabezado.</p>      </div>  </body>  </html> |

Tabla 4.6 código de barra de navegación versión 1

Abra este archivo en su navegador.

**Menú con BottStrap agregando funcionalidad con JS**

En el ejemplo del menú anterior (sección 4), el menú se muestra pero sin funcionalidad, agreguemos ahora código en JS para darle funcionalidad al menú:

|  |
| --- |
| index.html |
| <!DOCTYPE html>  <html lang="en">  <head>      <title>Ejemplo menú Bootstrap</title>      <meta charset="utf-8">      <meta name="viewport" content="width=device-width, initial-scale=1">      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">      <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>      <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>      <script>          $(document).ready(function(){              $("#home").click(function() {                  $('#area\_trabajo').html("estas en la ventana principal");              });              $("#pagina1").click(function() {                  $('#area\_trabajo').html("estas en la opcion 1");              });              $("#pagina2").click(function() {                  $('#area\_trabajo').html("estas en la opcion 2");              });          });      </script>  </head>  <body>      <nav class="navbar navbar-default">          <div class="container-fluid">              <div class="navbar-header">                  <a class="navbar-brand" href="#">Nombre WebSite</a>              </div>              <ul class="nav navbar-nav">                  <li class="active"><a href="#" id="home">Home</a></li>                  <li><a href="#" id="pagina1">Pagina 1</a></li>                  <li><a href="#" id="pagina2">Pagina 2</a></li>                  <li><a href="#" id="pagina3">Pagina 3</a></li>              </ul>          </div>      </nav>      <div class="container">          <h3>Ejemplo Basico Navbar</h3>          <div id="area\_trabajo"></div>          <p>Barra de navegación en el encabezado.</p>      </div>  </body>  </html> |

Tabla 4.7 Menú con funcionalidad

Ejecute en su navegador y pruebe hacer clic en las opciones.

## IV.5. Procedimiento

1. En la sección de ejercicios se describen una serie de problemas en los que es necesario aplicar estilos para mejorar la visibilidad de la página.
2. Elaborar el análisis del problema en base al tipo de usuario y proponer un diseño.
3. Codificar usando la biblioteca Bootstrap.
4. Probar en tres diferentes navegadores.

## IV.6. Prácticas

1. Diseñe un menú para las necesidades de una tienda pequeña usando BootStrap.
2. Diseñe un menú para la aplicación de su proyecto.
3. Diseñe un formulario de captura de datos para su proyecto, usando elementos BootStrap.
4. Diseñe una página con una tabla que muestre datos usando elementos BootStrap.
5. Diseñe una página de búsqueda de datos usando elementos de bootstrap.
6. A las ventanas de su proyecto aplique estilos CSS + BootStrap

# PRACTICA V. JS

## V.1. Objetivo

El alumno será capaz de elaborar aplicaciones computacionales que den solución a problemas que requieran páginas web dinámicas usando lenguaje JS.

## V.2. Equipo

Computadora personal, dispositivo de almacenamiento secundario (disquete, memoria USB, etc.), lenguaje de programación visual basic, c.

## V.3. Materiales

Hojas de papel bond o cuaderno, lápiz o boligrafo.

## V.4. Descripción

### Introducción:

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico es soportado por virtualmente todos los navegadores modernos y nos permite agregar funcionalidad del lado del cliente a las páginas WEB.

Aquí veremos una introducción a el uso de este lenguaje.

### 1 Java Script y HTML.

Con Js puede cambiar el contenido del HTML. En un archivo con extencion html introduce el código de la tabla 5.1 y ábrelo con el navegador Google Chrome o Mozilla, sobre el archivo da clic derecho y luego abrir con, y elige en navegador.

|  |
| --- |
| Cambiando  contenido en HTML, archivo: holamundo.html |
| <!DOCTYPE html> <html>     <body>         <h1>Que puedo hacer con JS?</h1>         <p id="demo">JavaScript puede cambiar el contenido de HTML.</p>         <button type="button" onclick="document.getElementById('demo').innerHTML = 'Hola Mundo!'">Haz clic aqui!</button>     </body> </html> |

Tabla 5.1.

Aquí el trabajo lo hace la función onclick que esta como atributo del boton:

***onclick="document.getElementById('demo').innerHTML = 'Hola Mundo!'"***

Onclick ejecuta una sentencia cuando se hace clic sobre el botón, en este caso la sentencia es:

**document.getElementById('demo').innerHTML = 'Hola Mundo!'**

En la sentencia; ***document*** se refiere a todo el documento, es decir la página web, ***getElementById***, indica, que queremos hacer referencia a un elemento por su nombre, como se le pasa como parámetro ***(‘demo’),*** document.getElementById(‘demo’) quiere decir que dentro del documento debemos seleccionar al elemento cuyo id sea ***demo*** y en este caso es el elemento:

***<p id="demo">JavaScript puede cambiar el contenido de HTML.</p>***

Que estamos seleccionando. El resto de la sentencia:

***.innerHTML = 'Hola Mundo!'"***

¡Indica que al elemento seleccionado dentro de su atributo innerHTML, debemos asignar la cadena de texto ‘Hola Mundo!’, el atributo ***innerHTML*** hace referencia al contenido que visualiza el control, sería como el atributo text de un control en visual basic.

También puede cambiar los atributos del HTML

|  |
| --- |
| Cambiado atributos, archivo: atributo.html |
| <!DOCTYPE html> <html>     <body>         <h1>Que puedo hacer con JS?</h1>         <p id="demo">JavaScript puede cambiar el estilo de un elemento HTML.</p>         <button type="button" onclick="document.getElementById('demo').style.fontSize='35px'">Haga clic!</button>     </body> </html> |

Tabla 5.2

Esta sentencia indica que, al hacer clic en el botón, el estilo del texto del párrafo deberá cambiarse a una fuente de tamaño 35 pixels.

### 2 Donde codificar el JS.

En los ejemplos anteriores, se vio como codificar JS incrustandolo en los tag de HTML, una mejor forma de hacerlo es dentro de un área epecial dentro del archivo HTML (ver tabla 5.3) en la tabla se muestra el uso de una fucnión de JS asociada a un Button, donde la función está definida en el área <script>...</script>

|  |
| --- |
| Ejemplo JS |
| <!DOCTYPE html> <html>     <head>         <script>             function myFunction() {                 document.getElementById("demo").innerHTML = "Parrafo cambiado.";             }         </script>     </head>     <body>         <h1>JavaScript in Head</h1>         <p id="demo">Mi parrafo.</p>         <button type="button" onclick="myFunction()">clic aqui</button>     </body> </html> |

Tabla 5.3 Archivo: js03.html

Pruebe el ejemplo de la tabla 5.3.

Otra forma de hacerlo (seguramente la mas 'limpia' desde el punto de vista de un programador organizado) es poner el código JS en un archivo aparte, como se muestra en la tabla 5.4.

|  |
| --- |
| Archivo: js04.html |
| <!DOCTYPE html> <html>     <head>         <script src="myscript.js"></script>     </head>     <body>         <h1>JavaScript in Head</h1>         <p id="demo">Mi parrafo.</p>         <button type="button" onclick="myFunction()">clic aqui</button>     </body> </html> |
| Archivo: myscript.js |
| function myFunction() {     document.getElementById("demo").innerHTML = "Parrafo cambiado."; } |

Tabla 5.4

El ejemplo de la tabla 5.4 consta de 2 archivos, lo puede encontrar en la carpeta: \ejemplos\js-puro\basicos, que puede descargar desde la carpeta de ejemplos.

Pruebe los ejemplos de la tabla 5.3 y 5.4 Comente sus dudas con el profesor.

### 3 Entrada Salida

En JS no existe E/S como tal, no puede como en c usar printf, en su lugar el JS se comunica con su página HTML

Ejemplo mostrar una ventana Alert:

|  |
| --- |
| Mostrar una ventana emergente |
| <!DOCTYPE html> <html> <body>     <h1>Ejemplo de alert</h1>     <p>mi parrafo.</p>     <script>         window.alert(5 + 6);     </script> </body> </html> |

Tabla 5.5 Alert en JS

También puede escribir al documento HTML

|  |
| --- |
| Escribir en el documento HTML |
| <!DOCTYPE html> <html>     <body>         <h1>mi Web Page</h1>         <p>mi parrafo.</p>         <script>             document.write(5 + 6);         </script>     </body> </html> |

Tabla 5.6 Escribir al documento.

|  |
| --- |
| Si usa document.write despues de que el HTML es cargado totalmente, elimina el contenido anterior del HTML, vea este ejemplo |
| <!DOCTYPE html> <html>     <body>         <h1>mi Web Page</h1>         <p>mi parrafo.</p>         <button onclick="document.write(5 + 6)">click aqui</button>     </body> </html> |

Tabla 5.7.

Puede también escribir dentro de un elemento HTML

|  |
| --- |
| Escribiendo el el elemento HTML con id = "demo" |
| <!DOCTYPE html> <html>     <body>         <h1>My First Web Page</h1>         <p>My First Paragraph</p>         <p id="demo"></p>         <script>             document.getElementById("demo").innerHTML = 5 + 6;         </script>     </body> </html> |

Tabla 5.8

Finalmente, y este es un método usado por desarrolladores, se puede escribir a la consola del navegador (puede visualizar la consola de su navegador haciendo clic en la tecla f12, y en la ventana que aparece seleccione la pestaña consola), esto es muy útil para depurar su código. Normalmente el usuario no visualiza la consola, así que use este método para dar seguimiento a su código mientras está desarrollando.

|  |
| --- |
| Escribiendo a la consola |
| <!DOCTYPE html> <html>     <body>         <h1>My First Web Page</h1>         <p>My first paragraph.</p>         <script>             console.log(5 + 6);         </script>     </body> </html> |

Tabla 5.9

### 3 Entrada JS

La entrada en JS tampoco existe, JS tomara sus datos de los tag de HTML

### 4 Variables, tipos y expresiones.

**Variables**

En JS las variables se declaran así:

***var mivariable;***

***var miotravariable=23;***

Si son variables globales

O asi si son locales:

***let mivariable;***

***let miotravariable=23;***

No se declarar el tipo, el tipo de una variable se asocias al asignarle un dato, dependiendo del, dato una variable puede ser: string, number, boolean, array y object.

Las cadenas en JS se representan entre comillas o entre apostrofes.

