



CJAVA

The logo consists of a stylized 'C' and 'JAVA' wordmark. Above the 'C', there is a graphic element composed of three interlocking 3D cubes forming a cube-like shape with a small cross on top.



CJAVA
siempre para apoyarte

010101010101010101010101010101
001010101010101010101010101010101
0101010101010101010101010101010101
01010101010101010101
1010101010101010
0101010101010101010101010101010101
101010101010101010101001010010101010101
01010101010101

Visión

Poder aportar al desarrollo del País usando tecnología Java.

Quienes Somos

Somos una organización orientada a **desarrollar, capacitar e investigar tecnología JAVA** a través de un prestigioso staff de profesionales a nivel nacional.





cjavaperu.com
info@cjavaperu.com

COMUNIDAD

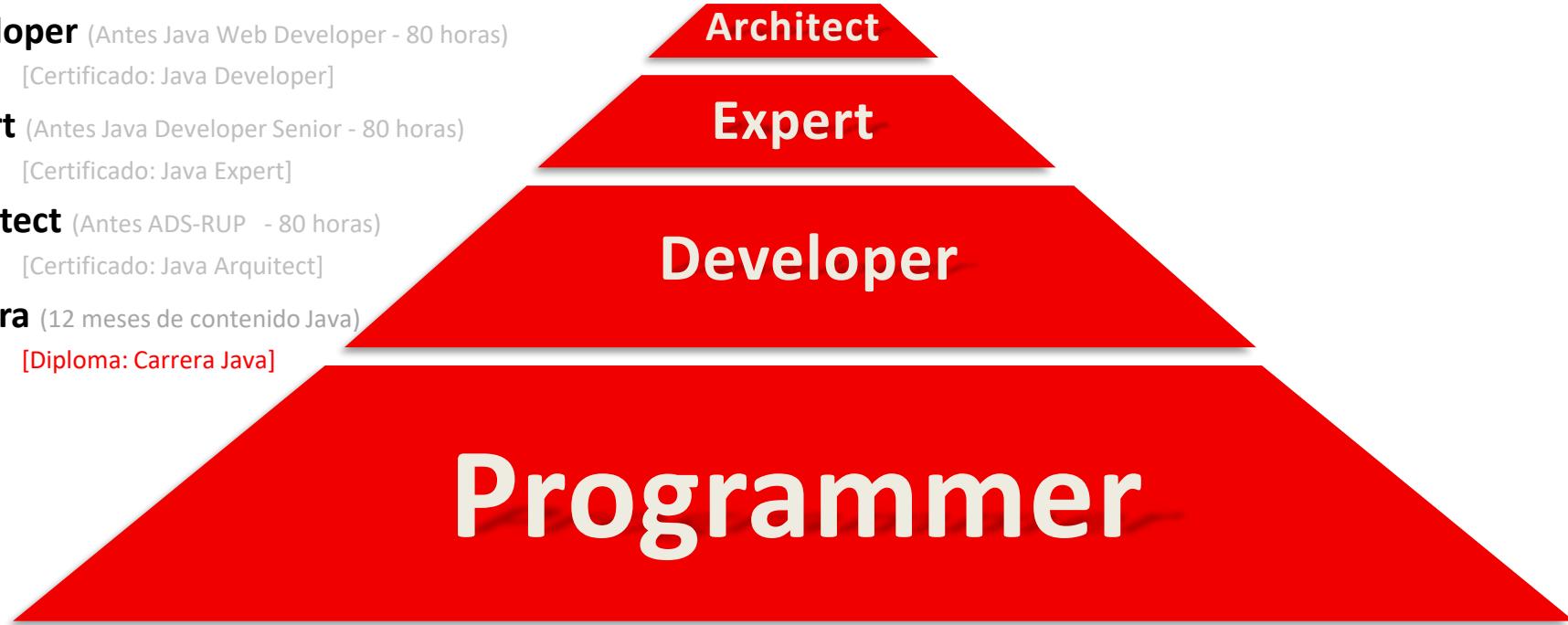
ACADEMICA

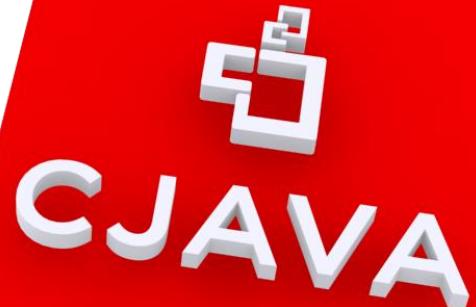


siempre para apoyarte

Servicio de Capacitación

- **Programer** (Antes Java Developer Junior - 80 horas)
[Certificado: Java Programer]
- **Developer** (Antes Java Web Developer - 80 horas)
[Certificado: Java Developer]
- **Expert** (Antes Java Developer Senior - 80 horas)
[Certificado: Java Expert]
- **Arquitect** (Antes ADS-RUP - 80 horas)
[Certificado: Java Arquitect]
- **Carrera** (12 meses de contenido Java)
[Diploma: Carrera Java]

