Q

What is non-blocking I/O, and why is it relevant ?

I/O

*A **blocking IO** means:*
*a given thread cannot do anything more until the **IO** is fully received (in the case of sockets this wait could be a long time).*

*Non-**blocking IO** means:*
*an **IO** request is queued straight away and the function returns. The actual **IO** is then processed at some later point by the kernel*

Q

Can JavaScript be used server-side?

Server-Side JavaScript

A substitute for CGI.

Server-dependent technology to process the Web page ***before*** passing it to the client.

An approach which started long ago (Netscape SSJS)

Then mostly forgotten, later revived by Rhino (a bridge between JS and Java) and more recently by **Node.js**

Node.js

an open-source, cross-platform JavaScript engine:
not a framework or a library, but a run-time environment based on Chrome's V8 JavaScript engine for executing JavaScript code server-side.

- **event-driven architecture**
- **asynchronous I/O**

Optimizes throughput and scalability
- in Web applications with many input/output operations,
- for real-time Web applications

https://www.w3schools.com/nodejs/default.asp

Node.js

1. The official Node.js website has installation instructions for Node.js:
   https://nodejs.org

1. Download and install.

2. Create a file called "node hello.js"

3. execute "node hello.js"

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

http.createServer(
    function (req, res) {
        res.writeHead(200, {'Content-Type': 'text/plain'});
        res.end('Hello World!');
    }
).listen(8080);
```

Node.js: built-in modules

Module	Description
cluster	To split a single Node process into multiple processes
crypto	To handle OpenSSL cryptographic functions
events	To handle events
fs	To handle the file system
http	To make Node.js act as an HTTP server
https	To make Node.js act as an HTTPS server.
os	Provides information about the operation system
path	To handle file paths
querystring	To handle URL query strings
timers	To execute a function after a given number of milliseconds
url	To parse URL strings
util	To access utility functions
zlib	To compress or decompress files

see https://www.w3schools.com/nodejs/ref_modules.asp for a full list

Node.js: modules

Create your own modules – save the following in "myModule.js

```
exports.myDateTime = function () {
  return Date();
};
```

Use your own modules

```
var dt = require('./myModule');
```

Obtain modules from the cloud

```
npm install upper-case
```

Use the obained modules

```
var dt = require('upper-case');
```

npm: the good and the ugly

The npm ecosystem is open in nature, allowing anyone to submit their packages.

This includes even those that aren’t open sourced via a repository, or those that may bundle backdoors or other malicious code.

With npm spanning more than 1.5 million packages, it isn’t an easy task to monitor and catch them all in time, creating a lucrative playground for attackers.

Express.js

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`)
})
```

