

Lenguajes de Programación

IIC1005

Avisos Varios

- Subiremos un video tutorial sobre git de aquí al final de la semana
- Recibirán el enunciado de tarea chica 1 el lunes

Hoy: Lenguajes de Programación

- Algo de Historia, y luego ...
 - Assembly
 - Fortran
 - C
 - Prolog
 - C++
 - Java / C#
 - Objective-C
 - Python
 - Ruby
 - PHP
 - Javascript
 - R/Matlab (Computacion Cientifica)
 - SQL

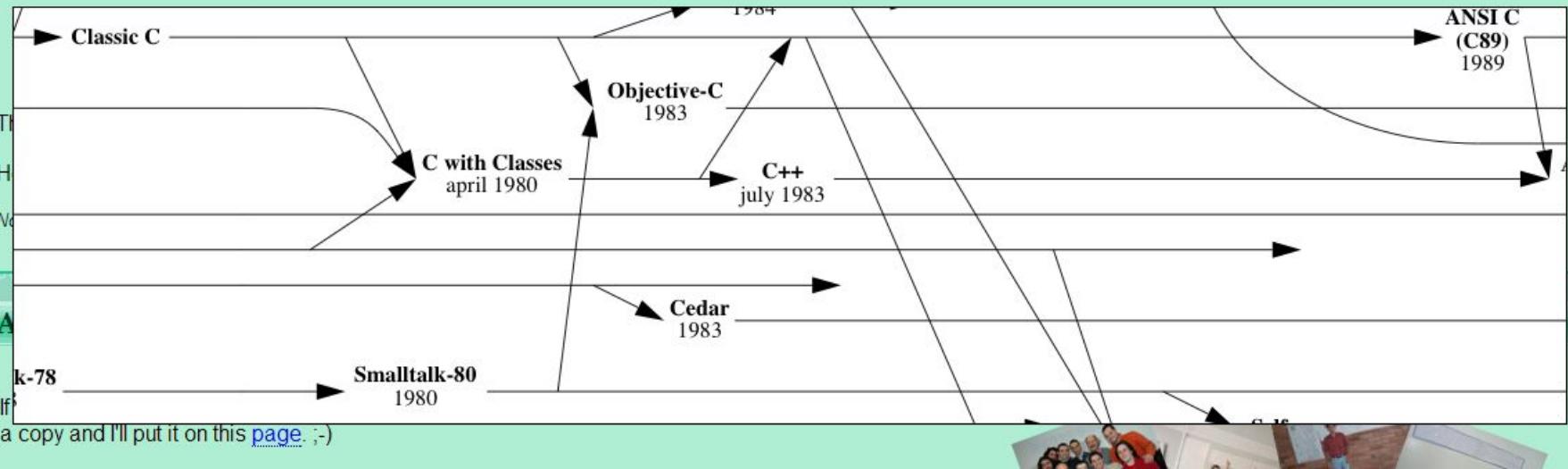
Un poquito de historia

<http://www.levenez.com/lang/>

Below, you can see the preview of the **Computer Languages History** (move on the white zone to get a bigger image):



If you want to print this timeline, you can **freely** download one of the following PDF files:



Lenguajes más populares

<https://octoverse.github.com/>



Otros rankings: TIOBE

<https://www.tiobe.com/tiobe-index/>

Mar 2020	Mar 2019	Change	Programming Language	Ratings	Change
1	1		Java	17.78%	+2.90%
2	2		C	16.33%	+3.03%
3	3		Python	10.11%	+1.85%
4	4		C++	6.79%	-1.34%
5	6	▲	C#	5.32%	+2.05%
6	5	▼	Visual Basic .NET	5.26%	-1.17%
7	7		JavaScript	2.05%	-0.38%
8	8		PHP	2.02%	-0.40%
9	9		SQL	1.83%	-0.09%
10	18	▲	Go	1.28%	+0.26%
11	14	▲	R	1.26%	-0.02%
12	12		Assembly language	1.25%	-0.16%
13	17	▲	Swift	1.24%	+0.08%
14	15	▲	Ruby	1.05%	-0.15%
15	11	▼	MATLAB	0.99%	-0.48%

¿Cuánto importa la popularidad?

