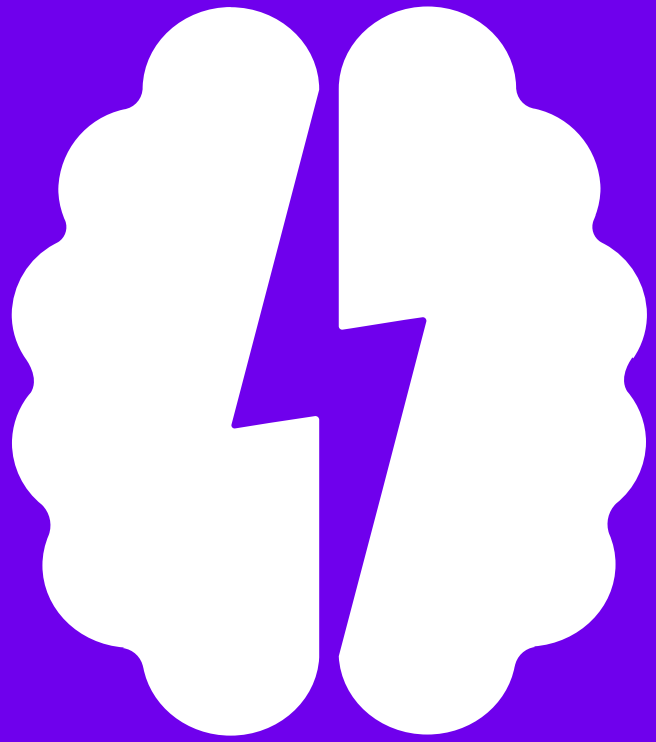
CURSO: DESARROLLO DE APLICACIONES WEB

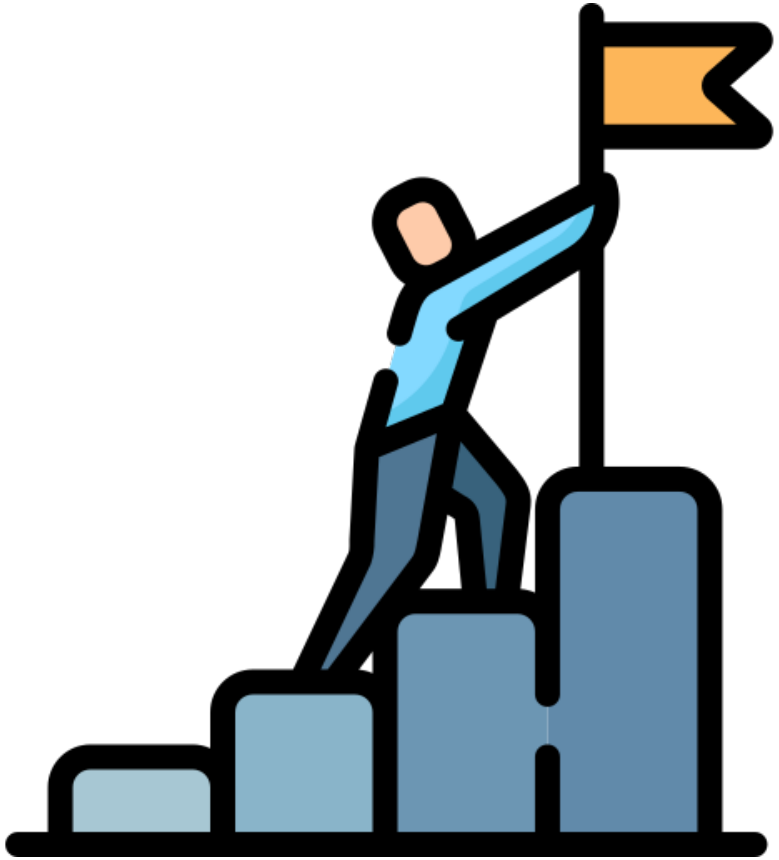
Sesión 5: JavaScript - II



OBJETIVOS

Al finalizar la sesión, el alumno logrará:

- Aplicar JavaScript en el desarrollo de aplicaciones para agregar características interactivas avanzadas a los sitios web.



AGENDA

- Funciones y eventos
- POO
- DOM – Document Object Model
- Validación de datos
- Fundamentos de AJAX

i d a t

inicio desarrollo aplicación término

FUNCIONES Y EVENTOS

¿QUÉ ES UNA FUNCIÓN?

