

Current state of geoprocessing development on the web applications:

NODE.js, R in the server , ArcGIS server and Geoserver



cooperación
española



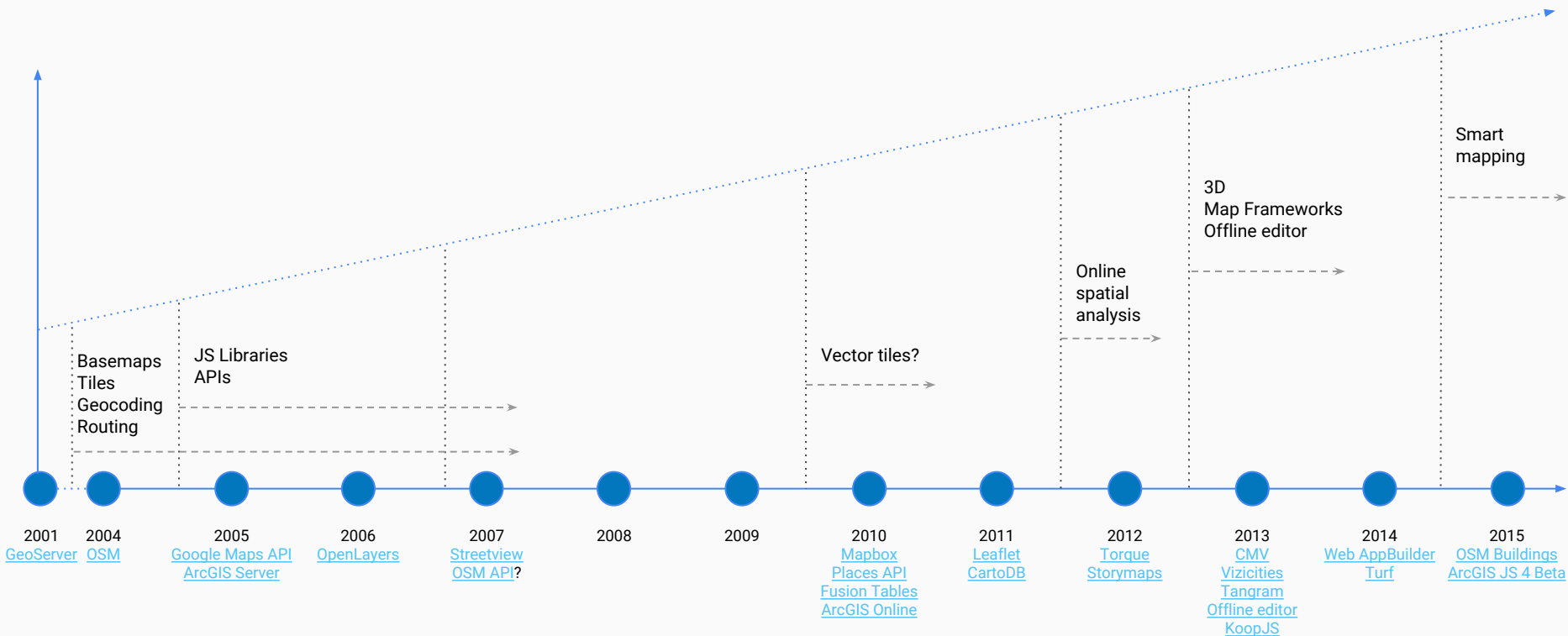
United Nations
Educational, Scientific and
Cultural Organization



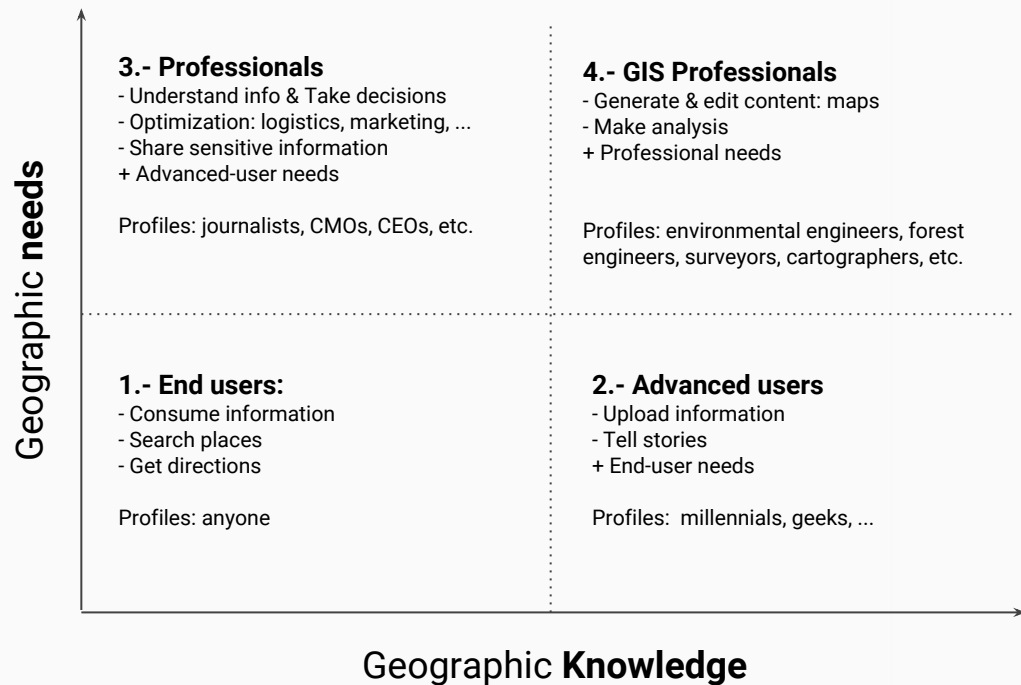
Intergovernmental
Oceanographic
Commission



