	<p>Servicio Nacional de Aprendizaje SENA CEAI - Centro de Electricidad y Automatización Industrial – Regional Valle Instructor : JAIR ROA VELÁSQUEZ</p>	<p>Fecha: Febrero de 2016 Página 1 de 6</p>
---	--	---

Json

Con el crecimiento abismal de la información y la necesidad de enviarla continuamente usando servicios web se requiere la presencia de formatos de datos muy ligeros que permitan optimizar los tiempos de transmisión.

Douglas Crockford, un experimentado ingeniero de software, propuso un nuevo formato de datos construido sobre JavaScript llamado JSON, JavaScript Object Notation (notación de objetos JavaScript), el cual utiliza un subconjunto de la sintaxis de JavaScript manejando literales de matrices y objetos.

La ventaja que tiene el Json frente a otros formatos como el XML, es que el Json al incluirse dentro de los archivos JavaScript puede acceder a la información sin la necesidad de realizar análisis adicionales como lo hace XML, por lo tanto el tratamiento de grandes volúmenes de información se hace de forma ágil, dinámica, escalable y robusta, permitiendo compartir la data almacenada entre diferentes los componentes y lenguajes de aplicaciones web, convirtiéndose en la notación más exitosa en el desarrollo web, adicionalmente el tratamiento de la big data que exige un formato bson, utiliza como capa de presentación al Json para la interacción humana.

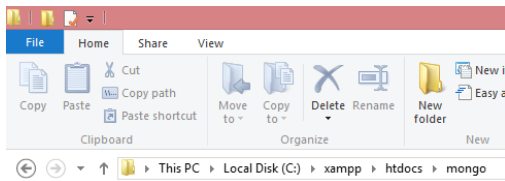
Para que podamos contextualizar la notación Json, vamos a usar un saber previo que es el PHP, el cual nos permitirá ambientarnos en la forma de almacenar, recuperar e interpretar la información.

Requisitos de software

Json_encode() permite crear una cadena, expresar un objeto u otro tipo de variable en expresión o notación Json.

Eval() permite almacenar el Json en una variable JavaScript sin almacenar la cadena sino el objeto que representa

Empecemos a comprender un poco la notación, para esto iniciaremos con lo más básico que es una variable.



Lo primero que hará será crear en su servidor web una carpeta que llamara mongo donde grabara un script php llamado Json.php

```
1 html:5
```

Inicie la codificación con un documento html5.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Document</title>
6 </head>
7 <body>
8
9 </body>
10 </html>
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Comprendiendo Json</title>
6 </head>
7 <body>
8
9 <?php
10
11     $var = "Prueba Json";
12     print_r(json_encode($var));
13
14 >?>
15
16 </body>
17 </html>
```

Como habíamos dicho, iniciaremos con lo más básico que es una variable y la llevamos a formato Json.

localhost/mongo/json.php
"Prueba Json"

Podemos observar que al visualizar el contenido de la variable lo muestra entre comillas dobles, ya que se trata de una cadena de caracteres.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Comprendiendo Json</title>
6 </head>
7 <body>
8
9 <?php
10
11     $var = "Prueba Json";
12     print_r(json_encode($var));
13
14     echo "<br><br>";
15
16     $arr = array(1,5,9,7,55,74,6,32,44,125,658);
17     print_r(json_encode($arr));
18
19     echo "<br><br>";
20
21 >?>
```

Ahora vamos a darle tratamiento a un vector que llamaremos \$arr y lo llevamos a notación Json

localhost/mongo/json.php
"Prueba Json"
[1,5,9,7,55,74,6,32,44,125,658]

Mire como nos indica con corchetes que se trata de un arreglo que en este caso es vectorial.

