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');
```

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}`)
})
```

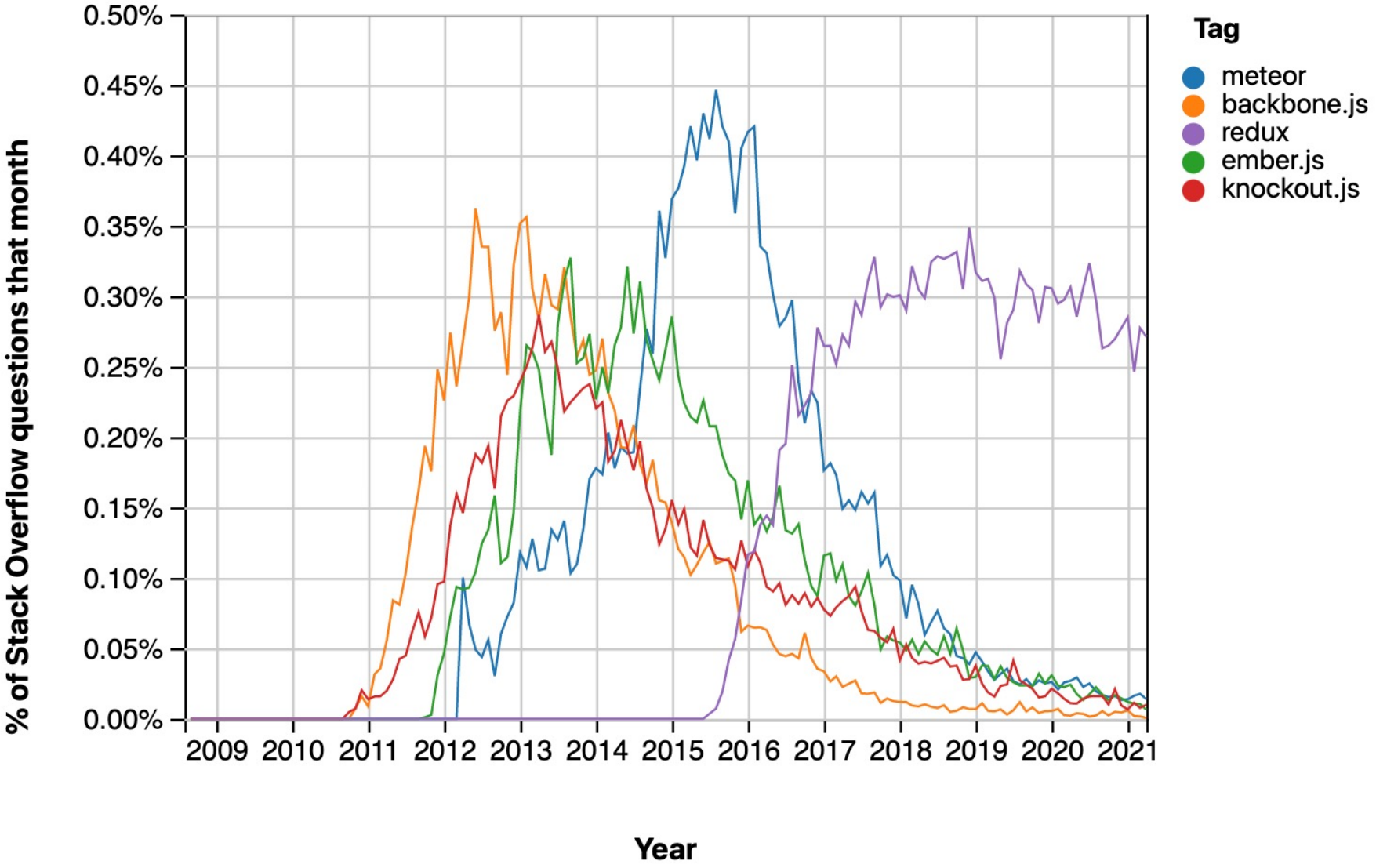
see https://expressjs.com/en/starter/hello-world.html