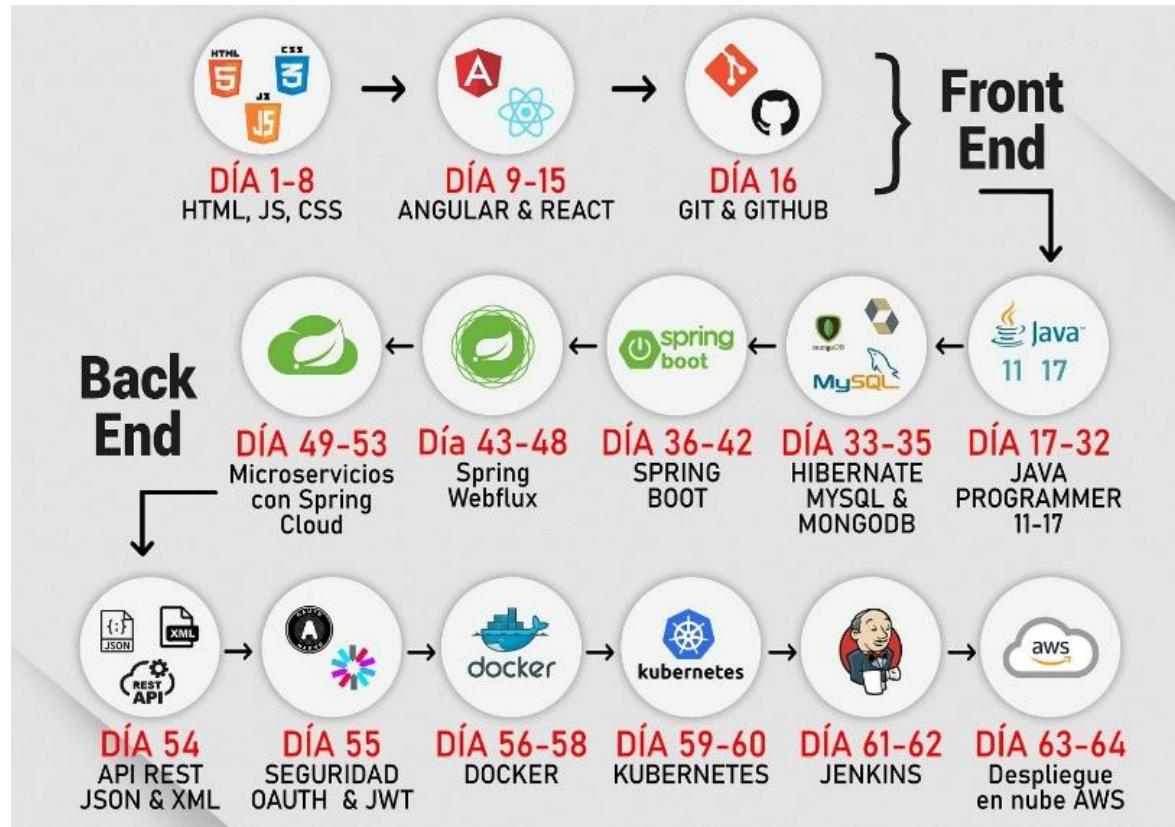




Java FullStack Developer

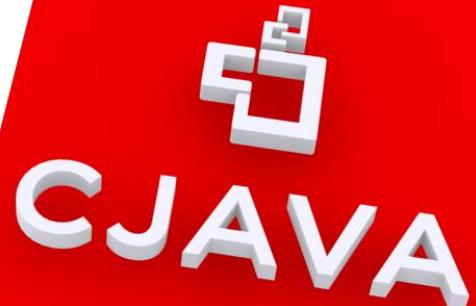
emaravi@cjavaperu.com

Introducción a Full Stack Development



Introducción a Full Stack Development

Front-end Development: Discutir las tecnologías y herramientas utilizadas en el desarrollo front-end, como HTML, CSS, JavaScript y React.



JavaScript

Edwin Maravi

Resultado de aprendizaje

- Al finalizar la sesión, el estudiante construye aplicaciones web con JavaScript diseñando formularios y sus respectivas validaciones.



Contenidos o temas

- Funciones de JavaScript para validar formularios.
- Validación capturando eventos de formulario.
- Validación con formato de datos.

Validar formularios con JavaScript

- Antes de **enviar** datos al servidor, es importante asegurarse de que se **completan** todos los controles de formulario requeridos, y en el **formato** correcto. Esto se denomina **validación** de formulario en el lado del **cliente** y ayuda a garantizar que los datos que se envían coinciden con los requisitos establecidos en los diversos controles de formulario.



Validar formularios con JavaScript

- Accede a cualquier sitio web popular que incluya un formulario de registro y observa que proporcionan **comentarios** cuando no introduces tus datos en el **formato** que se espera. Recibirás mensajes como:
 - «**Este campo es obligatorio**» (No se puede dejar este campo en blanco).
 - «**Introduzca su número de teléfono en el formato xxx-xxxx**» (Se requiere un formato de datos específico para que se considere válido).
 - «**Introduzca una dirección de correo electrónico válida**» (los datos que introdujiste no están en el formato correcto).
 - «**Su contraseña debe tener entre 8 y 30 caracteres y contener una letra mayúscula, un símbolo y un número**». (Se requiere un formato de datos muy específico para tus datos).

Validar formularios con JavaScript

- La principal **utilidad** de JavaScript en el manejo de los formularios es la **validación** de los datos **introducidos** por los usuarios. Antes de enviar un formulario al servidor, se recomienda **validar** mediante **JavaScript** los datos insertados por el usuario. De esta forma, si el usuario ha cometido algún error al llenar el formulario, se le puede **notificar** de forma instantánea, sin necesidad de esperar la respuesta del servidor.



Validar formularios con JavaScript

- A continuación se muestra el código JavaScript básico necesario para incorporar la **validación a un formulario**:

```
<form action="" method="" id="" name="" onsubmit="return validacion()">  
    ...  
</form>
```

Validar formularios con JavaScript

- Y el esquema de la función **validacion()** es el siguiente:

```
function validacion() {  
    if (condicion que debe cumplir el primer campo del formulario) {  
        // Si no se cumple la condicion...  
        alert('[ERROR] El campo debe tener un valor de...');  
        return false;  
    }  
    else if (condicion que debe cumplir el segundo campo del formulario) {  
        // Si no se cumple la condicion...  
        alert('[ERROR] El campo debe tener un valor de...');  
        return false;  
    }  
    ...  
    else if (condicion que debe cumplir el último campo del formulario) {  
        // Si no se cumple la condicion...  
        alert('[ERROR] El campo debe tener un valor de...');  
        return false;  
    }  
  
    // Si el script ha llegado a este punto, todas las condiciones  
    // se han cumplido, por lo que se devuelve el valor true  
    return true;  
}
```

Validar un campo de texto obligatorio

- Se trata de **forzar** al usuario a introducir **un valor** en un **cuadro de texto o textarea** en los que sea obligatorio. La condición en JavaScript se puede indicar como:

```
valor = document.getElementById("campo").value;  
  
if( valor == null || valor.length == 0 || /\^\\s+$/.test(valor) ) {  
    return false;  
}
```



Validar un campo de texto con valores numéricos

- Se trata de **obligar** al usuario a introducir un **valor numérico** en un **cuadro de texto**. La condición JavaScript consiste en:

```
valor = document.getElementById("campo").value;  
  
if( isNaN(valor) ) {  
    return false;  
}
```



