

# JavaScript for Science

Armenuhi.Abramyan@cern.ch

Inverted CERN School of Computing (16-18 March, 2020)

# Content

- Popularity of JavaScript
- Evolution of Node.js
- Rich ecosystem of JavaScript libraries
- JavaScript libraries for scientific tasks
- Performance of JavaScript

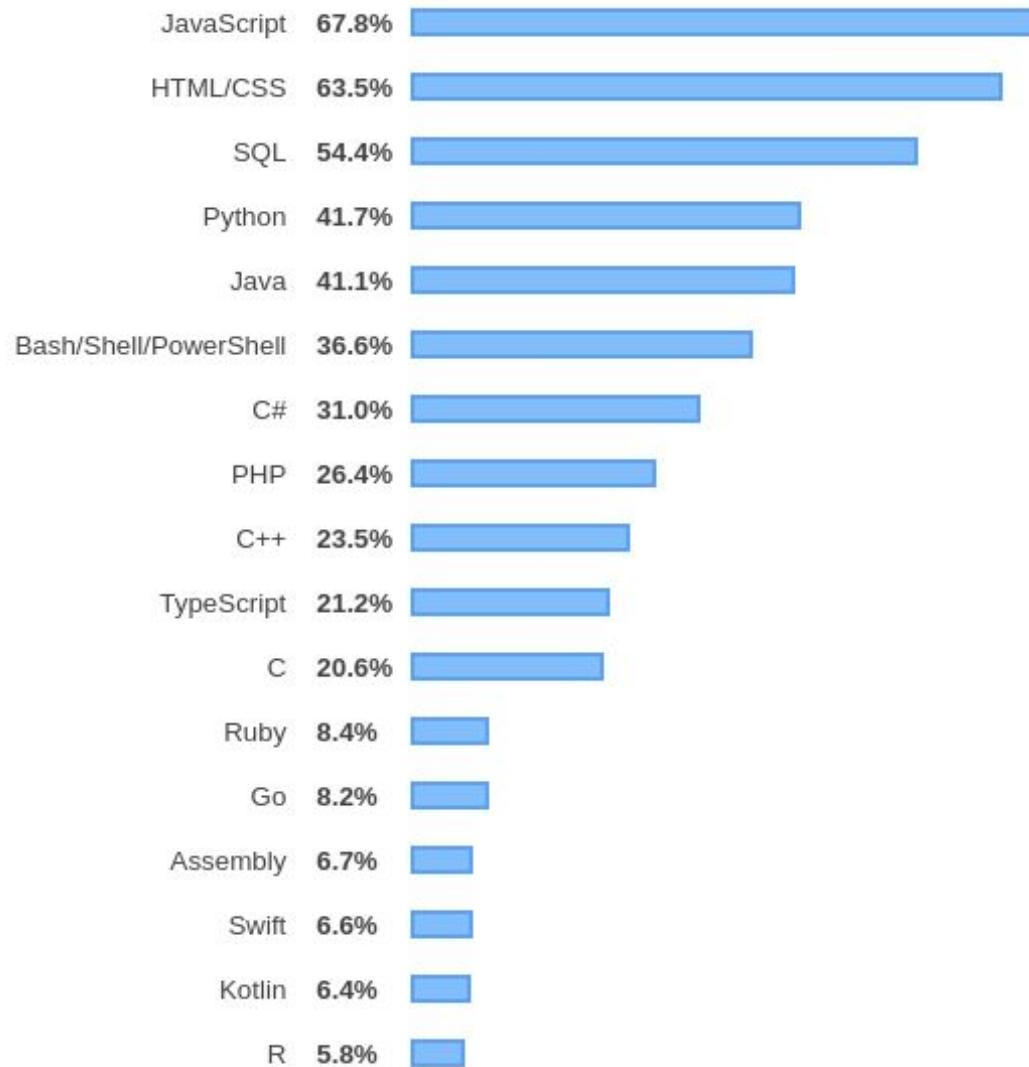
\* Most of the framed images are linked to their relevant content

# JavaScript, the native language of the Web

JavaScript (JS) is a high-level, object oriented, interpreted programming language.

- The **JS** was created by **Brendan Eich** in 1995 at **Netscape** as a scripting tool to manipulate web pages inside [Netscape Navigator](#) browser.
- Initially **JS** had another name: “**LiveScript**”. But **Java** was very popular at that time, so it was decided that positioning a new language as a “younger brother” of **Java** would help to make it noticeable.

