

JavaScript en Móviles,  
servidores, backend y  
para tu mascota.



Alberto Luebbert M.  
@almsx

alberto@ideashappy.com  
www.ideashappy.com



ideas *Happy*

# Objetivos



- Conocer el Lenguaje de Programación JavaScript a través de su estructura, variables, funciones y POO.
- Ejercicios de programación en JavaScript.
- Breve introducción al ecosistema móvil.
- Desarrollo de Apps Móviles con Titanium Appcelerator.
- NodeJS.
- Backend
- Mascotas





# ¿Quién soy?



Alberto Luebbert M.

alberto@ideashappy.com

@almsx

ISC Tecnológico de Estudios Superiores del Oriente del Estado de México.

Mobile Developer Android – iOS 2011

Titanium Appcelerator Certified 2012

Mobile Senior Global Hitss Grupo CARSO Ciudad de México

CEO en ideas Happy



JS

# JavaScript



ideas *Happy*



# Objetivo



- Conocer la sintaxis de JavaScript donde aprenderemos a implementar , recorrer estructuras de datos y funciones básicas de la programación.
- Crear sencillas aplicaciones usando HTML y JavaScript para ir ejecutando cada uno de los ejercicios.
- Implementar JavaScript en Móviles.
- JavaScript en Servidores.
- Para tu mascota.



# Introducción a JavaScript



- Creado por Brendan Eich de Netscape en 1995.
- Lenguaje script, interpretado y no compilado.
- JavaScript != Java





# Conceptos Basicos



- Multiparadigma
  - Lenguaje imperativo
  - Procedural
  - POO



# ¿Porqué usarlo?



- Soportado por todos los navegadores, incluso IE :)
- Criticado pero estandar por "default"
- Sintaxis muy sencilla
- Soporte en Frameworks





# Hola Mundo!



```
<script type="text/javascript">
```

```
alert("Hola Mundo");
```

```
</script>
```



ideas *Happy*

# Hola Mundo! Desde un archivo JS

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

```
<script type="text/javascript" src="js/hola.js" >  
alert("Esto no se va a ejecutar nunca!");  
</script>
```

Hola.js

```
alert("Esto si se va a ejecutar!");
```





# Hola mundo! En un link



```
<a href="javascript:alert('Hola mundo!');">
```

```
Soy un link y ejecuto una función de JS</a>
```



# Variables en JavaScript



```
var miVariable;
```

```
var miVariable=1;
```

```
var a,b,c;
```

```
var a,b,c=1;
```

```
var a=1,b=2,c=3;
```

```
var a=b=c=3;
```





# Ejemplo... Operaciones Básicas

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

```
var a=10;
```

```
var b=5;
```

```
document.write(a+" "+b+"="+(a+b)+"<br/>");
```

```
document.write(a+"-"+b+"="+(a-b)+"<br/>");
```

```
document.write(a+"*"+b+"="+(a*b)+"<br/>");
```

```
document.write(a+"/"+b+"="+(a/b)+"<br/>");
```



# Tipos de Datos en JavaScript



```
Xvariable = 345;
```

```
Xvariable = "Hola Mundo!";
```

```
var x;
```

```
var x = 5;
```

```
var x = "Carlos";
```





# Tipos de Datos en JavaScript



```
var carName = "Volvo XC60";
```

```
var carName = 'Volvo XC60';
```

```
var answer = "It's alright";
```

```
var answer = "He is called 'Johnny'";
```

```
var answer = 'He is called "Johnny"'
```



# Tipos de Datos en JavaScript



```
var x1 = 34.00;
```

```
var x2 = 34;
```

```
var y = 123e5;    // 123000000
```

```
var z = 123e-5;   // 0.00123
```





# Tipos de Datos en JavaScript



```
var x = true;
```

```
var y = false;
```



# Arreglos en JavaScript



```
var cars = ["Saab", "Volvo", "BMW"];
```





# Objetos en JavaScript



```
var person = {firstName:"Martin", lastName:"Perez",  
age:30, eyeColor:"black"};
```



# Datos indefinidos en JavaScript



```
var cars;
```

```
person = null;
```





# Datos y Objetos en JavaScript



```
var x = new String();  
var y = new Number();  
var z = new Boolean();
```



# Funciones en JavaScript



- Las funciones son pequeños fragmentos de código que pueden ser ejecutadas n numero de veces.

