

Front End

CFP
Programador
full-stack

JSON

Organización de los datos

Una forma de organizar las **variables** y **funciones** es a través de **objetos**.

Encapsulan **datos** y **comportamiento**

Son los objetos nativos en JavaScript

Su “clase” es de tipo “objeto” y son de estructura libre

Qué es JSON?

- Organizado
- Fácil de acceder

//objeto 'profesor' con dos atributos

```
{  
  "nombre": "javier",  
  "materia": "web",  
}
```

Se compone de

Registros (objetos): { }

Propiedades: "clave" : "valor"

Arreglos: [] (Contienen objetos)



No confundir un Objeto JSON con una función, tiene llaves, pero no tiene parámetros, ni código, ni la palabra **function** antes

Agregar y acceder a miembros

```
let curso = {  
  "nombre": "full-stack",  
  "alumno": [  
    {  
      "nombre": "juan",  
      "apellido": "perez"  
    },  
    {  
      "nombre": "maria",  
      "apellido": "garcia"  
    }  
  ]  
}
```

Agregar y acceder a miembros

Podemos leer los miembros:

```
let nombre = curso.nombre;  
let primerApellido = curso.alumnos[0].apellido;
```

Les podemos agregar miembros

```
curso.lugar = "cfp";
```

Claves

Las claves pueden o no ir entre comillas

```
{ valor: 4 }
```

O

```
{ "valor": 4 } //Good practice!,
```

evita problemas si tenemos un campo que es una palabra reservada

```
{  
    "error": 4,  
    "if": 6  
}
```

Valores en JSON

Los valores pueden ser de los siguientes tipos:

```
{  
  "cadena": "texto",  
  "numero": 5,  
  "otroObjeto": {...},  
  "arreglo": [5, "a", 1],  
  "verdadero": true,  
  "nada" : null  
}
```

Notar uso de comillas
en *value* sólo para
cadenas de texto

Ventajas

- Super - Liviano para transferir
- Datos auto-descriptos
- Legible por el humano
- Fácil de adoptar por los lenguajes orientados a objetos

{JSON}

{JSON}

```
{
  - Contacts: [
    - {
      FirstName: "Demis",
      LastName: "Bellot",
      Email: "demis.bellot@gmail.com"
    },
    - {
      FirstName: "Steve",
      LastName: "Jobs",
      Email: "steve@apple.com"
    },
    - {
      FirstName: "Steve",
      LastName: "Ballmer",
      Email: "steve@microsoft.com"
    },
    - {
      FirstName: "Eric",
      LastName: "Schmidt",
      Email: "eric@google.com"
    },
    - {
      FirstName: "Larry",
      LastName: "Ellison",
      Email: "larry@oracle.com"
    }
  ]
}
```

<xml />

```
<ContactsResponse xmlns:i="http://www.w3.org/2003/05/soap-envelope">
  <Contacts>
    <Contact>
      <Email>demis.bellot@gmail.com</Email>
      <FirstName>Demis</FirstName>
      <LastName>Bellot</LastName>
    </Contact>
    <Contact>
      <Email>steve@apple.com</Email>
      <FirstName>Steve</FirstName>
      <LastName>Jobs</LastName>
    </Contact>
    <Contact>
      <Email>steve@microsoft.com</Email>
      <FirstName>Steve</FirstName>
      <LastName>Ballmer</LastName>
    </Contact>
    <Contact>
      <Email>eric@google.com</Email>
      <FirstName>Eric</FirstName>
      <LastName>Schmidt</LastName>
    </Contact>
    <Contact>
      <Email>larry@oracle.com</Email>
      <FirstName>Larry</FirstName>
      <LastName>Ellison</LastName>
    </Contact>
  </Contacts>
</ContactsResponse>
```


