

Introducción a Julia

El “Ju” en Jupyter



Presentación

- ¡Bienvenidos y muchas gracias por su interés en este taller!
- Taller **introductorio** a

Sobre mí



Sobre mí



Sobre mí



Centro de Enseñanza
de Lenguas Extranjeras



UNAM

ENALLT

Escuela Nacional de Lenguas,
Lingüística y Traducción



Sobre ustedes

- ¿Por qué les interesó este taller?
- ¿Qué esperan del taller?
- ¿Qué lenguajes conocen?
- Lenguajes que usan para cómputo científico

Objetivo general

- Presentar una introducción completa a Julia
- ¿Qué hace a Julia único?
- Ejemplos de aplicaciones en análisis y presentación de datos, procesamiento de texto, aprendizaje automático y profundo

Objetivos específicos

- Instalar Julia en su computadora
- Instalar paquetes de Julia
- Trabajar en ambientes independientes

Objetivos específicos

- Elaborar sus propios *notebooks*
- Escribir sus propios programas
- Ejecutar Python desde Julia
- Continuar su aprendizaje de forma independiente

Actividades

Horario	Lunes 30/11/2020	Martes 01/12/2020	Miércoles 02/12/2020	Jueves 03/12/2020
10:00 - 11:00	Presentación e Instalación	Julia con "pilas incluidas"	Python desde Julia / Manejo de datos	Procesamiento de texto
11:00 - 12:00	Presentación del lenguaje (1)	Manejo de paquetes	Visualización (1)	Aprendizaje automático
12:00 - 13:00	Presentación del lenguaje (2)	Ambientes de desarrollo	Visualización (2) / Precompilado de paquetes	Aprendizaje profundo

Plataforma

Introducción a Julia: El "Ju" en Jupyter

Código de la clase 56soyb3 []

56soyb3

Introducción a Julia: El "Ju" en Jup...



Copiar enlace de
invitación



Instalación

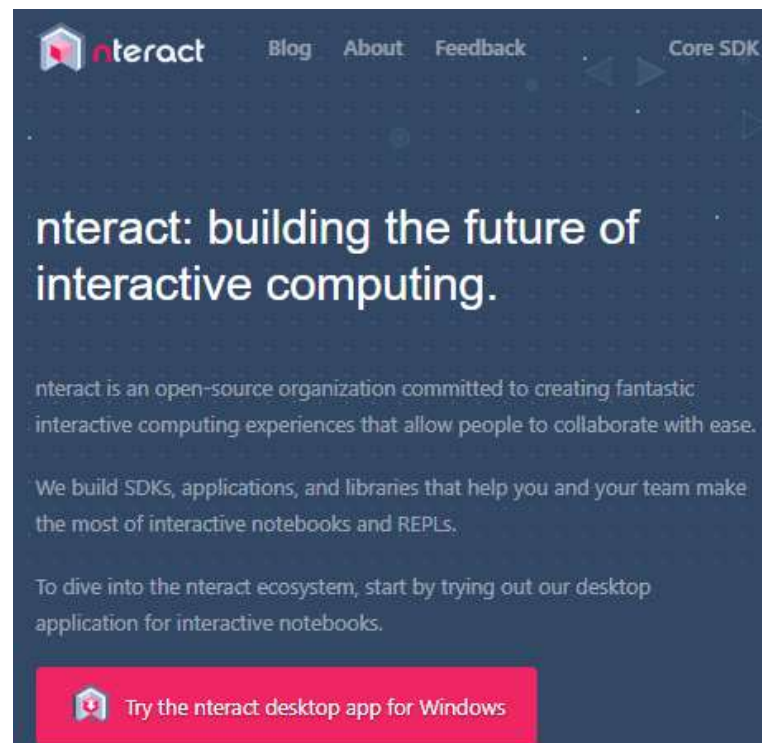
Julia

[Download](#)[Documentation](#)[Blog](#)[Community](#)[Learn](#)[Research](#)[JSoC](#)[♥ Sponsor](#)