Se declara de la siguiente manera:

```
function HolaMundo(){  
    alert ("Hola mundo desde una funcion");  
}
```





# Funciones en JavaScript



```
function Calculo(v1,v2) {  
    return v1 * v2;  
}
```



# Funciones en JavaScript



```
function Calculo(v1,v2) {  
    return v1 * v2;  
}
```

- La función anterior permite el paso de parámetros donde obtendríamos al final un resultado de la operación aritmética expuesta.





# Ejercicio



- A través de funciones desarrollar un JavaScript empleando un archivo externo que convierta de Grados Fahrenheit a Grados Celsius.



# Respuesta...



```
function toCelcius(f) {  
    return (f-32) / 1.8;  
}
```

```
Console.log("De Fahrenheit a Celsius "+toCelcius(112));
```

```
function toFahrenheit(g){  
    return g*(1.8)+32;  
}
```

```
document.write("<br/>De Celsius a Fahrenheit "+toFahrenheit(50));
```





# Eventos en JavaScript



- Los eventos en JavaScript permiten ejecutar alguna función al momento en que son mandados llamar; a través de un botón por ejemplo.
- Algunos ejemplos de uso son:
  - Cuando una pagina web ha terminado de cargar.
  - Cuando el valor de un input ha sido modificado.
  - Cuando un botón fue presionado.



# Eventos en JavaScript



- Los siguientes eventos son algunos de los empleados en JavaScript y HTML.

onchange	Cuando un elemento ha sido modificado
onclick	Cuando el usuario hace click sobre un elemento
onmouseover	Cuando el usuario mueve el puntero del mouse a otro elemento del sitio
onload	Cuando el navegador ha terminado de cargar el sitio





# Objetos en JavaScript

JS



## Propiedades

Car.name = Fiat;  
Car.model = 500;  
Car.weight = 850kg;  
Car.color = White;

## Metodos

Car.start();  
Car.drive();  
Car.brake();



# Objetos en JavaScript



```
var person = {  
  firstName: "Martin",  
  lastName : "Perez",  
  id       : 5566,  
  fullName : function () {return this.firstName + " " +  
this.lastName}  
};
```





# Cadenas en JavaScript



- Permite almacenar información, así como su posterior manipulación.
- ```
var modeloCarro = "Volvo XC60";  
var unSolocaracter= modeloCarro[7];
```



# Cadenas en JavaScript



```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
var tamaño = txt.length;
```

- La función **length** permite conocer el tamaño de una cadena de información.





# Cadenas en JavaScript



```
var str = "Localizar palabra 'localizar' por favor!";  
var pos = str.indexOf("localizar");
```



# Cadenas en JavaScript



```
var str = "Localizar palabra 'localizar' por favor!";  
var pos = str.search("localizar");
```





# Cadenas en JavaScript



```
var str = "Uso Windows en mi computadora!";  
var txt = str.replace("Windows", "Linux");
```



# Cadenas en JavaScript



```
var txt = "Hola mundo!";  
var txt1 = txt.toUpperCase();
```





# Cadenas en JavaScript



```
var txt = "HOLA MUNDO!";  
var txt1 = txt.toLowerCase();
```



# Cadenas en JavaScript



```
var str = "a,b,c,d,e,f";  
var arr = str.split(",");  
alert(arr[0]);
```

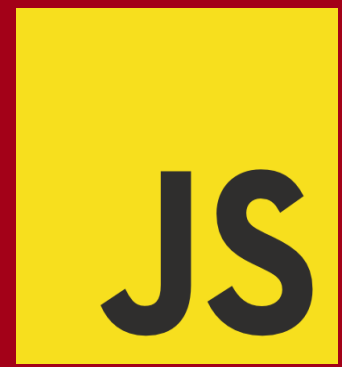
- También es posible utilizar este metodo con los siguientes caracteres

```
txt.split(" ");  
txt.split(" |");
```





# Ejercicio



- Con lo aprendido, desarrollar una aplicación donde el usuario pueda ingresar manualmente o en un arreglo fijo de datos sus datos personales como Nombre, Apellido Paterno, Apellido Materno y genere su RFC.