Una función ss un trozo de código reutilizable en el que hay un conjunto de instrucciones, este codigo solo se ejecuta cuando se llama a dicha función.

DECLARACIÓN

1 DECLARAR

* Palabra reservada function
* Nombre de la función
* Escribir los Parámetros

2 BLOQUE DE CÓDIGO

* Se escribe el conjunto de instrucciones que va a realizar la función

3 INVOCAR FUNCIÓN

* Llamar a la función
* Escribir los argumentos (valores)

```
// Function declaration //
function someName(param1, param2) {

// bunch of code as needed...
var a = param1 + "love" + param2;
return a;
}

// Invoke (run / call) a function
someName("Me", "You")
```

MGuedez

¿QUÉ ES UN EVENTO?

- Un evento es una acción dentro de un aplicativo que nos permiten definir un comportamiento cuando éstos ocurren. Una acción puede ser:
  - Click de ratón del usuario sobre un elemento de la página
  - Pulsación de una tecla específica del teclado
  - Reproducción de un archivo de audio/video
  - Scroll de ratón sobre un elemento de la página
  - El usuario ha activado la opción «Imprimir página»

- Existen varias formas diferentes de manejar eventos en Javascript los cuales los podemos resumir de la siguiente manera:

FORMA	EJEMPLO
Mediante **atributos HTML**	<button onClick="..."></button>
Mediante **propiedades Javascript**	.onclick = function() { ... }
Mediante addEventListener()	.addEventListener("click", ...)

FUNCIONES Y EVENTOS

Ejemplo de Funciones:

```
function Alerta(){

    alert("Mensaje....");
}

function NombreFuncion(){

    //Lo que queremos que haga esta funcion
}
```

Ejemplo de Eventos:

```
function Mensaje() {

  var Msj = document.getElementById('MensajeID'); //Se obtiene la etiqueta html

  //Primera Opcion
  Msj.addEventListener("click", Alerta());//Se produce el evento "Click" al clickear la etiqueta html

  //Segunda Opcion
  Msj.addEventListener("click", function(){

    alert("Mensaje...");

  });
}

function Alerta(){

    alert("Mensaje....");
}
```

[Recomendaciones de código limpio en JAVASCRIPT]

POO

- Uno de los aspectos más importantes del lenguaje Javascript es el concepto de objeto, puesto que prácticamente todo lo que utilizamos en Javascript, son objetos. Sin embargo, tiene ligeras diferencias con los objetos de otros lenguajes de programación, así que vamos a comenzar con una explicación sencilla y más adelante ampliaremos este tema en profundidad.
- En Javascript, existe un tipo de dato llamado objeto . Una primera forma de verlo es como una variable especial que puede contener más variables en su interior. De esta forma, tenemos la posibilidad de organizar múltiples variables de la misma temática en el interior de un objeto.
- En muchos lenguajes de programación, para crear un objeto se utiliza la palabra clave new. En Javascript también se puede hacer.

Ejemplos:

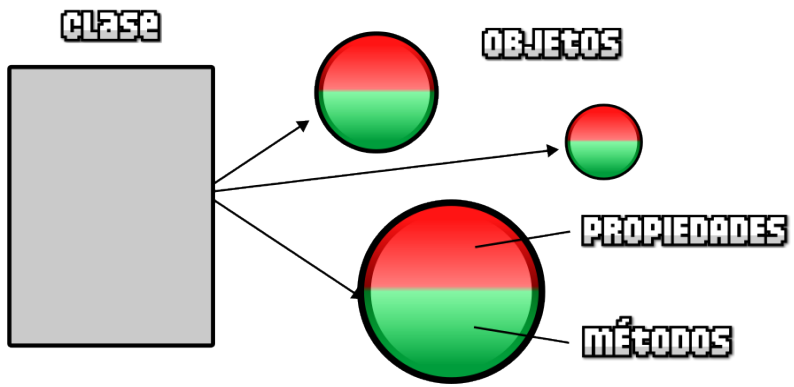
1
```
const objeto = new Object();    // Evitar esta sintaxis en Javascript (no se suele usar)
```

2
```
const objeto = {};    // Esto es un objeto vacío
```

3
```
const player = {
  name: "Manz",
  life: 99,
  power: 10,
};
```

POO

- La Programación Orientada a Objetos (POO) es un enfoque popular en la programación donde se utilizan estructuras de datos que imitan objetos del mundo real, lo que facilita la planificación y organización del código. Al dominar la POO, se simplifica la creación de estructuras de datos complejas y se mejora la organización del código a medida que los programas crecen.
- Aunque JavaScript no tenía soporte nativo para clases, con ECMAScript 2015 introdujo la posibilidad de utilizar clases, lo que permite a los programadores trabajar de manera más familiar y agradable, aunque internamente JavaScript sigue utilizando su sistema basado en prototipos.