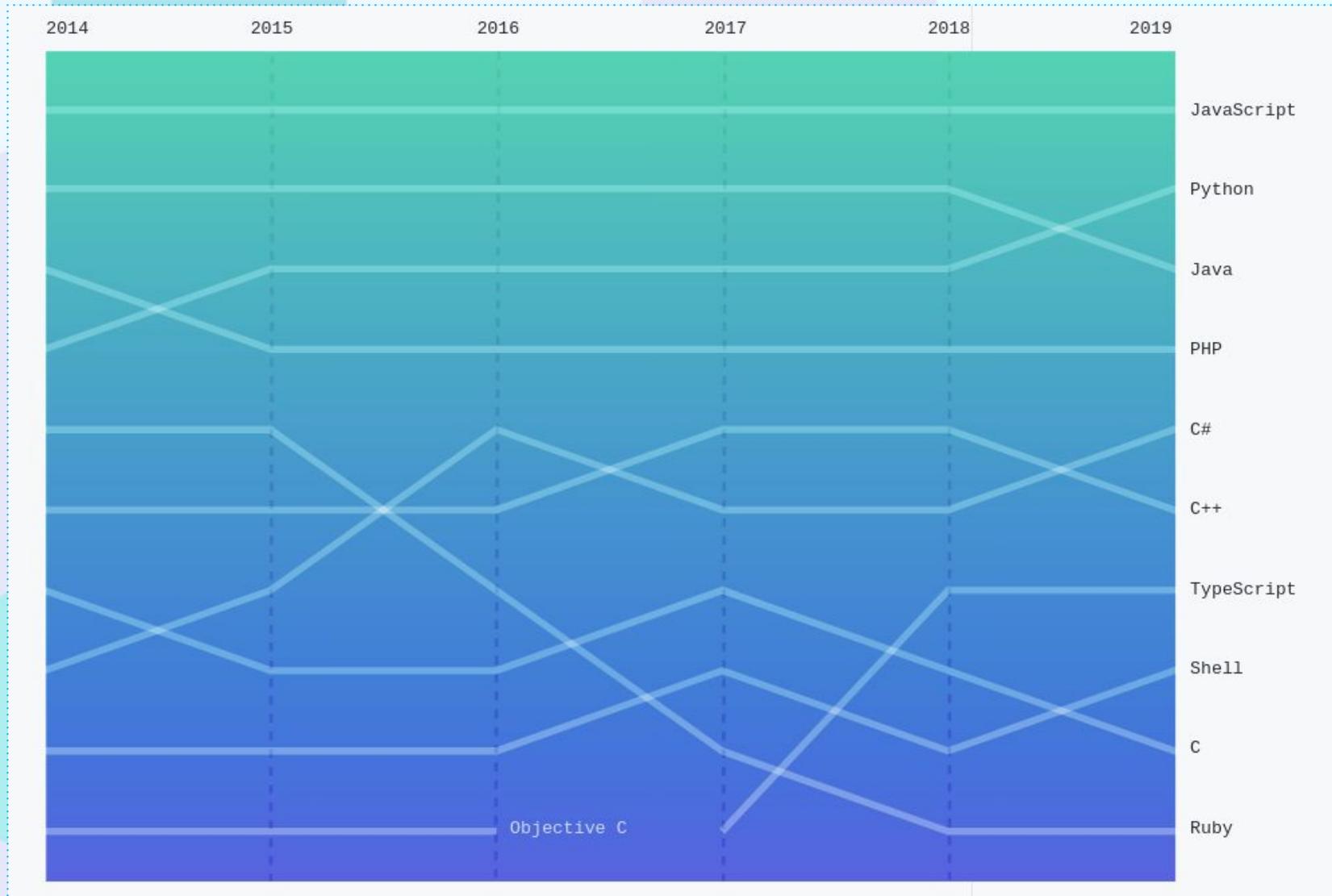
# Popular programming languages by StackOverflow



StackOverflow developers survey (2013-2019)

# Top programming languages by Github

Every year, GitHub releases the [Octoverse report](#), ranking the top technologies favored by its community of millions of software developers.