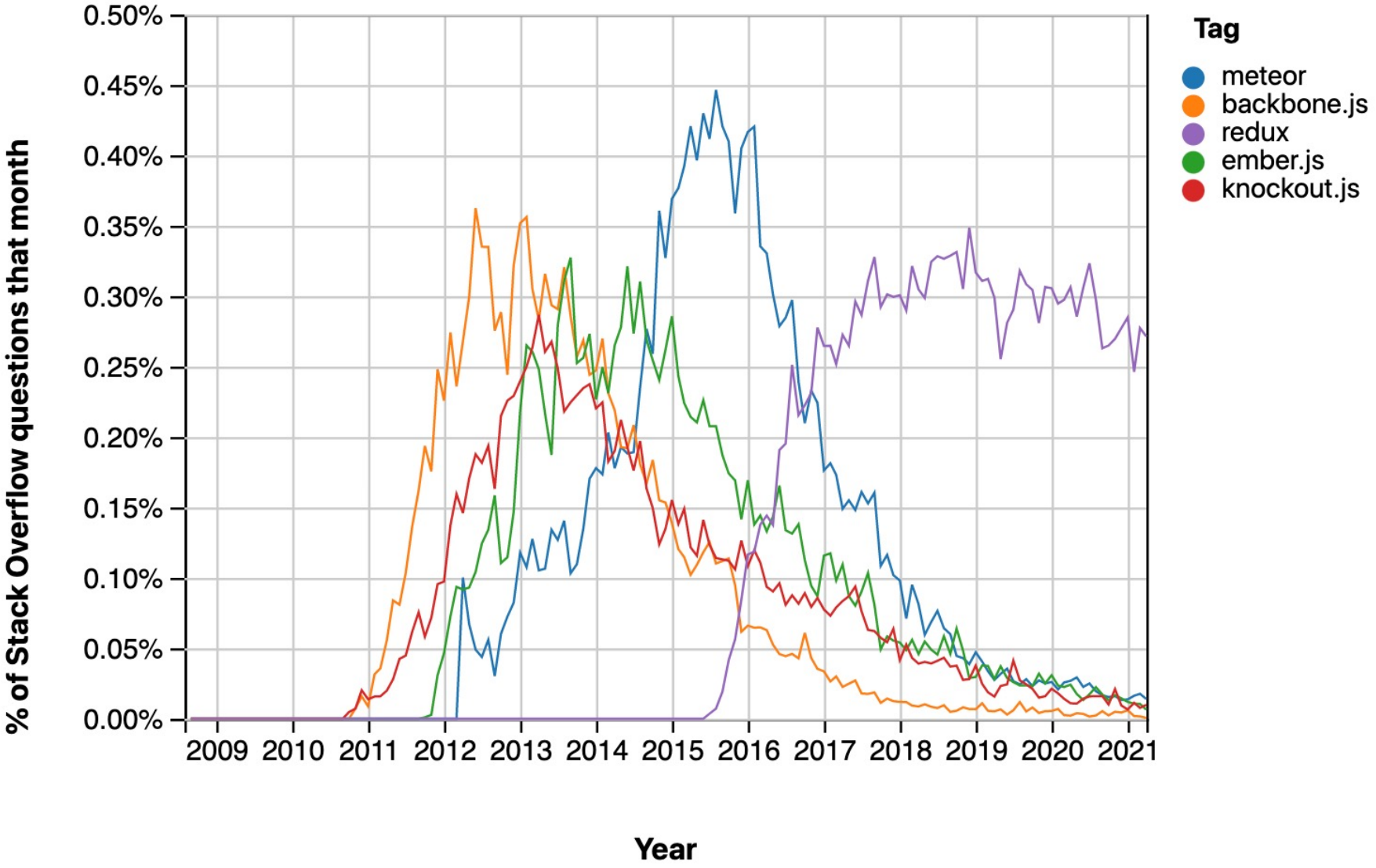
see https://expressjs.com/en/starter/hello-world.html

cookie-session

```
var cookieSession = require('cookie-session')
var express = require('express')
var app = express()
app.use(
  cookieSession(
      { name: 'session',
        keys: [/* secret keys */],
        // Cookie Options
        maxAge: 24 * 60 * 60 * 1000 // 24 hours
      }
  )
)
```