Web mapping: The history



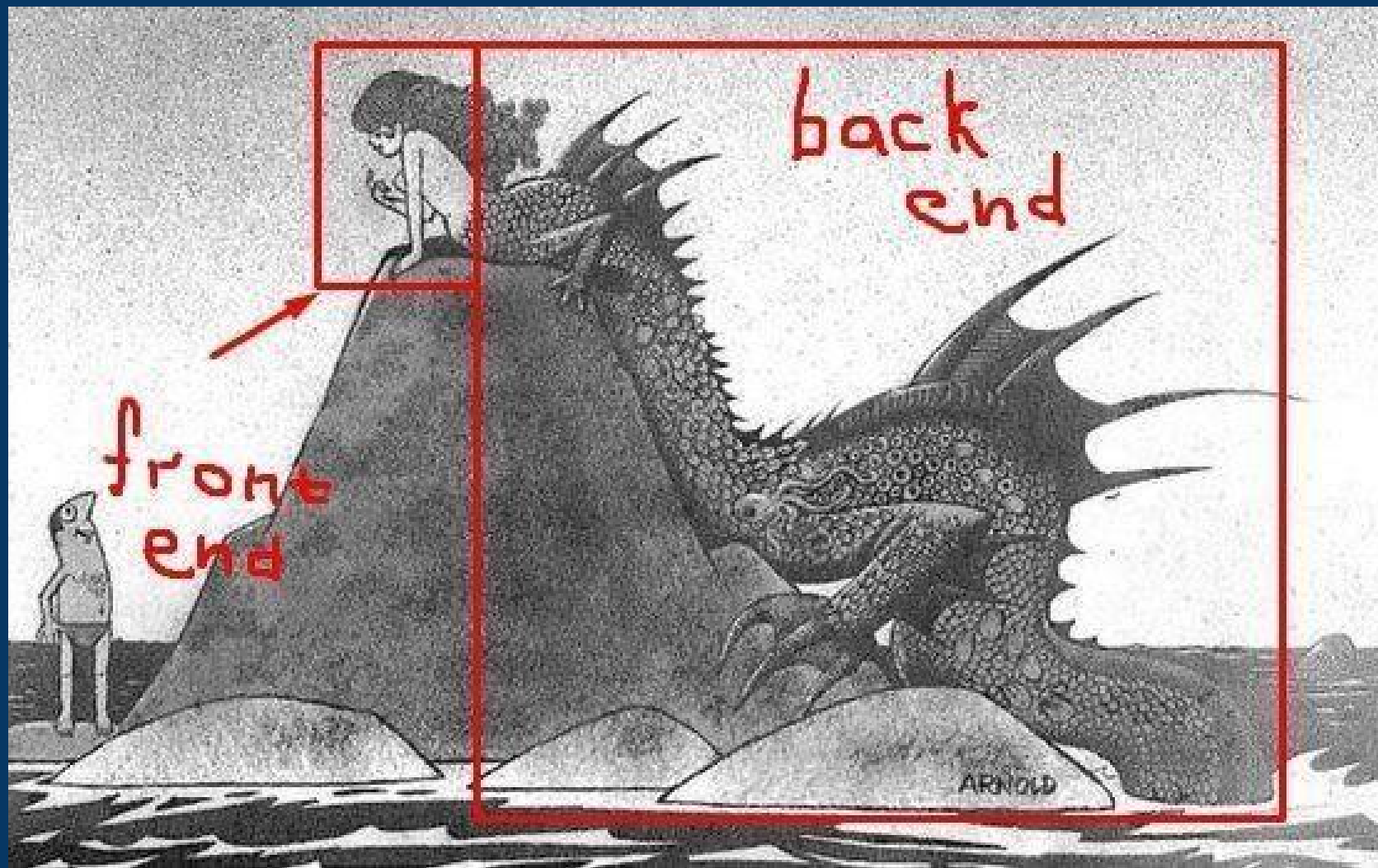
Why web mapping is so trendy



Many
NEEDS

Many
PROFESSIONAL PROFILES

Many
ROLES





Product	Short description
ArcGIS Online & Portal for ArcGIS	Online platform to manage and share: data, user, maps, etc.
ArcGIS Web App Templates	Javascript configurable applications : viewer, editor, social
Web AppBuilder for ArcGIS	GUI to create web app but it is also a framework to develop applications based on maps
ArcGIS API for JavaScript	JavaScript Library for ArcGIS
ArcGIS Online Content	Ready to use content available through APIs and GUI
ArcGIS Open Data	Online platform to search and manage Open Data
Koop	Koop extracts geographic data from third party providers, transforms it in various formats.
ArcGIS Server	Server for creating and managing GIS Web Services, applications and data
ArcGIS Online & ArcGIS Server REST APIs	

[Free developer accounts](#)



Name

ST_Buffer — (T) Returns a geometry covering all points within a given distance from the input geometry.