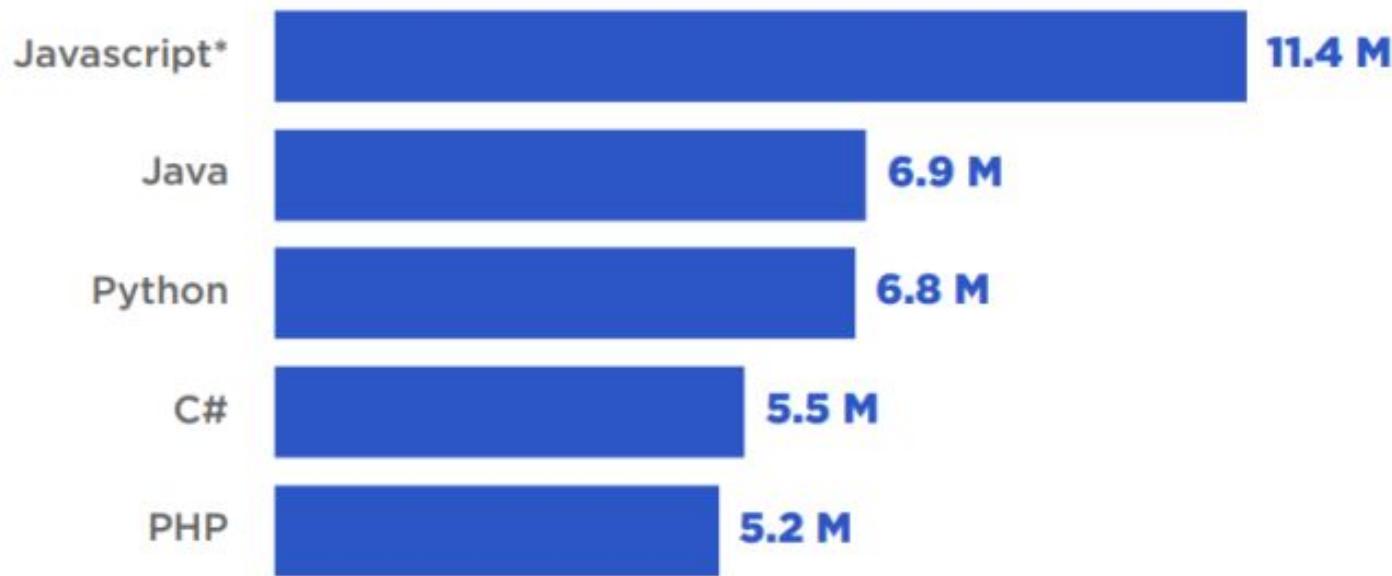


# Top 5 Programming languages by SlashData

Developer analyst and research company *SlashData*, which helps top 100 technology firms understand their developer audiences, surveyed **17000+** developers from **155** countries and revealed the following report:

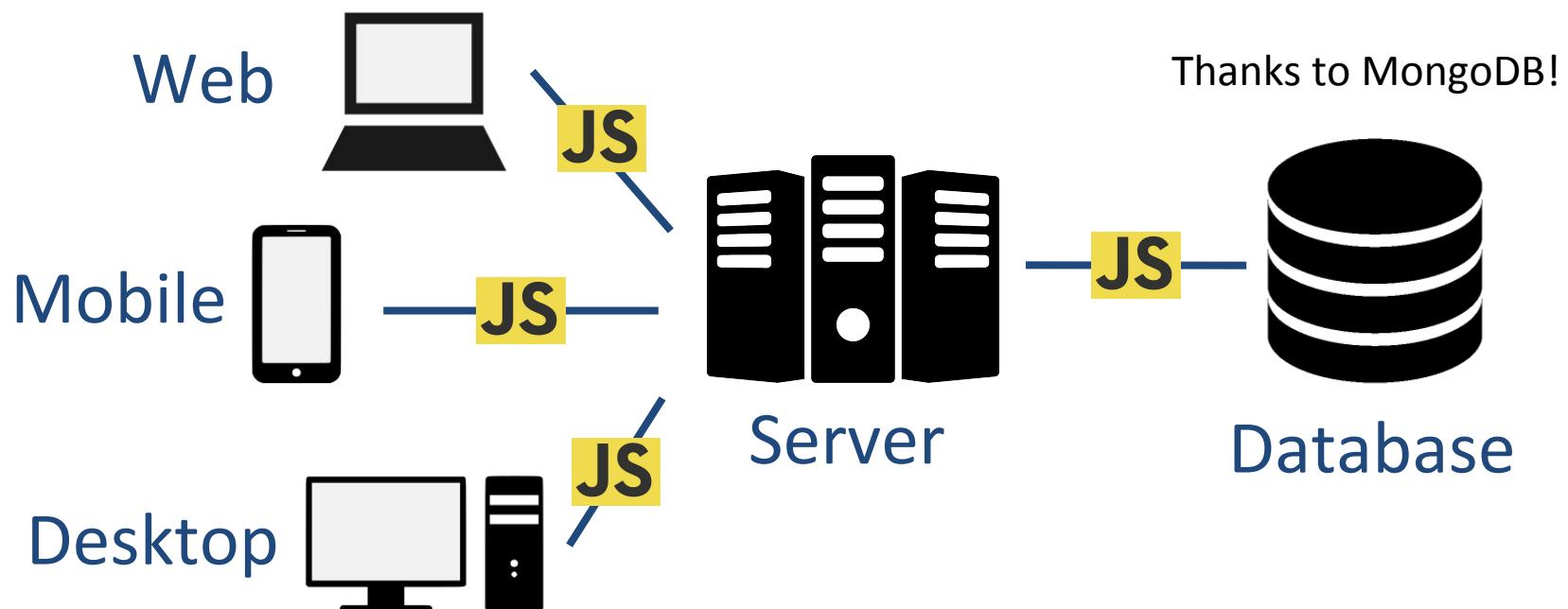
## Programming language communities Q2 2019