cookie-session

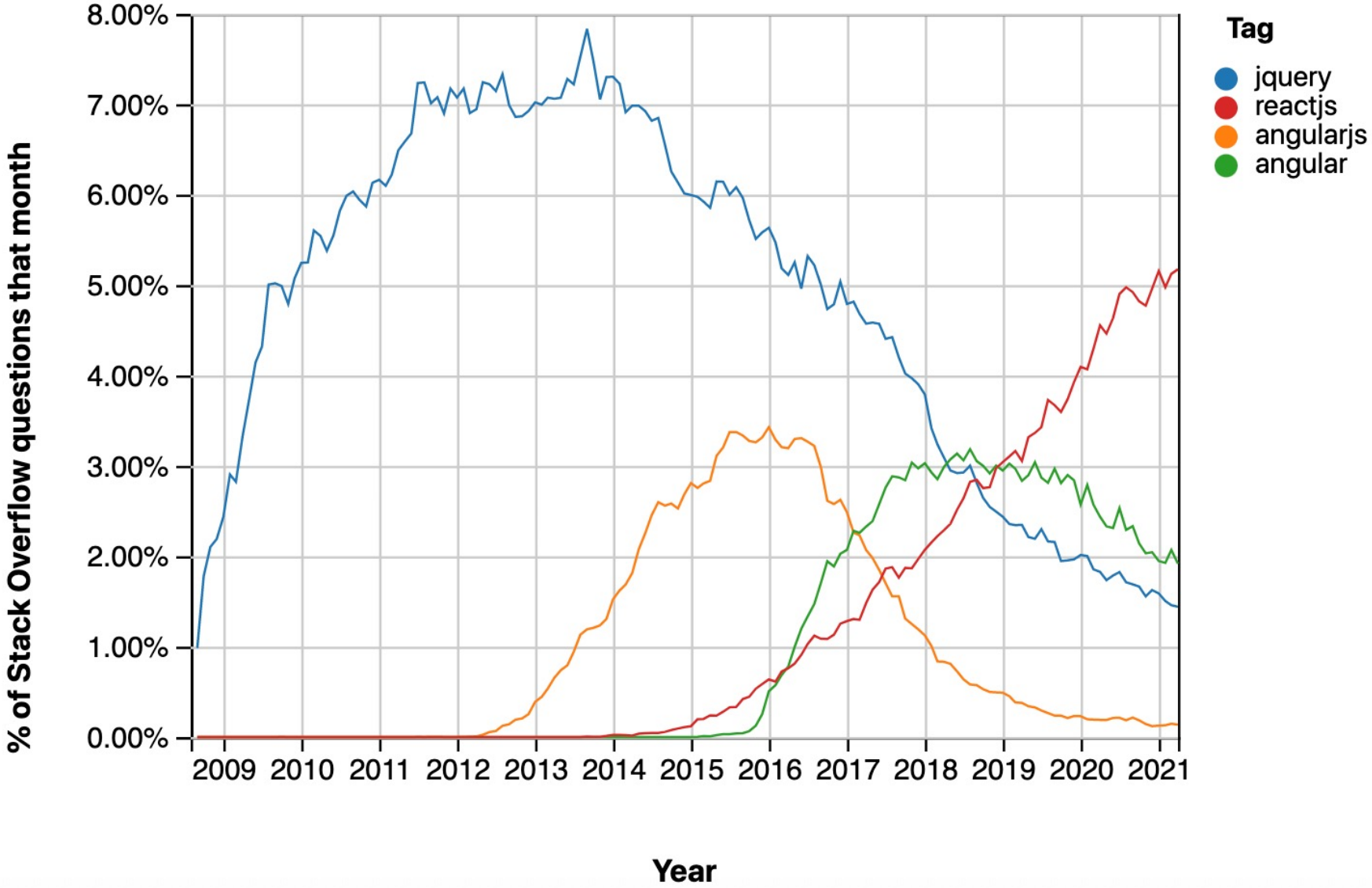
```
var cookieSession = require('cookie-session')
var express = require('express')
var app = express()
app.use(
  cookieSession(
      { name: 'session',
        keys: [/* secret keys */],
        // Cookie Options
        maxAge: 24 * 60 * 60 * 1000 // 24 hours
      }
  )
)
```

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

Other Node.js-based frameworks



Main js frameworks



jQuery

jQuery

The most widely deployed JavaScript library (3-4x more usage than any other).