CFP

Programador

full-stack

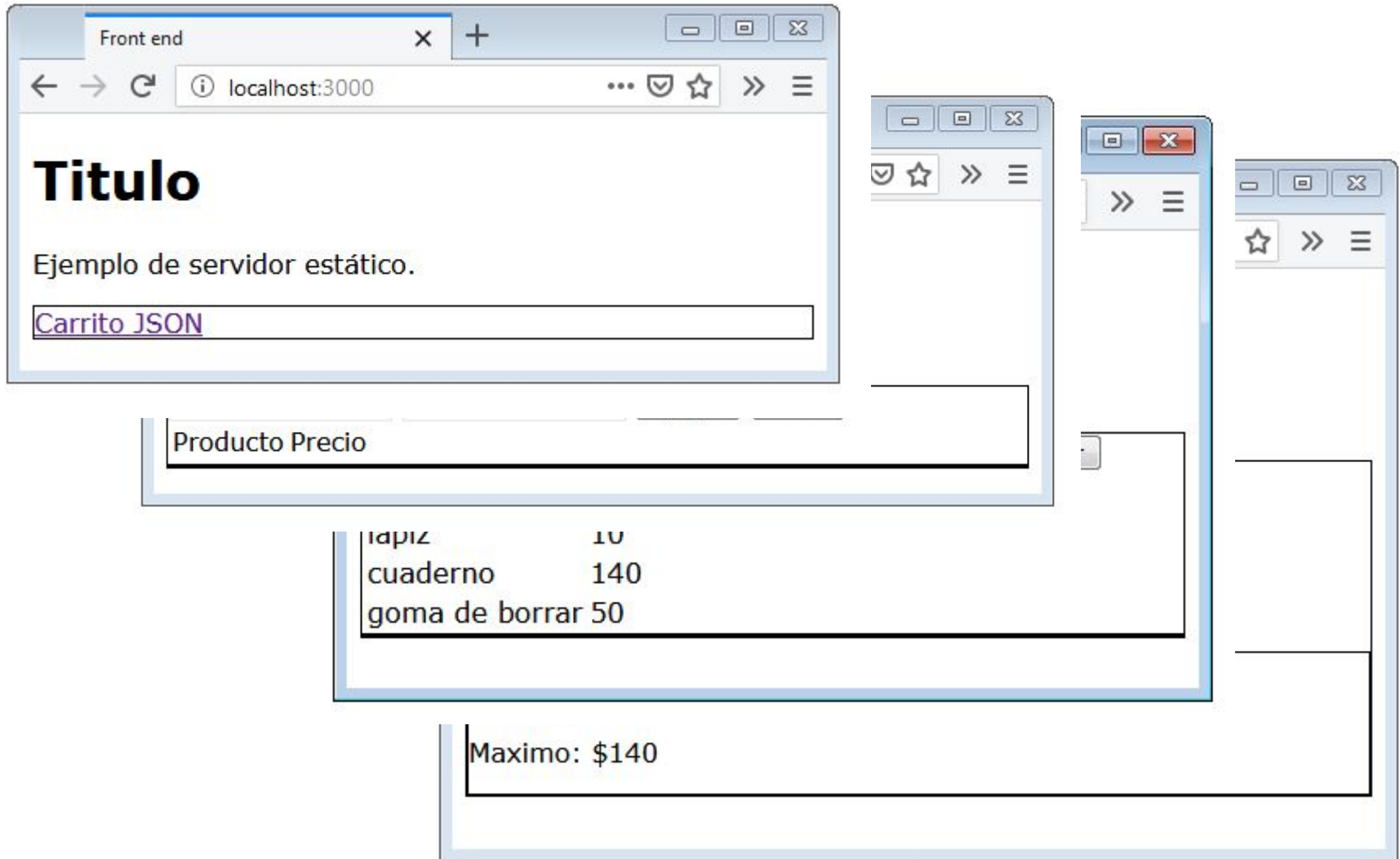
Ejemplo con JSON

Lista de compras

Hagamos una lista de compras (un carrito de compras) que tenga el precio de cada elemento y el total que gastamos llevamos gastado.