# Operadores Aritméticos en JS

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

| Operador | Descripción    | Ejemplo                | Valor              | Resultado          |
|----------|----------------|------------------------|--------------------|--------------------|
| +        | Adición        | $X = y + 2$            | $Y = 5$            | $X = 7$            |
| -        | Sustracción    | $X = y - 2$            | $Y = 5$            | $X = 3$            |
| *        | Multiplicación | $X = y * 2$            | $Y = 5$            | $X = 10$           |
| /        | División       | $X = y / 2$            | $Y = 5$            | $X = 2.5$          |
| %        | Modulo         | $X = y \% 2$           | $Y = 5$            | $X = 1$            |
| ++       | Incremento     | $X = ++y$<br>$X = y++$ | $Y = 6$<br>$Y = 6$ | $X = 6$<br>$X = 5$ |
| --       | Decremento     | $X = --y$<br>$X = y--$ | $Y = 4$<br>$Y = 4$ | $X = 4$<br>$X = 5$ |





# Concatenar cadenas



```
txt1 = "Que tengas un";
```

```
txt2 = "Excelente día!";
```

```
txt3 = txt1 + txt2;
```

```
alert(txt3);
```



# Concatenar cadenas



```
txt1 = "Que tengas un ";  
txt1 += "Excelente dia!";
```

```
alert(txt1);
```





# Operadores Matemáticos en JS



- `Math.random`

Genera un valor aleatorio entre 0 y 1. Puede ser implementado por ejemplo para la generación de un password aleatorio numérico.



# Operadores Matemáticos en JS



- `Math.min(0, 150, 30, 20, -8,-22);`

Al hacer uso de esta función es devuelto el valor mínimo de toda una cadena.





# Operadores Matemáticos en JS



- `Math.max(0, 150, 30, 20, -8);`

Por otro lado esta función devuelve el valor más grande encontrado en un arreglo de datos numérico



# Operadores Matemáticos en JS



- `Math.round()`

Redondea un valor numérico a su valor entero más próximo tomando como regla para ello un valor mayor a .5





# Operadores Matemáticos en JS



- `Math.ceil()`

Esta función incrementa a un numero decimal a su numero entero próximo, partiendo de un valor .01



# Operadores Matemáticos en JS



- `Math.floor();`

Por su parte este método devolverá al número entero más próximo cualquier valor otorgado.





# Operadores Matemáticos en JS



¿Qué resultado obtendremos en la siguiente función?

```
function Numero(){  
    var a = Math.floor(Math.random() * 11);  
    alert(a);  
}
```



# Fechas en JavaScript



- Podemos obtener el valor de la fecha actual a través de la función Date().

```
function ObtenerFecha(){  
    alert("La fecha de hoy es "+Date());  
}
```





# Fechas en JavaScript



```
var fecha = new Date();
```

- Obtener Año actual

```
fecha.getFullYear();
```

- Asignar un Año

```
fecha.setFullYear(2020, 10, 3);
```

- Comparar con un arreglo de datos

```
var dias = ["dom", "lun", "mar", "mie", "jue", "vie", "sab"];
```

```
dias[fecha.getDay()];
```



# Comparar Fechas en JS



```
var x = new Date();
```

```
x.setFullYear(2100, 0, 14);
```

```
var today = new Date();
```

```
if (x > today) {
```

```
    alert("Hoy es menor al 14 de Enero de 2100");
```

```
} else {
```

```
    alert("Hoy es después del 14 de Enero de 2100");
```

```
}
```