The Julia Programming Language

[Download v1.5.3](#)[Documentation](#)[★ Star](#) 30,792

ninteract



Visual Studio Code

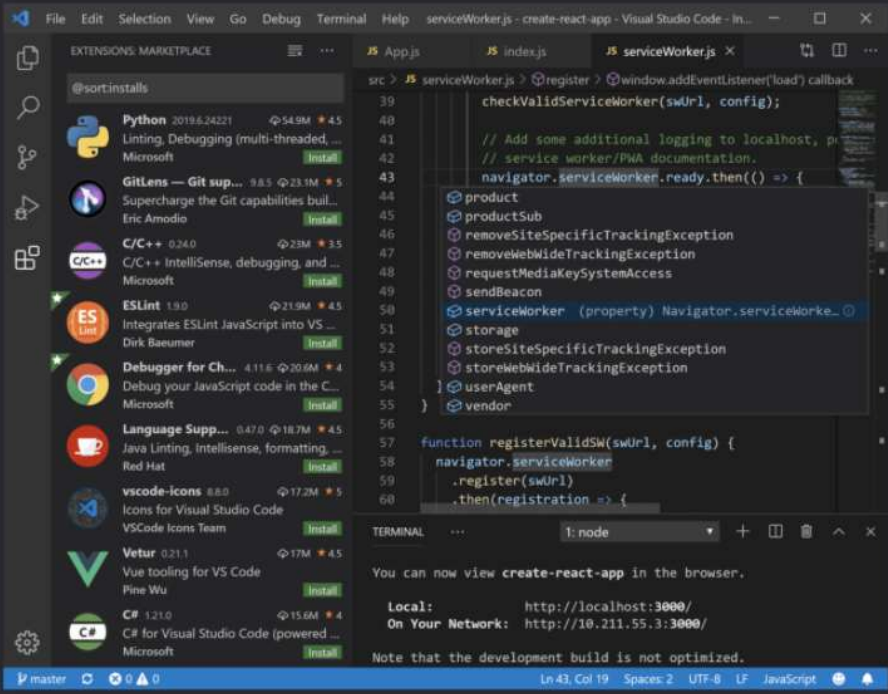
Code editing. Redefined.

Free. Built on open source. Runs everywhere.

[Download for Windows](#)
Stable Build

[Other platforms and Insiders Edition](#)

By using VS Code, you agree to its [license](#) and [privacy statement](#).



The screenshot displays the Visual Studio Code interface. On the left, the 'EXTENSIONS: MARKETPLACE' sidebar lists various extensions such as Python, GitLens, C/C++, ESLint, Debugger for Chrome, Language Support for Java, vscode-icons, Vetur, and C#. The main editor area shows a JavaScript file named 'serviceWorker.js' with code for registering a service worker. The terminal at the bottom displays the output of a command, indicating that the application is running on localhost:3000.

```
src > .\serviceWorker.js > register > window.addEventListener('load') callback
39
40
41 checkValidServiceWorker(swUrl, config);
42
43 // Add some additional logging to localhost, pointing
44 // to the service worker/PWA documentation.
45 navigator.serviceWorker.ready.then(() => {
46
47   product
48   productSub
49   removeSiteSpecificTrackingException
50   removeWebWideTrackingException
51   requestMediaKeySystemAccess
52   sendBeacon
53   serviceWorker (property) Navigator.serviceWorker
54   storage
55   storeSiteSpecificTrackingException
56   storeWebWideTrackingException
57   userAgent
58   vendor
59
60 function registerValidSW(swUrl, config) {
61   navigator.serviceWorker
62     .register(swUrl)
63     .then(registration => {
```

TERMINAL ... 1: node

You can now view create-react-app in the browser.

Local: http://localhost:3000/
On Your Network: http://10.211.55.3:3000/

Note that the development build is not optimized.