Sintaxis de POO:

```
// Declaración de una clase (de momento, vacía)
class Animal {}

// Crear (instanciar) un objeto basada en una clase
const pato = new Animal();
```

```
class Animal {
  name;               // Propiedad (variable de clase sin valor definido)

  constructor(name) {
    this.name = name;   // Hacemos referencia a la propiedad name del objeto instanciado
  }
}
```

```
class Animal {
  // Propiedades
  name = "Garfield";
  type = "cat";

  // Métodos
  hablar() {
    return "Odio los lunes."
  }
}
```

```
// Animal.js

export class Animal {
  /* Contenido de la clase */
}
```

```
// index.js

import { Animal } from "./Animal.js";

const pato = new Animal();
```

POO

Ejemplo de POO:

```
Class Celular{

      constructor( color, peso, ram, marca, resolución )
      {
            this.color = color;
            this.peso = peso;
            this.ram = ram;
            this.marca = marca;
            this.resolucion = resolucion;
            this.encendido = false;
      }

      PresionarBotonEncendido()
      {
          if(this.encendido == false){
            alert("Celular encendido");
            this.encendido = true
          }
          else{
            alert("Celular apagado");
            this.encendido = false;
          }
      }

      TomarFoto()
      {
          if(this.encendido == true){
            alert(`Foto tomada con una resolución de: ${this.resolucion}`);
          }
          else{
            alert("No se puede tomar una foto porque el celular esta apagado");
          }
      }

      InfoCelular()
      {
            Return `
            Color: <b> ${this.color} </b><br>
            Peso: <b> ${this.peso} </b><br>
            Ram: <b> ${this.ram} </b><br>
            Marca: <b> ${this.marca} </b><br>
            Resolucion: <b> ${this.resolucion} </b><br>
            `;
      }
}
```

```
//Con esto creamos un nuevo Celular
Celular1= new Celular("Negro", "50g", "16gb", "Samsung", "1920x1080");
Celular2= new Celular("Blanco", "80g", "30gb", "iPhone", "4K");
Celular3= new Celular("Rosado", "65g", "8gb", "Motorola", "Full HD");

//Llamamos a las funciones
Celular1.PresionarBotonEncendido();
Celular1.TomarFoto();

Celular2.TomarFoto();

//Al nosotros usar el document.write para mostrar la info es necesario usar el
backtick para poder
//Llamar al método y mostrar la información en la pagina

document.write(`
      ${Celular1.InfoCelular()}<br><br>
      ${Celular2.InfoCelular()}<br><br>
      ${Celular3.InfoCelular()}<br><br>
`)
```

POO

Ejemplo de POO:

```
//Hablaremos de clases para entendernos, aunque no lo son propiamente, siempre son objetos.

// Super Clase
function Persona(nombre, edad, peso) {
    this.nombre = nombre;
    this.edad = edad;
    this.peso = peso;
    this.getInfo = function() {
      return "Me llamo " + this.nombre +
             ", tengo " + this.edad + " años " +
             "y peso " + this.peso + " kilos.";
    }
}

// Sub Clase
function Empleado(nombre, edad, peso, sueldo){
    Persona.call(this, nombre, edad, peso); //llamamos al constructor de Persona
    this.sueldo = sueldo;
    let clave\_acceso = ""; //propiedad privada

    this.setClaveAcceso = function(key){
        clave\_acceso = key;
    }
    this.sueldo = sueldo;
    this.getInfo = function() {
      return "Me llamo " + this.nombre +
             ", tengo " + this.edad + " años " +
             ", peso "+ this.peso + " kilos y gano " + this.sueldo + " euros.";
    }
}
```