# Arreglos en JavaScript



- ```
var frutas = ["Banana", "Naranja", "Manzana", "Mango"];  
alert("Tamaño del Arreglo frutas "+frutas.length);
```



# Añadir Elementos al Arreglo



- `var frutas = ["Banana", "Naranja", "Manzana", "Mango"];`
- `frutas[frutas.length] = "Limon";`





# Recorrer un Arreglo



- `var frutas = ["Banana", "Naranja", "Manzana", "Mango"];`

```
for (index = 0; index < frutas.length; index++) {  
    alert(frutas[index]);  
}
```



# De arreglo a Cadena



- `var frutas = ["Banana", "Naranja", "Manzana", "Mango"];`

```
var cadena = frutas.toString();
```

```
alert(cadena);
```





# Añadir elementos a un arreglo



- `var frutas = ["Banana", "Naranja", "Manzana", "Mango"];`

```
var cadena = frutas.join(" * ");
```

```
alert(cadena);
```



# Eliminar elementos de un Arreglo

A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

- `var frutas = ["Banana", "Naranja", "Manzana", "Mango"];`

`frutas.pop();`

`alert(frutas)`





# Añadir elementos a un arreglo



- `var frutas = ["Banana", "Naranja", "Manzana", "Mango"];`

```
frutas.push("Kiwi");
```

```
alert(frutas);
```



# Añadir elementos a un arreglo



- `var frutas = ["Banana", "Naranja", "Manzana", "Mango"];`

```
frutas.push("Kiwi");
```

```
alert(frutas);
```





# Eliminar primer registro en Arreglo



- `var frutas = ["Banana", "Naranja", "Manzana", "Mango"];`

```
frutas.shift();
```

```
alert(frutas);
```

Elimina el primer elemento del arreglo.



# Añadir elemento al inicio de Arreglo



- `var frutas = ["Banana", "Naranja", "Manzana", "Mango"];`

```
frutas.unshift("Limón");
```

```
alert(frutas);
```

Añade el elemento Limón al arreglo al principio.





# Condicionales en JavaScript



If

Sirve para evaluar si un par de variables se cumple una condición, que puede ser un valor mayor, un valor menor o si el valor es igual entre ambas.

- If ( edad == 18)
- If ( edad < 18)
- If ( edad > 18)
- If ( edad <= 18)
- If ( edad >= 18)



# Condicionales en JavaScript



- Else

Es empleado cuando dentro de una estructura condicional que sucederá al dar un valor false.

```
If ( condición){  
    // código  
} else {  
    // código  
}
```





# Condicionales en JavaScript



```
• if (hora < 12) {  
    saludo = "Buenos días";  
} else {  
    saludo = "Buenas tardes";  
}
```



# Condicionales en JavaScript



```
if (time < 11) {  
    saludo = "Buenos dias";  
} else if (time < 20) {  
    saludo = "Buenas tardes";  
} else {  
    saludo = "Buenas noches";  
}
```





# Switch en JavaScript



```
switch(expression) {  
  case n:  
    code block  
    break;  
  case n:  
    code block  
    break;  
  default:  
    default code block  
}
```



# Ciclo while en JavaScript



- Permite que se ejecute una función mientras una condición se este cumpliendo.

```
while (i < 10) {  
    text += "El numero es " + i;  
    i++;  
}
```





# Ciclo do while en JavaScript



- Es una variante de while donde el código se sigue ejecutando mientras revisa si la condición es cumplida (en un estado true) y terminando el recorrido hasta encontrar un false.

```
do {  
    //codigo a ejecutar  
}  
while (condicion);
```



# Errores en JavaScript



- A través de las siguientes funciones podemos controlar los errores que existan ya sea en entorno de ejecución o provocados por el ingreso de información del usuario.
- Try. Seguirá ejecutando el código hasta encontrar un error.
- Catch Permitirá retener el error que se ha generado y darle tratamiento al mismo..