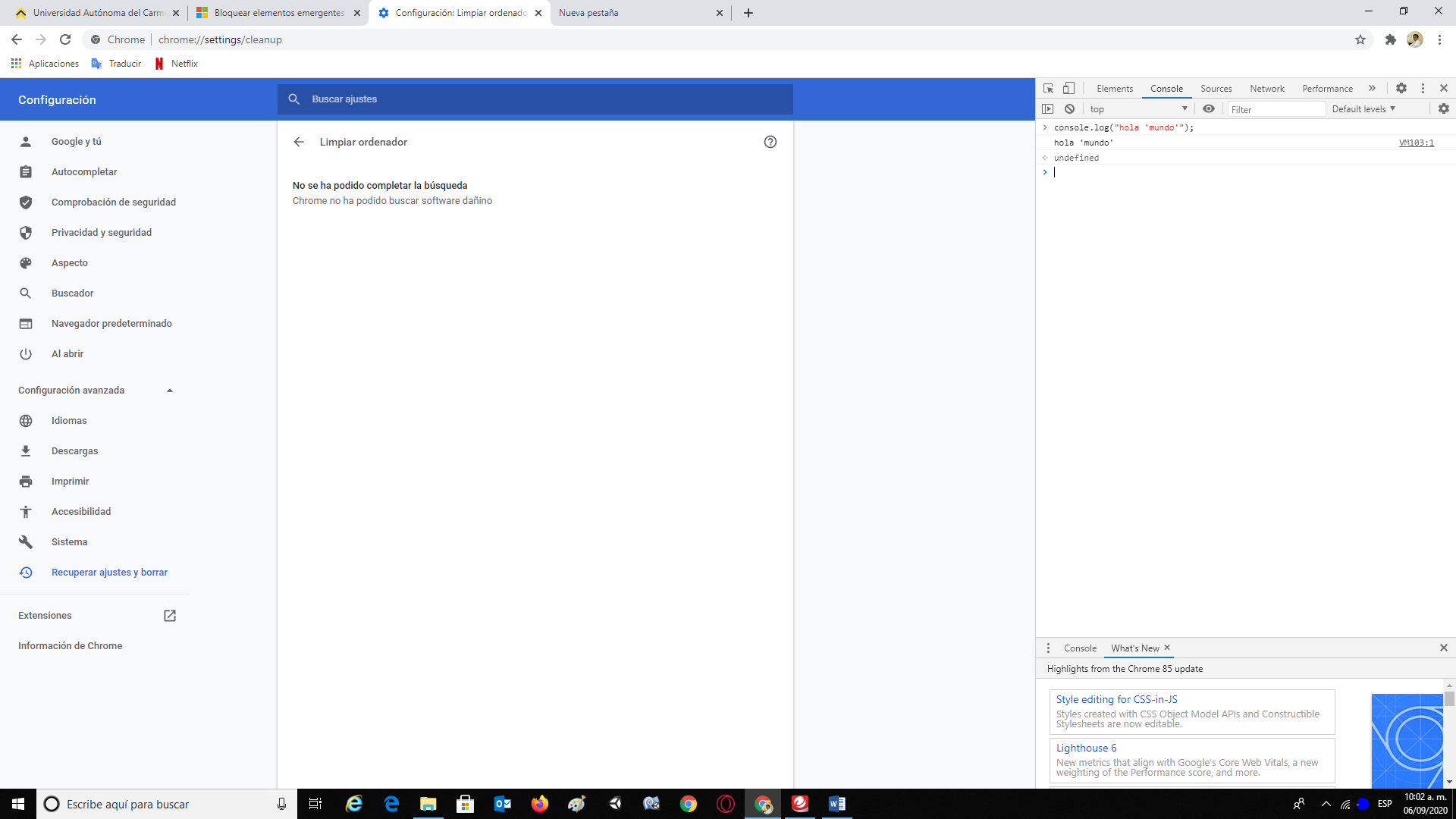
***"esta es una cadena"***

***'esta tambien es una cadena'***

***var nombre='Andrea Flores';***

Las cadenas se pueden anidar así:

Pruebe este código escribiéndolo en la consola de su navegador así:



Observe que los apostrofes dentro de la cadena se toman como parte de la cadena.

**Tipos**

Como dijimos el tipo de la variable se define cuando se le carga un dato ejemplo:

var length = 16;                               // Number  
var lastName = "Johnson";                      // String  
var cars = ["Saab", "Volvo", "BMW"];           // Array  
var x = {firstName:"John", lastName:"Doe"};    // JSON

pruebe este código en la consola del navegador:

|  |
| --- |
| ***let persona={nombre:'juan', apellido:'perez'};***  ***console.log(persona.nombre);*** |

**Concatenación de cadenas**

Las cadenas se concatenan con +

***var nombre= "Alex" +" "+"Flores";***

**Operadores**

|  |  |
| --- | --- |
| **Operador** | **Descripción** |
| + | Adición |
| - | Substracción |
| \* | Multiplicación |
| / | División |
| % | Modulo |
| ++ | Incremento |
| -- | Decremento |

|  |  |  |
| --- | --- | --- |
| **igualdad** | Comprueba si dos valores son iguales entre sí, y devuelve un valor de true/false(booleano). | == |
| **identidad** | Comprueba si dos valores son identicos entre sí, y devuelve un valor de true/false(booleano). | === |
| **Negación, distinto (no igual)** | En ocasiones utilizado con el operador de identidad, la negación es en JS el equivalente al operador lógico NOT — cambia true por false y viceversa. | !, !== |

Tabla 5.10

Ejemplos:

1 == 1 // true  
"1" == 1 // true  
1 == '1' // true  
0 == false // true  
1 != 2 // true  
1 != "1" // false  
1 != '1' // false  
1 != true // false  
0 != false // false  
3 === 3 // true  
3 === '3' // false  
3 !== '3' // true  
4 !== 3 // true

### 5 Funciones.

Una función en JS se declara de forma similar que, en C, ejemplo:

**//definición de función**

**function toCelsius(fahrenheit) {**  
**return (5/9) \* (fahrenheit-32);**  
**}**

**//llamado a la función**  
**document.getElementById("demo").innerHTML = toCelsius(9);**

Una función puede asignarse a una variable:

|  |
| --- |
| Función a una  variable |
| <!DOCTYPE html> <html>     <body>         <p id="demo"></p>         <script>             var x= function celsius(fahrenheit) {                 return (5/9) \* (fahrenheit-32);             }             document.getElementById("demo").innerHTML ="la temperatura es " + x(77) + " Celsius";     </script> </body> </html> |

Tabla 5.11

En este caso ya que la función esta referenciada por la variable x podemos omitir el nombre de la función a lo que se llama función anónima.

|  |
| --- |
| Función anónima |
| <!DOCTYPE html> <html>     <body>         <p id="demo"></p>         <script>             var x= function(fahrenheit) {                 return (5/9) \* (fahrenheit-32);             }             document.getElementById("demo").innerHTML ="la temperatura es " + x(77) + " Celsius";     </script> </body> </html> |

#### Tabla 5.12

#### 5.1 Funciones anónimas

Una función anónima es una función que no tiene nombre se declaran así:

**function () {**

**console.log('Esta función no tiene un nombre')**

**}**

No parece tener sentido ya que al no tener nombre no puedes llamarla, pero se usan cuando se quiere una función autoinvocada o enviar una función como parámetro.

5.2 Funciones autoinvocadas.

Una función anónima autoinvocada se ejecuta de forma automática, su sintaxis es:

(function(){

    // Código

})();

Su contenido se ejecuta automáticamente, ejemplo pruebe el ejemplo de la siguiente tabla:

|  |
| --- |
| Ejemplo función anónima autoinvocada |
| <!DOCTYPE html> <html>     <script>         (function() {             var primera = 'Hola',             segunda = 'Don Pepito';              function saludo() {                 return [primera, segunda].join(';');         }         document.write(saludo());     })();  </script> </html> |

Tabla 5.13

Para el resultado del ejemplo bastaba hacer:

**document.write(primera + segunda);**

en lugar de llamar a una función sin embargo quisimos hacerlo así para mostrar que una función puede contener a otra función. La función join() por otra parte une los elementos de una arreglo en una cadena separados por el separador enviado como parámetro a la función join.

### 6 Eventos.

En JS se pueden manejar varios eventos ejemplo:

|  |
| --- |
| <!DOCTYPE html> <html>     <body>         <p id="demo" onmouseover="saluda()">hola</p>             <button onclick="displayDate()">The time is?</button>     </body>     <script>         function displayDate() {             document.getElementById("demo").innerHTML = Date();             document.getElementById("demo").style.fontSize='12px';         }         function saluda() {             document.getElementById("demo").style.fontSize='35px';         }  </script> </html> |

Tabla 6.1

En la tabla 6.1 el evento onmouseover se dispara cuando pasa el cursor sobre el párrafo, el otro es un evento clic.

Otros eventos son:

|  |  |
| --- | --- |
| **Evento** | **Descripción** |
| onchange | Cuando un elemento  HTML cambió |
| onclick | Cuando el usuario hizo clic sobre un elemnto HTML |
| onmouseover | El usuario mueve el mouse sobre un elemento HTML |
| onmouseout | El usuario mueve el mouse fuera de un elemento HTML |
| onkeydown | El usuario oprime una tecla |
| onload | El navegador termina de cargar la página |

Tabla 6.2

#### **6.1 Donde asignar eventos**

Se puede asignar un evento en forma estática como en la tabla 6.1 en la línea:

***<button onclick="displayDate()">The time is?</button>***

Tambien se pueden asiganar eventos en forma deinámica dentro del código html:

|  |
| --- |
| Archivo: |
| <!DOCTYPE html>  <html>  <head>          <meta charset="utf-8">    <style>              table{                  width:300px;              }              th{                  background-color:#008080;                  color:#ffffff;              }  tr:nth-child(even){background-color:#f2f2f2;}              tr:nth-child(odd){background-color:#C0C0C0;}              th, td{                  text-align:center;                  padding:10px;              }              td:hover{                  background-color:#424242;                  color:#ffffff;              }              #miBoton {  background-color: #008CBA;  border: none;  color: white;  padding: 10px 25px;  text-align: center;  text-decoration: none;  display: inline-block;  font-size: 16px;  }  </style>  </head>  <body>  <div id="area\_trabajo">    </div>  <input type="button" id="miBoton" name="llenar" value="llenar tabla">  </body>  <script>    window.onload=function(){  document.getElementById("miBoton").addEventListener("click", function(){  document.getElementById("area\_trabajo").innerHTML="Hola mundo";  });  }  </script>  </html> |

Tabla 6.3

Explicación:

Aquí obrevamos dos cosas:

window.onload=function(){

}

Una que estamos usando la función Windows.onload, es recomendable usarla para encapsular su código JS, esta función ejecuta su código después de que la página se cargo totalmente, esto evita errores. De esta forma estamos seguros que nuestro código se ejecuta cuando todos los elementos de la página están ya disponibles.

La otra parte del código:

***document.getElementById("miBoton").addEventListener(evento , manejador);***

Agrega un evento al ejlemento del documento cuyo id es igual a “miboton”, los parámetros evento es una cadena que indica el evento que agregamos y el parámetro manejador es una función que maneja el evento.

Los parametyros que se ponen en la sentencia son:

"click"

Para agregar el envento clic y el otro parámetro es la función anónima:

function(){

document.getElementById("area\_trabajo").innerHTML="Hola mundo";

}

Que se ejecutará cuando se dispare el evento.

**Ejemplo de evento clic en menú**

Veamos ahora algoi un poco mas complejo, mostraremos un menú y asignaremos clic a las opciones del menú de manera que al hacer clci en alguna opción, se muestre en la consola, un mensaje que indique a que opción se hizo clic:

Este ejemplo lo puede ver en la carpeta de ejercicios: ***ejemplos\js-puro\eventos\event-mnu-0.html***

Y en la tabla siguiente:

|  |
| --- |
| Archivo: event-mnu-0.html |
| <!DOCTYPE html>  <html>  <style>  ul {  list-style-type: none; /\*sin la marca de la lista (el punto)\*/  margin: 0;  padding: 0;  overflow: hidden; /\* no mostrar barra de desplazamiento\*/  border: 1px solid #e7e7e7;  background-color: #f3f3f3;  }  li {  float: left; /\* los muestra en la misma linea \*/  }  li a {  display: block; /\*despliega la opcion en modo de una caja ocupando todo su espacio\*/  color: #666;  text-align: center;  padding: 14px 16px;  text-decoration: none; /\*sin subraya\*/  }  li a:hover:not(.active) { /\*cuando el cursos pasa sobre las opciones no activas\*/  background-color: #ddd;  }  li a.active { /\*la opcion activa\*/  color: white;  background-color: #4CAF50;  }  </style>  <head>    </head>  <body>      <ul>          <li><a id="op\_1" class="op\_menu active" href="#">Alta alumno</a></li>          <li><a id="op\_2" class="op\_menu" href="#">Buscar Alumno</a></li>          <li><a id="op\_3" class="op\_menu" href="#">Mostrar alumnos</a></li>          <li><a id="op\_4" class="op\_menu" href="#">Acerca de</a></li>      </ul>    </body>  <script>  function click\_menu(e){  let quien=e.target; //a quien se le hizo clic  console.log("clic en "+quien.getAttribute("id"));  }  window.onload=function(){  let ops=document.getElementsByClassName("op\_menu");  //agregamos el listener a las opciones del menu          for(i=0;i<ops.length;i++){              ops[i].addEventListener('click',click\_menu);          }  };  </script>  </html> |
| Ejecutelo dando clic derecho en el archivo y abrir con el navegador de su preferencia. Al hacer clci en alguna opción en la consola se verá un mensaje. |

Tabla 6.4

Explicación:

La sentencia

let ops=document.getElementsByClassName("op\_menu");

Obtiene una lista de elementos, en este caso la lista de elementos cuto nombre de clase es “op\_menu”.

La sentencia:

        for(i=0;i<ops.length;i++){

            ops[i].addEventListener('click',click\_menu);

        }

Recorre la lista ops y a cada elemento le asocia el evento clic e indica que el manejador de evento es la función click\_menu.

function click\_menu(e){

let quien=e.target; //a quien se le hizo clic

console.log("clic en "+quien.getAttribute("id"));

}

La función clic menú recibe como arguimento (de manera automática, usted no tiene que enviarlo) on objeto que tiene la información del evento, en este caso se recibe en la variable local e.