Synopsis

geometry **ST_Buffer**(geometry *g1*, float *radius_of_buffer*);

geometry **ST_Buffer**(geometry *g1*, float *radius_of_buffer*, integer *num_seg_quarter_circle*);






geometry **ST_Buffer**(geometry *g1*, float *radius_of_buffer*, text *buffer_style_parameters*);

geography **ST_Buffer**(geography *g1*, float *radius_of_buffer_in_meters*);

Description

Returns a geometry/geography that represents all points whose distance from this Geometry/geography is less than or equal to distance.

Geometry: Calculations are in the Spatial Reference System of the geometry. Introduced in 1.5 support for different end cap and mitre settings to control shape.

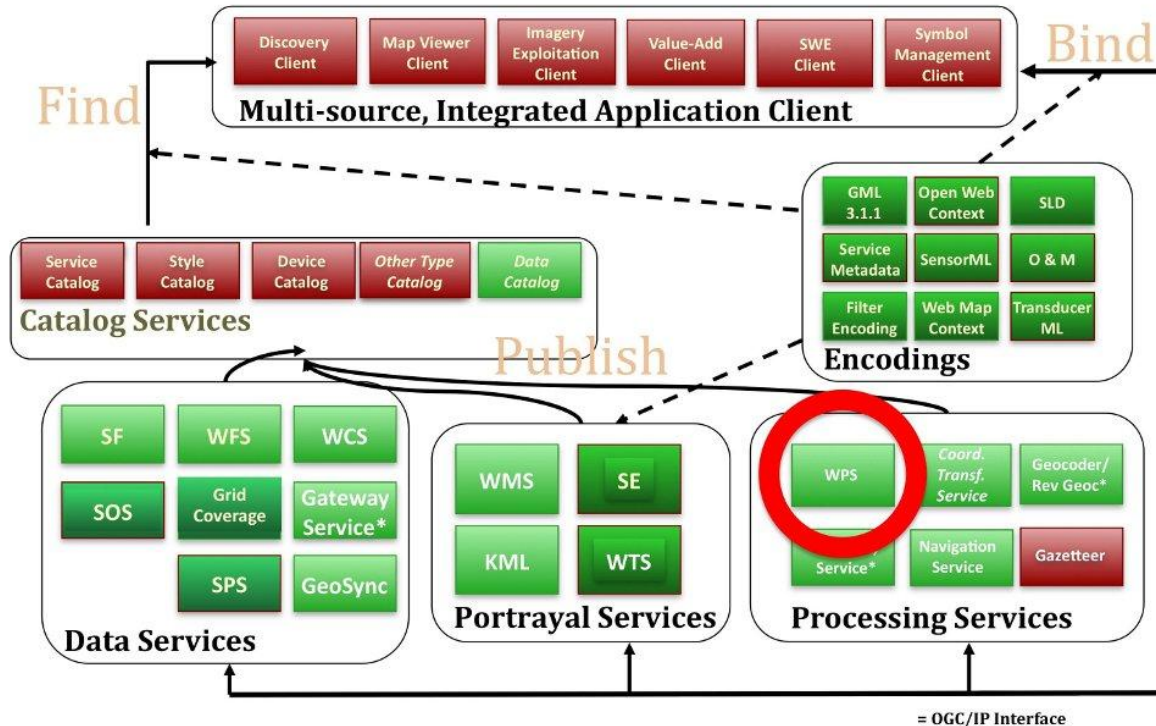
		
quad_segs=8 (default) <pre>SELECT ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50, 'quad_segs=8');</pre>	quad_segs=2 (lame) <pre>SELECT ST_Buffer(ST_GeomFromText('POINT(100 90)'), 50, 'quad_segs=2');</pre>	
		
endcap=round join=round (default) <pre>SELECT ST_Buffer(ST_GeomFromText('LINESTRING(80 80,180 180,180 80)'), 10, 'endcap=round,join=round');</pre>	endcap=square <pre>SELECT ST_Buffer(ST_GeomFromText('LINESTRING(80 80,180 180,180 80)'), 10, 'endcap=square,join=round');</pre>	endcap=flat <pre>SELECT ST_Buffer(ST_GeomFromText('LINESTRING(80 80,180 180,180 80)'), 10, 'endcap=flat,join=round');</pre>

SQL PostGIS Special Spatial Functions

One powerful method of performing spatial processing is through the *Web Processing Service*, or **WPS**. This OGC-based protocol, analogous to other protocols such as Web Map Service (WMS) and Web Feature Service (WFS), allows for client-server interaction with server-hosted “processes”. A server can provide WPS processes, which can then be executed by clients on data they supply or applied to existing server-side datasets.