Active software developers in millions (n=11,292)



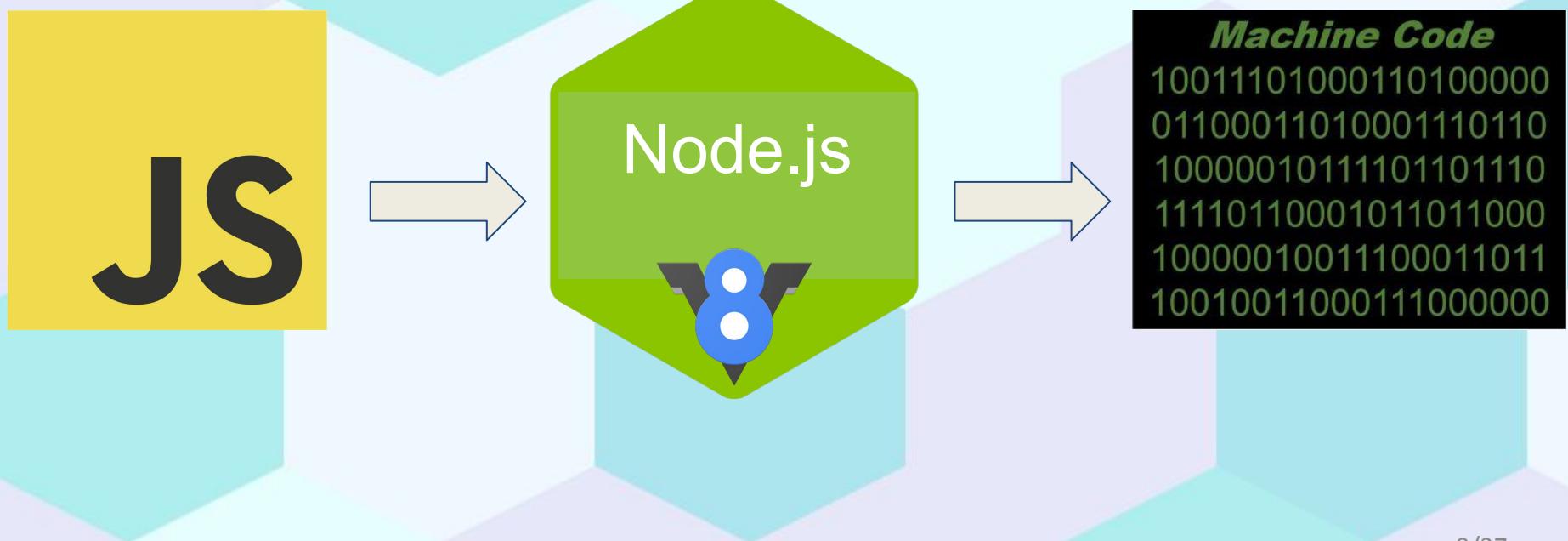
# Why JS became so popular?

- **JS** is the language of the Web.
- The introduction of **Node.js** allowed **JS** to go beyond the browser.
- **JS** can serve as a universal language to build cross-platform isomorphic software systems, which makes the language very cost-efficient.