# Convención de Código



- Las convenciones de codificación son las pautas de estilo de programación. Por lo general abarcan:

Nombrar y reglas de declaración de variables y funciones.

Reglas para el uso de espacios en blanco, la sangría y comentarios.

Prácticas y principios de programación



# Nombre de Variables



- W3schools recomienda la forma camelCase o notación Camello donde una de las letras esta declarada en mayúscula y el resto en minúsculas

```
firstName = "John";
```

```
lastName = "Doe";
```

```
price = 19.90;
```

```
discount = 0.10;
```

```
fullPrice = price * 100 / discount;
```





# Declarar variables al principio



- Por regla general debemos declarar nuestras variables a usar a principio de nuestro Script ya que si son declaradas en medio o al final podemos correr el riesgo de sobrescribir el valor en algún momento.

```
var firstName, lastName;
```

```
firstName = "John";
```

```
lastName = "Doe";
```



# Espacios al declarar operadores



`x = 5 + 6;`      `// Bien`

`x=5+6`            `// Mal`

`[40, 100, 1, 5]`   `// Bien`

`[40,100,1,5]`      `// Mal`





# Indentar el Código



```
function toCelsius(fahrenheit) {  
    return (5/9) * (fahrenheit-32);  
}
```

```
for (i = 1; i < 50; i++) {  
    sum += i;  
}
```



# Bloques de código pequeño



```
document.getElementById("demo").innerHTML =  
    "Hello Dolly.";
```

- Es recomendado para efectos de lectura que no sobrepase 80 caracteres.





# Algunas palabras reservadas...

JS

abstract	arguments	boolean	break	byte
case	catch	char	class*	const
continue	debugger	default	delete	do
double	else	enum*	eval	export*
extends*	false	final	finally	float
for	function	goto	if	implements
import*	in	instanceof	int	interface
let	long	native	new	null
package	private	protected	public	return
short	static	super*	switch	synchronized
this	throw	throws	transient	true
try	typeof	var	void	volatile
while	with	yield		



# JSON



- JavaScript Object Notation. Es un formato de transporte de datos. Creado en 2001. Estándar abierto basado en texto. Permite generar estructuras que sean soportadas por otros lenguajes de programación y poder llevar a cabo su manipulación.

```
{"employees":[  
  {"firstName":"Juan", "lastName":"Diaz"},  
  {"firstName":"Anna", "lastName":"Martinez"},  
  {"firstName":"Pedro", "lastName":"Juarez"}  
]}
```





# Desarrollo Móvil



PhoneGap



# Desarrollo Móvil - ionic



- Posee un número importante de librerías propias para el desarrollo de apps nativas usando HTML5 y JS.
- Implementa un patrón MVC para dividir capas y programar como Dios Manda.
- Utiliza Apache Cordova abajo.
- Funciona con Bower.