- Es importante, pero hay lenguajes que parecen poco populares y son muy usados en ciertas áreas:
- En Bancos y grandes compañías: COBOL
- Aplicaciones matemáticas: FORTRAN
- Etc...

Supongan este requerimiento

<<If somebody came to me and wanted to pay me a lot of money to build a large scale message handling system that really had to be up all the time, could never afford to go down for years at a time, I would unhesitatingly choose to build it in.>>

¿Qué lenguaje elegirían?

VOTEN en SLIDO.COM #10159

Supongan este requerimiento

<<If somebody came to me and wanted to pay me a lot of money to build a large scale message handling system that really had to be up all the time, could never afford to go down for years at a time, I would unhesitatingly choose **ERLANG** to build it in.>>

Tim Bray, director of Web Technologies at Sun Microsystems, keynote at OSCON in July 2008

ERLANG

- Primera versión de Erlang implementada en ProLog
- ¿Quién lo usa?
 - Amazon.com: Para su BD SimpleDB
 - WhatsApp: Para soportar el servicio de mensajería, con 2 millones de usuarios conectados por servidor
 - Bet365: Para el servicio de apuestas inPlay
 -

Erlang <-> Relación con ProLog

- ¿Qué tiene de especial Prolog que no tienen otros lenguajes que han usado antes?

```
-module(count_to_ten).  
-export([count_to_ten/0]).  
  
count_to_ten() -> do_count(0).  
  
do_count(10) -> 10;  
do_count(Value) -> do_count(Value + 1).
```

Relación con ProLog

- Hot Code Loading (Hot Swaping)

```
%% A process whose only job is to keep a counter.
%% First version
-module(counter).
-export([start/0, codeswitch/1]).

start() -> loop(0).

loop(Sum) ->
    receive
        {increment, Count} ->
            loop(Sum+Count);
        {counter, Pid} ->
            Pid ! {counter, Sum},
            loop(Sum);
        code_switch ->
            ?MODULE:codeswitch(Sum)
            % Force the use of 'codeswitch/1' from the latest MODULE version
    end.

codeswitch(Sum) -> loop(Sum).
```

PROLOG

- Logic programming language: asociado a lógica e inteligencia artificial.
- Lenguaje declarativo que expresa relaciones y permite realizar inferencias

mother_child(trude, sally).

father_child(tom, sally).

father_child(tom, erica).

parent_child(X, Y) :- father_child(X, Y).

parent_child(X, Y) :- mother_child(X, Y).

sibling(X, Y) :- parent_child(Z, X), parent_child(Z, Y).

?- sibling(sally, erica). Yes

LOGO

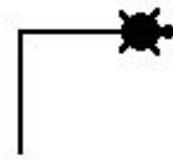
- Creado en 1969 por Seymour Papert, con propósito pedagógico



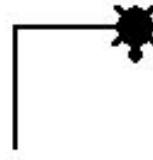
forward 50



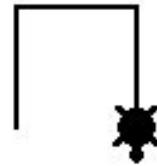
right 90



forward 50



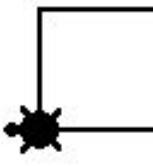
right 90



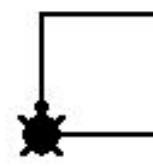
forward 50



right 90



forward 50



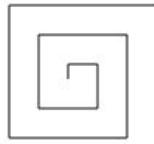
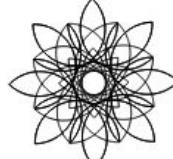
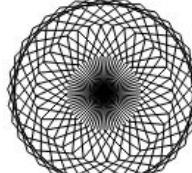
right 90