***e.target;***

nos devuelve el nombre del control que generó el evento, es decir a quien le hicimos clic.

Finalmente escribimos en la consola el atributo id del control que nos generó el evento.

### 7 Objetos (opcional).

Los objetos en JS son como variables, pero pueden contener mas de un valor.

var car = {type:"Fiat", model:"500", color:"white"};

Para acceder a las propiedades de u objeto se usa:

|  |
| --- |
| <!DOCTYPE html> <html>     <body>         <p id="demo"></p>         <p id="otro"></p>         <script>             var car = {tipo:"Fiat", modelo:"500", color:"white"};             document.getElementById("demo").innerHTML ="el auto es un " + car.tipo;             document.getElementById("otro").innerHTML =" otra forma de acceder a las propiedades es " + car["tipo"];         </script> </body> </html> |

Tabla 7.1

Los objetos también pueden tener operaciones:

|  |
| --- |
| <!DOCTYPE html> <html>     <body>         <p id="demo"></p>         <script>             var car = {                  tipo:"Fiat",                   modelo:"500",                   color:"white",                  muestra:function(){                      return [this.tipo, this.modelo, this.color].join(',');                  }             };             document.getElementById("demo").innerHTML ="el auto es un " + car.muestra();         </script>     </body> </html> |

Tabla 7.2

### 8 prototipos de objetos (opcional).

Como dijimos los objetos son variables, los valores en los objetos se escriben como nombre:valor, los objetos en JS son contenedores para los pares nombre:valor. Los pares nombre:valor en los objetos son llamados propiedades, Los métodos son acciones que pueden ser realizadas en objetos, puede acceder a un método de un objeto con la sintaxis:

***objectName.methodName()***

8.1 Creando Objetos

En JS puede:

* Definir y crear un objeto simple usando un objeto literal.
* Definir y crear un objeto simple con  new.
* Definir un constructor de objetos (object constructor)y crear objetos del tipo construido.

Forma 1: Objeto literal:

**var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};**

Espacios y CR no importan en la definición:

**var person = {  
    firstName:"John",  
    lastName:"Doe",  
    age:50,  
    eyeColor:"blue"  
};**

Forma 2: Otra forma de tener el mismo resultado (el objeto de arriba), es:

**var person = new Object();**  
**person.firstName = "John";**  
**person.lastName = "Doe";**  
**person.age = 50;**  
**person.eyeColor = "blue";**

Esta forma también crea un objeto persona con 4 propiedades, por rapidez de ejecución y legibilidad es recomendable la primera forma.

Forma 3: Constructor de objetos, también llamado prototipo de objetos:

**function person(first, last, age, eye) {  
    this.firstName = first;  
    this.lastName = last;  
    this.age = age;  
    this.eyeColor = eye;  
}  
var myFather = new person("John", "Doe", 50, "blue");  
var myMother = new person("Sally", "Rally", 48, "green");**

JS tiene un conjunto de objetos preconstruidos:

**var x1 = new Object();    // A new Object object  
var x2 = new String();    // A new String object  
var x3 = new Number();    // A new Number object  
var x4 = new Boolean();   // A new Boolean object  
var x5 = new Array();     // A new Array object  
var x6 = new RegExp();    // A new RegExp object  
var x7 = new Function();  // A new Function object  
var x8 = new Date();      // A new Date object**

Otro objeto es Math(), sin embargo este es un objeto global y de este no puede usar new.

8.2 Agregar propiedades y métodos a objetos.

Se puede agregar propiedades al objeto o al prototipo, veamos primero como agregar una propiedad a un objeto:

|  |
| --- |
| <!DOCTYPE html><html>    <body>          <p id="demo"></p>          <script>            function Person(first, last, age, eye) {                this.firstName = first;                this.lastName = last;                this.age = age;                this.eyeColor = eye;            }              var myFather = new Person("John", "Doe", 50, "blue");            var myMother = new Person("Sally", "Rally", 48, "green");              myFather.nationality = "English";              myFather.name = function() {                return this.firstName + " " + this.lastName;            };              document.getElementById("demo").innerHTML ="My father is " + myFather.nationality;        </script>      </body></html> |

Tabla 7.3

En este ejemplo se agregó la propiedad nationality y el método name al objeto myFather y no al objeto myMother.

8.3 Agregando propiedades y métodos al prototipo

|  |
| --- |
| <!DOCTYPE html><html>    <body>          <p id="demo"></p>          <script>            function Person(first, last, age, eye) {                this.firstName = first;                this.lastName = last;                this.age = age;                this.eyeColor = eye;                  this.nationality = "English";                this.name = function() {                    return this.firstName + " " + this.lastName                };            }              var myFather = new Person("John", "Doe", 50, "blue");            document.getElementById("demo").innerHTML ="My father is " + myFather.name();        </script>      </body></html> |

Tabla 7.4

En color azul se muestra la agregación de una propiedad y un método al prototipo.

### 9 Formato JSON

**JSON** (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del [Lenguaje de Programación JavaScript](http://javascript.crockford.com/), [Standard ECMA-262 3rd Edition - Diciembre 1999](http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf). JSON es un formato de texto que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

En la liga: <http://www.json.org/json-es.html> vea un diagrama para escritura de objetos JSON

En la liga:<http://jsonlint.com/>  puede verificar la sintaxis de sus objetos JSON.

**Ejemplo de una cadena json:**

***let personas=[{"id" : 01,"nombre": "ana"},{"id": 02,"nombre": "bety"},{"id : 03,"nombre" : "ceci"}];***

Observe que la variable se esta declarando con la sentencia let y no con var, let se utiliza para declarar variables locales y var se usa para declarar variables globales.

Para recorrer un elemento json podemos usar:

***<script>***

***let personas=[***

***{***

***"id" : 01,***

***"nombre": "ana"***

***},***

***{***

***"id": 02,***

***"nombre": "bety"***

***},***

***{***

***"id" : 03,***

***"nombre" : "ceci"***

***}***

***];***

***personas.forEach(function recorre(item, index){***

***console.log("index=" + index + " dato: " + item.nombre);***

***});***

***</script>***

Muestra todos los elementos del json en la consola.

Explicación:

Personas es el objeto josn, y los objetos json tienes asociada cienta funcionalidad en JS.

personas.forEach indica que queremos recorrer cada elemento del json, al forEach le podemos dar como parámetro una función que se encargará de hacer el recorrido:

***persona.forEach(function recorre(parametros){//codigo });***

los parámetros de la función son ítem que contiene el elemento corriente an cada iteración del for e index que contiene el índice o posición del elemento dentro del json, console.log ya sabemos que escribe a cosnola. Pruebe el ejercicio.

Dentro de la pestaña de ejercicios en la carpeta: ejemplos\js\js02, pruebe los ejercicios:

recorrer\_js.html

recorrer\_js2.html

recorrer\_js3.html

enseguida dos ejemplos mas de json:

### 10 Recorre JSON

|  |
| --- |
| json-avanzado0.html |
| <!DOCTYPE html>  <html>  <head>  </head>  <body>  </body>  <script>  let personas=[  {  "nombre" : "anita pacheco",  "edad" : 22  },  {  "nombre" : "bety",  "edad" : 33  },  {  "nombre" : "ceci",  "edad" : 44  }  ];  for(var {"nombre": n, "edad": e} of personas){  console.log(n, e);  }  </script>  </html> |

|  |
| --- |
| json-avanzado.html |
| <!DOCTYPE html>  <html>  <head>  </head>  <body>  </body>  <script>  let personas=[  {  nombre : "anita pacheco",  familia : {  padre : "anastacio",  madre : "nastasia"  },  edad : 22  },  {  nombre : "bety",  familia : {  padre : "beto",  madre : "beta"  },  edad : 33  },  {  nombre : "ceci",  familia : {  padre : "cesilio",  madre : "cecilia"  },  edad : 44  }  ];  for(var {nombre: n, familia:{padre: p}} of personas){  console.log(n, p);  }  </script>  </html> |

Ejemplo que llena una tabla html con datos de un arreglo JSON:

|  |
| --- |
| Archivo: ejemplos\js-puro\tablas |
| <!DOCTYPE html>  <html>  <head>          <meta charset="utf-8">  <style>              table{                  width:300px;              }              th{                  background-color:#008080;                  color:#ffffff;              }  tr:nth-child(even){background-color:#f2f2f2;}              tr:nth-child(odd){background-color:#C0C0C0;}              th, td{                  text-align:center;                  padding:10px;              }              td:hover{                  background-color:#424242;                  color:#ffffff;              }              #btnLlenar {  background-color: #008CBA;  border: none;  color: white;  padding: 10px 25px;  text-align: center;  text-decoration: none;  display: inline-block;  font-size: 16px;  }  </style>  </head>  <body>  <div id="area\_trabajo">  <table>  <thead>  <tr>  <th>id</th>  <th>nombre</th>  </tr>  </thead>  <tbody id="datosTabla">  </tbody>  </table>  </div>  <input type="button" id="btnLlenar" name="llenar" value="llenar tabla">  </body>  <script>          var personas=[{  "id":"01",  "nombre":"juan"  },{  "id":"02",  "nombre":"toño"  },  {  "id":"03",  "nombre":"sánchez"              },  {  "id":"04",  "nombre":"rodriguez"              },  {  "id":"05",  "nombre":"lopitos"              }          ];      window.onload=function(){  document.getElementById("btnLlenar").addEventListener("click", function(){  var content="";  personas.forEach(function(element) {  content+="<tr>";                      //se accede al json con:element["id"] o con:element.id  content+="<td>"+element["id"]+"</td><td>"+element.nombre+"</td>";  content+="</tr>";  });  document.getElementById("datosTabla").innerHTML=content;  });    }  </script>  </html> |

### 11 Acceso al DOM

El modelo de objeto de documento (DOM) es una interfaz de programación para los documentos HTML y XML. Facilita una representación estructurada del documento y define de qué manera los programas pueden acceder, al fin de modificar, tanto su estructura, estilo y contenido. El DOM da una representación del documento como un grupo de nodos y objetos estructurados que tienen propiedades y métodos. Esencialmente, conecta las páginas web a scripts o lenguajes de programación.

Una página web es un documento. Éste documento puede exhibirse en la ventana de un navegador o también como código fuente HTML. Pero, en los dos casos, es el mismo documento. El modelo de objeto de documento (DOM) proporciona otras formas de presentar, guardar y manipular este mismo documento. El DOM es una representación completamente orientada al objeto de la página web y puede ser modificado con un lenguaje de script como JavaScript.

El [W3C DOM](http://www.w3.org/DOM/) estándar forma la base del funcionamiento del DOM. Varios navegadores ofrecen extensiones más allá del estándar W3C, hay que ir con extremo cuidado al utilizarlas en la web, ya que los documentos pueden ser consultados por navegadores que tienen DOMs diferentes.

Por ejemplo, el DOM de W3C especifica que el étodo ***getElementsByTagName***en el código de abajo debe devolver una lista de todos los elementos ***<p>*** del documento:

paragraphs = document.getElementsByTagName ("p");

// paragraphs[0] es el primer elemento <p>

// paragraphs[1] es el segundo elemento <p>, etc.

alert (paragraphs [0].nodeName);

Todas las propiedades, métodos y eventos disponibles para la manipulación y la creación de páginas web están organizados dentro de objetos. Un ejemplo: el objeto ***document*** representa al documento mismo, el objeto ***table*** hace funcionar la interfaz especial ***HTMLTableElement*** del DOM para acceder a tablas HTML, y así sucesivamente. Ésta documentación procura una relación objeto-por-objeto del DOM.

Dentro de un script JS tiene acceso a objetos de DOM, por ejemplo los objetos ***document*** o ***window***.