# Desarrollo Móvil – jQuery Mobile

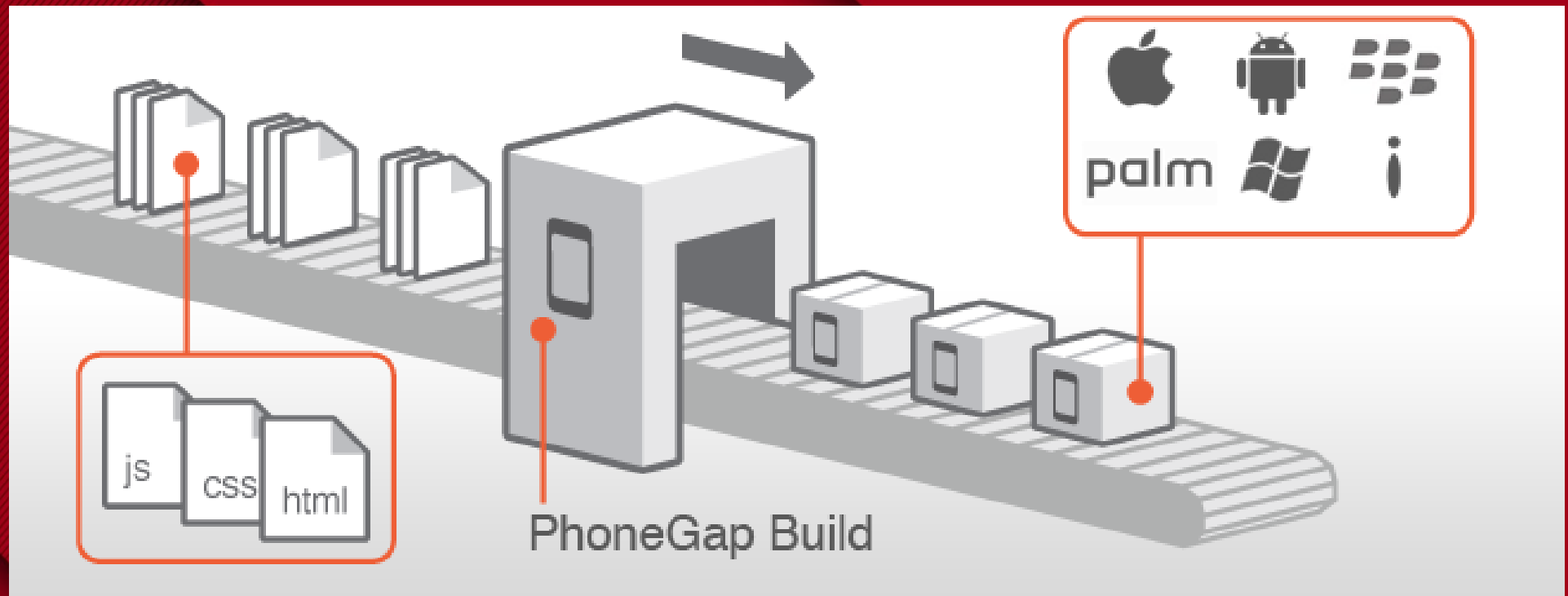


- Themes personalizados.
- Tamaño reducido.
- Facilidad de Uso.
- Múltiples Plataformas.
- Soporte HTML5 .... y obvio JavaScript



# Desarrollo Móvil - Phonegap

JS





# Desarrollo Móvil - Appcelerator



- Desarrollo Nativo (iOS, Android y Windows Phone).
- Conexión a Appcelerator Cloud Service (ACS).
- Soporte Empresarial Appcelerator Studio.