Por si alguien quiere probar

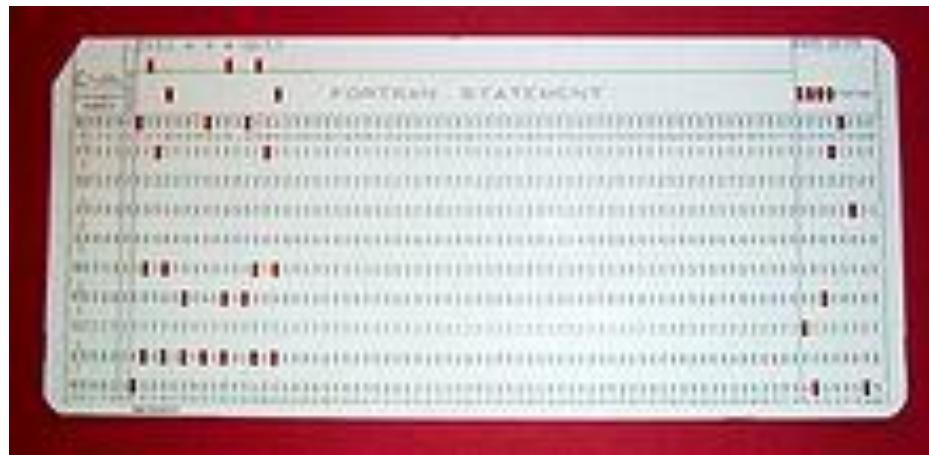
- Turtle academy <http://turtleacademy.com/>

The screenshot shows the homepage of Turtle Academy. At the top, there is a navigation bar with a turtle icon, links for Lessons, User programs, Playground, News, About, a language selector (English), and a Login/Sign Up button. The main title "Turtle Academy" is displayed prominently, followed by the tagline "The easy way to learn programming". Below this, there is a brief description: "Turtle Academy makes it surprisingly easy to start creating amazing shapes using the LOGO language". It also mentions "Here are some examples for easy and fun programming". Three examples are shown in boxes:

- Create a Spiral**
for [i 10 100 10] [fd :i rt 90]
ht

- Cool flower**
repeat 8 [rt 45 repeat 6 [repeat 90 [fd 1 rt 2] rt 90]]
ht

- Crazy octagon**
cs repeat 36 [rt 10 repeat 8 [fd 25 lt 45]] ht


Cuándo fueron creados estos lenguajes?

- Assembly
- FORTRAN
- C
- C++
- Java
- Javascript
- Python
- C#



Cuándo fueron creados estos lenguajes?

- Assembly (1949)
- FORTRAN
- C
- C++
- Java
- Javascript
- Python
- C# (2001)



¿Qué año se creó el lenguaje Python?

VOTEN en SLIDO.COM #10159

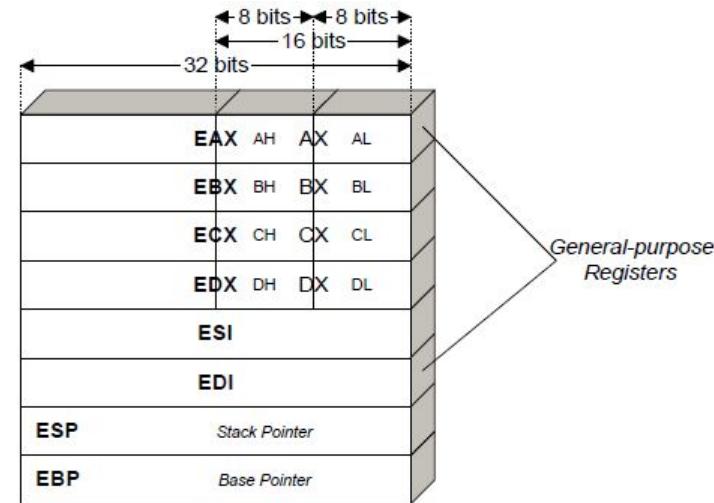
Cuándo fueron creados estos lenguajes?

- Assembly (1949)
- FORTRAN (1953)
- C (1969)
- C++ (1980)
- Java (1995)
- Javascript (1995)
- Python (1991)
- C# (2001)



Ejemplo de Assembly x86

- Así es como sumas dos números:
- Poner primer numero en registro
- Poner segundo numero en registro
- Sumar los registros
- Retornar resultado



```
;n1 db 3; n2 db 7  
mov eax, 3 ; podria ser mov eax, [n1]  
mov ecx, 7; podria ser mov eax, [n2]  
add eax, ecx  
ret
```

```
// equivalente en C  
int a = 3;  
int c = 7;  
a += c;  
return a;
```

Considerando la Arquitectura: Assembly

- Es un lenguaje de bajo nivel, y no porque sea de mala calidad ;-)
- Los lenguajes “Assembly” consideran directamente la arquitectura del equipo (número y tamaño de los registros) y por lo tanto no son portables a otras arquitecturas (como Java, por ejemplo)

Compiladores

- Gracias a los compiladores, podemos escribir instrucciones en lenguajes parecidos al lenguaje natural, los que el computador traduce a lenguajes de más bajo nivel.