Validar que se ha seleccionado una opción de una lista

- Se trata de **obligar** al usuario a **seleccionar** un elemento de una **lista desplegable**. El siguiente código JavaScript permite conseguirlo:

```
indice = document.getElementById("opciones").selectedIndex;  
if( indice == null || indice == 0 ) {  
    return false;  
}  
  
<select id="opciones" name="opciones">  
    <option value="">- Selecciona un valor -</option>  
    <option value="1">Primer valor</option>  
    <option value="2">Segundo valor</option>  
    <option value="3">Tercer valor</option>  
</select>
```

Validar una dirección de email

- Se trata de **obligar** al usuario a introducir una **dirección de email** con un **formato** válido. Por tanto, lo que se comprueba es que la dirección parezca válida, ya que no se comprueba si se trata de una cuenta de correo electrónico real y operativa. La condición JavaScript consiste en:

```
valor = document.getElementById("campo").value;  
  
if( !(/\\w+([-+.'])\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)/.test(valor)) {  
    return false;  
}
```

Validar una fecha

- Las **fechas** suelen ser los campos de formulario más complicados de validar por la multitud de formas diferentes en las que se pueden introducir. El siguiente código asume que de alguna forma se ha obtenido el año, el mes y el día introducidos por el usuario:

```
var ano = document.getElementById("ano").value;
var mes = document.getElementById("mes").value;
var dia = document.getElementById("dia").value;

valor = new Date(ano, mes, dia);

if( isNaN(valor) ) {
    return false;
}
```

Validar que un checkbox ha sido seleccionado

Si un elemento de tipo **checkbox** se debe seleccionar de forma obligatoria, JavaScript permite comprobarlo de forma muy sencilla:

```
elemento = document.getElementById("campo");
if( !elemento.checked ) {
    return false;
}
```



Validar que un radiobutton ha sido seleccionado

Aunque se trata de un caso similar al de los *checkbox*, la validación de los ***radiobutton*** presenta una diferencia importante: en general, la comprobación que se realiza es que el usuario haya seleccionado algún *radiobutton* de los que forman un determinado grupo. Mediante JavaScript, es sencillo determinar si se ha seleccionado algún *radiobutton* de un grupo:

```
opciones = document.getElementsByName("opciones");

var seleccionado = false;
for(var i=0; i<opciones.length; i++) {
    if(opciones[i].checked) {
        seleccionado = true;
        break;
    }
}

if(!seleccionado) {
    return false;
}
```



Eventos utilizados en la Validación de Formularios con JavaScript

Los **eventos** más utilizados en el manejo de los formularios son los siguientes:

onclick: Evento que se produce cuando se pincha con el ratón sobre un elemento. Normalmente se utiliza con cualquiera de los tipos de botones que permite definir XHTML (<input type="button">, <input type="submit">, <input type="image">).

onchange: Evento que se produce cuando el usuario cambia el valor de un elemento de texto (<input type="text"> o <textarea>). También se produce cuando el usuario selecciona una opción en una lista desplegable (<select>). Sin embargo, el evento sólo se produce si después de realizar el cambio, el usuario pasa al siguiente campo del formulario, lo que técnicamente se conoce como que “el otro campo de formulario ha perdido el foco”.

Eventos utilizados en la Validación de Formularios con JavaScript

Los **eventos** más utilizados en el manejo de los formularios son los siguientes:

onfocus: Evento que se produce cuando el usuario selecciona un elemento del formulario.

onblur: Evento complementario de onfocus. Este se produce cuando el usuario ha deseleccionado un elemento por haber seleccionado otro elemento del formulario. Técnicamente, se dice que el elemento anterior “**ha perdido el foco**”.



Expresiones regulares en JavaScript

Las **expresiones**, son un lenguaje utilizado para describir **patrones** en cadenas de caracteres. Forman un pequeño y separado lenguaje, que está incluido en JavaScript (y en la gran mayoría de lenguajes de programación). No es un lenguaje fácil de leer, pero es una herramienta muy poderosa que simplifica mucho tareas de procesado de cadenas de caracteres.



Expresiones regulares en JavaScript

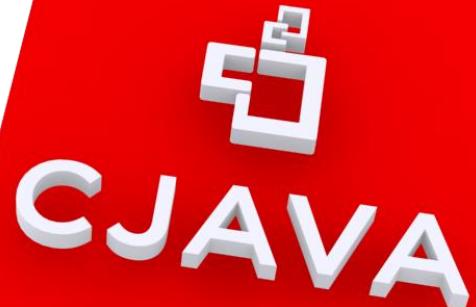
¿Qué significa cada elemento de la **expresión regular** de este ejemplo?

```
function literal() {
    var m = document.getElementById("matricula").value;
    var expreg = /^[A-Z]{1,2}\s\d{4}\s([B-D][F-H][J-N][P-T][V-Z])\{3\}$/;

    if(expreg.test(m))
        alert("La matrícula es correcta");
    else
        alert("La matrícula NO es correcta");
}
```



^ : el emparejamiento se debe realizar desde el principio de la cadena.
[A-Z] : cualquier carácter entre la A mayúscula y la Z mayúscula.
{1,2} : uno o dos caracteres.
\s : un espacio en blanco.
\d : un dígito.
{4} : cuatro dígitos.
\s : un espacio en blanco.
([B-D][F-H][J-N][P-T][V-Z]) : cualquier carácter entre la B mayúscula y la Z mayúscula, excepto las vocales.
{3} : tres caracteres.
\$: el emparejamiento se debe realizar hasta el final de la cadena.



JavaScript

Edwin Maravi



Contenidos o temas

- Introducción al framework
- Formas de uso con bootstrap
- Formularios con bootstrap
- Contenidos con bootstrap
- Componentes con bootstrap

Introducción a Bootstrap

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**.



Introducción a Bootstrap

- Bootstrap, es un **framework** originalmente creado por Twitter, que permite crear interfaces web, cuya particularidad es la de **adaptar** la interfaz del sitio web al tamaño del dispositivo en que se visualice.
- Bootstrap es el segundo proyecto más destacado en GitHub y es usado por la NASA y la MSNBC, entre otras organizaciones.



Introducción a Bootstrap

- Bootstrap, originalmente llamado Blueprint de Twitter, fue desarrollado por **Mark Otto** y **Jacob Thornton** de Twitter, como un marco de trabajo (framework) para fomentar la consistencia entre las herramientas internas.
- La **versión 5** de Bootstrap fue anunciada por Mark Otto el 21 de diciembre de 2018.