see http://expressjs.com/en/resources/middleware/cookie-session.html

Other Node.js-based frameworks



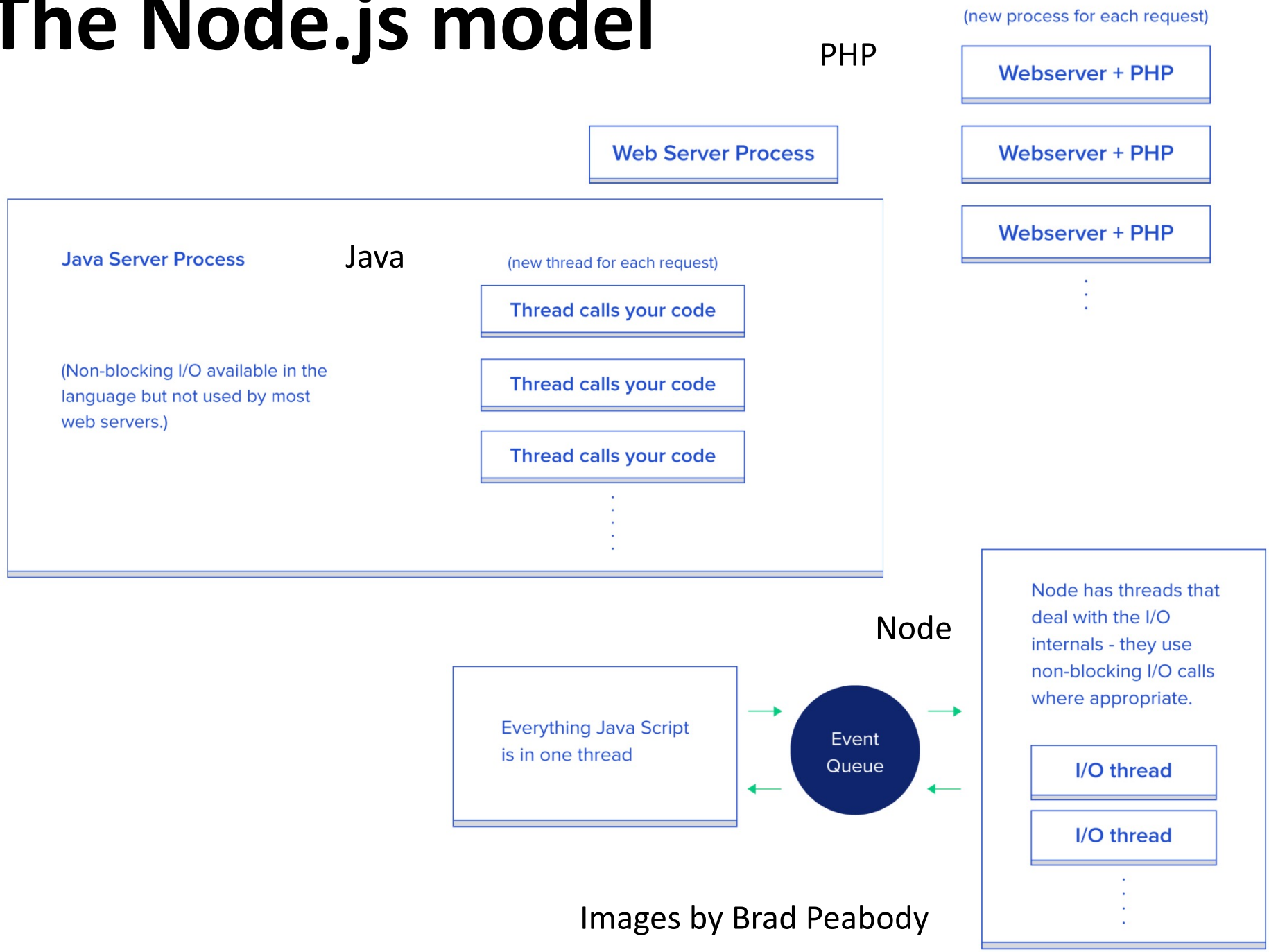
Node.js stories

Nexflix used JavaScript and NodeJS to transform their website into a single page application.

Traditionally, Netflix has been an enterprise Java shop, but “as we migrated out of the data center to the cloud we moved to a more service-based architecture,” Trott said.