Processes fall into three categories: vector, raster, and geometry, referring to the type of geospatial content used as the Process’s input. These categories are broad, as processes can take multiple types of input.

The OGC® Web Processing Service Interface (WPS) Standard provides standardized inputs and outputs for geospatial processing services.

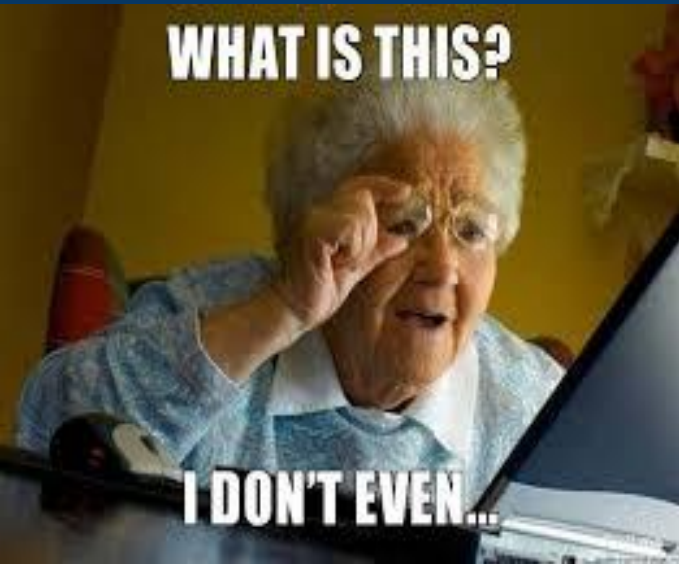


- 1. Hard to implement**
- 2. Slow**
- 3. Difficult to combine**



NODE.JS SERVICES





Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Node.js uses an **event-driven, non-blocking I/O** model that makes it lightweight and efficient.

Node.js' package ecosystem, **npm**, is the largest ecosystem of open source libraries in the world.

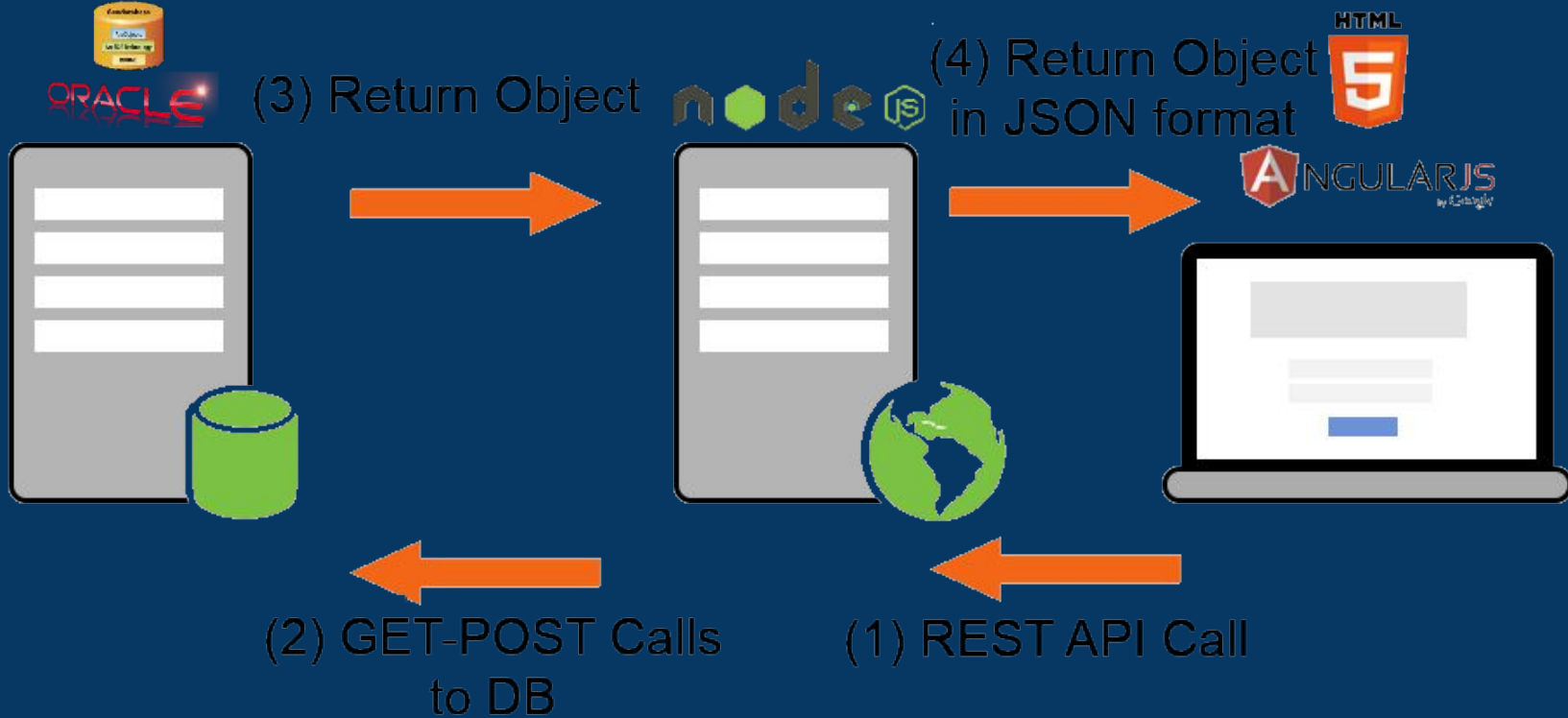
1. Web development in a dynamic language (JavaScript) on a VM that is incredibly fast (V8). It is **much faster than Ruby, Python, or Perl**. Ability to handle thousands of concurrent connections with minimal overhead on a single process.
2. **JavaScript is perfect for event loops** with first class function objects and closures. People already know how to use it this way having used it in the browser to respond to user initiated events. It is arguably **the most popular web programming language**.
3. **Using JavaScript on a web server as well as the browser** reduces the impedance mismatch between the two programming environments which **can communicate data structures via JSON** that work the same on both sides of the equation. Duplicate form validation code can be shared between server and client, etc.

