

Profesor: *Neiner Maximiliano*

### Parte 1 - Ejercicios Simples con JSON

#### Aplicación N° 1 (Generar un Json)

Crear un Json que represente información acerca de un **producto** (codigoBarra, nombre, precio). Diseñarlo en el `<script type="text/javascript">`.

Validar su buen diseño ingresando en "<http://jsonviewer.stack.hu/>".

Una vez validado el Json, mostrar todos sus atributos en un **alert()** y en el **console.log()**.

#### Aplicación N° 2 (Generar una colección de Json)

Tomando como punto de partida el ejercicio anterior, diseñar una colección de tres elementos de tipo producto. Diseñarlo en el `<script type="text/javascript">`.

Validar su buen diseño ingresando en "<http://jsonviewer.stack.hu/>".

Una vez validado el Json, mostrar todos sus atributos en un **alert()** y en el **console.log()**.

#### Aplicación N° 3 (Enviar Json por Ajax)

Diseñar una aplicación que envíe por Ajax un **producto** hacia la página **mostrarJson.php**. En dicha página, mostrar el valor recibido utilizando la función **var\_dump()**.

Luego, transformar lo recibido en un objeto standard de PHP y mostrar cada uno de sus atributos. Utilizar las funciones **json\_encode()** y **json\_decode()**.

#### Aplicación N° 4 (Enviar colección de Json por Ajax)

Tomando como punto de partida el ejercicio anterior, enviar una colección de tres elementos de tipo **producto** por Ajax (hacia la página **mostrarColeccionJson.php**) y mostrar lo recibido con **var\_dump()** y luego de transformarlo en un objeto standard de PHP, mostrar todos los atributos de todos los objetos.

#### Aplicación N° 5 (Recibir Json por Ajax)

Diseñar una aplicación que reciba por Ajax un **producto** desde la página **recibirJson.php**.

Crear una instancia de **stdClass** y asignarle los atributos y valores correspondientes.

Desde javascript, mostrar el valor recibido utilizando la función **alert()** y en el **console.log()**.

#### Aplicación N° 6 (Recibir colección de Json por Ajax)

Tomando como punto de partida el ejercicio anterior, recibir una colección de tres elementos de tipo **producto** por Ajax (desde **recibirColeccion.php**) y mostrar lo recibido con **alert()** y en el **console.log()**.

### Parte 2 - Ejercicios con Archivos .json

#### Aplicación N° 7 (Leer un archivo .json)

Realizar una aplicación web que, a través de Ajax, lea el archivo **auto.json** desde la página **traerAuto.php** y muestre el JSON recibido por **alert()** y en el **console.log()**.

### Aplicación N° 8 (Leer un archivo .json II)

Tomando como punto de partida el ejercicio anterior, armar una página que posea un `<input type="button">` que al pulsarlo, muestre el JSON recibido por Ajax en los elementos de tipo `<input type="text">` (uno por cada atributo del objeto recibido).

### Aplicación N° 9 (Armar listado desde un .json)

Realizar una aplicación web que muestre un listado de autos, tomando como origen de datos el archivo ***autos.json***. La aplicación tendrá sólo un botón (`<input type="button">`), que al pulsarlo, generará dinámicamente un listado de los autos (armar una tabla html) que se reciban como objetos JSON desde el archivo. Como página nexa, utilice ***traerAutos.php***.

### Aplicación N° 10 (Agregar elemento a un .json II)

Examinar cuidadosamente el archivo ***city.list.min.json***, luego realizar una aplicación web, similar a la del ejercicio anterior, que permita armar un listado con el contenido completo de las ciudades. El archivo *.php* se deberá nombrar como ***administrarCiudades.php***, pasándole como parámetro la opción "traerCiudades".

### Aplicación N° 11 (Agregar elemento a un .json)

A la aplicación del punto anterior, agregarle otro `<input type="button">` que permita agregar un nuevo JSON al archivo ***city.list.min.json***. La aplicación deberá tener tantos elementos de tipo `<input type="text">` como atributos tenga el JSON. Como página nexa, utilice ***administrarCiudades.php***, pasándole la opción "agregarCiudad" y el JSON a ser agregado.