# What is Node.js

- An open-source, cross-platform, **JS** runtime environment written in C++
- Uses V8 **JS** engine to execute **JS** code outside of a browser, i.e. server-side
- Implements an event-driven, non-blocking I/O model that makes it perfect for data-intensive, real-time applications.



# Most famous JS engines

***JS engines*** are programs that convert ***JS*** code into lower level or machine code and execute them.



Chakra



v8



SpiderMonkey



JSCore

For the full list refer to [List of ECMAScript engines](#) wiki page

# Evolution of Node.js

In 2009, by the initiative of Ryan Dahl, the Node.js was born.

**MongoDB** (JSON based NoSQL database)

Feb 2009

Nov 2009

The first presentation of Node.js by its creator Ryan Dahl at **JSCConf 2009**

**NPM**

Jan 2010

July 2010

Ryan Dahl's **GoogleTechTalks** about JS on the Server.

**ExpressJS, Mongoose, AngularJS (MEAN)**

Late 2010 and early 2011

Node.js support for **Windows**

July 2011

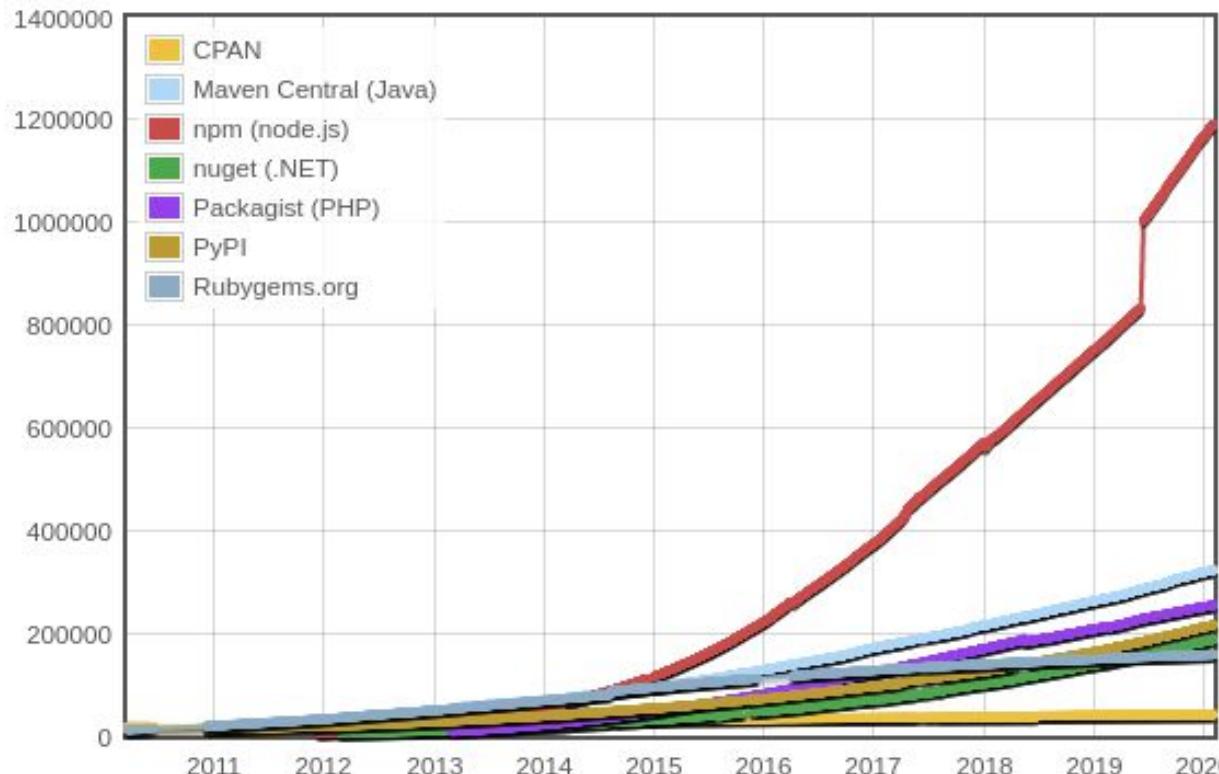
June 2015

Node.js Foundation

# Open-source ecosystem of Node.js

Node.js currently has the largest in the world ecosystem of open-source software packages.