JSON APIs

Building light-weight REST / JSON api's is something where node.js really shines. Its non-blocking I/O model combined with JavaScript make it a great choice for wrapping other data sources such as databases or web services and exposing them via a JSON interface.

Single page apps

If you are planning to write an AJAX heavy single page app (think gmail), node.js is a great fit as well. The ability to process many requests / seconds with low response times, as well as sharing things like validation code between the client and server make it a great choice for modern web applications that do lots of processing on the client.



WHY USE NODE.JS?

- 1 **Faster web Apps: Asynchronous events & non-blocking I/O**
- 2 **NPM: Bigger open source libraries in the world**
- 3 **Javascript Frontend/Backend. JSON Apis.**
- 4 **Really now is COOL.**



<http://codecondo.com/11-awesome-things-you-can-build-with-node-js/>



★ arc-node public

Node module to work with ArcGIS Online and ArcGIS Server

How to install it

Just write this in your prompt: `npm install --save arc-node`

And you are ready to go, just instantiate the object like this:

```
var ArcNode = require('arc-node'),
    service = new ArcNode(<config object>);
```

Check here the description of the *<config object>* parameter.

Documentation

When you have instantiate the service you will have available methods to:

- [Get a new token](#)
- [Check if a feature service exists](#)
- [Create an empty feature service](#)
- [Determine the SQLType for an EsriType](#)
- [Create a JSON object describing a field in a layer](#)
- [Create a JSON object describing a layer](#)
- [Add layers to a feature service](#)
- [Add features to a layer](#)

 `npm install arc-node`

 [esries](#) published 3 months ago

0.2.9 is the latest of 27 releases

github.com/esri-es/ArcNode

AGPL-3.0 license

Collaborators



Stats

44 downloads in the last day

790 downloads in the last week

1,048 downloads in the last month

No open issues on GitHub

No open pull requests on GitHub

Try it out

NODE FOR GIS OPENSOURCE



no parts, none

npm private modules npm On Site documentation blog npm weekly jobs support

find packages

search

★ pg public

PostgreSQL client - pure javascript & libpq with the same API

mode-postgres

build failing no.js supported

PostgreSQL client for node.js. Pure JavaScript and optional native libpq.

npm install pg

Install

```
$ npm install pg
```

Examples

Client pooling

Generally you will access the PostgreSQL server through a pool of clients to establish a new connection. A client also consumes a non-trivial amount of memory, so something you want to do on every http request. Good news: node-postgres has a `pg-pool` module.

```
var pg = require('pg');
var conString = "postgres://username:password@localhost:5432/dbname";
```

```
var pg = require('pg');
var conString = "postgres://username:password@localhost:5432/dbname";

//this initializes a connection pool
//it will keep idle connections open
//and set a limit of 20 connections
pg.connect(conString, function(err, client) {
  if(err) {
    return console.error('Error connecting to database', err);
  }
  client.query('SELECT 1', function(err, result) {
    //call `done()` to release the connection back to the pool
    done();

    if(err) {
      return console.error('Error executing query', err);
    }
    console.log(result.rows[0].id);
    //output: 1
  });
});
```

notify president madagascar

npm private modules npm On Site documentation blog npm weekly jobs support

find packages

★ leaflet public

JavaScript library for mobile-friendly interactive maps

Leaflet is an open source JavaScript library for **mobile-friendly interactive maps**. It is developed by **Vladimir Agafonkin** of MapBox with a team of dedicated **contributors**. Weighing just about 30 KB of gzipped JS code, it has all the **features** most developers ever need for online maps.

Leaflet is designed with **simplicity, performance and usability** in mind. It works efficiently across all major desktop and mobile platforms out of the box, taking advantage of HTML5 and CSS3 on modern browsers while being accessible on older ones too. It can be extended with a huge amount of **plugins**, has a beautiful, easy to use and **well-documented** API and a simple, readable **source code** that is a joy to **contribute** to.

For more info, docs and tutorials, check out the **official website**.

For **Leaflet downloads** (including the built master version), check out the **download page**.