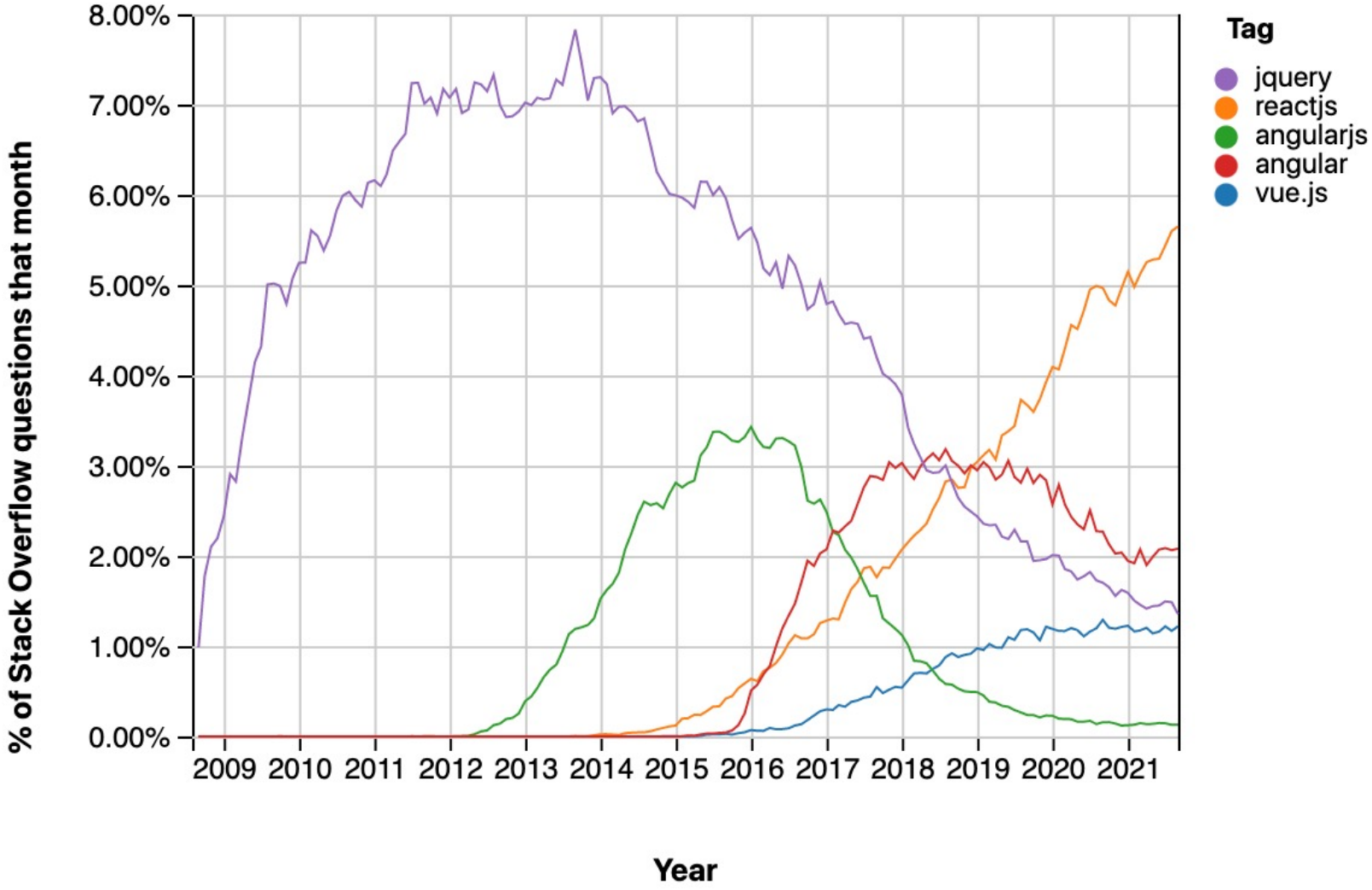
Java still powers the backend of Netflix, but all the stuff that the user sees comes from Node.

The Node.js model



Images by Brad Peabody

Main js frameworks



https://insights.stackoverflow.com/trends

Reactive programming

Reactive programming

Reactive programming is programming with asynchronous data streams.

A stream is a sequence of ongoing events ordered in time. It can emit three different things: a value (of some type), an error, or a "completed" signal.