Introducción a Bootstrap

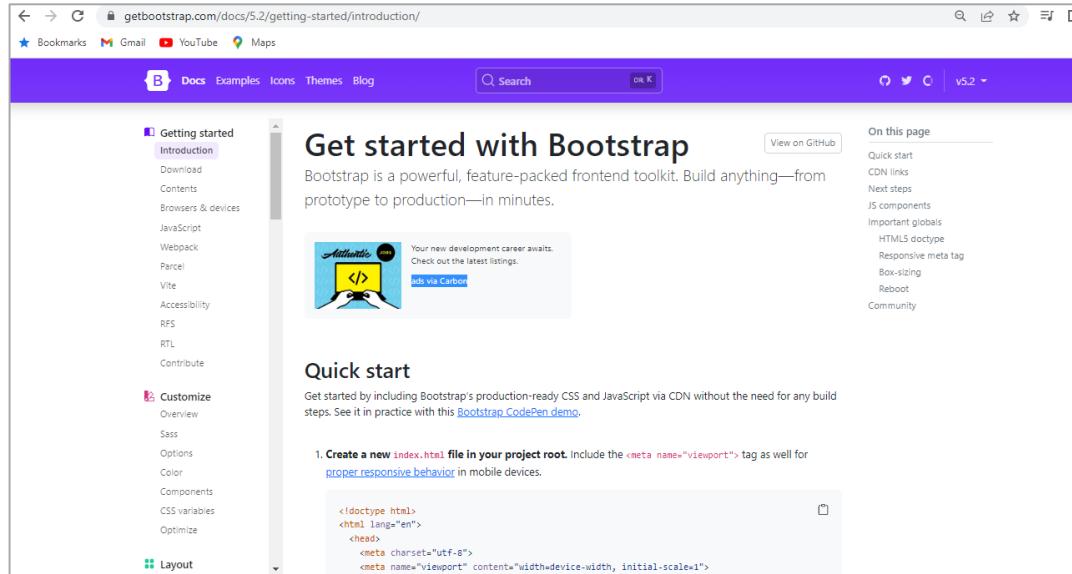
Características:

- Facilidad de uso
- Responsive
- Mobile-first
- Reutilización de código
- Compatibilidad con los navegadores
- Sistema de imágenes de Bootstrap
- Comunidad activa
- Documentación de Bootstrap



Formas de uso con Bootstrap

Documentación oficial: <https://getbootstrap.com/docs/5.2/getting-started/introduction/>



The screenshot shows the official Bootstrap documentation website at <https://getbootstrap.com/docs/5.2/getting-started/introduction/>. The page has a purple header with navigation links for Docs, Examples, Icons, Themes, and Blog. A search bar and a version selector (v5.2) are also present. The main content area features a sidebar with sections like 'Getting started' (Introduction, Download, Contents, Browsers & devices, JavaScript, Webpack, Parcel, Vite, Accessibility, RFS, RTL, Contribute), 'Customize' (Overview, Sass, Options, Color, Components, CSS variables, Optimize), and 'Layout'. The main content area displays the 'Get started with Bootstrap' section, which includes a brief introduction, a 'Quick start' guide, and a code snippet for creating an index.html file. On the right side, there's a sidebar titled 'On this page' with links to various Bootstrap components and resources.

Formas de uso con Bootstrap

- Comience incluyendo el **CSS** y **JavaScript** listos para producción de **Bootstrap** a través de **CDN** sin necesidad de ningún paso de compilación.
- Como referencia, aquí están nuestros enlaces principales de CDN:

Description	URL
CSS	https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css
JS	https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/js/bootstrap.bundle.min.js

Formas de uso con Bootstrap

Coloque la `<link>`etiqueta en el `<head>`para nuestro CSS y la `<script>`etiqueta para nuestro paquete de JavaScript (incluido Popper para colocar menús desplegables, ventanas emergentes e información sobre herramientas) antes del cierre `</body>`.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css" re
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/js/bootstrap.bundle.min.j
  </body>
</html>
```

Formas de uso con Bootstrap

Bootstrap requiere el uso del tipo de **documento HTML5**. Sin él, verás un estilo funky e incompleto.

```
<!doctype html>
<html lang="en">
  ...
</html>
```

Formas de uso con Bootstrap

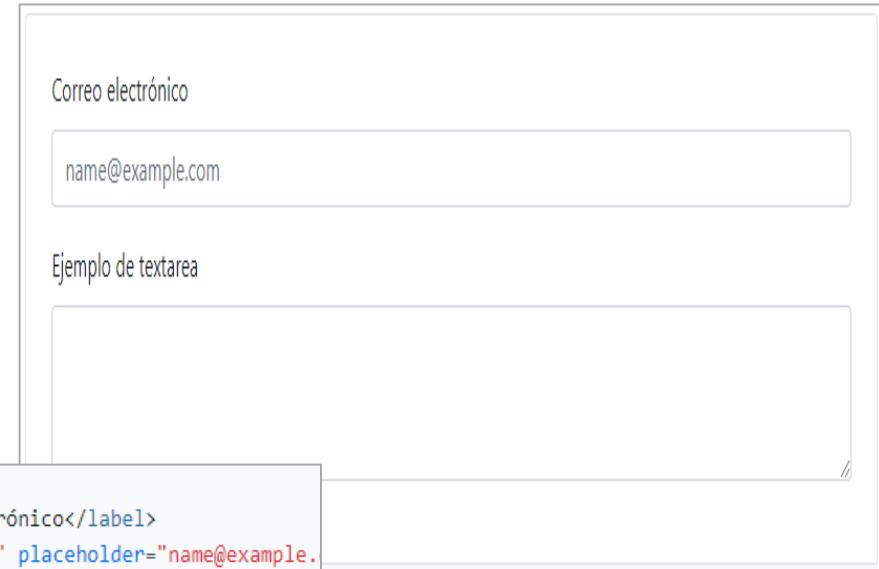
- Bootstrap se desarrolla **mobile first**, una estrategia en la que primero optimizamos el código para dispositivos móviles y luego escalamos los componentes según sea necesario utilizando consultas de medios CSS. Para garantizar una representación adecuada y el zoom táctil para todos los dispositivos, agregue la metaetiqueta de ventana gráfica sensible a su archivo <head>.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Controles de formulario con Bootstrap

- Proporciona a controles de formulario como `<input>` y `<textarea>` una actualización con estilos personalizados, tamaño, estados de enfoque y más. Veamos el siguiente ejemplo:

```
<div class="mb-3">
  <label for="exampleFormControlInput1" class="form-label">Correo electrónico</label>
  <input type="email" class="form-control" id="exampleFormControlInput1" placeholder="name@example.com" required>
</div>
<div class="mb-3">
  <label for="exampleFormControlTextarea1" class="form-label">Ejemplo de textarea</label>
  <textarea class="form-control" id="exampleFormControlTextarea1" rows="3"></textarea>
</div>
```



The screenshot shows a user interface with two form fields. The first field is a text input labeled "Correo electrónico" with the placeholder "name@example.com". The second field is a text area labeled "Ejemplo de textarea". Both fields are styled with rounded corners and a light gray background.

Controles de formulario con Bootstrap

- Si deseas tener elementos `<input readonly>` en tu formulario con estilo de texto sin formato, usa la clase `.form-control-plaintext` para eliminar el estilo de campo de formulario predeterminado y conservar el margen y padding correctos.

```
<div class="mb-3 row">
  <label for="staticEmail" class="col-sm-2 col-form-label">Email</label>
  <div class="col-sm-10">
    <input type="text" readonly class="form-control-plaintext" id="staticEmail" value="email@example.com" />
  </div>
</div>
<div class="mb-3 row">
  <label for="inputPassword" class="col-sm-2 col-form-label">Contraseña</label>
  <div class="col-sm-10">
    <input type="password" class="form-control" id="inputPassword" />
  </div>
</div>
```



A screenshot of a web form demonstrating Bootstrap styling. It shows two fields: an email input and a password input. The email input has a placeholder "Email" and contains the value "email@example.com". The password input has a placeholder "Contraseña". Both inputs are styled with the `.form-control` class, which provides a standard look and feel for form controls.



Controles de formulario con Bootstrap

- Los menús `<select>` personalizados solo necesitan una clase personalizada, `.form-select` para activar los estilos personalizados. Los estilos personalizados están limitados a la apariencia inicial de `<select>` y no pueden modificar `<option>` debido a las limitaciones del navegador.



A screenshot of a dropdown menu. The visible part of the menu contains the text "Abre este menú select". Below it is a small downward arrow indicating it can be expanded. The expanded menu shows the following HTML code:

```
<select class="form-select" aria-label="Default select example">
  <option selected>Abre este menú select</option>
  <option value="1">Uno</option>
  <option value="2">Dos</option>
  <option value="3">Tres</option>
</select>
```



Controles de formulario con Bootstrap

- Las **casillas de verificación y radios predeterminadas** del navegador se reemplazan con la ayuda de `.form-check`, una serie de clases para ambos tipos de entrada que mejoran el diseño y el comportamiento de tus elementos HTML.



Controles de formulario con Bootstrap

- Casilla de verificación:

- Casilla de verificación por defecto
- Casilla de verificación marcada

```
<div class="form-check">
  <input class="form-check-input" type="checkbox" value="" id="flexCheckDefault">
  <label class="form-check-label" for="flexCheckDefault">
    Casilla de verificación por defecto
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="checkbox" value="" id="flexCheckChecked" checked>
  <label class="form-check-label" for="flexCheckChecked">
    Casilla de verificación marcada
  </label>
</div>
```

Controles de formulario con Bootstrap

- Botón de radios:

- Radio por defecto
 Radio marcado por defecto

```
<div class="form-check">
  <input class="form-check-input" type="radio" name="flexRadioDefault" id="flexRadioDefault1">
  <label class="form-check-label" for="flexRadioDefault1">
    Radio por defecto
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="flexRadioDefault" id="flexRadioDefault2" checked="checked">
  <label class="form-check-label" for="flexRadioDefault2">
    Radio marcado por defecto
  </label>
</div>
```

Contenidos con Bootstrap

- Usando el marcado de **tabla** más básico, así es como se ven las tablas basadas en .table en Bootstrap.

#	Primerº	Último	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

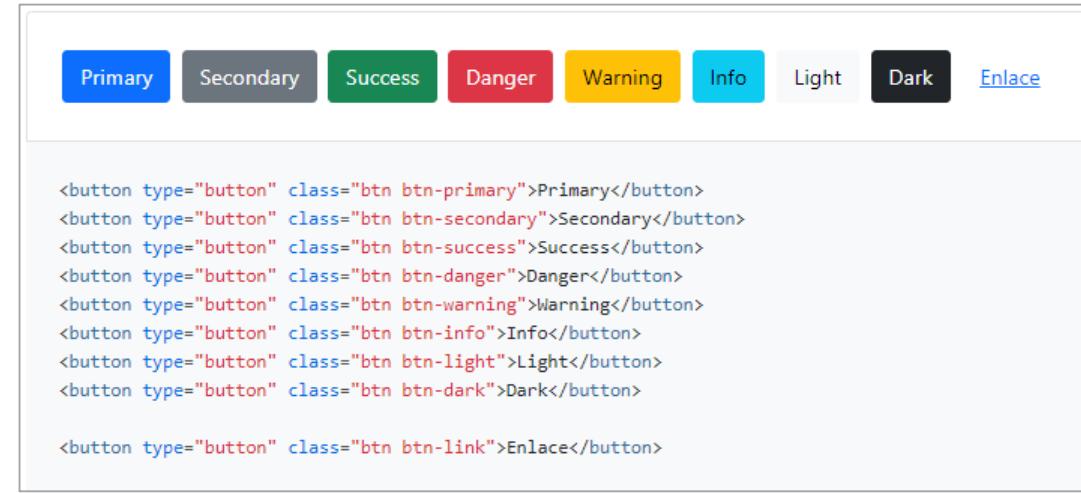
Contenidos con Bootstrap

Usando el marcado de **tabla** más básico, así es como se ven las tablas basadas en .table en Bootstrap.

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">Primero</th>
      <th scope="col">Último</th>
      <th scope="col">Handle</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1</th>
      <td>Mark</td>
      <td>Otto</td>
      <td>@mdo</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>Jacob</td>
      <td>Thornton</td>
      <td>@fat</td>
    </tr>
    <tr>
      <th scope="row">3</th>
      <td colspan="2">Larry the Bird</td>
      <td>@twitter</td>
    </tr>
  </tbody>
</table>
```

Componentes con Bootstrap

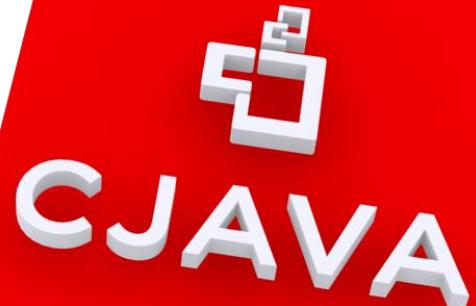
Bootstrap incluye varios estilos de **botones** predefinidos, cada uno con su propio propósito semántico, con algunos extras incluidos para un mayor control.