Los anteriores ejemplos son muy sencillos y no apreciamos la fortaleza de la notación Json, con el siguiente ejemplo podemos comenzar a notar un cambio y poder comprender un poco más el objetivo del formato. Para esto vamos a utilizar un arreglo asociativo donde ya podemos incorporar atributos acompañados del dato, en *Json* se conoce como literales de objeto que se utilizan para almacenar información en parejas “*nombre-valor*” creando de esta forma un objeto.

```

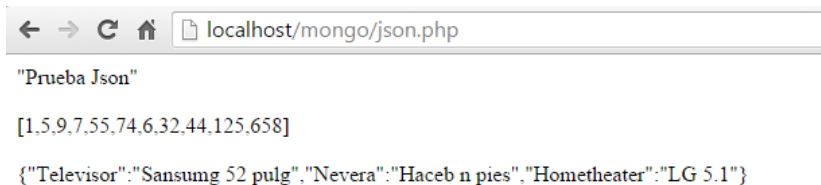
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Comprendiendo Json</title>
6 </head>
7 <body>
8
9 <?php
10
11     $var = "Prueba Json";
12     print_r(json_encode($var));
13
14     echo "<br><br>";
15
16     $arr = array(1,5,9,7,55,74,6,32,44,125,658);
17     print_r(json_encode($arr));
18
19     echo "<br><br>";
20
21     $arra = array("Televisor" => "Sansumg 52 pulg" , "Nevera" => "Haceb n pies" , "Hometheater" => "LG 5.1");
22     print_r(json_encode($arra));
23
24 >?
25
26
27
28 </body>
29 </html>

```

Recordemos la definición de objeto:

“Es la unidad de código que contiene características en forma de datos y que tiene la capacidad de realizar abstracción de las responsabilidades de su clase o dominio origen dándole vida a través de la acción.”

Nos quedaremos con la primera parte de la definición (*es la unidad de código que contiene características en forma de datos*)



```

localhost/mongo/json.php

"Prueba Json"

[1,5,9,7,55,74,6,32,44,125,658]

{"Televisor":"Sansumg 52 pulg","Nevera":"Haceb n pies","Hometheater":"LG 5.1"}

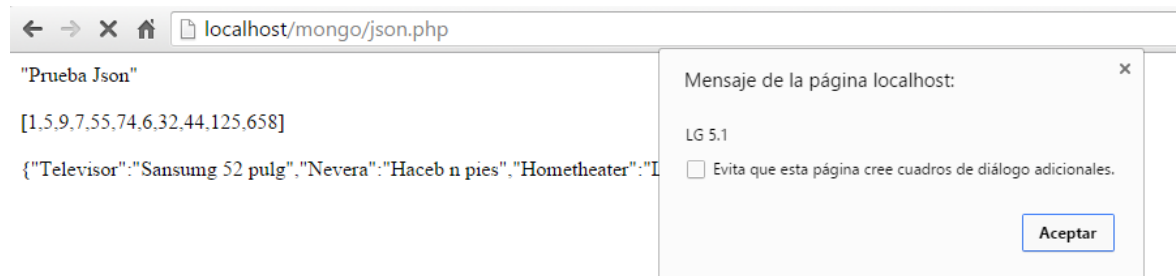
```

Este resultado ya tiene una forma parecida a lo que oferta el formato ***Json***.



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Comprendiendo Json</title>
6 </head>
7 <body>
8
9   <?php
10
11     $var = "Prueba Json";
12     print_r(json_encode($var));
13
14     echo "<br><br>";
15
16     $arr = array(1,5,9,7,55,74,6,32,44,125,658);
17     print_r(json_encode($arr));
18
19     echo "<br><br>";
20
21     $arra = array("Televisor" => "Sansumg 52 pulg" , "Nevera" => "Haceb n pies" , "Hometheater" => "LG 5.1");
22     print_r(json_encode($arra));
23
24   ?>
25
26   <script>
27     jsonelectro = eval(<?php echo json_encode($arra); ?>);
28     alert(jsonelectro.Hometheater);
29   </script>
30
31
32
33 </body>
34 </html>
```

Ahora utilizando la función JavaScript Eval() podemos almacenar el dato en Json luego podemos este componente Json como lo que es un objeto constituido por datos y para llegar a estos datos nos referimos a sus propiedades. ***Jsonelectro.hometheater*** y nos dará como resultado el dato contenido en la propiedad del objeto Json.



El mensaje nos muestra el dato “LG 5.1” que pertenece a la propiedad ***hometheater*** que pertenece al objeto ***jsonelectro***.



Un poco de programación orientada a objetos.

Vamos a crear una clase e instanciamos objetos de ella.