Free, open-source (MIT License)

Designed to simplify:

- HTML DOM tree traversal and manipulation,
- event handling,
- CSS manipulation, effects and animation
- Ajax

Include jQuery

1) Download it from https://jquery.com/download/
2) Use it:

```
<head>
<script src="jquery-3.4.1.min.js"></script>
</head>
```

Most CDN's will serve the request from the closest server, which leads to faster loading time.

CDN:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

```
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.4.1.min.js"></script>
```

Many users already have downloaded jQuery from Google or Microsoft when visiting another site.
So they already have it in the cache

Basic jQuery syntax

Basic syntax is: **$(*selector*).*action*()**

where:

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)
  - or *event* performed on the element(s)

```
$(this).hide() - hides the current element.
$("p").hide() - hides all <p> elements.
$(".test").hide() - hides all elements with class="test".
$("#test").hide() - hides the element with id="test".
```

Document.ready (actions/events)

In general, all jQuery methods are inside a document ready event:

```
$(document).ready(function(){

  // jQuery methods go here...

});
```

There are other shortcuts, but this writing is easier to read

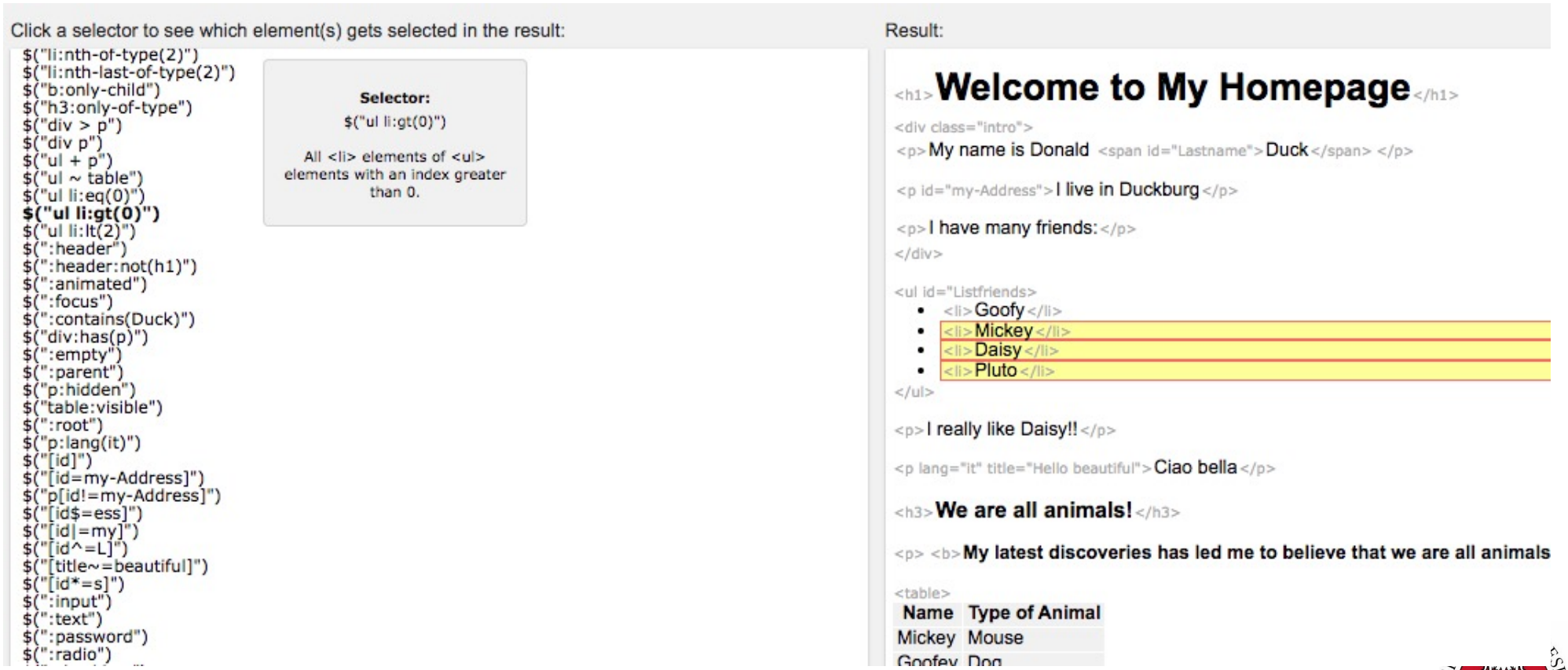
This prevents any jQuery code from running before the document is finished loading (is ready).