Algunos objetos importantes para manipular el DOM son:

|  |  |
| --- | --- |
| document | Cuando un miembro devuelve un objeto del tipo document (por ejemplo, la propiedad **ownerDocument** de un elemento devuelve el documento "document" al cual pertenece), este objeto es la raíz del objeto documento en sí mismo. |
| element | element se refiere a un elemento o a un nodo de tipo de elemento "element" devuelto por un miembro del API de DOM. Dicho de otra manera, por ejemplo, el método document.createElement() devuelve un objeto referido a un nodo, lo que significa que este método devuelve el elemento que acaba de ser creado en el DOM. Los objetos element ponen en funcionamiento a la interfaz Element del DOM y también a la interfaz de nodo "Node" más básica, las cuales son incluidas en esta referencia. |
| nodeList | Una "nodeList" es una serie de elementos, parecido a lo que devuelve el método document.getElementsByTagName(). Se accede a los items de la nodeList de cualquiera de las siguientes dos formas:   * list.item (1) * lista [1]   Ambas maneras son equivalentes. En la primera, **item()** es el método del objeto nodeList. En la última se utiliza la típica sintaxis de acceso a listas para llegar al segundo ítem de la lista. |
| attribute | Cuando un atributo ("attribute") es devuelto por un miembro (por ej., por el método **createAttribute()**), es una referencia a un objeto que expone una interfaz particular (aunque limitada) a los atributos. Los atributos son nodos en el DOM igual que los elementos, pero no suelen usarse así. |
| NamedNodeMap | Un namedNodeMap es una serie, pero los ítems son accesibles tanto por el nombre o por un índice, este último caso es meramente una conveniencia para enumerar ya que no están en ningún orden en particular en la lista. Un NamedNodeMap es un método de ítem() por esa razón, y permite poner o quitar ítems en un NamedNodeMap |

A continuación, una lista de métodos comunes de acceso al DOM:

document.getElementById(id)

Devuelve una referencia aun elemento por si id, ejemplo:

|  |
| --- |
| Archivo: getElementById.html |
| <html>  <head>  <title>Ejemplo getElementById</title>  </head>  <body>  <p id="para">Cualquier texto acá</p>  <button onclick="changeColor('blue');">Azul</button>  <button onclick="changeColor('red');">Rojo</button>  </body>  <script>  function changeColor(newColor) {  var elem = document.getElementById('para');  elem.style.color = newColor;  }  </script>  </html> |

element.getElementsByTagName(name)

Devuelve una lista de elementos con un nombre determinado.

|  |
| --- |
| Archivo: getElementByTagName.html |
| <html>  <head>  <title>ejemplo de getElementsByTagName</title>  <script>  function getAllParaElems(){  var allParas = document.getElementsByTagName("p");  var num = allParas.length;  alert("Hay " + num + " <p> elementos en este documento");  }  function div1ParaElems(){  var div1 = document.getElementById("div1")  var div1Paras = div1.getElementsByTagName("p");  var num = div1Paras.length;  alert("Hay " + num + " <p> elementos en el elemento div1");  }  function div2ParaElems(){  var div2 = document.getElementById("div2")  var div2Paras = div2.getElementsByTagName("p");  var num = div2Paras.length;  alert("Hay " + num + " <p> elementos en el elemento div2");  }  </script>  </head>  <body style="border: solid green 3px">  <p>Algo de texto</p>  <p>Algo de texto</p>  <div id="div1" style="border: solid blue 3px">  <p>Algo de texto en div1</p>  <p>Algo de texto en div1</p>  <p>Algo de texto en div1</p>  <div id="div2" style="border: solid red 3px">  <p>Algo de texto en div2</p>  <p>Algo de texto en div2</p>  </div>  </div>  <p>Algo de texto</p>  <p>Algo de texto</p>  <button onclick="getAllParaElems();">  muestra todos los elementos p en el documento</button><br />  <button onclick="div1ParaElems();">  muestra todos los elementos p en div1</button><br />  <button onclick="div2ParaElems();">  muestra todos los elementos p en div2</button>  </body>  </html> |

## V.5. Procedimiento

1. En la sección de ejercicios se describen una serie de problemas en los que es necesario aplicar lenguaje JS para darles solución.
2. Elaborar el análisis del problema codificar adecuadamente una solución.
3. Codificar el programa en lenguaje JS.
4. Probar el programa en 3 navegadores diferentes.

## V.6. Prácticas

1. Esciba una página web que muestre un saludo en una ventana emergente al iniciar.
2. Repita el ejercicio anterior, pero el saludo debe mostrarse al presionar un boton.
3. Defina una cadena JSON para representar la información de los alumnos del curso de cliente servidor 2, los datos requeridos son matricula, nombre, sexo y tres calificaciones por alumno.
4. Defina una cadena JSON para representar la información de los medicamentos en una farmacia, con datos id, nombre, descripción, caducidad, existencia, laboratorio que lo produce, costo de compra y costo de venta.
5. Defina una cadena JSON para representar la información del plan de estudio de las carreras de la FCI (clave y nombre de carrera, gestor y lista de materias por semestre).
6. Escriba una página web que capture los datos de 3 alumnos con los siguientes campos: matricula, nombre y edad, los datos deberá almacenarlos en una variable de memoria JSON y mostrarlos en una tabla en la misma página al presionar un botón.
7. En la carpeta “ejemplos\js-puro\json” pruebe los ejercicios que ahí se encuentran y que muestran el uso de JSON.

# PRACTICA VI. Mysql

## VI.1. Objetivo

El alumno será capaz de implementar sus bases de datos en MySql.

## VI.2. Equipo

Computadora personal, dispositivo de almacenamiento secundario (disquete, memoria USB, etc.), lenguaje de programación pascal, c, java, etc.

## VI.3. Materiales

Hojas de papel bond o cuaderno, lápiz o boligrafo.

## VI.4. Descripción

*HeidiSQL*es un software libre y de código abierto que permite conectarse a servidores *MySQL*, *MariaDB*, *Percona Server*, *Microsoft SQL Server* y *PostgreSQL*. Nosotros lo usaremos para trabajar con *MariaDB*.

Descargue Heidi desde: sitio:http://www.heidisql.com/download.php

e instálelo.

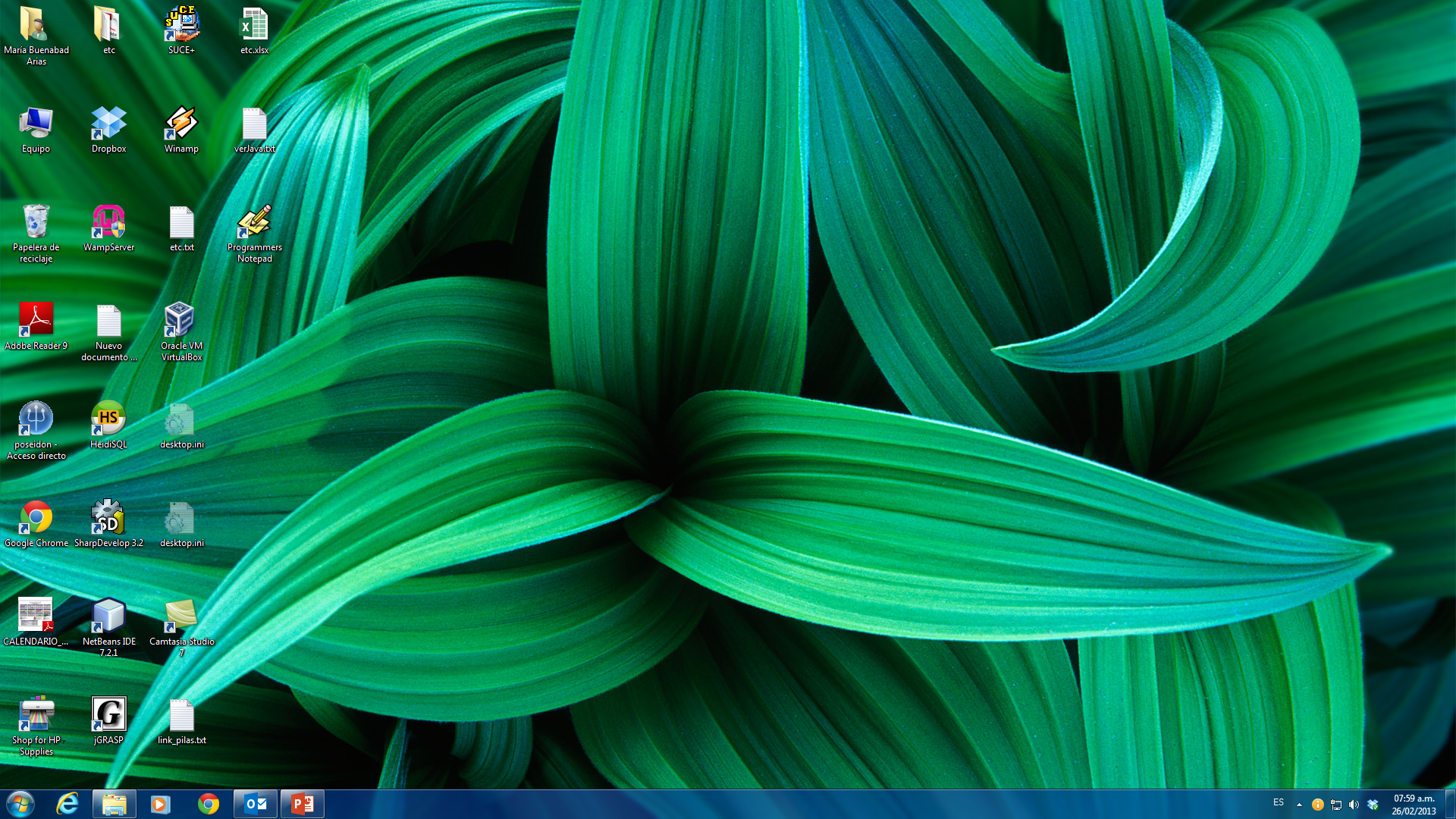
Necesitamos:

* HeidiSQL
* MySql

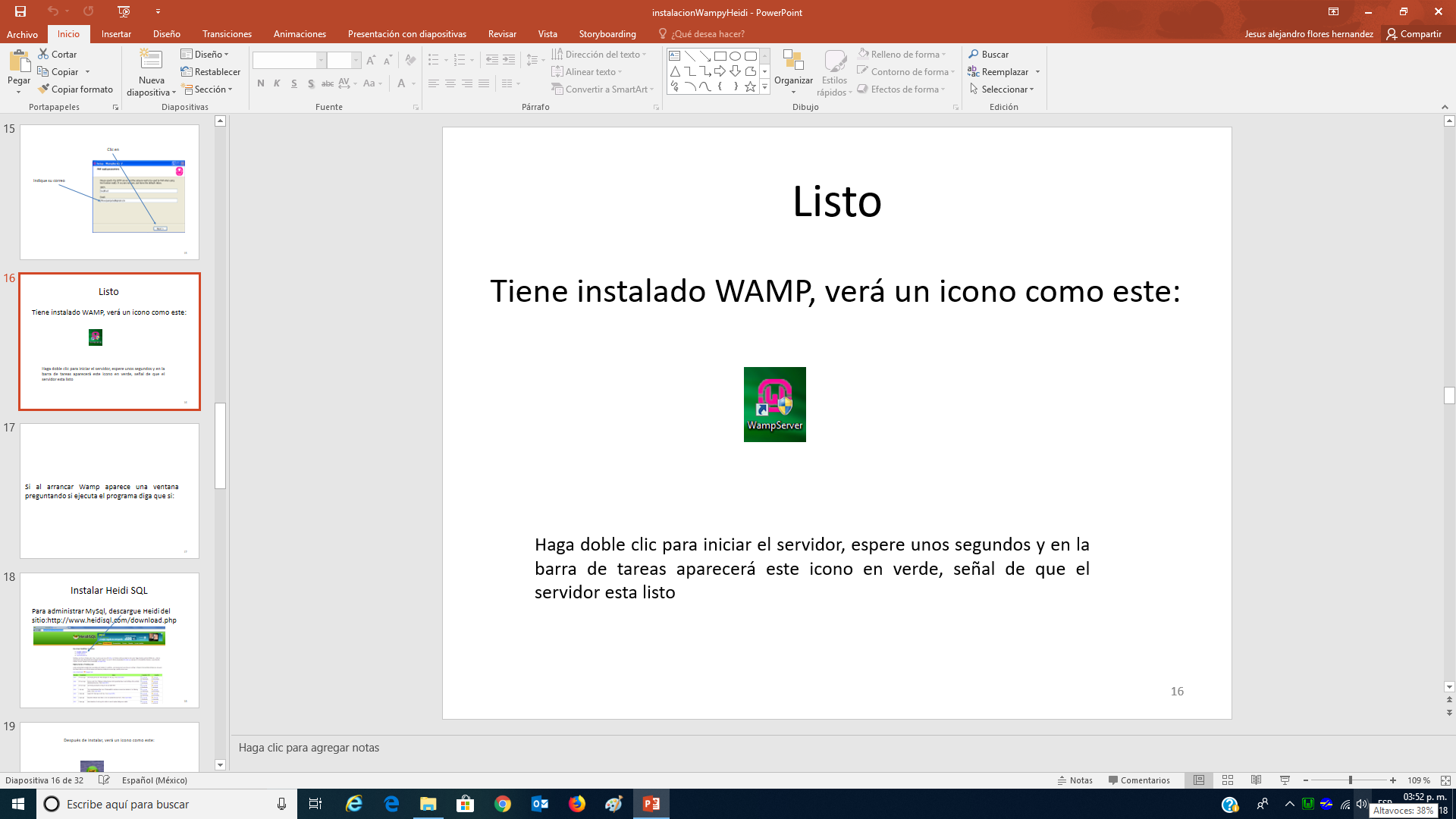
**Iniciar MySql**

Si no tiene iniciado MySql, inicialó ejecutando su WAMPSERVER. Es recomendable no tener otro *server* en ejecución, así que, si tiene alguno activo, desactívalo.