Introducción

Historia



Fernando Pérez
@fperez_org

2001

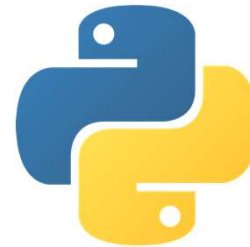
IP[y]:

IPython

Historia

IP[y]:
IPython

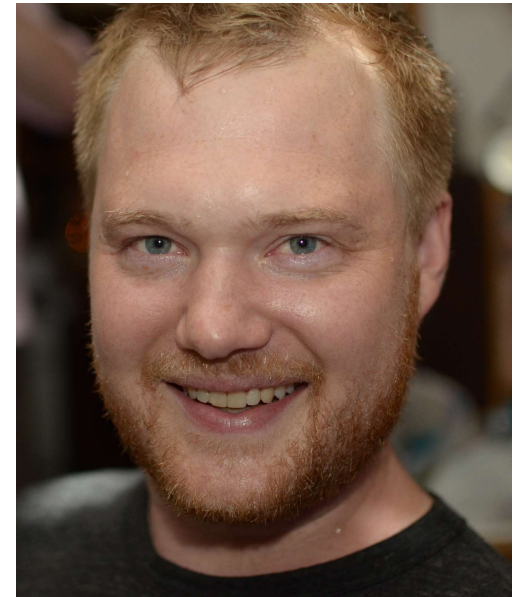
julia



Julia



Jeff Bezanson
@JeffBezanson



Stefan Karpinski
@StefanKarpinski

Julia



Viral B. Shah
@Viral_B_Shah



Alan Edelman

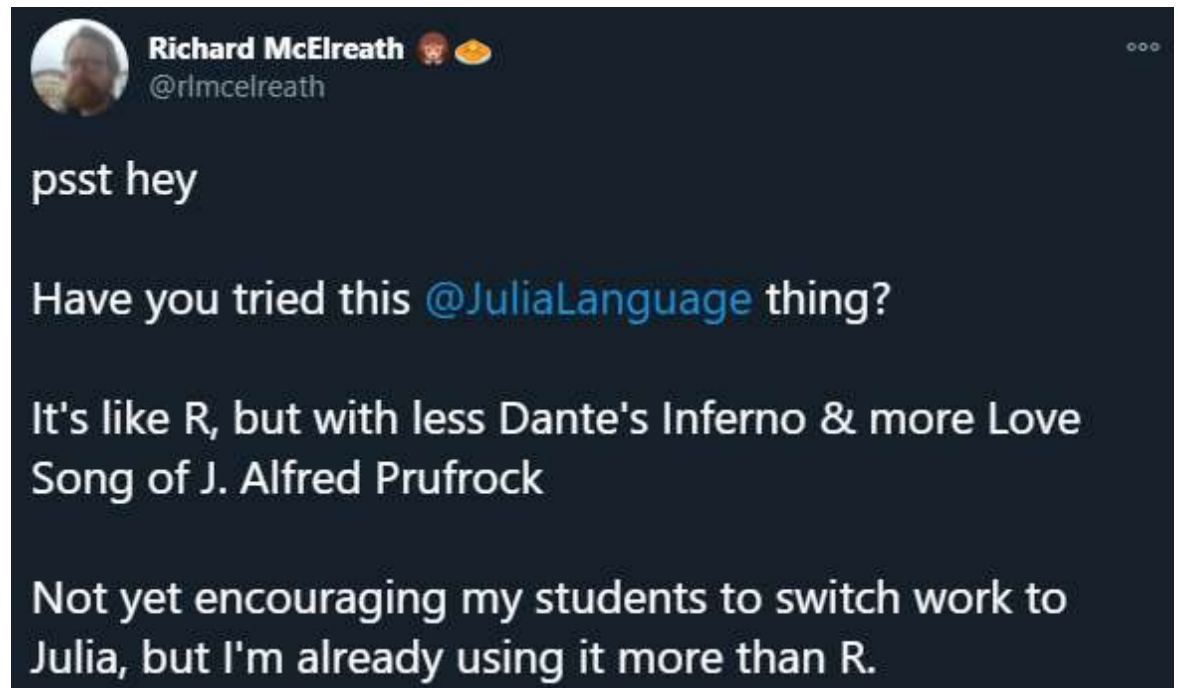
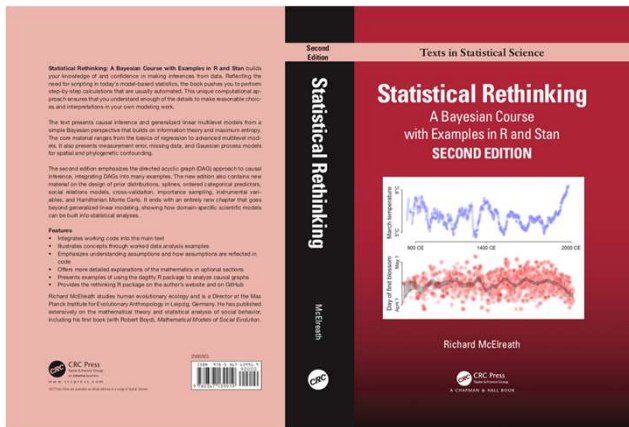
¿Por qué Julia?