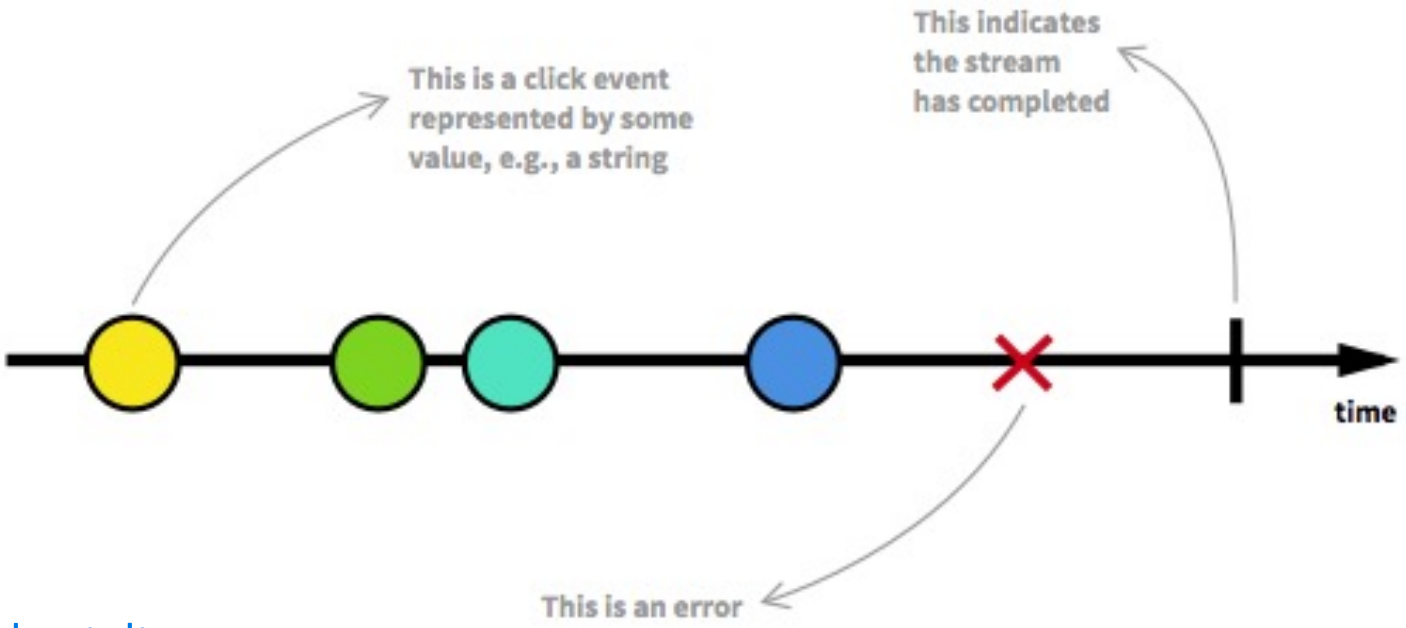


image by [andrestaltz]

Reactive programming

We capture these emitted events asynchronously, by defining functions that will execute :