We're happy to meet new contributors. If you want to **get involved** with Leaflet development, check out the **contribution guide**. Let's make the best mapping library that will ever exist, and push the limits of what's possible with online maps!

build passing

npm install leaflet

moumer published 2 days ago

0.7.7 is the latest of 18 releases

github.com/Leaflet/Leaflet

none license

Collaborators

Stats

1,228 downloads in the last day

7,629 downloads in the last week

27,842 downloads in the last month

170 open issues on GitHub

46 open pull requests on GitHub

Try it out

Test leaflet in your browser.

Keywords

map, gis



find packages

★ socket.io public

node.js realtime framework server

 npm package **1.3.7** downloads **2M/month**

How to use

The following example attaches socket.io to a plain Node.JS HTTP server listening on port 3000.

```
var server = require('http').createServer();
var io = require('socket.io')(server);
io.on('connection', function(socket){
  socket.on('event', function(data){});
  socket.on('disconnect', function(){});
});
server.listen(3000);
```

Standalone

```
var io = require('socket.io')();
io.on('connection', function(socket){});
io.listen(3000);
```

 npm install socket.io rauchg published a month ago**1.3.7** is the latest of 84 releasesgithub.com/Automattic/socket.io**none** license

Collaborators



Stats

80.851 downloads in the last day**450.463** downloads in the last week**1.968.772** downloads in the last month**273** open issues on GitHub**46** open pull requests on GitHub

NODE FOR REALTIME APPs



socket.io

#SERVER:

```
var e = require('express');
var sio = require('socket.io');
var http = require('http');

var app = e();
var server = http.createServer( app );

var io = sio.listen(server);

io.sockets.on('connection', function(socket) {
  socket.on('ping', function(msg) {
    var x=35+Math.random();
    var y=-106+Math.random();
    socket.emit('pong', { "x":x,"y":y });
    //socket.emit('pong', { "x":35.10418,"y":-106.62987 } );
  });
});

app.get('/', function(req, res) {
  res.sendFile('client.html',{root:'Users/Me/LeafletWebsocket/pu
});

server.listen(3000);
```

#CLIENT:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Leaflet.js Socket.io</title>
    <link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7.2/leaflet.css" />
    <style>
      html, body, #map {padding: 0; margin: 0; height: 100%;}
    </style>
  </head>
  <body>
    <script src="http://cdn.leafletjs.com/leaflet-0.7.2/leaflet.js"></script>
    <script src="/socket.io/socket.io.js"></script>
    <div id="message"></div>
    <div id="map"></div>

    <script>
      var socket = io.connect('http://DomainName:3000');
      socket.on('connect', function() {
        alert("Connected to WebSocket Server");
      });

      socket.on('pong', function(msg) {
        //document.getElementById("message").innerHTML=msg;
        L.marker([msg.x,msg.y]).addTo(map).bindPopup(""+msg.x+","+msg.y+"").openPopup();
      });

      var map = L.map('map', {
        center: [35.10418, -106.62987],
        zoom: 9
      });

      L.tileLayer('http://{s}.tile.osm.org/{z}/{x}/{y}.png').addTo(map);

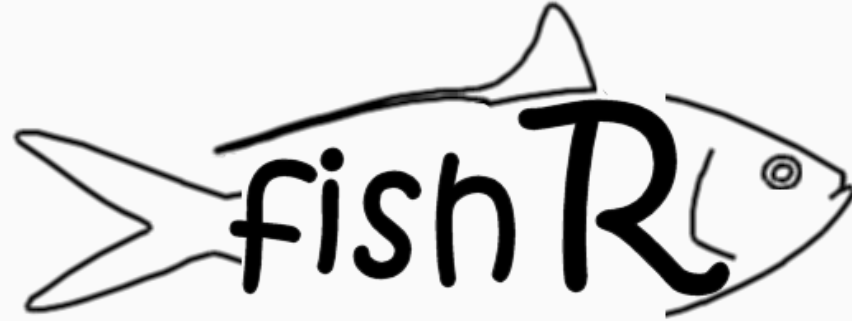
      map.on("click", function(){
        socket.emit('ping', {msg: 'Hello'});
      });