## Module Counts



Over **million** libraries available in npm, and over **10,000** new ones being published weekly.

Number of packages over time

(Data is collected by scraping the relevant websites)

# How to use JS libraries/packages?

- Loading or downloading the library from a CDN
- Installing the library with a package manager

A Content Delivery Network (**CDN**) consists of distributed servers that deliver pages and other static content (e.g. images, CSS, JS) to a user, based on their geographic location.



# Top 3 open source CDN providers



One of the most famous public CDN services used by **9.2%** of all worldwide websites.



Open Source CDN, which can serve web files directly from the npm and GitHub.



Fast, global content delivery network for everything on npm

# Usage of libraries via CDNs

```
<html>
  <head>
    <title>Awesome web app</title>
    <meta tags...>
    <link rel="stylesheet" type="text/css" href="css/style.css">
    <link rel="stylesheet" href="//cdn.url/path/to/css/lib/style.css">
    <script src="//cdn.url/path/to/js/lib/script.min.js"></script>
    <script src="js/script.js" type="text/JS"></script>
  </head>
  <body>
    ...
    <script src="//cdn.url/path/to/js/lib/otherScript.js"></script>
  </body>
</html>
```

# Top 3 Node package managers



Node(JS) Package Manager with the richest open source libraries in the world.



Yarn is developed by Facebook as a fast and secure alternative of npm, although relying on npm registry.



Fast, disk space efficient package manager.

# Usage of libraries by installing with a package manager

## Installation of Node.js is a prerequisite

1. Navigate to [nodejs.org](https://nodejs.org) website
2. Download and install the Node.js binary on your machine.

## Open a terminal and install the package with npm

```
$> npm install package_name
```

## Include the package in the top of your app code

```
require("package_name") or import "package_name"
```

# How to choose the "right" JavaScript package?



Git repo with over 44M repositories across the globe.  
*(Presentation, Contributors, Sponsors, Maintenance)*



Compare packages, spot trends, pick the winner!  
*(Weekly downloads, stars, issues, last activity, age, size)*



Lookup the npm packages you want to compare  
*(Dependencies, Licenses)*



Largest online Q&A platform for developers.  
*(Documentation)*



Check for known **vulnerabilities** in public GitHub repos, npm packages and Docker images

Categorized list of JS packages.  
*(Popularity in its category)*

# How to choose the "right" JavaScript package!

- Popularity
- Age and maturity
- Community
- Number of plugins
- Maintenance
- Size
- Dependencies
- Vulnerabilities
- Versioning policy
- Licences
- Comparison with similar packages

# JS for Science?



# Biggest (past) limitations of JS

- Only for the Web
- Single-threaded - performs only one computation at a time
- Good precision costs additional computation
- Poor file IO support
- Depends a lot on the browser.
- JS DOM (Document Object Model) is slow.
- Source code and other static resources are visible to anyone

# How can we rely on JS when...?

$(1 < 2 < 3)$  condition returns **true**

$(3 > 2 > 1)$  condition returns **false**

Because  $(3 > 2 == \text{true})$  and  $(\text{true} > 1)$  is **false**

$(0.1 + 0.2 == 0.3)$  condition returns **false**

Because  $(0.1 + 0.2 == 0.3000000000000004)$

**NaN === NaN** condition returns **false**

Because: I really don't know!

The biggest (past) gaps of JS are covered by its ecosystem and the community of developers and enthusiasts.



JS is now for Science too!

# Node.js kickstarters

## HACKATHON STARTER

A kickstarter for Node.js web applications

Build Node.js projects simple and fast during Hackathons or Tech meetups.



## RealWorld

Pick the technology and build your project on the base of Real World examples.



## YEOMAN

Generate the code for various parts of your project on the base of best practices.

# Web, Mobile and Desktop apps

Web



Mobile



Desktop



ELECTRON



nw.js

# Data visualization and charting



**D3 (*Data-Driven Documents*)** is the most famous open-source JS library for interactive data visualizations in the browser.



A powerful, rich and easy configurable JS charting library, used by over **80%** of the largest companies in the world.



Generates diagrams, charts, graphs or flows from markdown-like text.



Automated tool to build dynamic data visualization and analytics dashboards.



A dynamic, easy to use, interactive, browser based visualization library, to handle large amounts of dynamic data.



Provides essential tools and charts for your UI.