The screenshot shows a collection of Bootstrap buttons arranged horizontally. From left to right, they are: Primary (blue), Secondary (gray), Success (green), Danger (red), Warning (yellow), Info (light blue), Light (very light gray), Dark (black), and Enlace (link). Below the buttons is a block of HTML code demonstrating their creation:

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
<button type="button" class="btn btn-danger">Danger</button>
<button type="button" class="btn btn-warning">Warning</button>
<button type="button" class="btn btn-info">Info</button>
<button type="button" class="btn btn-light">Light</button>
<button type="button" class="btn btn-dark">Dark</button>

<button type="button" class="btn btn-link">Enlace</button>
```



JavaScript

Edwin Maravi



Contenidos o temas

- Introducción al jQuery
- Formas de uso jQuery
- Sintaxis de jQuery
- Elementos y eventos de jQuery

Introducción a jQuery

jQuery es una **biblioteca de JavaScript**, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos **HTML**, manipular el **árbol DOM**, manejar eventos, desarrollar animaciones y agregar interacción con la técnica **AJAX** a páginas web.



Introducción a jQuery

jQuery incluye las siguientes características:

Selección de elementos DOM

Interactividad y modificaciones del árbol DOM

Eventos.

Manipulación de la hoja de estilos CSS.

Efectos y animaciones.

Animaciones personalizadas.

AJAX.

Objetos diferidos y de promesa para controlar el procesamiento asíncrono

Soporta extensiones (JSON).

Soporte para múltiples navegadores



Formas de uso jQuery

Hay varias formas de iniciar el uso de jQuery en su sitio web. Usted puede:

Descargar la librería jQuery de **jQuery.com**

Incluir jQuery desde una **CDN** (Content Delivery Network), como Google



Formas de uso jQuery

- **Descarga de jQuery:** Hay dos versiones de jQuery disponibles para descargar:
 - **Versión de producción** - esto es para su sitio en línea.
 - **Versión de desarrollo** - esto es para pruebas y desarrollo (código sin comprimir y legible).
- Ambas **versiones** se pueden descargar desde **jQuery.com**.



Formas de uso jQuery

La **librería** jQuery es un **solo archivo** de JavaScript, y hace referencia a ella con la etiqueta HTML <script> (nótese que la etiqueta <script> debe estar dentro de la sección <head>):

```
<head>
  <script src="jquery-1.11.2.min.js"></script>
</head>
```



Formas de uso jQuery

- Si no desea descargar y alojar jQuery, puede incluir desde un **CDN** (Content Delivery Network).
- Tanto Google como Microsoft tienen alojado el jQuery.
- Para utilizar jQuery de **Google** utilice:

```
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/jque
ry/1.11.2/jquery.min.js"></script>
</head>
```



Sintaxis de jQuery

La **sintaxis** de jQuery sido elaborado expresamente para la selección de los elementos HTML y realizar alguna acción sobre los elementos.

Sintaxis básica es: **`$(selector).action()`**

Un signo **\$** para definir/acceder jQuery

Un **(selector)** para "consulta (o encontrar)" elementos HTML

Una acción **jQuery()** para realizarse sobre los elementos



Sintaxis de jQuery

Ejemplos:

- **`$(this).hide()`**: Oculta el elemento actual.
- **`$("p").hide()`**: Oculta todos los elementos `<p>`
- **`$(".test").hide()`**: Oculta todos los elementos con class = "test".
- **`$("# test").hide()`**: Oculta el elemento con id= "test"



El evento **\$(document).ready()**

No es posible interactuar de forma segura con el contenido de una página hasta que el documento no se encuentre preparado para su manipulación. jQuery permite detectar dicho estado a través de la declaración **\$(document).ready()** de forma tal que el bloque se ejecutará sólo una vez que la página este disponible.

```
$(document).ready(function() {  
    console.log('el documento está preparado');  
});
```



Selección de elementos

El concepto más básico de jQuery es el de "seleccionar algunos elementos y realizar acciones con ellos". La biblioteca soporta gran parte de los selectores CSS3 y varios más no estandarizados.

En api.jquery.com/category/selectors se puede encontrar una completa referencia sobre los selectores de la biblioteca.

Puede seleccionar todos `<p>` elementos de una página como esta:
 `$("p")`



Selección de elementos

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js">
</script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
</script>
</head>
<body>
    <h2>Encabezado</h2>
    <p>Este es un parrafo.</p>
    <p>Este es otro parrafo.</p>
    <button>Click aqui</button>
</body>
</html>
```



Selección de elementos: #id

- El selector jQuery **#id** utiliza el atributo **id** de una etiqueta HTML para encontrar el elemento específico.
- Un **id** debe ser único dentro de una página, por lo que debe utilizar el selector **#id** cuando se quiere encontrar un único elemento.
- Para encontrar un elemento con un id específico, escriba una almohadilla, seguido por el id del elemento HTML: `$("#test")`



Selección de elementos: #id

- **Ejemplo:**

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js">
</script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("#test").hide();
    });
});
</script>
</head>
<body>
<h2>Encabezado</h2>
<p>Primer parrafo.</p>
<p id="test">Segundo Parrafo.</p>
<button>Click Aquí</button>
</body>
</html>
```



Selección de elementos: .class

- El selector **.class** de jQuery encuentra los elementos con una clase específica.
- Para encontrar los elementos con una clase específica, escriba un carácter de punto, seguido del nombre de la clase: Ejemplo:
\$(".test")
- Cuando un usuario hace clic en un botón, los elementos con **class = "test"** se ocultarán:

```
$(document).ready(function(){
    $("button").click(function(){
        $(".test").hide();
    });
});
```

Selección de elementos

Sintaxis	Descripción
<code>\$("")</code>	Selecciona todos los elementos
<code>\$(this)</code>	Selecciona el elemento HTML actual
<code>\$("p.intro")</code>	Selecciona todos los elementos <code><p></code> con <code>class="intro"</code>
<code>\$("p:first")</code>	Selecciona el primer elemento <code><p></code>
<code>\$("ul li:first")</code>	Selecciona el primer elemento <code></code> del primer <code></code>
<code>\$("ul li:first-child")</code>	Selecciona el primer elemento <code></code> de cada <code></code>
<code>\$("[href]")</code>	Selecciona todos los elementos con atributo href
<code>\$("a[target='_blank']")</code>	Selecciona todos los elementos <code><a></code> con un atributo target con valor igual a <code>_blank</code>
<code>\$("a[target!='_blank']")</code>	Selecciona todos los elementos <code><a></code> con un atributo target cuyo valor no sea igual a <code>_blank</code>
<code>\$(":button")</code>	Selecciona todos los elementos <code><button></code> y elementos <code><input></code> de tipo="button"
<code>\$("tr:even")</code>	Selecciona todos los elementos <code><tr></code> pares
<code>\$("tr:odd")</code>	Selecciona todos los elementos <code><tr></code> impares

Los eventos

- Un **evento** representa el preciso momento en que **sucede algo**.
- **Ejemplos:**
 - Mover el ratón encima de un elemento
 - Seleccionar un botón de radio
 - Al hacer clic en un elemento
- El término "**disparar**" se utiliza a menudo con los eventos.
- Ejemplo: "El evento keypress dispara el momento en que se pulsa una tecla".



Los eventos

Aquí hay algunos eventos DOM comunes:

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload



Los eventos

- En jQuery, la mayoría de los **eventos DOM** tienen un **método** jQuery equivalente.
- Para asignar un evento click a todos los párrafos de una página, usted puede hacer esto: `$(“p”).click();`
- El siguiente paso es definir lo que debería suceder cuando se activa el evento. Se debe pasar una función para el evento:

```
$(“p”).click(function(){  
    // action goes here!!  
});
```

Los eventos

- El método **\$(document).ready()** nos permite ejecutar una función cuando el documento se ha cargado completamente.
- El método **click()** atribuye una función de controlador de eventos para un elemento HTML. La función se ejecuta cuando el usuario hace click en el elemento HTML.
- El siguiente ejemplo dice: Cuando un evento click se desencadena en un elemento **<p>**; ocultar el actual elemento **<p>**:

```
$( "p" ).click(function(){
    $(this).hide();
});
```

Los eventos

- El método **dblclick()**. La función se ejecuta cuando el usuario hace doble clic en el elemento HTML:

```
 $("p").dblclick(function(){
    $(this).hide();
});
```

- El método **mouseenter()**. La función se ejecuta cuando el puntero del ratón entra en el elemento HTML:

```
 $("#p1").mouseenter(function(){
    alert("Entraste al parrafo..!");
});
```

- El método **mousemove()**. La función se ejecuta cuando el puntero del ratón sale del elemento HTML:

```
 $("#p1").mouseleave(function(){
    alert("Adios! Dejaste el parrafo..!");
});
```

Los eventos

- El método **mousedown()**. La función se ejecuta cuando se pulsa el botón izquierdo del ratón, mientras que el ratón está sobre el elemento HTML:

```
$("#p1").mousedown(function(){
    alert("El mouse bajo desde el parrafo!");
});
```

- El método **mouseup()**. La función se ejecuta cuando se suelta el botón izquierdo del ratón, mientras que el ratón está sobre el elemento HTML:

```
$("#p1").mouseup(function(){
    alert("El mouse subio desde el parrafo!");
});
```

Los eventos

- El método **hover()** realiza dos funciones y es una combinación de los métodos **mouseenter()** y **mouseleave()**.
- La primera función se ejecuta cuando el ratón entra en el elemento HTML, y la segunda función se ejecuta cuando el ratón sale del elemento HTML:

```
$("#p1").hover(function(){
    alert("Entraste al parrafo..!");
},
function(){
    alert("Adios! Dejaste el parrafo..!");
});
```

Los eventos

- El método de **focus()** concede una función de controlador de eventos para un campo del formulario HTML. La función se ejecuta cuando el campo de formulario obtiene el enfoque:

```
$(document).ready(function(){
    $("input").focus(function(){
        $(this).css("background-color", "#cccccc");
    });
    $("input").blur(function(){
        $(this).css("background-color", "#ffffff");
    });
});
```

Los eventos

- El método **blur()** concede una función de controlador de eventos para un campo en un formulario HTML. La función se ejecuta cuando el campo del formulario pierde el enfoque:

```
<script  
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/  
jquery.min.js"></script>  
<script>  
$(document).ready(function(){  
    $("input").focus(function(){  
        $(this).css("background-color", "#cccccc");  
    });  
    $("input").blur(function(){  
        $(this).css("background-color", "#555555");  
    });  
});  
</script>
```

Los eventos

- El método **on()** concede uno o más controladores de eventos para los elementos seleccionados.