Compiladores

- ¿Quién escribió el primer compilador y cuándo?

VOTEN en SLIDO.COM #10159

Compiladores

- ¿Quién escribió el primer compilador ?
 - Grace Hopper, científica estadounidense.
 - El Sistema A-0 fue escrito por Grace Hopper en 1951 y 1952 para UNIVAC I. Fue el primer compilador desarrollado para una computadora electrónica.
 - Luego inventó el B-O y el FLOWMATIC, del que se deriva el lenguaje COBOL.

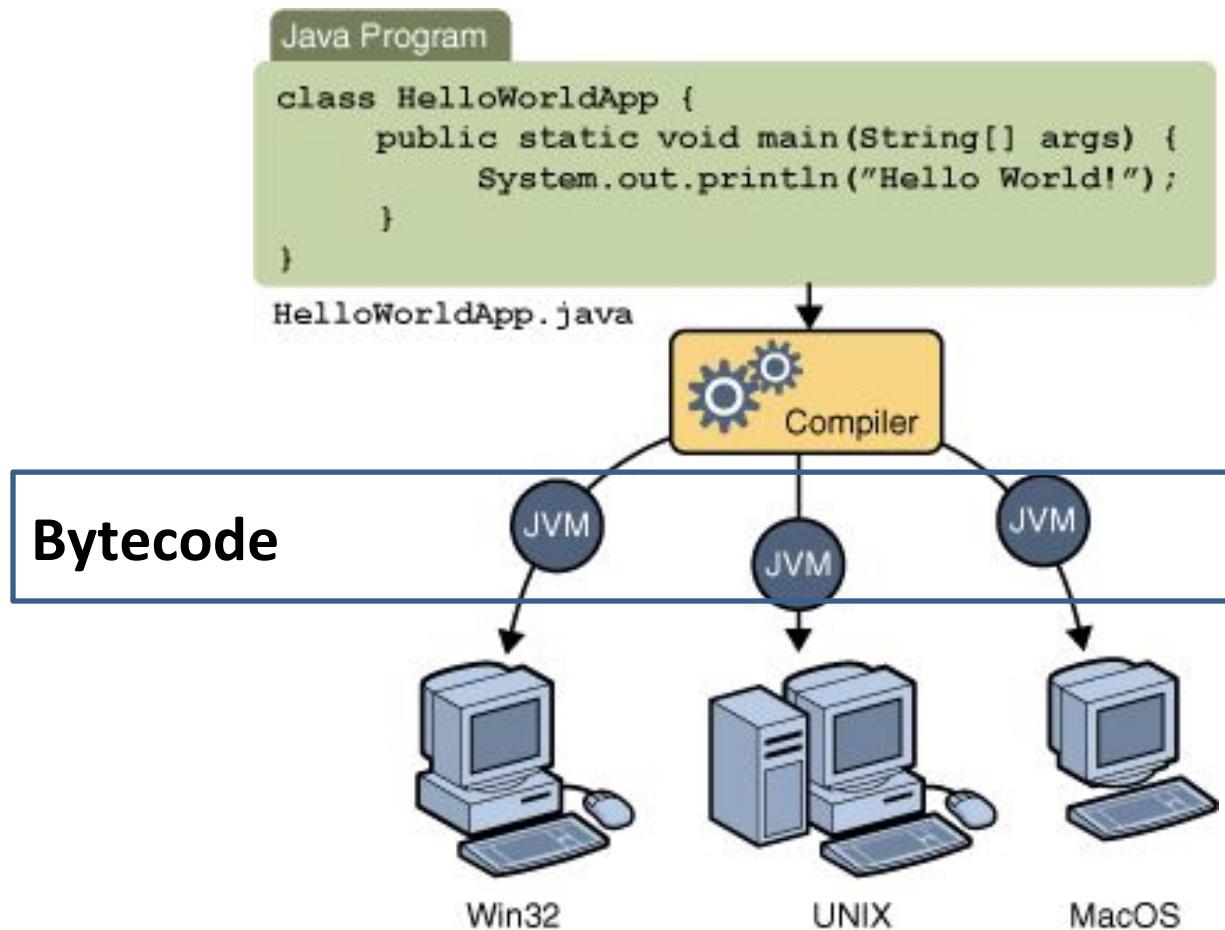


Primera diferencia: Lenguaje Compilado versus Interpretado

- Compilado: FORTRAN, Pascal, C, C++
- Interpretado: Python, Ruby
- Compilado es generalmente más rápido porque apuntan directamente a la máquina/arquitectura en la cual se ejecutan.
- Interpretado tiende a ser más portable.
- Versión de lenguajes “interpretados” es más fáciles de crear porque escribir compiladores es algo difícil.

¿Cómo funciona JAVA?

- Con la Java Virtual Machine



POO: Hello World en C++ y Java

===== C++ =====

```
#include <iostream>
int main() {
    //comentario
    std::cout << "Hello World!";
}
```

===== JAVA =====

```
public class HelloWorld { //comentario
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

C / C++ / Java

Manejo de memoria: C

```
1#include <stdlib.h> // needed for malloc and free!