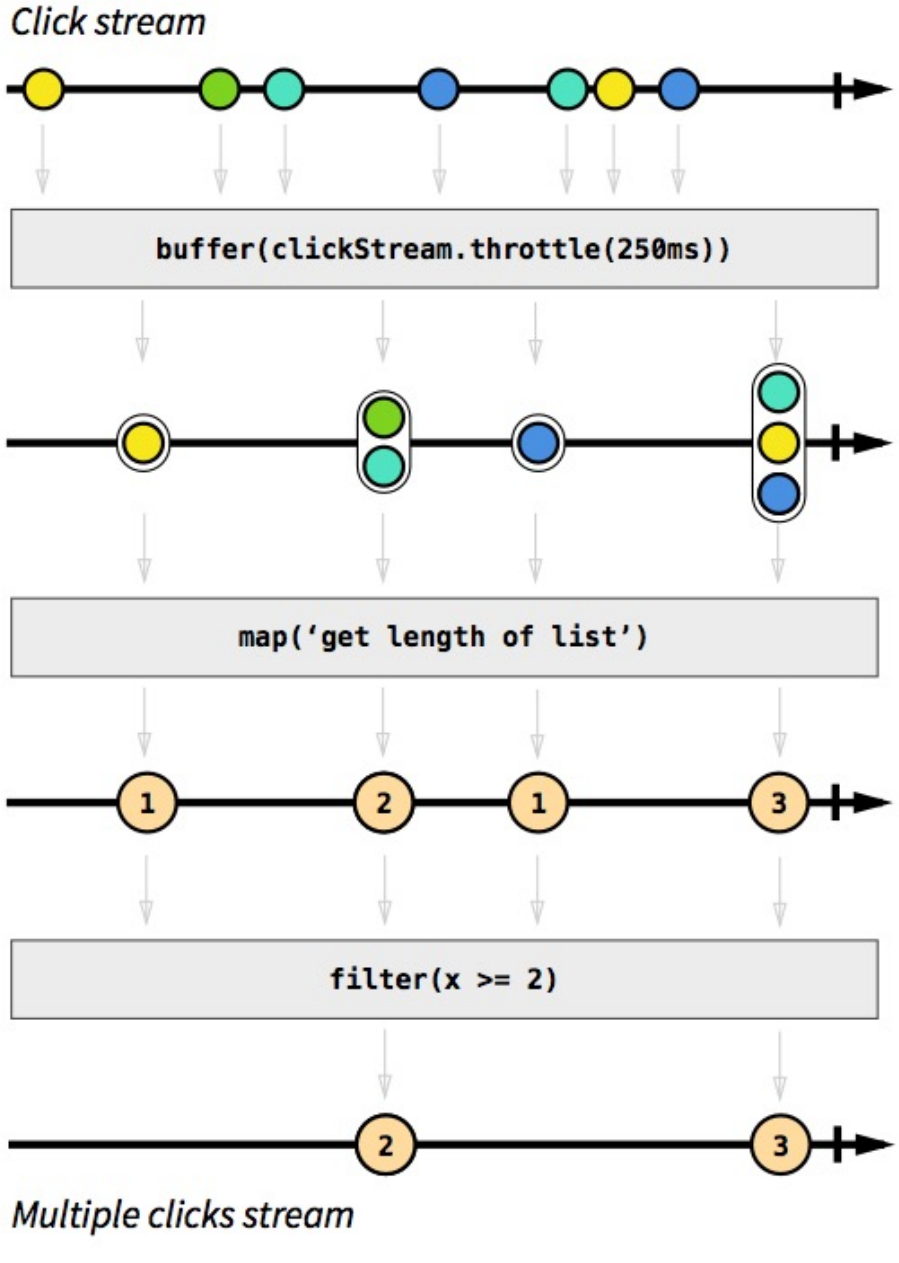
- when a value is emitted,
- when an error is emitted,
- when 'completed' is emitted.

According to the Observer Design Pattern:

- The stream "observable" being observed.
- The "listening" to the stream is called subscribing.
- The functions we are defining are "observers".

Reactive programming

In a reactive programming environment, you are generally given a toolbox of functions to combine, create and filter any of those streams.



RxJS

RxJS is a library for composing asynchronous and event-based programs by using observable sequences.
It provides:

- one core type, the Observable,
- satellite types (Observer, Schedulers, Subjects)
- operators (map, filter, reduce, every, etc) to allow handling asynchronous events as collections.