```
$(document).ready(function(){
    $("p").on({
        mouseenter: function(){
            $(this).css("background-color", "lightgray");
        },
        mouseleave: function(){
            $(this).css("background-color", "lightblue");
        },
        click: function(){
            $(this).css("background-color", "yellow");
        }
    });
});
```



JavaScript

Edwin Maravi

Contenidos o temas

- Introducción Programación orientada a objetos (POO)
- Clase y objeto
- POO con JavaScript
- Instanciar una clase

Introducción Programación Orientada a Objetos (POO)

La Programación Orientada a Objetos (**POO**, en español; **OOP**, según sus siglas en inglés) es un **paradigma de programación** que parte del concepto de "**objetos**" como base, los cuales contienen información en forma de campos (a veces también referidos como atributos o propiedades) y código en forma de métodos.



Introducción Programación Orientada a Objetos (POO)

- Los **objetos** son capaces de **interactuar y modificar** los valores contenidos en sus campos o atributos (estado) a través de sus métodos (comportamiento).
- Muchos de los objetos prediseñados de los lenguajes de programación actuales permiten la agrupación en bibliotecas o librerías, sin embargo, muchos de estos lenguajes permiten al usuario la creación de sus propias bibliotecas.
- Algunas **características clave** de la programación orientada a objetos son *herencia, abstracción, polimorfismo y encapsulamiento*.

Introducción Programación Orientada a Objetos (POO)

Ventajas:

- Proximidad de los conceptos modelados respecto a objetos del mundo real.
- Facilita la reutilización de código. Y por tanto el mantenimiento del mismo.
- Se pueden usar conceptos comunes durante las fases de análisis, diseño e implementación.
- Disipa las barreras entre el qué y el cómo.



Introducción Programación Orientada a Objetos (POO)

- **Desventajas:**

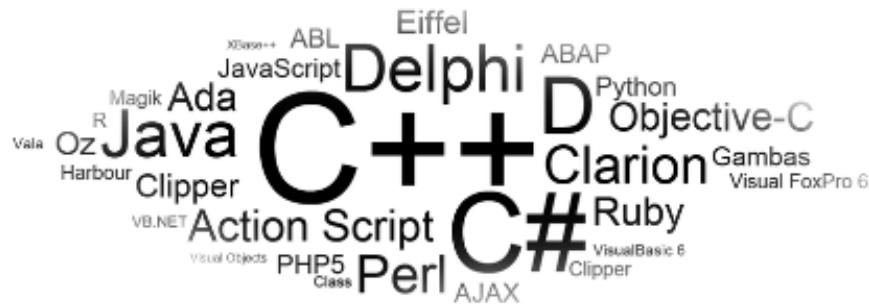
- Mayor complejidad a la hora de entender el flujo de datos
 - Pérdida de linealidad
- Requiere de un lenguaje de modelización de problemas más elaborado:
 - Unified Modelling Language (UML)
 - Representaciones gráficas más complicadas



Introducción Programación Orientada a Objetos (POO)

Algunos lenguajes orientados a objetos:

- Abap
- C++
- C#
- Java
- **JavaScript**
- Objective-C
- VB.NET
- Python
- Ruby
- Php



Clase y Objeto

- Una **clase** es una **estructura** preliminar que describe un objeto y define **atributos** y **operaciones** para el objeto
- Las clases utilizan abstracción para poner a disposición únicamente los elementos esenciales que definen el objeto
- Las clases utilizan encapsulación para garantizar que se cumple una abstracción

Lo que ve el usuario:

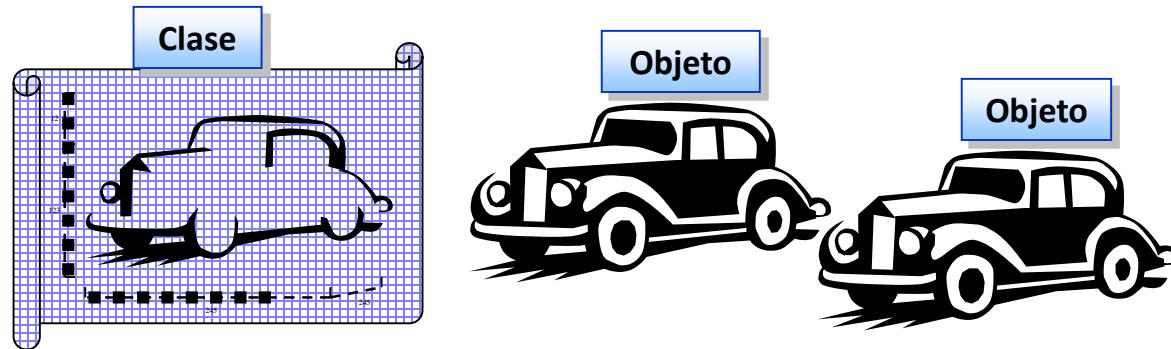


Lo que está encapsulado:



Clase y Objeto

- Un **objeto** es una **instancia** de una clase
- Los objetos tienen las siguientes cualidades:
 - **Identidad:** los objetos se distinguen uno de otro
 - **Comportamiento:** los objetos pueden realizar tareas
 - **Estado:** los objetos almacenan información que puede cambiar con el tiempo.

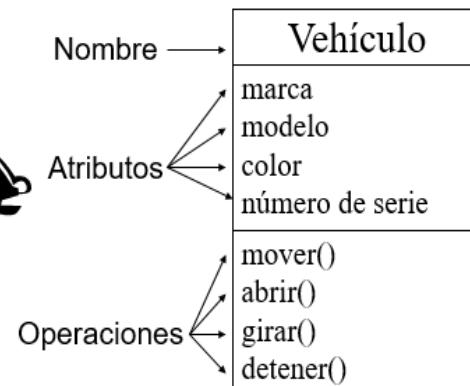


Clase y Objeto

Una clase se compone de:

Campos (atributos, propiedades)

Métodos (operaciones, funciones)



Nombre →

Perro

Atributos →

Nombre
Raza
Altura

Operaciones →

Comer()
Dormir()
Ladrar()

POO con JavaScript

Tradicionalmente, Javascript no soportaba clases de forma nativa, pero en **ECMAScript 2015** se introdujo la posibilidad de **usar clases** simulando como se utilizan en otros lenguajes de programación. Internamente, Javascript traduce estas clases al sistema basado en prototipos que usa en realidad.



POO con JavaScript

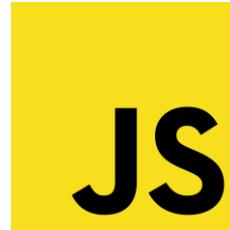
Un **ejemplo básico** donde se incluye propiedades y métodos en JavaScript:

```
class Animal {  
    // Propiedades  
    name = "Garfield";  
    type = "cat";  
  
    // Métodos  
    hablar() {  
        return "Odio los lunes."  
    }  
}
```



Instanciar una clase con JavaScript

Se le llama **instanciar** una clase, **crear un objeto** o crear una instancia a la acción de crear un nuevo objeto basado en una clase particular. Esta acción la realizamos a través de la palabra clave **new**, seguida del nombre de la clase, la cuál puede tener parámetros, en cuyo caso se controlarían desde un constructor.



Instanciar una clase con JavaScript

En Javascript, para instancia una clase, se utiliza una sintaxis muy similar a otros lenguajes como, por ejemplo, Java. Es tan sencillo como escribir lo siguiente:

```
// Declaración de una clase  
class Animal {}  
// Crear (instanciar) un objeto basada en una clase  
const pato = new Animal();
```





CJAVA

siempre para apoyarte

Gracias