2int *p_int = malloc(sizeof(*p_int));
3// use p_int
4free( p_int );
```

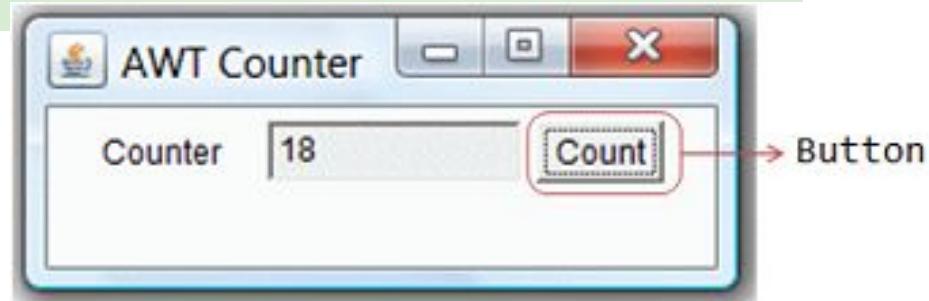
Manejo de memoria: C++

```
1int *p_int = new int;
2// use p_int
3delete p_int;
```

Manejo de memoria Java: Garbage Collector!

Java

```
import java.awt.Frame;  
// Using Frame class in package java.awt  
// A GUI program is written as a subclass of Frame - the top-level container  
// This subclass inherits all properties from Frame, e.g., title, icon, buttons, content-pane  
public class MyGUIProgram extends Frame {  
// Constructor to setup the GUI components  
public MyGUIProgram() {  
.....  
}  
  
// Other methods ..... ....  
// The entry main() method  
public static void main(String[] args) {  
// Invoke the constructor (to setup the GUI) by allocating an instance  
new MyGUIProgram();  
}  
}
```



SQL

- **Structured Query Language** orientado especialmente para DBMS relacionales
- Cuando la información está almacenada de forma estructurada, SQL es el estándar para consultas.

SQL – 3 ejemplos

Considerando estas tablas:

TABLA CUSTOMERS

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

TABLA ORDERS

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

SQL – EJEMPLO 1: SELECT

- SELECT [campos] FROM [tabla] WHERE
[condiciones]