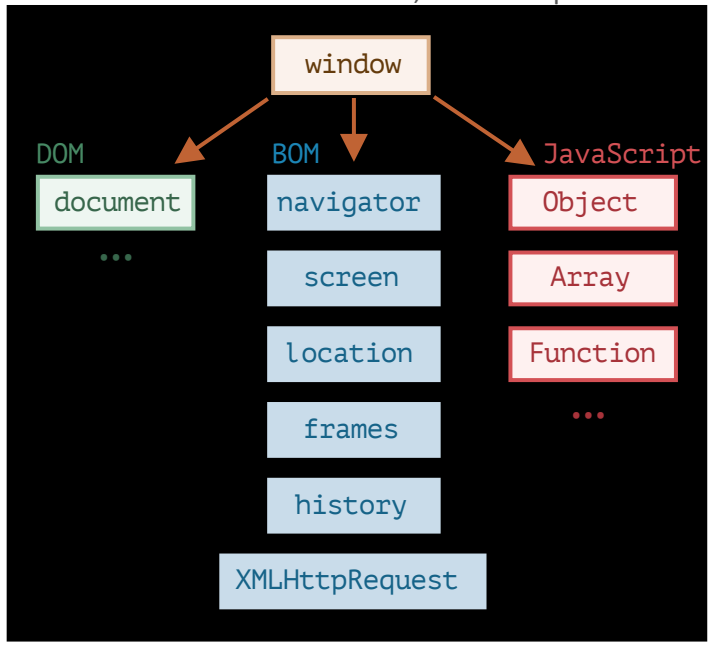
```
// Herencia
Empleado.prototype = Object.create(Persona.prototype);
Empleado.prototype.constructor = Empleado;

// Polimorfismo
// objeto puede ser de tipo Persona o de tipo Empleado, y getInfo llamará a la implementación
function showInfo(objeto) {
    console.log(objeto.getInfo());
}

// Instancias de un clase
let persona = new Persona("Javi", 31, 90);
let empleado = new Empleado("David", 32, 91, 23000);
showInfo(persona);
showInfo(empleado);
```

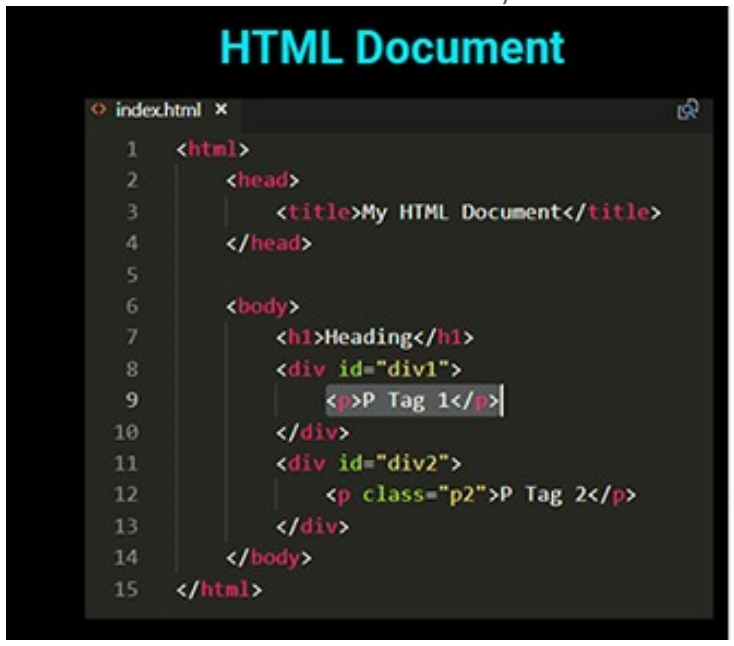
Laboratorio Nº 1: Crea acciones JS usando POO