```
26
27 <script>
28     jsonelectro = eval(<?php echo json_encode($arra); ?>);
29     alert(jsonelectro.Hometheater);
30 </script>
31
32 <?php
33
34     // Creamos Nodo Inicial.
35     class padre
36     {
37         var $nombre;
38         var $apellido;
39         var $hijos;
40
41         function __construct($nombre,$apellido){
42             $this->nombre = $nombre;
43             $this->apellido = $apellido;
44         }
45
46         function anadirhijo($nodohijo, $nombre)
47         {
48             // añadir un hijo
49             if(!isset($this->hijos))
50             {
51                 $this->hijos = array();
52             }
53             $this->hijos[$nombre] = $nodohijo;
54         }
55     }
56
57     $papa = new padre("Robert Tulio", "Zapata");
58     $miNodo = new padre("Luis Carlos", "Castillo");
59     $papa->anadirhijo($miNodo, "Adoptado");
60     $miNodo2 = new padre("Jesus jair", "Cifuentes");
61     $papa->anadirhijo($miNodo2, "Negado");
62
63     print_r(json_encode($papa));
64 ?>
65
66 </body>
67 </html>
```

Hemos creado una clase padre con la que nos proponemos a crear objetos anidados entre si y poder llegar a desarrollar un árbol genealógico.

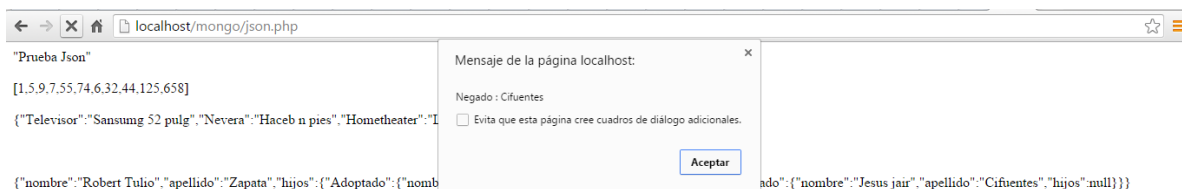
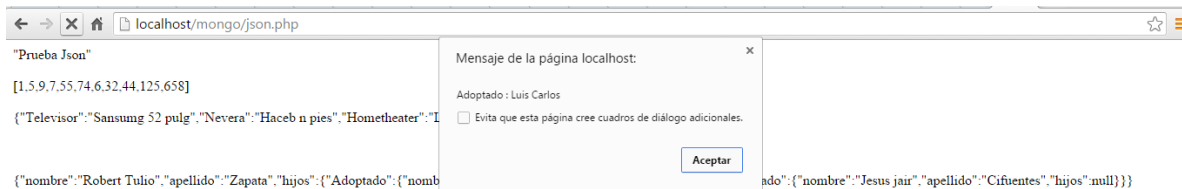
```
localhost/mongo/json.php
"Prueba Json"
[1,5,9,7,55,74,6,32,44,125,658]
{"Televisor":"Samsung 52 pulg","Nevera":"Haceb n pies","Hometheater":"LG 5.1"}

{"nombre":"Robert Tulio","apellido":"Zapata","hijos":{"Adoptado":{"nombre":"Luis Carlos","apellido":"Castillo","hijos":null},"Negado":{"nombre":"Jesus jair","apellido":"Cifuentes","hijos":null}}}
```



```
41 function __construct($nombre,$apellido){
42     $this->nombre = $nombre;
43     $this->apellido = $apellido;
44 }
45
46 function anadirhijo($nodohijo, $nombre)
47 {
48     // añadir un hijo
49     if(!isset($this->hijos))
50     {
51         $this->hijos = array();
52     }
53     $this->hijos[$nombre] = $nodohijo;
54 }
55 }
56
57 $papa = new padre("Robert Tulio", "Zapata");
58 $miNodo = new padre("Luis Carlos", "Castillo");
59 $papa->anadirhijo($miNodo, "Adoptado");
60 $miNodo2 = new padre("Jesus jair", "Cifuentes");
61 $papa->anadirhijo($miNodo2, "Negado");
62
63 print_r(json_encode($papa));
64
65 $father = json_encode($papa);
66
67 ?>
68
69 <script>
70     var genealogico = eval(<?php echo $father; ?>);
71     alert('Adoptado : '+genealogico.hijos.Adoptado.nombre);
72     alert('Negado : '+genealogico.hijos.Negado.apellido);
73 </script>
74
75 </body>
76 </html>
```

Ahora podemos ubicar objetos en el cliente para que estos puedan ser procesados por él.



ACTIVIDAD

Construir el esquema requerido para registrar una atención veterinaria, al menos 20 documentos