Para iniciar WAMP



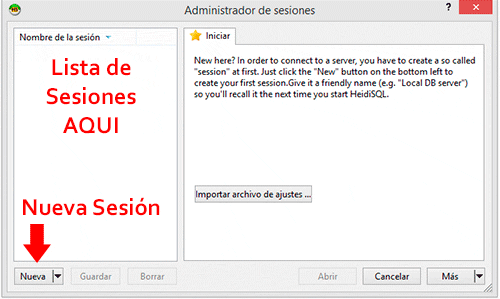
Haga doble clic en este ícono, para iniciar el servidor, espere unos segundos y en la barra de tareas aparecerá este icono en verde, señal de que el servidor está listo:



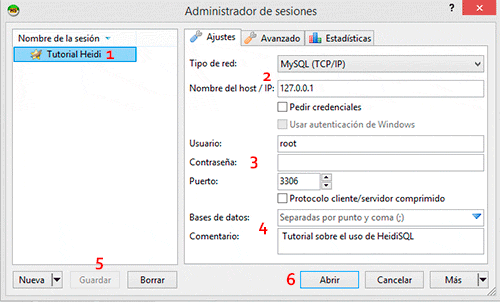
Unos instantes después deberá ver este ícono en color verde, indicando que está corriendo el servidor apache y el Mysql.

**Sesión en Heidi**

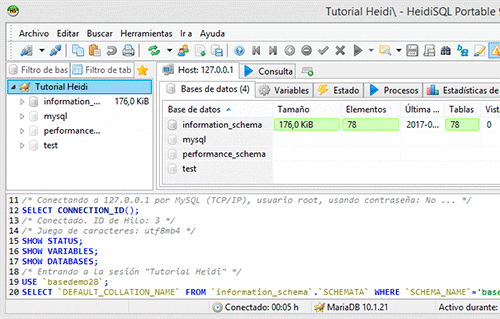
Arranque el HeidiSql

[](https://www.jc-mouse.net/wp-content/uploads/2017/04/sesion.gif)

HeidiSQL cuenta un sistema de sesiones para gestionar el trabajo. Cuando iniciamos el programa, debemos seleccionar con qué sesión queremos trabajar o en su defecto, crear una nueva sesión presionando el botón «***Nueva***».

[](https://www.jc-mouse.net/wp-content/uploads/2017/04/sesion_configuracion.gif)

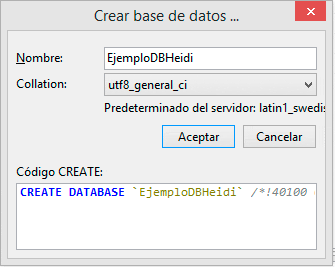
1: Escribimos el nombre de la sesión, por ejemplo «**Tutorial Heidi**»  
2: Seleccionamos el Tipo de Red (MySQL TCP/IP) y el nombre del Host: 127.0.0.1  
3: Escribimos el nombre de usuario *(root por defecto en MariaDB)* y contraseña *(si tuviera)*en blanco si no tiene. Podemos indicar también otro puerto de conexión.  
4: Seleccionamos el o las bases de datos con lo que trabajaremos*(no es obligatorio)*, caso contrario dejamos en blanco.  
5: Presionamos «*Guardar*» para grabar los cambios  
6: Y finalmente presionamos «*Abrir*»

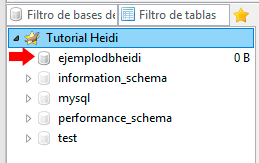
[](https://www.jc-mouse.net/wp-content/uploads/2017/04/gui_heidi.gif)

La interfaz de Heidi es intuitiva y además esta en español.

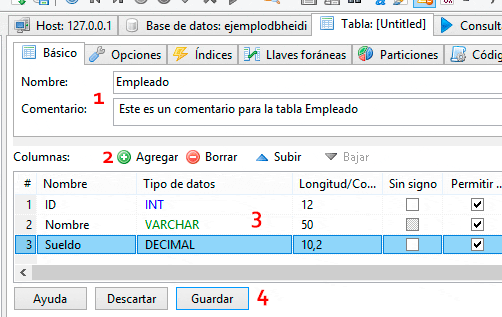
**Crear Base de Datos**

Hacemos clic derecho sobre el nombre de la sesión, elegimos «*Crear Nuevo*» → «*Base de Datos*«, se nos presenta una ventana de dialogo, donde escribimos el nombre de la base de datos «*EjemploDBHeidi*«, Seleccionamos el «*Collation*» (juego de caracteres asociadas a una base de datos).

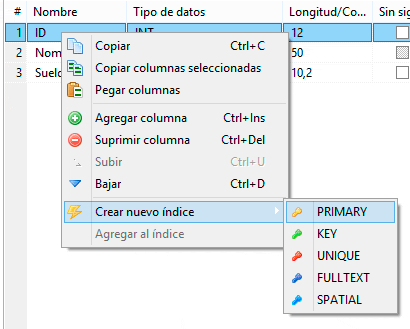
[](https://www.jc-mouse.net/wp-content/uploads/2017/04/mariadb_database.gif)  
Para terminar la creación, presionamos «*Aceptar*»

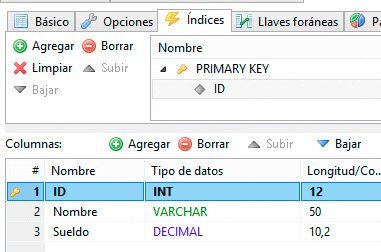
[](https://www.jc-mouse.net/wp-content/uploads/2017/04/database.gif)  
**Crear Tabla**

Clic derecho sobre la base de datos donde se quiere crear la tabla, seleccionar «*Crear nuevo*» → «*Tabla*«.  
1: Escribimos un nombre para la tabla  
2: Para crear nuevas columnas (campos) en la tabla, presionamos el botón «*Agregar*»  
3: Escribimos nombre de la columna e indicamos sus propiedades  
4: Para salvar los cambios, presionamos «*Guardar*»

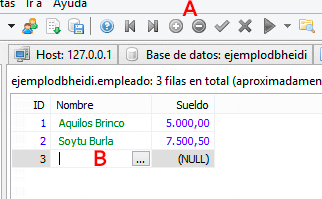
[](https://www.jc-mouse.net/wp-content/uploads/2017/04/table_maria.gif)  
**Llave primaria**

Clic derecho sobre el campo que queramos definir como Llave Primaria, «*Crear Nuevo Indice*» → **PRIMARY**

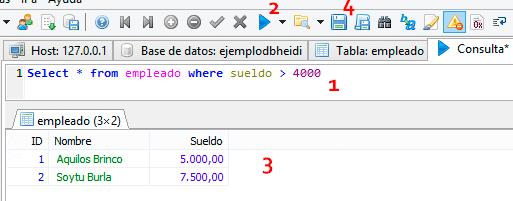
[](https://www.jc-mouse.net/wp-content/uploads/2017/04/indice_tabla.gif)  
El campo será marcado con una llave para indicar que corresponde a una **PRIMARY KEY**, además en la pestaña «*Indices*» se puede gestionar las llaves.

[](https://www.jc-mouse.net/wp-content/uploads/2017/04/primary_id.gif)  
**Agregar datos a tabla**

Para agregar datos a una tabla, seleccionamos la misma, y abrimos la pestaña «*Datos*»  
A: Para agregar o remover filas, utilizamos lo botones (+) y (-)  
B: Para agregar datos a cada campo, doble clic y escribir su contenido

[](https://www.jc-mouse.net/wp-content/uploads/2017/04/insert_mariadb.gif)  
**Consultas**

Para realizar consultas a la base de datos, contamos con la pestaña «*> Consulta*»  
1: Área donde escribir SQL  
2: Botón para ejecutar la consulta  
3: Área de resultados  
4: Te permite guardar la consulta en archivo con extensión \*.sql

[](https://www.jc-mouse.net/wp-content/uploads/2017/04/consulta_SQL.gif)

## VI.5. Procedimiento

1. En la sección de ejercicios se describen una serie de problemas en los que es necesario implementar bases de datos.
2. Diseñar sus bases de datos aplicxando formas normales.
3. Implementarlas en Mysql usando Heidi y WAMP.

## VI.6. Prácticas

1. Crear una base de datos que tenga las tablas de alumnos, carreras y profesores, isertar 3 datos de prueba en cada tabla.
2. Diseñar una base de datos para ventas a detalle de una tienda pequeña, deberá tener, productos, tipos de producto, proveedores y pedidos, impleméntelo en MySql.
3. Diseñar la base de datos para su proyecto de curso e implementarlo en Mysql.

# PRACTICA VII. PHP

## VII.1. Objetivo

El alumno será capaz de elaborar rutinas que enlacen la interfaz de una aplicación web con una base de datos.

## VII.2. Equipo

Computadora personal, dispositivo de almacenamiento secundario (disquete, memoria USB, etc.), lenguaje de programación pascal, c, java, etc.

## VII.3. Materiales

Hojas de papel bond o cuaderno, lápiz o boligrafo.

## VII.4. Descripción

1 Introducción.

En PHP vamos a comprobar qué tipo de navegador está utilizando el usuario visitante. Para hacerlo, vamos a comprobar el string del agente de usuario que el navegador envía como parte de la petición HTTP. Esta información es almacenada en una [variable](http://php.net/manual/es/language.variables.php). En PHP, las variables siempre comienzan con un signo de dólar. La variable que nos interesa ahora es [*$\_SERVER['HTTP\_USER\_AGENT']*](http://php.net/manual/es/reserved.variables.server.php).

[*$\_SERVER*](http://php.net/manual/es/reserved.variables.server.php) es una variable especial reservada por PHP que contiene toda la información del servidor web. Es conocida como una superglobal. Antes se podían usar en su lugar los antiguos arrays *$HTTP\_\*\_VARS*, tales como *$HTTP\_SERVER\_VARS*. A partir de PHP 5.4.0, estos antiguos arrays han sido eliminados.

pruebe el programa:

|  |  |
| --- | --- |
| index.php |  |
| <?php echo $\_SERVER['HTTP\_USER\_AGENT']; ?> |  |

Tabla 7.1 Código

Para el código de la tabla 7.1, captúrelo en su editor, guárdelo en la carpeta “C:\wamp\www\pruebas" o “C:\wamp64\www\pruebas" según haya instalado el WAMPO de 32 o 64 bits. Arranque el servidor y:

ejecútelo así: En su navegador escriba: <http://localhost/pruebas/index.php>

deberá ver lo siguiente:

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36

2 Bases de datos

Para los ejemplos de este apartado requerirá:

1. Tener instalado alguna versión de MySql con Apache, en estos ejemplos se usa WAMP server. Si no lo ha instalado vea el video:
   * <https://www.youtube.com/watch?v=5X7symb1QdE&list=PLExyZ7hD-J5b8ftq57xTwBceLXP6ErJzc&index=1>
2. Tener instalado Heidi SQL
3. Tener implementada la base de datos "provpar2", vea el código en la carpeta “ejercicios/baseDeDatos/provPar2.sql", puede generar la base de datos ejecutando este código en una ventana de consulta de HEIDI. Tamnbien puede crear esta base de datos a partir de los videos:
   * <https://www.youtube.com/watch?v=40YYmufKXus&list=PLExyZ7hD-J5b8ftq57xTwBceLXP6ErJzc&index=2>
   * <https://www.youtube.com/watch?v=5byoF-qF86U&list=PLExyZ7hD-J5b8ftq57xTwBceLXP6ErJzc&index=3>
4. Si desea saber cómo instalar Wamp y Heidi, vea el archivo de apoyo instalar WAMP y Heidi.