```
SELECT CustomerID, ContactName  
FROM Orders  
WHERE Country = "Mexico"
```

Paradigma de programación en SQL

- ¿Le dijeron en algún momento a la instrucción de SQL cómo ir a buscar la información?
- NO -> **SQL es un tipo de lenguaje declarativo**
- ¿Qué otros paradigmas de lenguajes de programación hay?
- (y por que nos interesa saber...)

Paradigmas de Programación

- Lenguajes más comunes responden a varios paradigmas

Paradigm	Description	Main characteristics	Related paradigm(s)	Critics	Examples
Imperative	Computation as statements that directly change a program state (data fields)	Direct assignments , common data structures , global variables		Edsger W. Dijkstra , Michael A. Jackson	C , C++ , Java , PHP , Python
Structured	A style of imperative programming with more logical program structure	Structograms , indentation , either no, or limited use of, goto statements	Imperative		C , C++ , Java
Procedural	Derived from structured programming, based on the concept of modular programming or the procedure call	Local variables , sequence, selection, iteration , and modularization	Structured, imperative		C , C++ , Lisp , PHP , Python
Functional	Treats computation as the evaluation of mathematical functions avoiding state and mutable data	Lambda calculus , compositionality , formula , recursion , referential transparency , no side effects			Erlang , Haskell , Lisp , Clojure , Scala , F#

Por que sería útil forzar programación funcional? $rt(x) = rt(y)$ if $x = y$

```
globalValue = 0;

integer function rq(integer x)
begin
    globalValue = globalValue + 1;
    return x + globalValue;
end

integer function rt(integer x)
begin
    return x + 1;
end
```

Si escribiéramos un loop con la F(x)

- Un compilador podría detectar una función dentro de un loop y optimizar la forma de referenciarla y escribirla en código de máquina SOLAMENTE si la función no depende del estado de ejecución del programa

```
While (i < 1000)  
    rq(i)
```

Depende de variable global
definida en tiempo de ejecución

```
While (i < 1000)  
    rt(i)
```

Este sí cumple con
transparencia referencial

Otros lenguajes - LoLCode

- Inspirado en lolspeak
- Extension de archivo: .lol , .lols

The screenshot shows the official website for LOLCODE. At the top left, there's a grey diagonal banner with the text "Now with doge!". In the center is a circular logo featuring a stylized cat's face split vertically, with the left half white and the right half red. Below the logo, there are two columns of text. The left column reads: "LOLCODE is an esoteric programming language inspired by the funny things that cats say on the Internet." It includes a blue button labeled "Learn more about the language". The right column reads: "Ici is a correct, portable, fast, and precisely documented interpreter for LOLCODE written in C." It includes green and orange buttons labeled "Download source" and "GitHub project page". At the bottom, there's a link to a mailing list and a bug report page.

Now with doge!

LOLCODE is an esoteric programming language inspired by the funny things that cats say on the Internet.

Ici is a correct, portable, fast, and precisely documented interpreter for LOLCODE written in C.

[Learn more about the language](#)

[Download source](#)

[GitHub project page](#)

Problems? Check out the [mailing list](#) or file a [bug report](#).

LOLCODE

Hagamos un “Hola Mundo” en LOLCODE

LOLCODE 1

HAI 1.2

VISIBLE "Hai world"

KTHXBYE

LOLCODE 2

- Declarar e inicializar una variable
- Mostrarla en Pantalla

I HAS A VARIABLE ITZ <var>

LOLCODE 3

- Agregar comentarios

BTW

LOLCODE 4

- Solicitar al usuario input desde teclado

GIMMEH

LOLCODE 5

- Una bifurcación (IF)

..., O RLY?

YA RLY

...

NO WAI

...

OIC

LOLCODE 6

- CONTADOR

IM IN YR LOOP

...

IM OUTTA YR LOOP

...otros

- Switch... case

<expression>

WTF?

OMG <value literal>

 <code block>

[OMG <value literal>

 <code block> ...]

[OMGWTF

 <code block>]

OIC

Volviendo a la realidad

- La tarea 1 grande incluye
- Incluye programación en:
 - Python
 - HTML
 - CSS
 - Javascript

¿Cómo prepararme rápidamente?

- Venir a ayudantías
- CodeCademy: Basic Web Projects
 - <https://www.codecademy.com/en/tracks/projects>
- W3Schools: tutorial javascript
 - <https://www.w3schools.com/js/default.asp>
- Libros de referencia:
 - Guia paso a paso: [You Don't Know Javascript](#)
 - Tradicional: [Javascript The Definitive Guide](#)

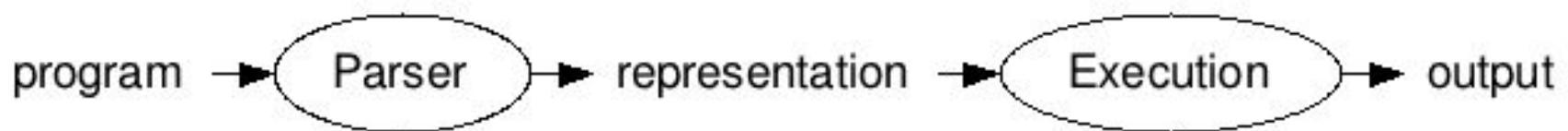
Gracias!

¿Pero cómo empezar?

- El mejor lugar: la consola del navegador que me permite ejecutar código en javascript.
- Click con el botón derecho sobre cualquier parte de la página y seleccionar “Inspect”
 - Hola Mundo
 - alert
 - Cambiar estilos

¿Y cómo escribo mi propio lenguaje?

- Necesitas un parser (sintaxis) y luego otro programa que implementa la ejecución