### Aplicación N° 12 (Quitar elemento a un .json)

A la aplicación del punto anterior, agregarle al listado una columna extra (ACCION) que posea un elemento `<a>` (cuyo texto sea Eliminar) que permita quitar el elemento seleccionado del archivo ***city.list.min.json***. Como página nexa, utilice ***administrarCiudades.php***, pasándole la opción "quitarCiudad" y el código de ciudad (`objJSON._id`) a ser eliminado.

### Aplicación N° 13 (Cargar ComboBox desde .json)

Generar una aplicación web que cargue de forma estática el combo ***paises*** (`<select>`) con las siguientes opciones:

- ARGENTINA (value="AR")
- FRANCIA (value="FR")
- GRECIA (value="GR")
- USA (value="US")

En el evento *onchange* de dicho combo, invocar a una función que por AJAX, permita la lectura del archivo ***paises.json***, retornando un array de objetos JSON.

De acuerdo a la selección del primer combo, se pide: cargar dinámicamente el combo ciudades (`<select>`) sólo con las ciudades correspondientes al país seleccionado.

## Parte 3 - Ejercicios Complejos con JSON

### Aplicación N° 14 (Armar listado desde archivo .json)

Realizar una aplicación web que muestre un listado de remeras, tomando como origen de datos el archivo ***remeras.json***. La aplicación tendrá sólo un botón (`<input type="button">`), que al pulsarlo, generará dinámicamente un listado de las remeras (armar una tabla html en

javascript) que se reciban como objetos JSON desde el archivo *.php*. Como página nexa, utilice **administrarRemeras.php**, pasándole la opción "traerRemeras".

### Aplicación N° 15 (Filtrar listado desde archivo .json)

Tomando como punto de partida el ejercicio anterior, agregarle un `<input type="text">` y un `<input type="button">` que al pulsarlo, generará dinámicamente un listado de las remeras filtradas por país de fabricante. Como página nexa, utilice **administrarRemeras.php**, pasándole la opción "traerRemerasFiltradas".

### Aplicación N° 16 (Filtrar listado desde archivo .json II)

Tomando como punto de partida el ejercicio anterior, agregarle un `<select>` (que posea como opciones *tamaño*, *color* y *país*) el cuál indicará el campo del filtrado. En el `<input type="text">` agregar el valor de filtrado (de acuerdo al campo de filtrado) y al pulsar el `<input type="button">`, generar dinámicamente un listado de las remeras filtradas. Como página nexa, utilice **administrarRemeras.php**, pasándole la opción "traerRemerasFiltradasPorCampo".

### Aplicación N° 17 (Agregar elemento a un archivo .json)

Realizar una aplicación web que permita agregar un nuevo JSON al archivo **remeras.json**. La aplicación deberá tener tantos elementos de tipo `<input type="text">` como atributos tenga el JSON. Como página nexa, utilice **administrarRemeras.php**, pasándole la opción "agregarRemera" y el JSON a ser agregado.

**Nota:** Para simplificar el ejercicio, 'hardcodee' el valor del campo **logo** con el siguiente valor: `"https://robohash.org/authicperferendis.bmp?size=50x50&set=set1"`.

### Aplicación N° 18 (Quitar elemento a un archivo .json)

A la aplicación del punto anterior, agregarle al listado una columna extra (ACCION) que posea un elemento `<a>` (cuyo texto sea Eliminar) que permita quitar el elemento seleccionado del archivo **remeras.json**, previa confirmación por parte del usuario. Como página nexa, utilice **administrarRemeras.php**, pasándole la opción "quitarRemera" y el ID de la remera (`objJSON.id`) a ser eliminado.

### Aplicación N° 19 (Modificar elemento a un archivo .json)

A la aplicación del punto anterior, agregarle un elemento `<a>` (cuyo texto sea Modificar) a la columna ACCION para permitir modificar el elemento seleccionado del archivo **remeras.json**. Al hacer click sobre el link Modificar, se deberán mostrar, en los correspondientes `<input type="text">`, los valores correspondientes al elemento seleccionado. Al mismo tiempo, deberá 'aparecer' (atributo `display = block`) un `<input type="button">` que, al pulsarlo y con la confirmación del usuario, modifique el archivo **remeras.json**. Como página nexa, utilice **administrarRemeras.php**, pasándole la opción "modificarRemera" y el objeto JSON a ser modificado.