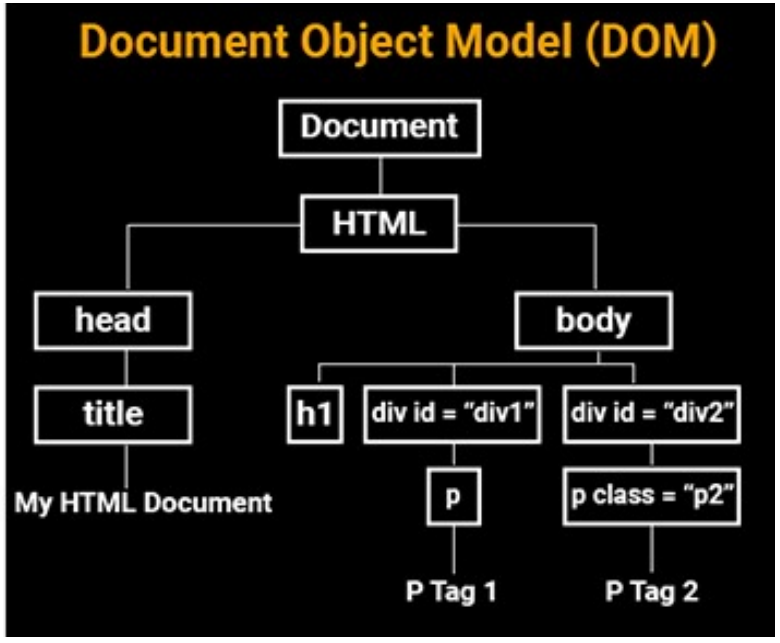
- Elaborar una clase Celular en la que se agregar nuevos Celulares manualmente y se mostraran mediante un botón.
- En este laboratorio, el alumno podrá aplicar los conceptos de POO en JS.

DOM – DOCUMENT OBJECT MODEL

- El **DOM** (Document Object Model) representa la estructura de un documento **HTML** en forma de árbol de etiquetas. En JavaScript, podemos acceder y modificar los elementos del **HTML** utilizando el **DOM**. Podemos añadir, modificar o eliminar etiquetas, cambiar atributos, agregar clases, editar contenido de texto, entre otras acciones. Además, podemos automatizar estas tareas y vincularlas a eventos del usuario, como clics, movimientos del ratón o entrada de texto.
- En **Javascript**, la forma de acceder al **DOM** es a través de un objeto llamado **document**, que representa el árbol DOM de la página o más concretamente la página de la pestaña del navegador donde nos encontramos. En su interior pueden existir varios tipos de elementos, pero principalmente serán **element** (representación genérica de una etiqueta) o **node** (una unidad más básica, la cuál puede ser element o un nodo de texto).