```
>> program = "(begin (define r 3) (* 3.141592653 (* r r)))"
```

```
>>> parse(program)
['begin', ['define', 'r', 3], ['*', 3.141592653, ['*', 'r', 'r']]]
```

```
>>> eval(parse(program))
28.274333877
```

code from (How to Write a (Lisp) Interpreter (in Python))
<http://norvig.com/lispy.html>

¿Y cómo escribo mi propio lenguaje?

- Necesitas un parser (sintaxis) y luego otro programa que implementa la ejecución
- Debes partir definiendo tu lenguaje: símbolos, reglas... y debe ser Context-free

$$G = (\{S\}, \{a, b\}, P, S)$$

$$S \rightarrow aSb$$

$$S \rightarrow ab$$

Equivalente a $\{a^n b^n : n \geq 1\}$

Si el tiempo da

- Ver ejemplo en

http://en.wikipedia.org/wiki/Context-free_grammar

Algebraic expressions [edit]

Here is a context-free grammar for syntactically correct [infix](#) algebraic expressions in the variables x , y and z :

1. $S \rightarrow x$
2. $S \rightarrow y$
3. $S \rightarrow z$
4. $S \rightarrow S + S$
5. $S \rightarrow S - S$
6. $S \rightarrow S * S$
7. $S \rightarrow S / S$
8. $S \rightarrow (S)$

This grammar can, for example, generate the string

$(x + y)^* x - z^* y / (x + x)$

as follows:

S (the start symbol)
 $\rightarrow S - S$ (by rule 5)
 $\rightarrow S^* S - S$ (by rule 6, applied to the leftmost S)
 $\rightarrow S^* S - S / S$ (by rule 7, applied to the rightmost S)

Paso de ejecución

Here is the definition of eval. Each of the nine cases in the table above has a line or two or three here, and the definition of eval needs nothing but those nine cases. The eval function takes two arguments: an expression, x, and an *environment*, env. An environment is a mapping from variable names to their values and will be covered in depth in the next section.

```
def eval(x, env=global_env):
    """Evaluate an expression in an environment."""
    if isa(x, Symbol):          # variable reference
        return env.find(x)[x]
    elif not isa(x, list):       # constant literal
        return x
    elif x[0] == 'quote':         # (quote exp)
        (_, exp) = x
        return exp
    elif x[0] == 'if':           # (if test conseq alt)
        (_, test, conseq, alt) = x
        return eval((conseq if eval(test, env) else alt), env)
    elif x[0] == 'set!':          # (set! var exp)
        (_, var, exp) = x
        env.find(var)[var] = eval(exp, env)
    elif x[0] == 'define':        # (define var exp)
        (_, var, exp) = x
        env[var] = eval(exp, env)
    elif x[0] == 'lambda':        # (lambda (var*) exp)
        (_, vars, exp) = x
        return lambda *args: eval(exp, Env(vars, args, env))
    elif x[0] == 'begin':         # (begin exp*)
        for exp in x[1:]:
            val = eval(exp, env)
        return val
    else:                        # (proc exp*)
```

¡Gracias!