# Algunas Apps hecha con JS

JS



ideas *Happy*



# BackEnd en JavaScript

JS



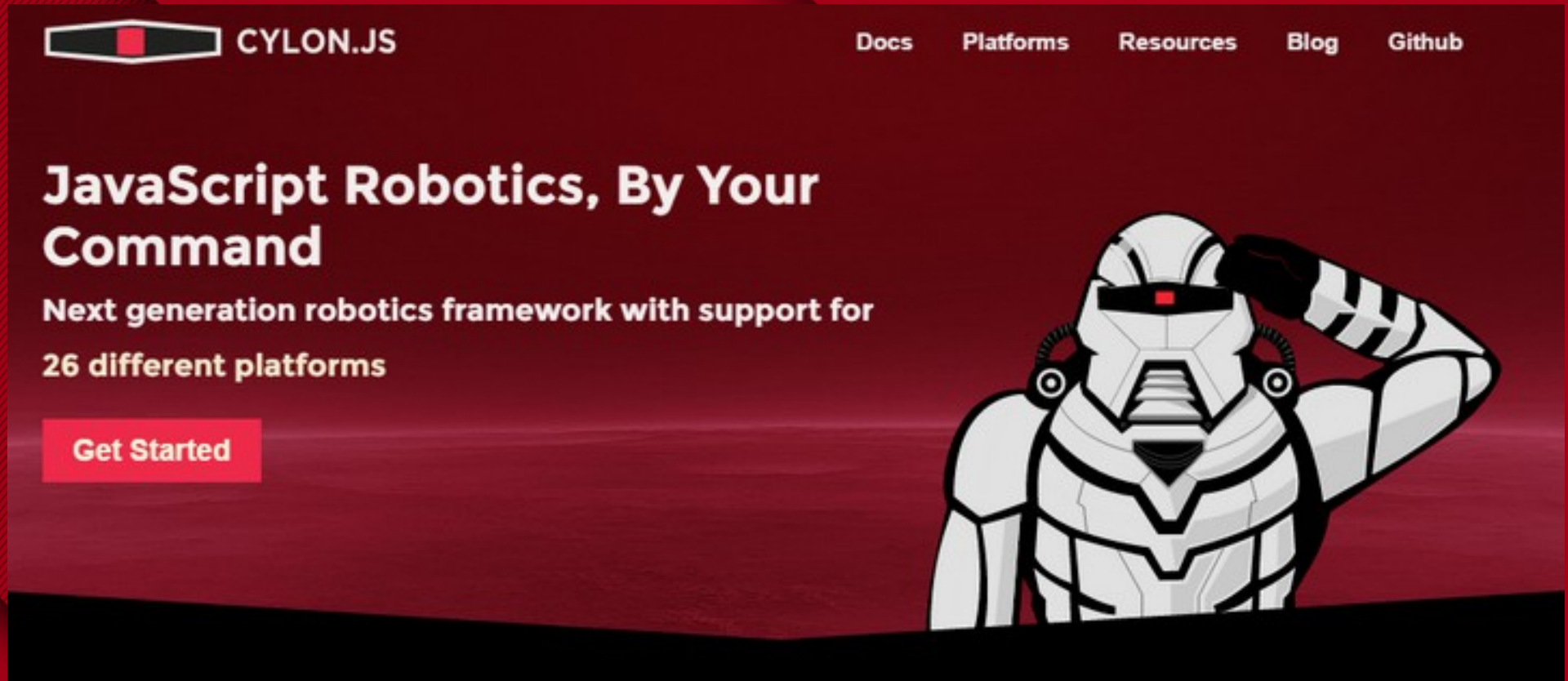
 **Scala** *vs* 



ideas *Happy*

# JavaScript para tu Mascota

JS



The screenshot shows the homepage of the CYLON.JS website. At the top left is the CYLON.JS logo, which consists of a stylized robot head icon followed by the text "CYLON.JS". To the right of the logo is a navigation menu with links for "Docs", "Platforms", "Resources", "Blog", and "Github". The main heading is "JavaScript Robotics, By Your Command". Below this is a subheading: "Next generation robotics framework with support for 26 different platforms". A red button with the text "Get Started" is positioned to the left of a large illustration of a white Cylon robot. The robot is shown from the chest up, wearing a white suit with black accents, and has its right hand raised to its forehead in a salute-like gesture. The background of the website is a dark red gradient.

**CYLON.JS**

[Docs](#) [Platforms](#) [Resources](#) [Blog](#) [Github](#)

## JavaScript Robotics, By Your Command

Next generation robotics framework with support for  
26 different platforms

[Get Started](#)



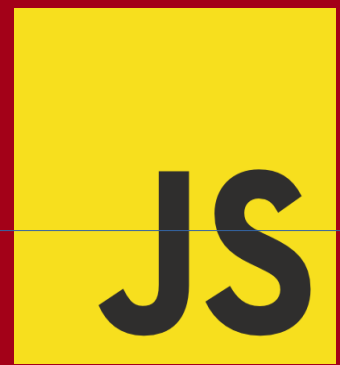
ideas *Happy*



# ¿Dónde Aprender Más?



ideas *Happy*



# JavaScript en Móviles, servidores, backend y para tu mascota.



Alberto Luebbert M.  
@almsx

alberto@ideashappy.com  
www.ideashappy.com



ideas *Happy*