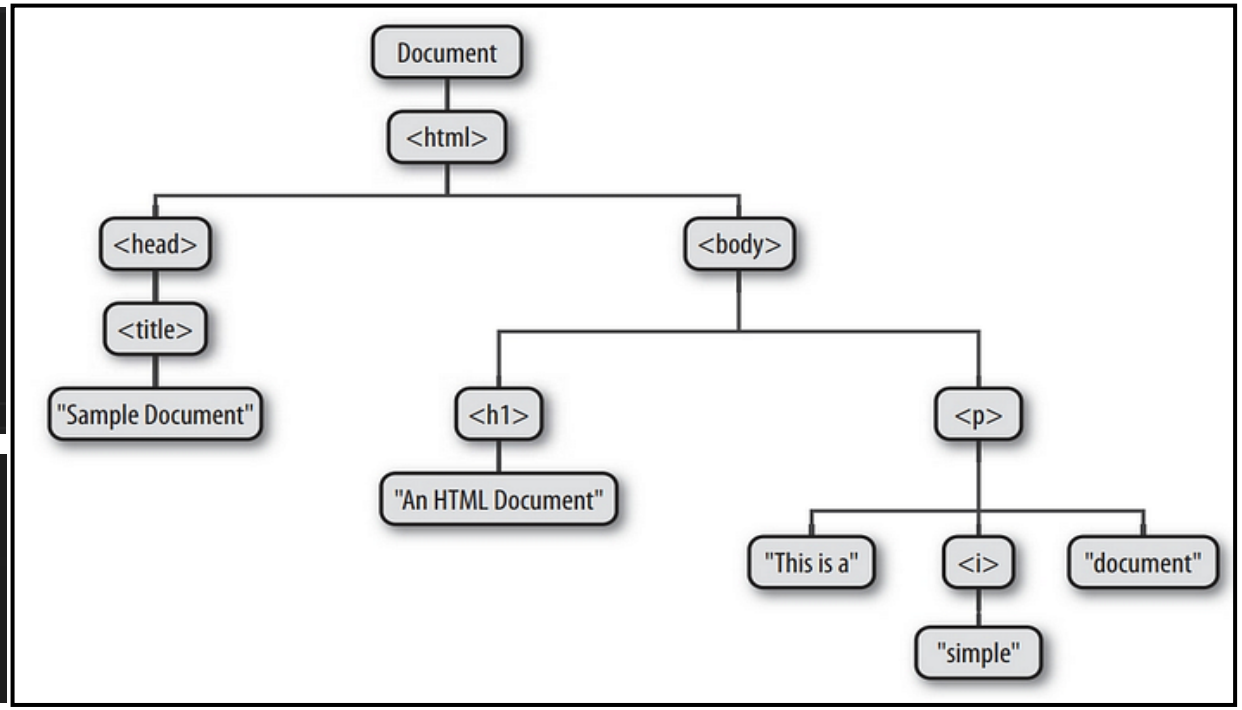
DOM – DOCUMENT OBJECT MODEL

Ejemplo de uso DOM:

aula10 > revisao > artigo.html > html

```html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sample Document</title>
</head>
<body>

    <h1>An HTML Document</h1>
    <p>This is a <i>simple</i> document.</p>

</body>
</html>
```

```html
<script>

    var paragrafo = document.getElementsByTagName('p')[0]
    document.write(paragrafo.innerText)


</script>
```



VALIDACIÓN DE DATOS

- La validación de formularios en JavaScript se utiliza para asegurarse de que los datos ingresados por los usuarios en un formulario cumplan con ciertos criterios o requisitos específicos.

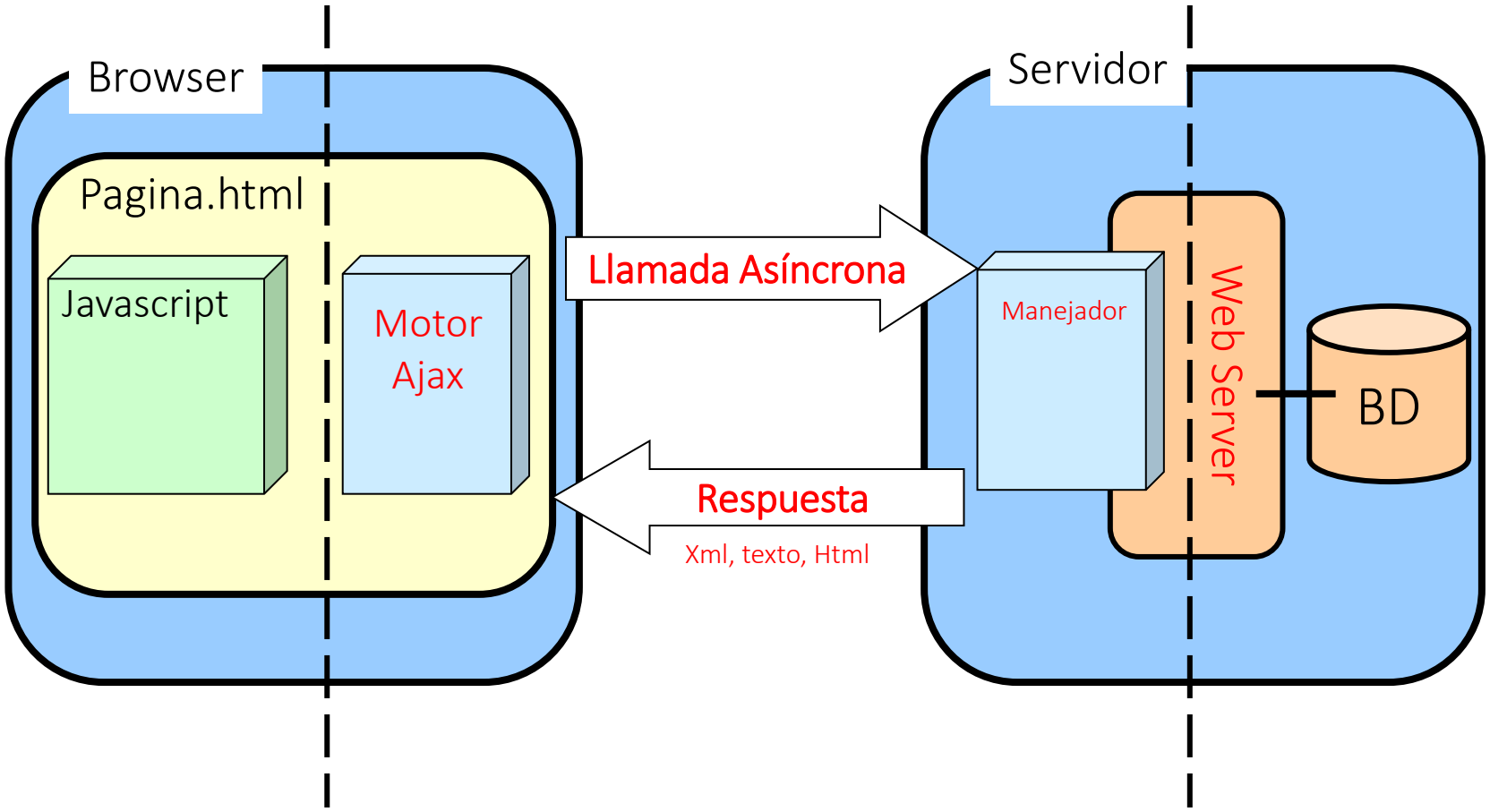
```
//El evento onsubmit sirve para ejecutar la función al darle al botón guardar
//Si la función retorna false, no se ejecuta el submit. Sin embargo, si retorna
true, se enviarán los datos

<form onsubmit="return ValidarFormulario();">

      <label>Nombre</label><br>
      <input type="text" Id="NombreID" ><br><br>

      <label>Apellido</label><br>
      <input type="text" Id="ApellidoID" ><br><br>

      <label>Telefono</label><br>
      <input type="text" Id="TelefonoID"><br><br>

      <label>Email</label><br>
      <input type="text" Id="EmailID"><br><br>

      <button type="submit" >Guardar</button>

</form>
```

```
function ValidarFormulario() {

      var Nombre = document.getElementById('NombreID').value;
      var Apellido = document.getElementById('ApellidoID').value;
      var Email = document.getElementById('TelefonoID').value;
      var Telefono = document.getElementById('EmailID').value;

      FormatoCorreo = /\w+@\w+\.+[a-z]/;

      if( Nombre.trim().length == 0 || Apellido.trim().length == 0 ||
          Email.trim().length == 0 || Telefono.trim().length == 0){

            alert("Todos los campos son obligatorios y no deben ser vacíos");
            return false;
      }
      else if(!FormatoCorreo.test(Email)){
       //Si no se cumple con el formato retornara false
       //.test() valida si lo que esta dentro de los paréntesis tiene una
       //coincidencia con la Expresión regular, en nuestro caso, FormatoCorreo

       alert("El correo no tiene un formato valido");
       return false;

      }
      else if(Telefono.trim().length > 9){
          alert("No puedes agregar más de 9 dígitos en el teléfono");
          return false;

      }
      else{
         alert("Datos guardados correctamente");//Por defecto retorna true
      }
}
```

