Ruby Tutorial

...o como pasar un buen rato programando

Lección 1

- Introducción
- Instalación
- El Primer Programa
- Números en Ruby
- Strings y diversión
- Variables
- Alcance de las variables

Lección 2

- Introduciendo Datos
- Normas en los nombres

- Los métodos
- Los métodos: argumentos
- Rangos
- Arrays

Lección 3

- Bloques
- Más malabares con strings
- Expresiones Regulares
- Condicionales
- Bucles
- Números Aleatorios

Lección 4

- Clases y Objetos
- Accesores
- Ficheros: lectura/escritura
- Cargando librerías
- Herencia de clases
- Modificando clases
- Congelando objetos
- Serializando objetos

Lección 5

- Control de acceso
- Excepciones
- Módulos
- Constantes
- Hashes y Símbolos
- La clase Time

Lección 6

contacto

e-mail

Self

En cada instante de la ejecución del programa, hay uno y sólo un **`self`**: el objeto que se está usando en ese instante.

Contexto del nivel superior

El contexto del nivel superior se produce si no se ha entrado en otro contexto, por ejemplo, la definición de una clase. Por la tanto, el término "nivel superior" se refiere al código escrito fuera de las clases o módulos. Si abres un fichero de texto y escribes:

```
x = 1
```

habrás creado una variable local en el nivel superior. Si escribes:

```
def m
end
```

habrás creado un método en el nivel superior: un método que no es definido como un método de una clase o módulo. Si nada más arrancar el intérprete, tecleas:

```
puts self
```

La respuesta es **`main`**, un término que se refiere al objeto que se crea al iniciar el intérprete.

self dentro de clases y módulos

En una clase o definición de módulo, `self` es la clase o el módulo al que pertenece el objeto:

```
class S
  puts 'Comenzó la clase S'
  puts self
  module M
    puts 'Módulo anidado S::M'
    puts self
  end
  puts 'De regreso en el nivel más superficial de S'
  puts self
end
```

La salida es:

```
Comenzó la clase S
S
Módulo anidado S::M
S::M
De regreso en el nivel más superficial de S
S
```

self dentro de los métodos

```
class S
  def m
    puts 'Clase S, metodo m:'
```

```
    puts self # <S:0x2835908>
  end
end
s = S.new
s.m
```


Other interesting sites

Oppimateriaali

Käsikirjoittamisen käsitteitä ja tehtäviä

The Gamer Dome

Community Game Design Projects

Chavez Brain Trust