Base de datos provpar2:

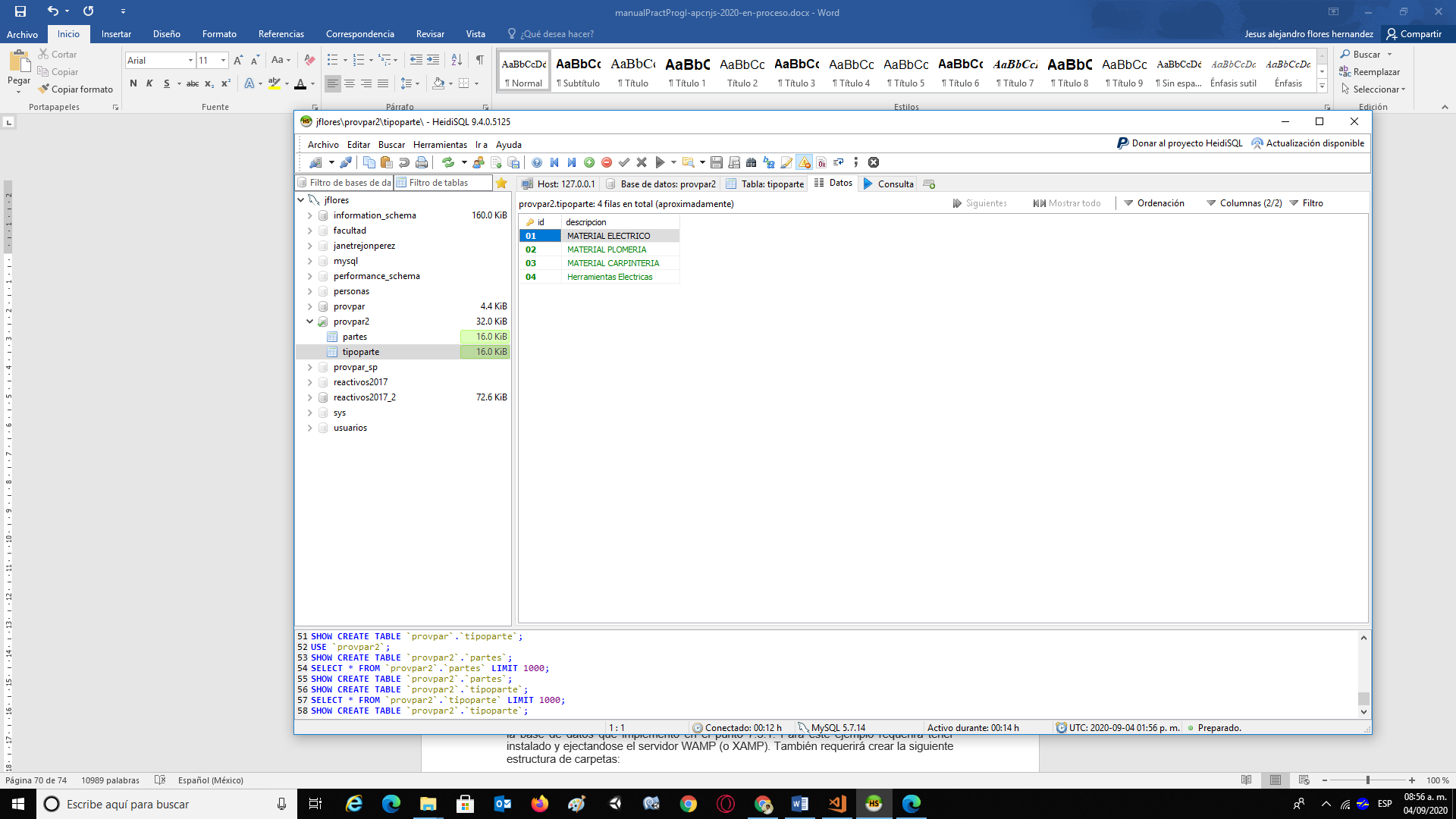


Tabla tipoparte de la base de datos provpar2



Tabla partes de la base de datos provpar2.

Aquí el código para crear la base de datos:

|  |
| --- |
| Copie este código en una ventana de consulat de heidisql y ejecútelo, esto creará su base de datos provpar2. |
| -- --------------------------------------------------------  -- Host: 127.0.0.1  -- Versión del servidor: 5.7.14 - MySQL Community Server (GPL)  -- SO del servidor: Win64  -- HeidiSQL Versión: 9.4.0.5125  -- --------------------------------------------------------  /\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;  /\*!40101 SET NAMES utf8 \*/;  /\*!50503 SET NAMES utf8mb4 \*/;  /\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;  /\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;  -- Volcando estructura de base de datos para provpar2  CREATE DATABASE IF NOT EXISTS `provpar2` /\*!40100 DEFAULT CHARACTER SET latin1 COLLATE latin1\_spanish\_ci \*/;  USE `provpar2`;  -- Volcando estructura para tabla provpar2.partes  CREATE TABLE IF NOT EXISTS `partes` (  `clave` varchar(10) COLLATE latin1\_spanish\_ci NOT NULL DEFAULT '',  `nombre` varchar(50) COLLATE latin1\_spanish\_ci DEFAULT NULL,  `tipo` varchar(10) COLLATE latin1\_spanish\_ci DEFAULT NULL,  `color` varchar(20) COLLATE latin1\_spanish\_ci DEFAULT NULL,  `existencia` int(11) DEFAULT NULL,  `sminimo` int(11) DEFAULT NULL,  `smaximo` int(11) DEFAULT NULL,  `peso` decimal(10,2) DEFAULT NULL,  `cc` decimal(10,2) DEFAULT NULL,  `cv` decimal(10,2) DEFAULT NULL,  PRIMARY KEY (`clave`)  ) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_spanish\_ci;  -- Volcando datos para la tabla provpar2.partes: ~8 rows (aproximadamente)  /\*!40000 ALTER TABLE `partes` DISABLE KEYS \*/;  INSERT INTO `partes` (`clave`, `nombre`, `tipo`, `color`, `existencia`, `sminimo`, `smaximo`, `peso`, `cc`, `cv`) VALUES    ('01', 'tuerca 4/8', '01', 'natural', 300, 200, 900, 1.00, 24.00, 30.00),    ('02', 'tuerca 3/8', '01', 'natural', 600, 700, 900, 2.00, 54.00, 60.00),    ('03', 'perno 1/2', '01', 'natural', 200, 100, 500, 3.00, 30.00, 35.00),    ('04', 'puerta madera', '02', 'blanco', 300, 50, 900, 5.00, 800.00, 1200.00),    ('05', 'taladro de banco', '02', 'blanco', 600, 100, 900, 10.00, 1400.00, 2200.00),    ('06', 'taladro de mano', '02', 'negro', 2, 1, 7, 7.00, 500.00, 700.00),    ('07', 'llave allen #2', '01', 'negro', 20, 10, 40, 1.00, 50.00, 80.00),    ('08', 'perno 3/8', '01', 'natural', 190, 200, 300, 1.00, 3.00, 20.00);  /\*!40000 ALTER TABLE `partes` ENABLE KEYS \*/;  -- Volcando estructura para tabla provpar2.tipoparte  CREATE TABLE IF NOT EXISTS `tipoparte` (  `id` varchar(10) COLLATE latin1\_spanish\_ci NOT NULL DEFAULT '',  `descripcion` varchar(80) COLLATE latin1\_spanish\_ci DEFAULT NULL,  PRIMARY KEY (`id`)  ) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1\_spanish\_ci;  -- Volcando datos para la tabla provpar2.tipoparte: ~3 rows (aproximadamente)  /\*!40000 ALTER TABLE `tipoparte` DISABLE KEYS \*/;  INSERT INTO `tipoparte` (`id`, `descripcion`) VALUES    ('01', 'MATERIAL ELECTRICO'),    ('02', 'MATERIAL PLOMERIA'),    ('03', 'MATERIAL CARPINTERIA'),    ('04', 'Herramientas Electricas');  /\*!40000 ALTER TABLE `tipoparte` ENABLE KEYS \*/;  /\*!40101 SET SQL\_MODE=IFNULL(@OLD\_SQL\_MODE, '') \*/;  /\*!40014 SET FOREIGN\_KEY\_CHECKS=IF(@OLD\_FOREIGN\_KEY\_CHECKS IS NULL, 1, @OLD\_FOREIGN\_KEY\_CHECKS) \*/;  /\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/; |

Tabla 7.2

### 3 Conexión a la base de datos.

Para conectarnos a la base de datos en PHP. Un programa en php se escribe en un archivo entre las marcas:

<?php

?>

Construyamos un código php para conectarnos a nuestra base de datos provpar:

|  |
| --- |
| Archivo: abre\_provpar.php |
| <?php      $hostname="localhost";      $dbname="provpar";      $username="root";      $pw="";      try{  $dbh = new PDO("mysql:host=$hostname;dbname=$dbname", $username, $pw);  echo '{"conexión abierta"}';      }catch(PDOException $e){          $error="conexion fallida: ".$e;          echo '{"error":'.$error.'}';          //echo "conexion fallida:".$e;      }  $dbh = null;  ?> |

Tabla 7.3

Este código abre la base de datos, y después la cierra con la sentencia ***$dbh=null,*** se agrega el try para capturar errores. Si todo va bien deberá ver en la ventana del navegador el mensaje “conexión abierta”. En caso de error se muestra en la ventana el motivo del error.

Como se ejecuta el programa:

1. Debe tener en ejecución el WAMP server.
2. Dentro de la carpeta ***wamp64/www/*** cree una nueva carpeta llamada: ***cursoapcnjs***
3. Abra su editor de código Miocrosoft Visual studio code (o programersNotepad si lo desea).
4. Escriba o copie el código del programa ***abre\_provpar.php*** que aquí se muestra, en el editor, guardelo en la carpeta ***wamp64/www/cursoapcnjs,*** con el nombre ***abre\_provpar.php.***
5. Abra una ventana de su navegador (Chrome, mozilla, Edge u opera)
6. En su ventana del navegador se ejecuta así:
   * En su navegador escriba: <http://localhost/cursoapcnjs/abre_provpar.php>
   * Deberá ver un mensaje en su ventana

Veamos ahora una consulta a la tabla ***tipoparte***:

|  |
| --- |
| Archivo: traeDatos.php |
| <?php      $hostname="localhost";      $dbname="provpar2";      $username="root";      $pw="";      try{  $dbh = new PDO("mysql:host=$hostname;dbname=$dbname", $username, $pw);  // la consulta  $sql = 'select \* from tipoparte';  // usar la sentencia prepara para verificar la sentencia  $stmt = $dbh->prepare( $sql );  // ejecuta la sentencia  if($stmt->execute()){              // poner el resultado de la consulta en el arreglo result              $result = $stmt->fetchAll( PDO::FETCH\_ASSOC );              // convertir a formato json              $json = json\_encode( $result );              // mandar la salida              echo $json;          }else{              echo '{"error":"no se pudo realizar la consulta"}';          }      }catch(PDOException $e){          $error="conexion fallida: ".$e;          echo '{"error":'.$error.'}';          //echo "conexion fallida:".$e;      }      $dbh = null;  ?> |

Tabla 7.4

Aquí se abre la base de datos como en la tabla 7.3 y se agrega el código para realizar la consulta, la sentencia:

**$sql = 'select \* from tipoparte';**

Almacena la consulta en la variable $sql

La sentencia:

$stmt = $dbh->prepare( $sql );

No es obligatoria, pero es recomendable para verificar algunos errores.