# 3D modeling



three.js

A cross-browser JS library and API used to create and display animated 3D computer graphics in a web browser.



An ease-to-use library to create world-class 3D globes and maps with the best possible performance, precision, visual quality.



For creative coding, with a focus on making coding accessible and inclusive for artists, designers, educators, beginners.



Used to build interactive animations, product configurators, engaging presentations of any kind.

# Math

**math.js**

An extensive math library for JavaScript and Node.js.

**bignumber.js**

A library for arbitrary-precision decimal and non-decimal arithmetic.

**Numbers.js**

An advanced mathematics JS library. Supports: Calculus, Matrix operations, Prime numbers, Statistics.

**KATEX**

Provides fast TeX math rendering on the web.

**MathJax**

A cross-browser library that displays math notations and uses markup like LaTeX, ASCIIMathML, and MathML.

# Machine learning



A WebGL-accelerated, browser-based JS library for training and deploying ML models.

Used to train Neural Networks entirely in the browser.

To build advanced statistical models and machine learning libraries.

Brain.js is a GPU accelerated JS library of Neural Networks for Browsers and Node.js. It is simple, fast and easy to use.

WebDNN is an open source software framework for executing deep neural network pre-trained model on web browser.

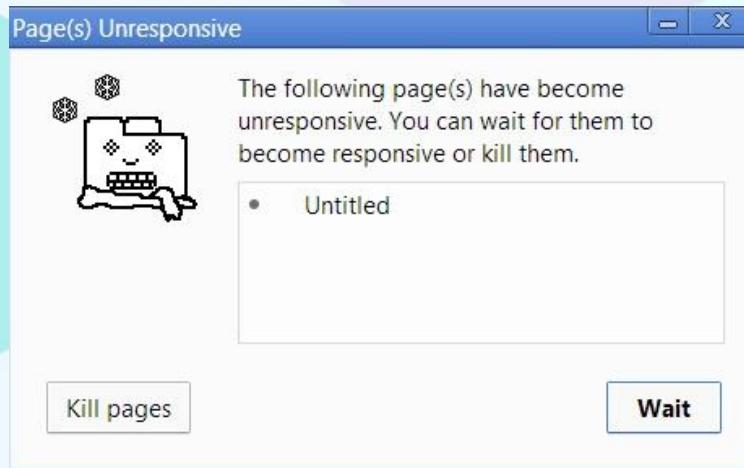
# High performance computations?



**JS** provides High performance by its non-blocking behavior and by the technologies allowing to run JS on multiple **threads**, **CPUs** and **GPUs** as in the client's browser as well as on the server.

# Multithreading in the browser with Web Workers

When executing scripts in the web page, all other operations (DOM manipulation, rendering, etc) become unresponsive until the script is finished.