Manejar un arreglo de objetos en JS

Objetivo final



Modelo de datos

[illegible]

Código HTML

...

```
<body>
  <h1>Carrito</h1>
  <p>Ejemplo de arreglo de objetos en JS.</p>
  <input type="text" name="" value="" id="producto">
  <input type="text" name="" value="" id="precio">
  <button type="button" id="btnAgregar">Agregar</button>
  <button type="button" id="btnTotal">Sumar</button>
  <table>
    <thead>
      <td>Producto</td>
      <td>Precio</td>
    </thead>
    <tbody id="tblCompras"></tbody>
  </table>
  <div id="total"></div>
  <script src="js/carrito.js"></script>
</body>
```

...

Código JS

```
let btnAgregar = document.querySelector("#btnAgregar");  
btnAgregar.addEventListener("click", agregar);  
let btnTotal = document.querySelector("#btnTotal");  
btnTotal.addEventListener("click", sumar);  
  
let compras = [];
```

Código JS

```
function agregar() {  
    console.log("Funcion Agregar");  
    let producto = document.querySelector('#producto').value;  
    let precio =  
    parseInt(document.querySelector('#precio').value);  
  
    let renglon = {  
        "producto": producto,  
        "precio": precio  
    }  
    compras.push(renglon);  
  
    mostrarTablaCompras();  
}
```

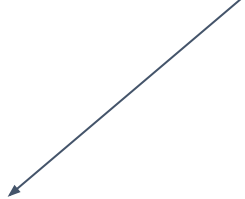
Código JS

```
function sumar() {  
  console.log("Funcion Sumar");  
  let total = 0;  
  for (let i = 0; i < compras.length; i++) {  
    total += compras[i].precio;  
  }  
  let max = compras[0].precio;  
  for (let r of compras) {  
    if(max < r.precio)  
      max = r.precio;  
  }  
  document.querySelector("#total").innerHTML =  
    "<p>Total: $" + total + "</p>" +  
    "<p>Maximo: $" + max + "</p>"  
}
```


Código JS

```
function mostrarTablaCompras() {  
  html = "";  
  for (let r of compras) {  
    html += `  
      <tr>  
        <td>${r.producto}</td>  
        <td>${r.precio}</td>  
      </tr>  
    `;  
  }  
  document.querySelector("#tblCompras").innerHTML = html;  
}
```

Esto es un Template Literal.
La comilla es el acento inverso.
Es una manera de incluir saltos de línea
y el \${} se reemplaza por la evaluación
de la expresión que engloba.



CFP

Programador full-stack

Funciones: pasaje de parámetros y ámbitos

Parámetros

- Tipos primitivos: se pasan por copia-valor
- Tipos objetos: se pasan por referencia

Ejemplo: La función **aumentar** suma 1 a ambos parámetros

```
let primitivo = 5
```

```
let objeto = { valor: 5 }
```

```
aumentar(primitivo,objeto)
```

```
// primitivo: 5 (se copio)
```

```
// objeto.valor: 6 (se usó el mismo)
```

```
function aumentar(primitivo, objeto){  
  primitivo++;  
  objeto.valor++;  
}
```

<https://codepen.io/webUnicen/pen/KmyoXP>

Parámetros

- Cualquier argumento puede ser omitido o agregado.

```
function sumar(a, b, c)
{
    return a + b + c;
}
```

```
sumar(1, 2, 3); //6
```

```
sumar(1, 2); //NaN : Hace 1+2+undefined
```

```
sumar(1, 2, 3, 4, 5, 6); //6
```

Arrow Functions

Es una forma abreviada para escribir funciones:



Ejemplo:

```
unArreglo.forEach(elem => {  
    console.log(elem)  
})
```

<https://codepen.io/webUnicen/pen/KmyGxG>

CFP

Programador

full-stack

Ejercicios

Ejercicio

Hacer una lista de documentos, con un formulario para agregar documentos con información correspondiente al título, al autor, al contenido temático y a la fecha de creación.

Sobre esa lista proveer al menos 3 de los siguientes servicios de consulta:

- autor con mas documentos.
- cantidad de documentos con más de un año de antigüedad.
- lista de documentos de un tema determinado.
- título más moderno
- título más antiguo
- tema mas tratado