La sentencia:

***$stmt->execute()***

Ejecuta la sentencia, devuelve null si no pudo ejecutarla o asocia un arreglo asociativo de php a la variable $stmt con los datos encontrados. Por tanto, al meterla al if verificamos si hay datos al momento de ejecutarla. Importante si la consulta esta vacia no devuelve null, devuelve un arreglo asociativo vacio, si no hay conexión o no existe la tabla o esta mal la consulta devuelve null y se ejecuta el else.

if($stmt->execute()){

            // poner el resultado de la consulta en el arreglo result

            $result = $stmt->fetchAll( PDO::FETCH\_ASSOC );

            // convertir a formato json

            $json = json\_encode( $result );

            // mandar la salida

            echo $json;

        }else{

            echo '{"error":"no se pudo realizar la consulta"}';

        }

Dentro del if, si trae datos la consulta, con la sentencia:

$result = $stmt->fetchAll( PDO::FETCH\_ASSOC );

Devuelve los datos del arreglo asociativo y los almacena en la variable $result.

Finalmente, la sentencia:

***$json = json\_encode( $result );***

***echo $json;***

Codifica el arreglo asociativo en formato json y con ***echo*** lo enviamos al navegador.

Aunque no lo usaremos, por curiosidad podemos ver como se muestran los valores sin convertirlos a json, cambie las líneas de código:

|  |
| --- |
| ***$json = json\_encode( $result );***  ***echo $json;*** |
| Por estas: |
| var\_dump($result); |

Tabla 7.5

Ahora debe ver los datos en el formato de arreglo asociativo.

Deje el código como antes para que devuelva un arreglo json.

**Insertar datos**

Para insertar datos primero veremos como enviar datos desde un archivo HTML, primera forma:

Necesitamos 2 archivos, el primero un html con un formulario, el segundo un php que reciba los datos:

|  |
| --- |
| Archivo: inserta.html |
| <!DOCTYPE html>  <html>  <body>  <form action="inserta.php" method="POST">  id<input type="text" name="id">  descripcion<input type="text" name="descripcion">  <input type="submit">  </form>  </body>  </html> |
| Archivo: inserta.php |
| <?php      //require\_once('php/clsPartes.php');      //$objpartes=new clsPartes();      if(!empty($\_POST)){  $id = $\_POST['id'];  $descripcion = $\_POST['descripcion'];  echo "id=".$id." desc=".$descripcion;  }else{  echo "sin datos";  }  ?> |

Tabla 7.6

El artchivo html tiene un formulario con 2 campos, con atributo name: ***id*** el primero y ***descripción*** el segundo. El formulario está sociando en su atributo action al archivo inserta.php, de manera que al hacer clic en el botón submit, se ejecutara el archivo ***inserta.php*** enviando los parametros ***id*** y ***descripción*** que se obtienen del contenido de los cuadros de texto.

En el archivo inserta.php con el ***if(empty($\_POST))*** verifica que se hayan recibido datos con el método post, de ser así, con las sentencias:

$id = $\_POST['id'];

$descripcion = $\_POST['descripcion'];

Obtiene los datos y con ***echo*** los muestra en la ventana.

Copie estos archivos a su navegador, guárdelos con el nombre indicado en la carpeta ***wamp64/www/cursoapcnjs*** , pruebe así:

En su navegador ponga la sentencia: <http://localhost/cursoapcnjs/inserta.html> , ponga información en los dos campos y haga clic en el botón, deberá ver lainformación de los campos en una ventana nueva. Con esto esta comprobando que el PHP recibe y devuelve bien la información desde/hacia JS.

### 4 JS-fetch

En la insercion de datos, en el ejemplo de la tabla 7.6 se muestra la forma clásica de usar php, sin embargo, nosotros no lo usaremos de esta forma, trabajaremos así:

1. Craremos la interface en html+css+bootstrap.
2. Craremos las consultas de inserción de datos en php.
3. Crearemos rutinas en JS que tomen los datos de inserción de los formularios de html y los envíen a php.
4. Crearemos rutinas JS que tomarán los resultados de la consulata y los enviaremos a html.

Para el punto 3 usaremos la sentencia fetch que es muy eficiente a la hora de enviar y recibir datos desde el back end, la sentencia fetch trabaja de manera asíncrona usando un concepto en JS llamado promesas (Promise), para entender mejor las promesas vea una explicación en:

* Promesas: <https://docs.google.com/document/d/1JZg8oCON5LG7j42r9l2sKdfRqvz4qf9DWKsOL2XEKVk/edit?usp=sharing>
* Una explicación de fetch y php: <https://docs.google.com/presentation/d/1WNqsq268kCPe4XUBkIKx5f-Dx4S7yorjiVyGnpnltKA/edit?usp=sharing>
* fetch envía datos a php: <https://docs.google.com/presentation/d/1Q3nFXiDohNoaFkwbcs7LSncihtR1DHFGwOaITnO-CJ0/edit?usp=sharing>
* fetch envía formulario: <https://docs.google.com/presentation/d/1ukDY8TcTNGk5F1AnX1MEPLyFC0sJKsjr9SqlpnMNoaE/edit?usp=sharing>
* Fetch enviar parámetros en url:<https://docs.google.com/presentation/d/1HMD4jTeg3zOOiKsrEqDlsvKUmEQfMMHGcksR9OUoL4o/edit?usp=sharing>
* fetch leer json: <https://docs.google.com/presentation/d/1PsWqwIe---sbqWFGBd4ormqIjZNJQS0U5k1CNmMphMA/edit?usp=sharing>

Trabajaremos con fetch así:

HTML 🡨🡪 JS-fetch 🡨🡪PHP🡨🡪 Base de datos

Veamos como tomar datos de HTML para enviarlos a php.

HTML 🡪 JS-fetch 🡪PHP

Para esto dberemos:

Crear el archivo html para capturar datos:

|  |
| --- |
| Archivo: fetch-basico-envia-formulario.html |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  </head>  <body>  <form id="miformulario">  <input type="number" name="movimiento" placeholder="1">  <input type="text" name="id" placeholder="id">  <input type="text" name="nombre" placeholder="nombre">  <input type="text" name="apellido" placeholder="apellido">  <input type="number" name="edad" placeholder="18">  <input type="button" id="mi\_boton" onclick="enviarDatos()" value="enviar datos">  </form>  <div id="div1"></div>  </body>  <script>  function enviarDatos(){  const data = new FormData(document.getElementById('miformulario'));  var url = 'recibe\_Form\_devuelveTXT.php';  fetch(url, {  method: 'POST',  body: data  }).then(res => res.text())  .catch(error => console.error('Error:', error))  .then(response => {  console.log('Success:', response);  document.getElementById("div1").innerText=response;  });  }  </script>  </html> |

Archivo php que recibe los datos del formulario y los envía a la base de datos, en este ejemplo para probar el funcionamiento, no se envían los datos a la base de datos, si no que se devuelven al JS para verificar la comunicación:

|  |
| --- |
| Archivo: recibe\_Form\_devuelveTXT.php |
| <?php  header("Content-Type: text/html;charset=utf-8");  if(! empty($\_POST)){  //se enviaron parametros por el método \_GET  $movimiento=(isset($\_POST["movimiento"])?$\_POST["movimiento"]:"ninguno");  $id=(isset($\_POST["id"])?$\_POST["id"]:"");  $nombre=(isset($\_POST["nombre"])?$\_POST["nombre"]:"");  $apellido=(isset($\_POST["apellido"])?$\_POST["apellido"]:"");  $edad=(isset($\_POST["edad"])?$\_POST["edad"]:0);  echo "recibi mov:".$movimiento.", id:".$id.",nombre: ".$nombre.", apellido:".$apellido.", edad:".$edad;  }else{  echo "sin datos recibidos";  }  exit();  ?> |

Pruebelo así:

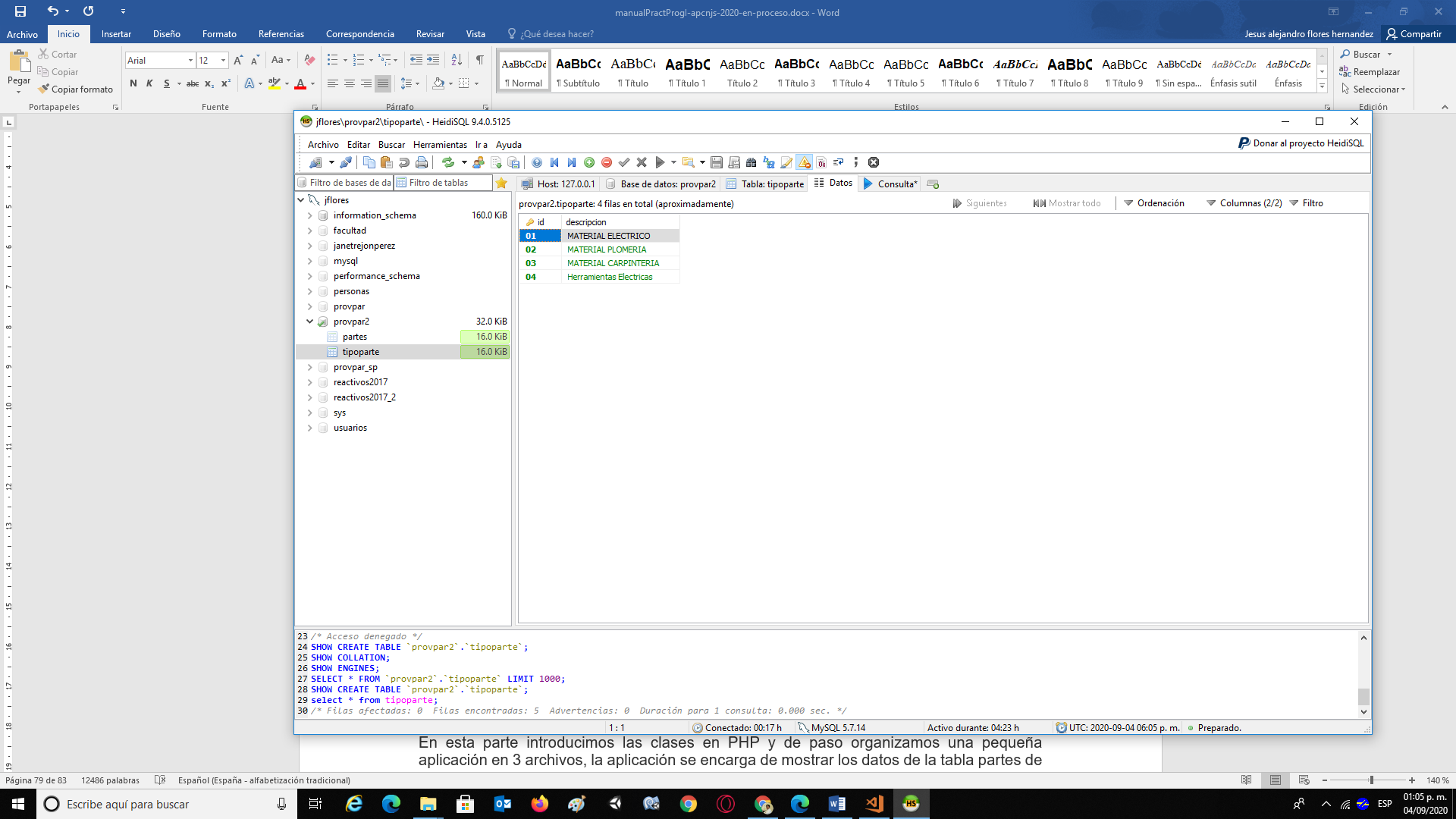
1. Guarde los dos archivos en la carpeta: C:\wamp64\www\cursoapcnjs
2. En su navegador ejecute así: <http://localhost/cursoapcnjs/fetch-basico-envia-formulario.html>
3. Como resultado debe ver que el html devuelve la misma información que recibe.

### 5 Insertar registro con fetch

Veamos como insertar un registro.