Laboratorio Nº 2: Validar formularios con JS

- Elaborar un formulario con HTML y luego validar sus imputs con JS.
- En este laboratorio, el alumno podrá aplicar los conceptos DOM para validar formularios con JS.

FUNDAMENTOS DE AJAX



FUNDAMENTOS DE AJAX

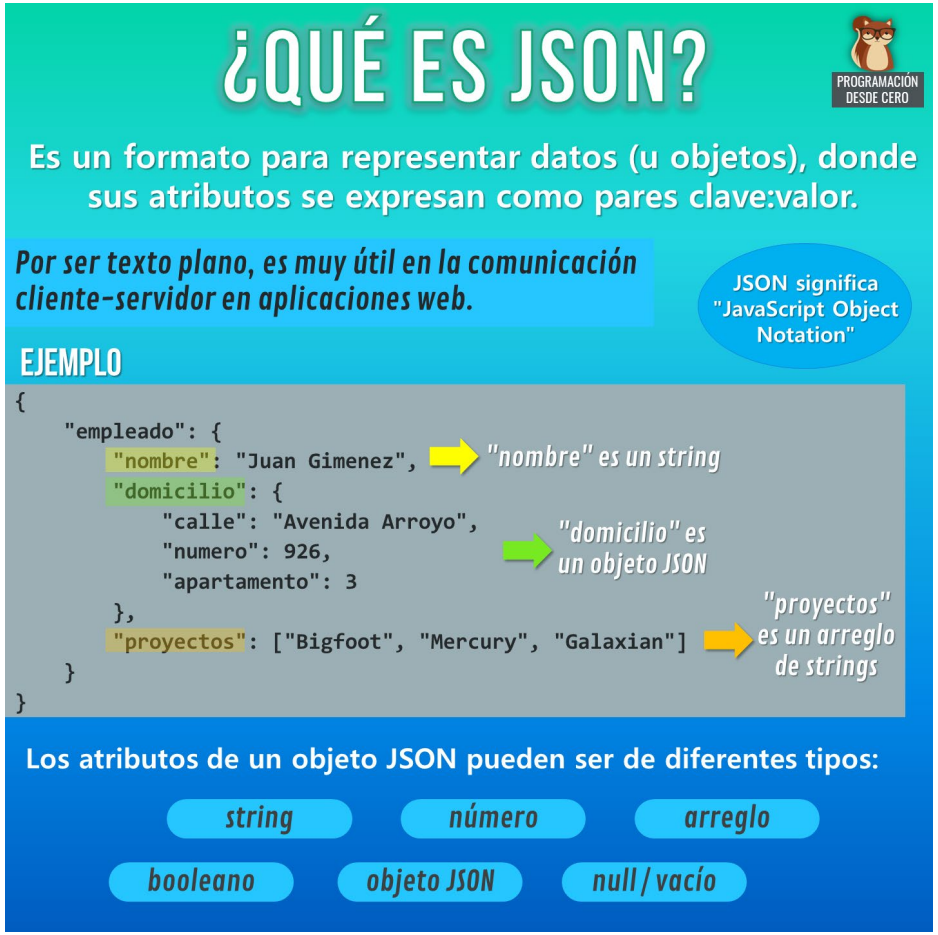
JQUERY :: AJAX

```
$.ajax({
    url: '/api/posts'
    type: 'POST',
    data: {},
    success: function () {},
    error: function () {}
});
```

FUNDAMENTOS DE AJAX

Ejemplo de uso de Ajax:

```
function getAjax(data) {
    $.ajax({
        type: "POST",
        url: "stats.aspx/getSC",
        data: JSON.stringify({ data: data }),
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: function (result) {  result = {d: "[{"TOTSC":241,"TOTALE":1695.7}]"}
            var data = $.parseJSON(result);
            $('#sc-gg').text(data.TOTSC);
        },
        error: function (xhr) {
            alert(xhr.status);
        }
    });

}
```



Laboratorio Nº 3: Crea una pequeña consulta AJAX

- Implementar peticiones AJAX usando jQuery.
- En este laboratorio, el alumno logra realizar peticiones AJAX usando jQuery.

LINKS IMPORTANTES

- https://platzi.com/clases/1814-basico-javascript/
- https://lenguajejs.com/javascript/
- https://uniwebsidad.com/libros/javascript/capitulo-4/funciones
- https://blog.hubspot.es/website/funciones-javascript
- https://somospnt.com/blog/191-eventos-con-javascript
- https://davidinformatico.com/poo-en-javascript
- https://medium.com/@jmz12/javascript-el-paradigma-de-poo-b04d19b6322c
- https://programadorviking.com.br/o-que-e-dom-o-guia-completo-sobre-document-object-model/
- https://rfcosta85.medium.com/conhecendo-o-document-object-model-dom-parte-i-b6e859fd0413
- https://it.javascript.info/browser-environment
- https://desarrolloweb.com/articulos/1767.php
- http://patriciaemiguel.com/conceptos/2021/01/11/json.html
- https://medium.com/@ipenywis/ajax-json-explained-examples-c3e983da8311

¿Qué te llevas de la sesión?

idat

PREGUNTAS

!GRACIASi

REFERENCIAS:

- Curso Básico de JavaScript (Diego De Granda)
- Lenguaje Javascript (Manz.dev)
- Funciones de JavaScript (Maria Coppola)
- Eventos con Javascript (Federico Lloves)
- POO en Javascript (David Poza Suárez)
- El paradigma de POO (Jorge Mendez)
- ¿Qué es DOM? (Programador Viking)
- Conociendo el Modelo de Objetos de Documento (Rodrigo Fernández)
- Browser come ambiente, specifiche (Javascript Info)
- Representar datos con el formato JSON (Programación desde cero)
- AJAX & JSON Explained (Islem Maboud)