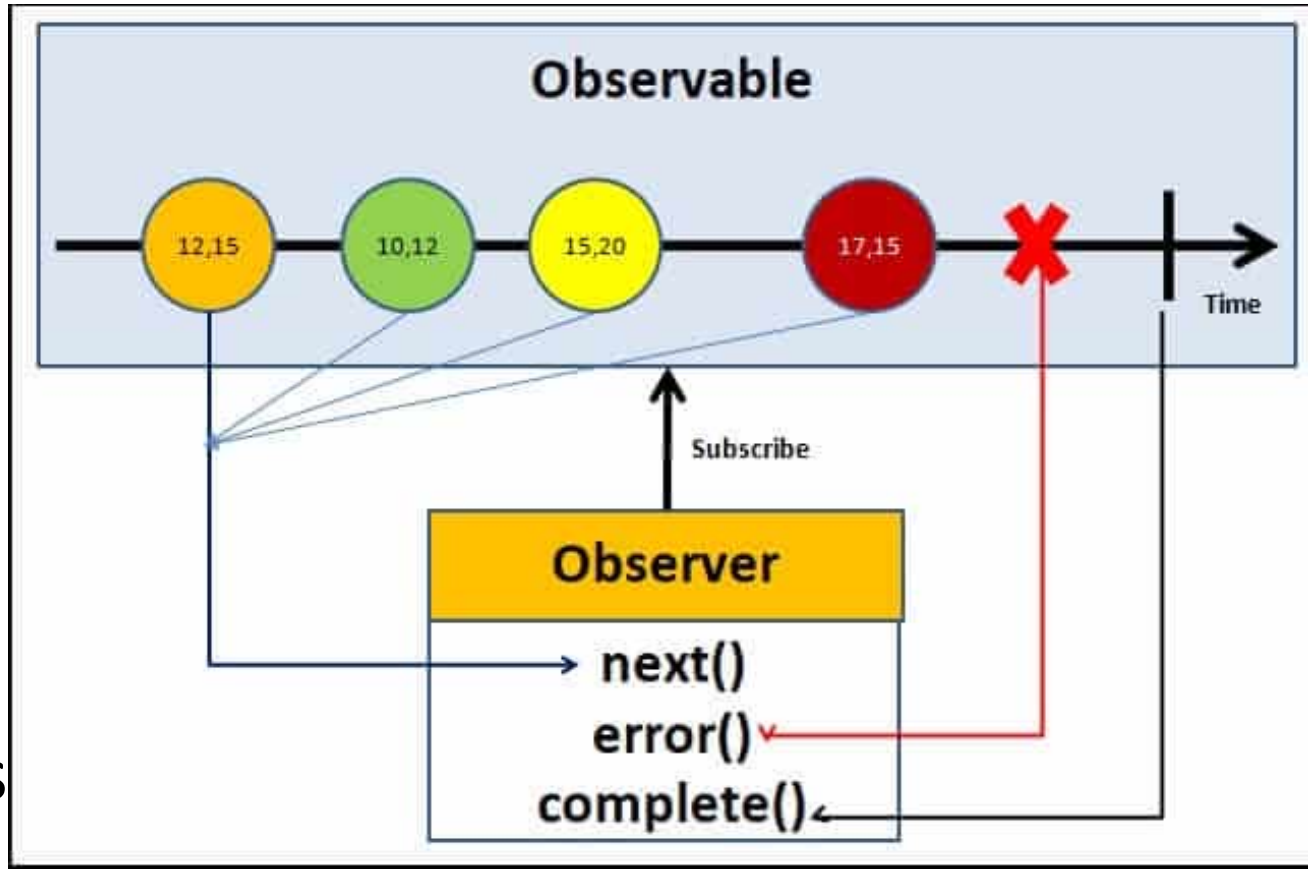
see https://rxjs-dev.firebaseapp.com/guide/overview

RxJS - operators

AREA	OPERATORS
Creation	`from`, `fromEvent`, `of`
Combination	`combineLatest`, `concat`, `merge`, `startWith` , `withLatestFrom`, `zip`
Filtering	`debounceTime`, `distinctUntilChanged`, `filter`, `take`, `takeUntil`
Transformation	`bufferTime`, `concatMap`, `map`, `mergeMap`, `scan`, `switchMap`
Utility	`tap`
Multicasting	`share`