    </script>
  </body>
</html>
```

USING R IN WEB SERVICES

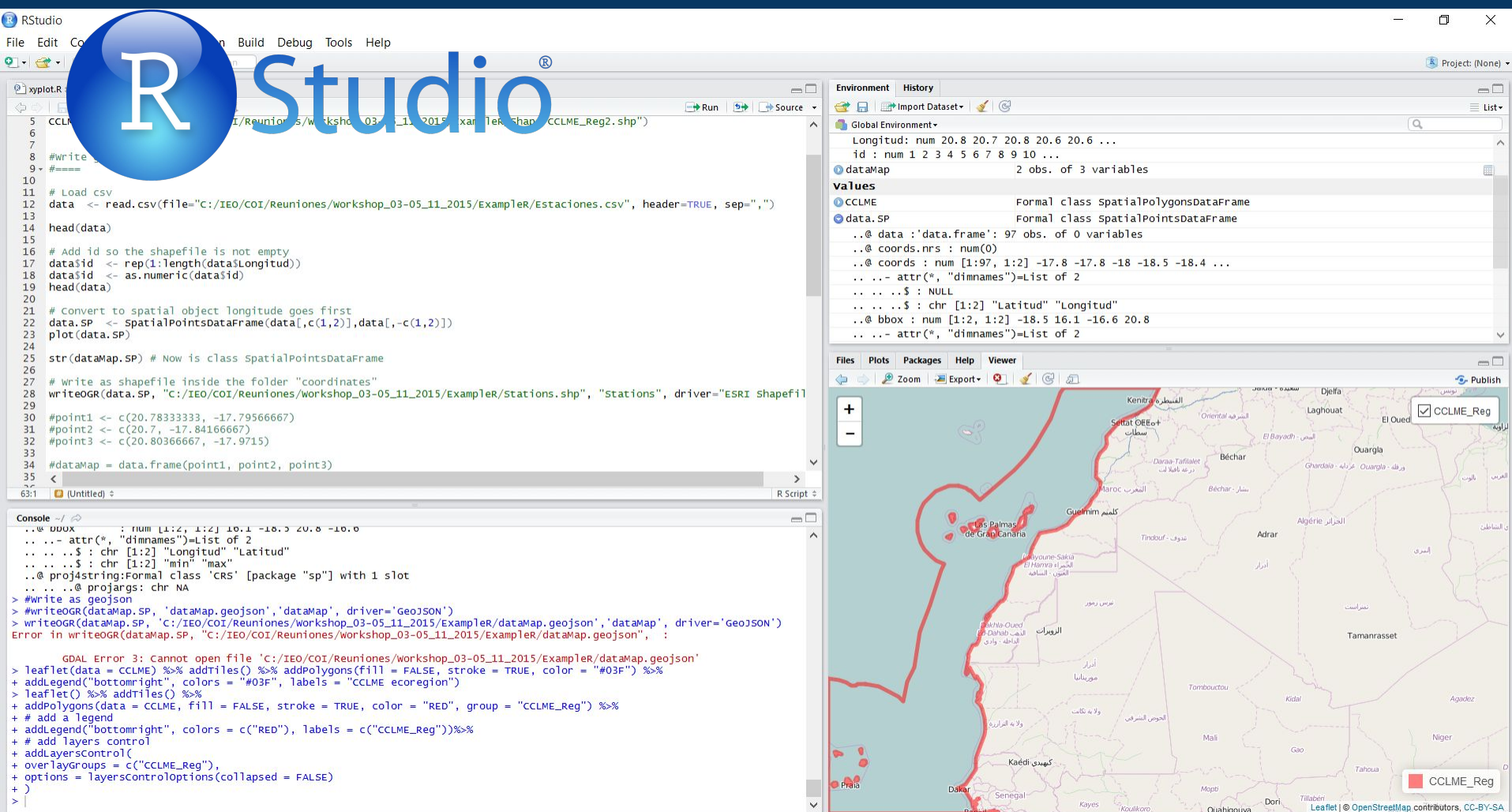


In oceanography and fisheries we have a lot of scripts developed with R



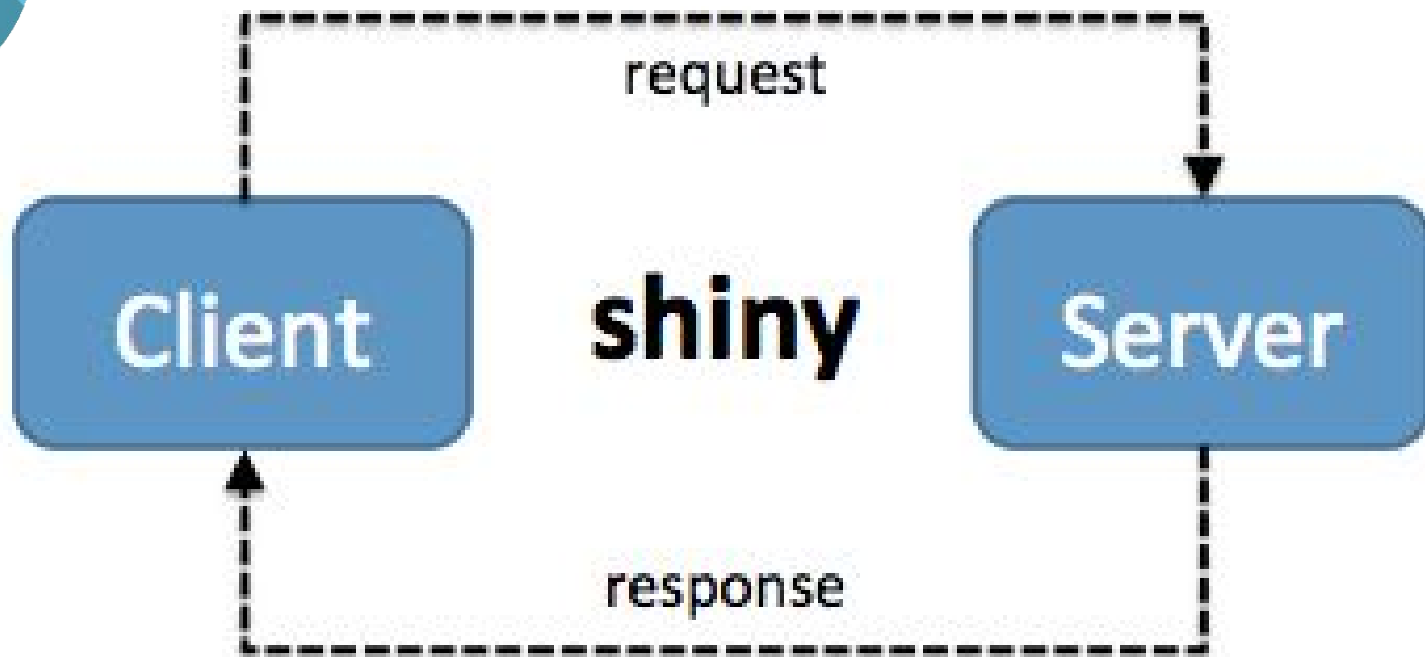
FLR-project

oceR





In my opinion, not enough





- Ubuntu 14.04 Droplet with 1 GB of RAM and 1 GB of swap space or 2 GB of RAM
- The latest version of R installed

Shiny Server is useful not only for hosting Shiny applications, but also for hosting interactive [R markdown documents](#).

By setting up Shiny Server, we are able to host Shiny applications and interactive R documents on the web in a way that is accessible to the public.



Gallery

This gallery contains useful examples to learn from. Visit the [Shiny User Showcase](#) to see an inspiring set of sophisticated apps.

Interactive visualizations

Shiny is designed for fully interactive visualization, using JavaScript libraries like [d3](#), [Leaflet](#), and [Google Charts](#).



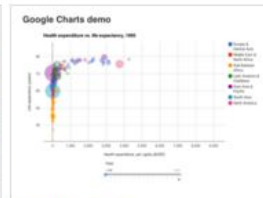
SuperZip example



Bus dashboard



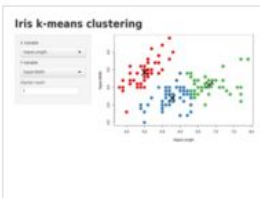
Movie explorer



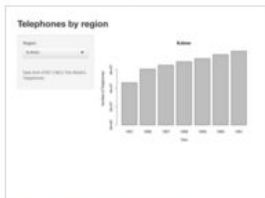
Google Charts

Start simple

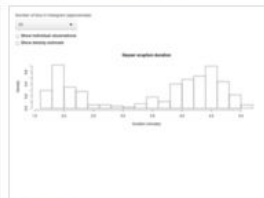
If you're new to Shiny, these simple but complete applications are designed for you to study.



Kmeans example



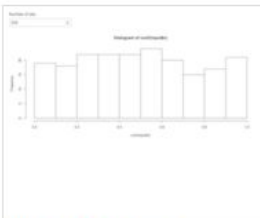
Telephones by region



Faithful



Word cloud



Single-file shiny app



Shiny Server

	Open Source Edition	Professional Edition
Overview	<ul style="list-style-type: none">• Great for hosting lightweight public applications• Does not support authentication or SSL• Single R process per application	<ul style="list-style-type: none">• All of the features of open source; plus:• Perfect for workgroups and enterprises• Supports authentication and SSL• Includes admin dashboard with both realtime and historical performance data• Can use multiple R processes per app