HTML 🡪 JS-fetch 🡪PHP🡪 Base de datos

Insertaremos un nuevo tipo de partes en la tabla tipoparte de provpar que tiene los campos id y descripción:



Necesitaremos 2 archivos:

|  |
| --- |
| Archivo: inserta\_tipo\_articulo.html |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  </head>  <body>  <p>Inserción de datos a la tabla tipo de articulo</p>  <form id="miformulario">  <input type="text" name="id" placeholder="id">  <input type="text" name="descripcion" placeholder="nombre">  <input type="button" id="mi\_boton" onclick="enviarDatos()" value="enviar datos">  </form>  <div id="div1"></div>  </body>  <script>  function enviarDatos(){  //http://localhost/ejercicios/json-fetch/fetch-basico/fetch-basico-envia-formulario.html  const data = new FormData(document.getElementById('miformulario'));  var url = 'inserta\_tipo\_articulo.php';  fetch(url, {  method: 'POST',  body: data  }).then(res => res.text())  .catch(error => console.error('Error:', error))  .then(response => {  console.log('Success:', response);  document.getElementById("div1").innerText=response;  });  }  </script>  </html> |
| Archivo: inserta\_tipo\_articulo.php |
| <?php        header('Content-type: text/html; charset=UTF-8') ;      //tabla a la que se le hacen los movimientos      $mi\_tabla='tipoparte';    $hostname="localhost";      $dbname="provpar2";      $username="root";  $pw="";  if (isset($\_POST) && count($\_POST)>0){          $id=(isset($\_POST["id"])?$\_POST["id"]:"");          $descripcion=(isset($\_POST["descripcion"])?$\_POST["descripcion"]:"");  //  $sql="insert into $mi\_tabla(id,descripcion)";  $sql.=" values('$id','$descripcion');";  try{  $dbh = new PDO("mysql:host=$hostname;dbname=$dbname", $username, $pw);  // usar la sentencia prepare para verificar la sentencia  $stmt = $dbh->prepare( $sql );  // ejecuta la sentencia  if($stmt->execute()){  echo '{"error":"insersion sin error"}';  }else{  echo '{"error":"no se pudo realizar la consulta"}';  }  }catch(PDOException $e){  $error="conexion fallida: ".$e;  echo '{"error":'.$error.'}';  //echo "conexion fallida:".$e;  }  $dbh = null;      }      exit();  ?> |

Explicación

En el archivo html. Tenemos un formulario y un botón asociado en su evento clic a la función enviarDatos(). Al dar clic se ejecuta la función de JS, el la función de JS enviarDatos() se realízalo siguiente, la sentencia:

const data = new FormData(document.getElementById('miformulario'));

Crea la variable data con la información del formulario, la sentencia:

***var url = 'inserta\_tipo\_articulo.php';***

declara una variable con el nombre del archivo php a ejecutar.

La sentencia:

fetch(url, {

method: 'POST',

body: data

}).then(res => res.text())

.catch(error => console.error('Error:', error))

.then(response => {

console.log('Success:', response);

document.getElementById("div1").innerText=response;

});

Realiza todo el trabajo, Indica que se enviarán los datos por el método post, en el body pone los datos obtenidos del formulario, recuerde que el url tiene la dirección del archivo php a ejecutar.

A continuación, una lista de documentos que explica a mas detalle el uso de fetch y las promesas:

### 6 Recibir datos con fetch desde php

HTML 🡨 JS-fetch 🡨PHP🡨Base de datos

A continuación, unejemplo para recibir datos desde php en JS y ponerlos en una tabla de html:

|  |
| --- |
| Archivo: muestraPersonas.html |
| <!DOCTYPE html>  <html>  <head>  <style>  table{ width:50%; }  td{ text-align:center; }  tr:hover{ background-color:#edd; }  tr:nth-child(even){background-color:#f2f2f2;}  </style>  </head>  <body>  <div id="area\_trabajo">  <table>  <thead>  <tr>  <th>id</th>  <th>nombre</th>  <th>edad</th>  <th>ciudad</th>  <th>trabajo</th>  </tr>  </thead>  <tbody id="datosTabla">  </tbody>  </table>  </div>  </body>  <script>  //http://localhost/ejercicios/json-fetch/ejemploPersonas/muestraPersonas.html  var personas={};  //fetch('traePersonas-mysqli.php')  fetch('traePersonasPDO')  .then(function(response) {    return response.json();  })  .then(function(data) {  //console.log('data = ', data);  personas=data.salida;  //console.log(personas);  var content="";  personas.forEach(function(element) {  content+="<tr>";                      //nota: como se ve a continuación, se accede al json con: element["id"] o con:element.id  content+="<td>"+element["id"]+"</td><td>"+element.nombre+"</td>";  content+="<td>"+element["edad"]+"</td><td>"+element['ciudad']+"</td>";  content+="<td>"+element.trabajo;  content+="</tr>";  });  document.getElementById("datosTabla").innerHTML=content;  })  .catch(function(err) {  console.error(err);  });  </script>  </html> |
| Archivo:traePersonasPDO.php |
| <?php  //http://localhost/ejercicios/json-fetch/ejemploPersonas/traePersonasPDO.php      //crear la conexion a la base de datos      try{          //intenta la conexión          $conexion = new PDO("mysql:host=localhost;dbname=personas;charset=utf8", "root", "");      } catch (PDOException $e) {          //atrapa el error          echo "Falla al obtener un manejador de BD: ".$e->getMessage() . "\n";          exit();      }        // realizar la consulta y llenar la tabla      $query = $conexion->prepare("SELECT \* FROM persona;");      $query->execute();      if($query->rowCount() > 0){          $userData = $query->fetchAll(PDO::FETCH\_ASSOC);          $datos["status"]="ok";  $datos["salida"]=$userData;      }else{          $datos["status"]="ok";  $datos["salida"]=NULL;      }      unset($conexion);  echo json\_encode($datos);      //var\_dump($userData);  ?> |

### 7 Clases en PHP (tema opcional)

En esta parte introducimos las clases en PHP y de paso organizamos una pequeña aplicación en 3 archivos, la aplicación se encarga de mostrar los datos de la tabla partes de la base de datos que implementó en el punto 7.3.1. Para este ejemplo requerirá tener instalado y ejectandose el servidor WAMP (o XAMP). También requerirá crear la siguiente estructura de carpetas:

Dentro de la carpeta: C:\wamp\www\ agregue la estructura de carpetas:

pruebas

└─ cursoweb

└─ aplic01

└─ php

Dentro de la carpeta PHP agregue los archivos:

1. clsConexion.php
2. clsPartes.php
3. modeloPartes.php

El contenido de los archivos se muestra en la tabla Tabla 7.3.3.1.

Los datos son mostrados en una ventana WEB con formato JSon. No generamos el formato con HTML pues el formato JSon lo usaremos más adelante en el punto de Angular.

Para ejecutar, en su navegador ejecute: http://localhost/pruebas/cursoweb/aplic01/modelopartes.php

Si todo va bien, deberá ver:

{"registros":[{"clave":"01","nombre":"tuerca 4\/8","tipo":"01","color":"natural","existencia":"300","sminimo":"200","smaximo":"900","peso":"1.00","cc":"24.00","cv":"30.00"},{"clave":"02","nombre":"tuerca 3\/8","tipo":"01","color":"natural","existencia":"600","sminimo":"700","smaximo":"900","peso":"2.00","cc":"54.00","cv":"60.00"},{"clave":"03","nombre":"perno 1\/2","tipo":"01","color":"natural","existencia":"200","sminimo":"100","smaximo":"500","peso":"3.00","cc":"30.00","cv":"35.00"},{"clave":"04","nombre":"puerta madera","tipo":"02","color":"blanco","existencia":"300","sminimo":"50","smaximo":"900","peso":"5.00","cc":"800.00","cv":"1200.00"},{"clave":"05","nombre":"taladro de banco","tipo":"02","color":"blanco","existencia":"600","sminimo":"100","smaximo":"900","peso":"10.00","cc":"1400.00","cv":"2200.00"}],"mensaje":"ok"}

Mostrando mejor los datos vemos su estructura JSon:

|  |
| --- |
| {"registros":[      {"clave":"01","nombre":"tuerca 4\/8","tipo":"01","color":"natural","existencia":"300","sminimo":"200","smaximo":"900","peso":"1.00","cc":"24.00","cv":"30.00"},      {"clave":"02","nombre":"tuerca 3\/8","tipo":"01","color":"natural","existencia":"600","sminimo":"700","smaximo":"900","peso":"2.00","cc":"54.00","cv":"60.00"},      {"clave":"03","nombre":"perno 1\/2","tipo":"01","color":"natural","existencia":"200","sminimo":"100","smaximo":"500","peso":"3.00","cc":"30.00","cv":"35.00"},      {"clave":"04","nombre":"puerta madera","tipo":"02","color":"blanco","existencia":"300","sminimo":"50","smaximo":"900","peso":"5.00","cc":"800.00","cv":"1200.00"},      {"clave":"05","nombre":"taladro de banco","tipo":"02","color":"blanco","existencia":"600","sminimo":"100","smaximo":"900","peso":"10.00","cc":"1400.00","cv":"2200.00"}  ],  "mensaje":"ok"} |

|  |
| --- |
| Archivo: clsConexion.php |
| <?php class clsConexion{     public $dbh=NULL;     public $hostname="localhost";     public $dbname="provpar2";     public $username="root";     public $pw="";     public $error="";     function \_\_construct() {         try{             $this->dbh = new PDO("mysql:host=$this->hostname;dbname=$this->dbname", $this->username, $this->pw);             $this->error="";         } catch (PDOException $e) {             $this->error="Falla al obtener un manejdor de BD: ".$e->getMessage() . "\n";             $this->dbn=NULL;         }     }     public function desconecta(){         unset($this->dbh);         $this->dbn=NULL;     } } |
| Archivo: clsPartes.php |
| <?php require\_once('clsConexion.php'); class clsPartes{     public $miTabla="partes";     //devuelve la lista de datos de la tabla $this->miTabla     public function traePartes(){         $respuesta = "";         try{             $sql = "select \* from $this->miTabla";             $conexion = new clsConexion();             $stmt = $conexion->dbh->prepare( $sql );             $stmt->execute();             $result = $stmt->fetchAll( PDO::FETCH\_ASSOC );             if($stmt->rowCount()>0){                 $arr=array('registros'=> $result, 'mensaje'=>'ok');                 $json = json\_encode($arr);                 echo $json;             }else{                 $arr=array('registros'=> '', 'mensaje'=>'0 registros');                 $json = json\_encode( $arr );                 $respuesta=$json;             }          }catch(Exception $e){             $arr=array('registros'=> NULL, 'mensaje' =>$e->getMessage());             $json = json\_encode( $arr );             $respuesta=$json;         }         $conexion->desconecta();         return $respuesta;     }//fin de traePartes } |
| Archivo: modeloPartes.php |
| <?php     require\_once('clsPartes.php');     $partes=new clsPartes();     echo $partes->traePartes(); |

Tabla 7.3.3.1.

## VII.4. Prácticas

1. Escriba una rutina en php que consulte una tabla de base de datos y muestre la información en formato json.
2. Escriba una rutina en php que reciba datos con formato GET y los inserte en una tabla de base de datos, pruébela enviando datos desde la línea de ejecución del navegador.
3. Escriba una rutina en PHP que reciba datos en formato json y los envie a la ventana del navegador.
4. Escriba las rutinas de insertar datos de su proyecto en php, debe recibir información en post o json e insertar en la tabla adecuada.
5. Escriba las rutinas de consulta de datos de su proyecto, debe realizar la consulta a la base de datos con un select y devolver una cadena json.
6. Escriba las rutinas de buscar datos de su proyecto.

# FUENTES CONSULTADAS

**[Joyanes1998]** Joyanes Aguilar, Luis y Zahoneri Martínez, Ignacio, (1998) Estructura de datos. Algoritmos, abstracción y objetos. Ed. McGrawHill, España.

**[Joyanes2003]** Joyanes Aguilar, Luis (2003) Fundamentos de la programación algoritmos y estructuras de datos. Tercera edición, Ed. McGrawHill. España.

**[Oviedo2003]** Oviedo Regino, Efrain (2003) Lógica de programación. Ed. Ecoediciones. España.

**[Tenembaum1996]** Tenembaum, Aaron M. Tenembaum, Aarón M., Langsam, Yedidhah y Augenstein, Moshe A., (1996) Estructuras de datos con C y C++. Segunda Edición – Prentice hall, México.

**[Goodrich2002]** Goodrich, Michael T., Tamassia, Robert, (2002). Estructura de datos y algoritmos en Java. Segunda Edición – ed. CECSA, México.

**[www3schools/php]** <https://www.w3schools.com/php/DEFAULT.asp>

**[mozila/html]** <https://developer.mozilla.org/es/docs/Web/HTML>

**[mozila/js]**  <https://developer.mozilla.org/es/docs/Web/JavaScript>

**[www3schools/bootstrap]** <https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp>

**[php]** <https://www.php.net/manual/es/intro-whatis.php>