see https://angular.io/guide/rx-library#operators

RxJS

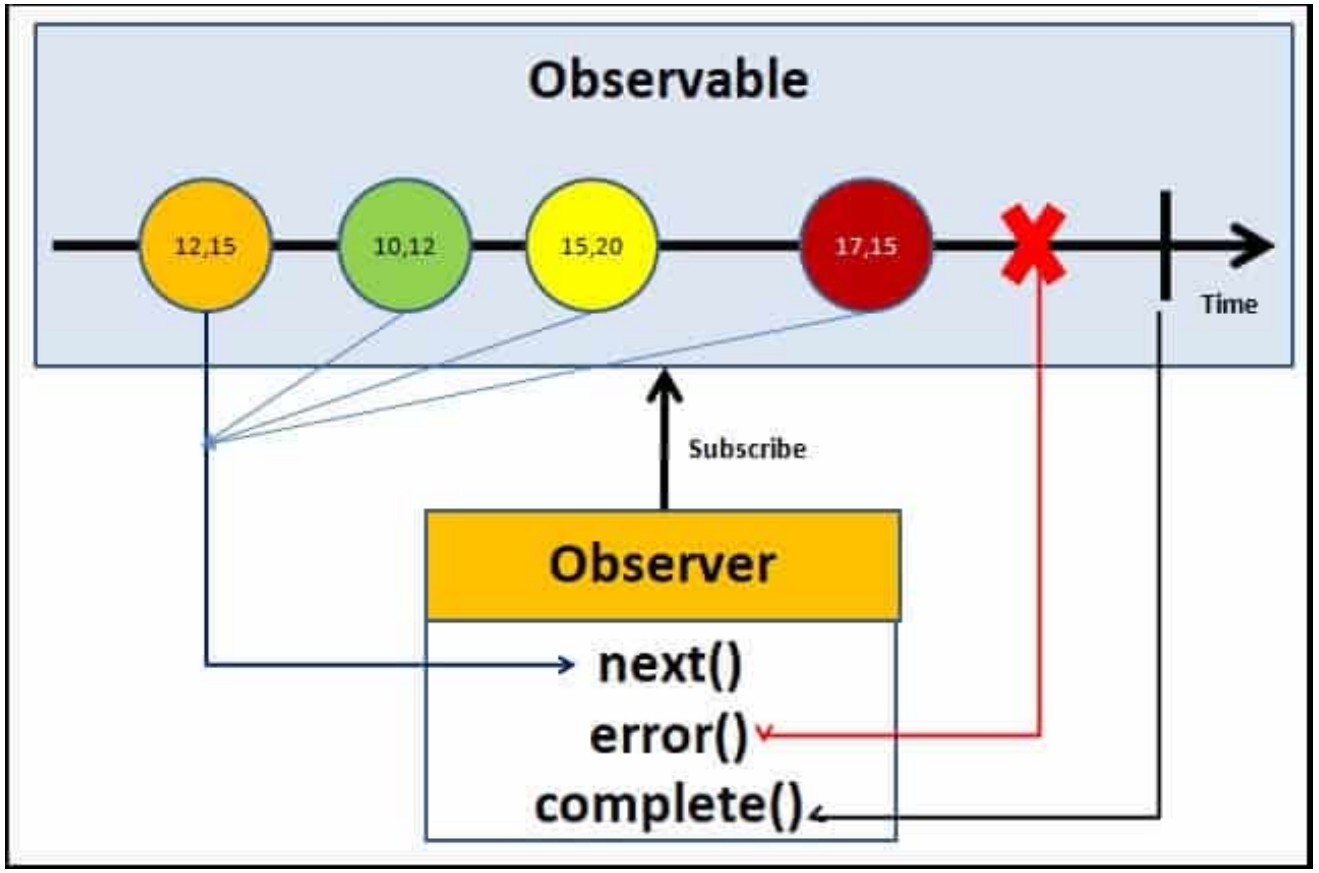


The observable:

- invokes the next() callback whenever a value arrives in the stream. It passes the value as the argument to the next callback.
- If the error occurs, then the error() callback is invoked.
- It invokes the complete() callback when the stream completes.

RxJS

Observers subscribe to Observables.



Observer registers three callbacks with the observable at the time of subscribing:

- next(),
- error()
- complete()

All three callbacks are optional.

The observer receives the data from the observer via the next() callback

They also receive the errors and completion events via the error() & complete() callbacks

Vue

A progressive framework.

Go to https://vuejs.org/ and click on Why Vue.js for a short video.