Documentation	Getting Started with Shiny Server	Shiny Server Professional Admin Guide
Support	Community forums only	<ul style="list-style-type: none">• Priority Email Support• 8 hour response during business hours (ET)
License	Open Source AGPL v3	RStudio License Agreement
Pricing	Free	<ul style="list-style-type: none">• 20 concurrent users: \$9,995/server/year• Additional 20 concurrent users: \$4,995/server/year• Additional 150 concurrent users: \$14,995/server/year* <p>Academic and Small Business discounts available</p>



★ rstats public

A node.js interface for statistical programming language R

npm package 0.3.1 build passing coverage 100% dependencies out-of-date

An interface for node.js to statistical programming language R based on the fabulous [Rcpp package](#)

Installation

```
var exec = require('child_process').exec;
exec('pwd', function callback(error, stdout, stderr){
  // result
});
```

With these prerequisites satisfied, one can simply install `rstats` using npm

```
npm install rstats
```

Getting Started

After installation, the package can be loaded as follows:

```
var rstats = require('rstats');
```

Once the package is loaded, we can create an R session by the command

```
var R = new rstats.session();
```

npm install rstats

planeshifter published 7 months ago

0.3.1 is the latest of 14 releases

github.com/Planeshifter/node-Rstats

ISC license

2 downloads in the last day

352 downloads in the last week

520 downloads in the last month

3 open issues on GitHub

One open pull request on GitHub

Try it out

Test rstats in your browser.

Keywords

Rcpp, R, statistics





cooperación
española



United Nations
Educational, Scientific and
Cultural Organization



Intergovernmental
Oceanographic
Commission



Luis Miguel Agudo Bravo
Imagudo@gmail.com



<http://2carto.com/>



es.linkedin.com/pub/luis-miguel-agudo-bravo/7b/630/8b2/