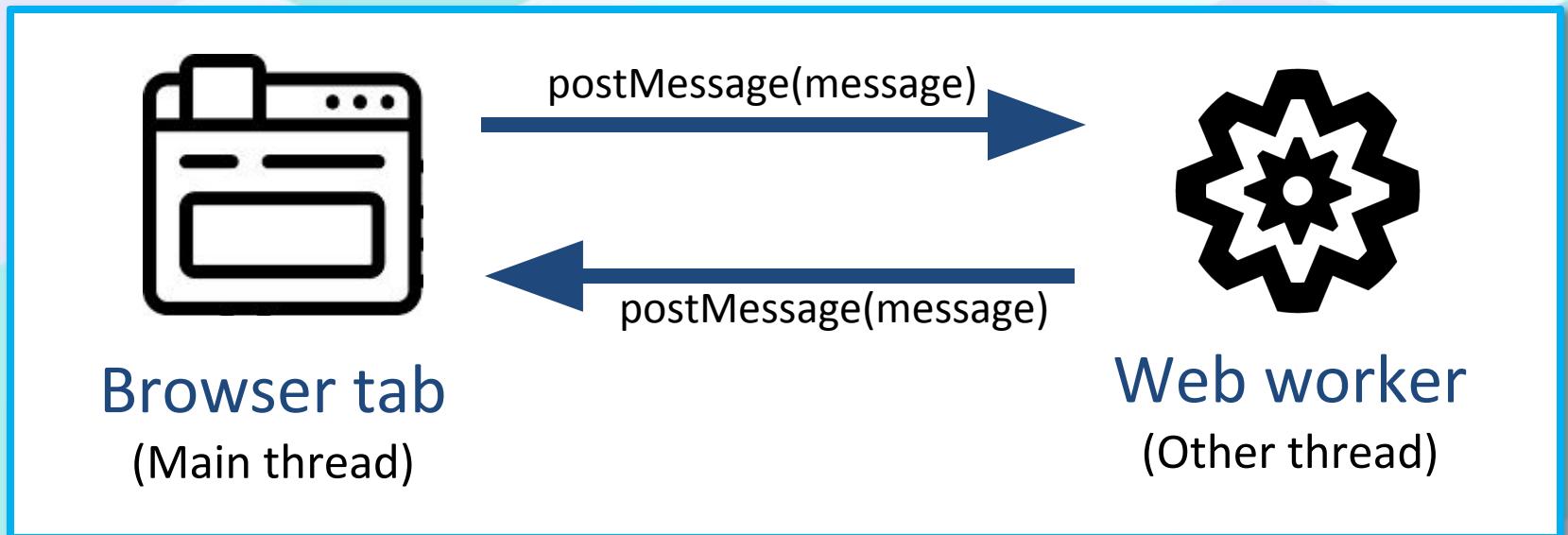


This was the case until **HTML5** introduced a [Web Workers](#) API, collaborative effort of **Mozilla** and **Google** to make their browsers more powerful.

**Now, most of the major browsers support [Web Workers](#).**

# What is Web Worker?

- Web Worker allows to run **JS** in background threads, in parallel with main thread.
- Worker thread can perform tasks without interfering with the user interface.
- Main thread can spawn an “**unlimited**” number of worker threads.
- On the browser tab close the main thread “**dies**” and “**kills**” all its worker threads.
- A worker can send (post) messages to its parent thread (and vice versa).



# A web page without a Web Worker

```
<html>
  <head>
    <title>JS computation without a Web Worker</title>
  </head>
  <body>
    <div rel="result">Loading...</div>
    <script>
      let finalMessage = '';
      function performOperations(nbOperations) {
        let start = Date.now(); // milliseconds
        let i = 1;
        while (i <= nbOperations) i++;
        finalMessage = 'Took: ' + (Date.now() - start) / 1000 + ' second';
        document.getElementById("result").innerHTML = finalMessage;
      }
      performOperations(1000000000); // For me it took ~12 seconds
    </script>
  </body>
</html>
```

index.html

# A web page with a Web Worker (1)

Structure:

```
✓ └── workers
      └── computation.js
      └── index.html
```

```
let finalMessage = '';

function performOperations(nbOperations) {
    var start = Date.now(); // milliseconds
    let i = 1;

    while (i <= nbOperations) i++;

    finalMessage = 'Took: ' + (Date.now() - start) / 1000 + ' seconds';
}

performOperations(10000000000);
self.postMessage(finalMessage);
```

./workers/computation.js

# A web page with a Web Worker (2)

```
<html>
  <head>
    <title>JS computation with a Web Worker</title>
  </head>
  <body>
    <div id="result">Loading...</div>
    <script>
      if(window.Worker) {
        let myWorker = new Worker('./workers/computation.js');
        myWorker.addEventListener('message', function (event) {
          document.getElementById('result').innerHTML = event.data;
        });
        myWorker.addEventListener('error', function (event) {
          console.error(event);
        });
      }
    </script>
  </body>
</html>
```

index.html

# Performance references

- [Simple web worker](#) - Simple usage example of Web Workers
- [Web Workers Examples](#) - More advanced examples of Web workers
- [Comlink](#) - JS Library which makes usage of Web Workers enjoyable :)
- [Worker Threads](#) - enables the use of threads in Node.js
- [Cluster](#) - contains a set of functions and properties that help to fork processes to take advantage of multi-core systems.
- [GPU.js](#) - JS Acceleration library, which automatically transpiles simple JS functions into shader language and compiles them so they run on GPU.

# Summary

- **JS** is at the top of the programming languages and its popularity is growing.
- **JS** currently has the largest in the world ecosystem of open-source libraries and the vast community of developers.
- **JS** is an universal language for **Web, Mobiles, Desktops, Servers, and Databases**.
- Being a High-level language, **JS** can be used for various scientific tasks.
- Most of the limitations of **JS** are covered by its libraries.
- **JS** can be run on multiple **threads, CPUs** and **GPUs** as in the client's browser as well as on the server.



# THANK YOU!

aabramya@{cern.ch, gmail.com}