- Dinámico
- Expresivo
- Rápido
- Código abierto, multiplataforma...

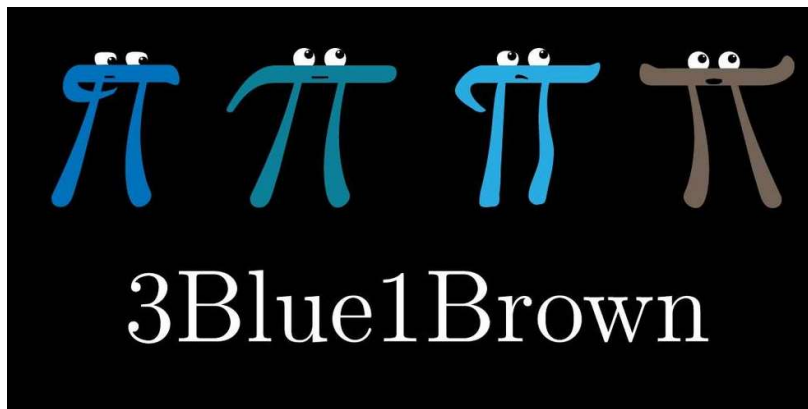
¿Quién lo usa?



¿Quién lo usa?



¿Quién lo usa?



¿Quién lo usa?



Demo

- Python vs Julia

```
In [1]: from math import pi
```

```
In [2]: def estimate_pi(n):  
...:     s = 1.0  
...:     for i in range(1, n+1):  
...:         s += (-1 if i % 2 else 1) / (2*i + 1)  
...:     return 4*s  
...:
```

```
In [3]: %time print(f"The error estimate is {estimate_pi(100_000_000) - pi}")  
The error estimate is 9.999532757376528e-09  
Wall time: 27.6 s
```


Demo

- Python vs Julia

```
In [1]: from math import pi

In [2]: def estimate_pi(n):
...:     s = 1.0
...:     for i in range(1, n+1):
...:         s += (-1 if i % 2 else 1) / (2*i + 1)
...:     return 4*s
...:

In [3]: %time print(f"The error estimate is {estimate_pi(100_000_000) - pi}")
The error estimate is 9.999532757376528e-09
Wall time: 27.6 s
```

```
julia> function estimate_pi(n)
    s=1.0
    for i in 1:n
        s += (isodd(i) ? -1 : 1) / (2i + 1)
    end
    return 4s
end
estimate_pi (generic function with 1 method)

julia> @time print("The error estimate is $(estimate_pi(100_000_000) - π)")
The error estimate is 9.999532757376528e-9  0.223183 seconds (8 allocations: 688 bytes)
```

Demo

- Python vs Julia

```
In [1]: from math import pi

In [2]: def estimate_pi(n):
...:     s = 1.0
...:     for i in range(1, n+1):
...:         s += (-1 if i % 2 else 1) / (2*i + 1)
...:     return 4*s
...:

In [3]: %time print(f"The error estimate is {estimate_pi(100_000_000) - pi}")
The error estimate is 9.999532757376528e-09
Wall time: 27.6 s
```

```
julia> function estimate_pi(n)
    s=1.0
    for i in 1:n
        s += (isodd(i) ? -1 : 1) / (2i + 1)
    end
    return 4s
end
estimate_pi (generic function with 1 method)

julia> @time print("The error estimate is $(estimate_pi(100_000_000) - π)")
The error estimate is 9.999532757376528e-9 0.223183 seconds (8 allocations: 688 bytes)
```