jQuery selectors

Similar to CSS:

See here of an excellent contextual demo:

https://www.w3schools.com/jquery/trysel.asp



jQuery DOM traversing

Selection, iterations and related operations

- find() each() map() slice() filter() end() add()

Tree relations

- parent() parents() parentsUntil() children() siblings()

Set relations

- first() last() closest()
- next() nextAll() nextUntil()
- prev() prevAll() prevUntil()

Conditions

- has() is() not() eq()

jQuery events/actions

Reference:

- https://www.w3schools.com/jquery/jquery_ref_traversing.asp

Examples:

- https://www.w3schools.com/jquery/jquery_traversing.asp

jQuery events/actions

```
$("p").on("click", function(){
  alert("The paragraph was clicked.");
});
```

```
$("button").click(function(){
  $("p").off("click");
});
```

Info about the event and event management:

event.target

event.timeStamp

event.type

event.which

event.stopPropagation()

jQuery: list of common events/actions

- mousedown(), mouseenter(), mouseleave(), mousemove(), mouseout(), mouseover(), mouseup()
- keydown(), keypress(), keyup()
- click(), dblclick()
- focus(),focusin(), focusout(), blur(), hover()
- change()

jQuery events/actions

Reference:

- https://www.w3schools.com/jquery/jquery_ref_events.asp

Examples:

- https://www.w3schools.com/jquery/jquery_events.asp

jQuery effects

- animate()
- fadeIn(), fadeOut(), fadeTo(), fadeToggle()
- hide(), show(), toggle()
- slideDown(), slideToggle(), slideUp()
- delay() ,stop(), finish()
- queue(), clearQueue(), dequeue()

jQuery effects

Reference:

- https://www.w3schools.com/jquery/jquery_ref_effects.asp

Examples:

- https://www.w3schools.com/jquery/jquery_hide_show.asp
- e seguenti:

jQuery DOM/CSS manipulation

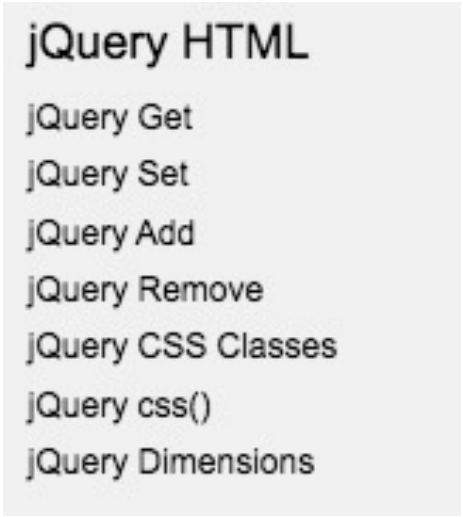
- Get/Set text, html
- Get/Set attributes
- Add/Remove elements
- Add/Remove/Inspect CSS classes
- Read/Set dimension of elements

jQuery DOM/CSS manipulation

Examples

- https://www.w3schools.com/jquery/jquery_dom_get.asp

e seguenti



Reference

- https://www.w3schools.com/jquery/jquery_ref_html.asp

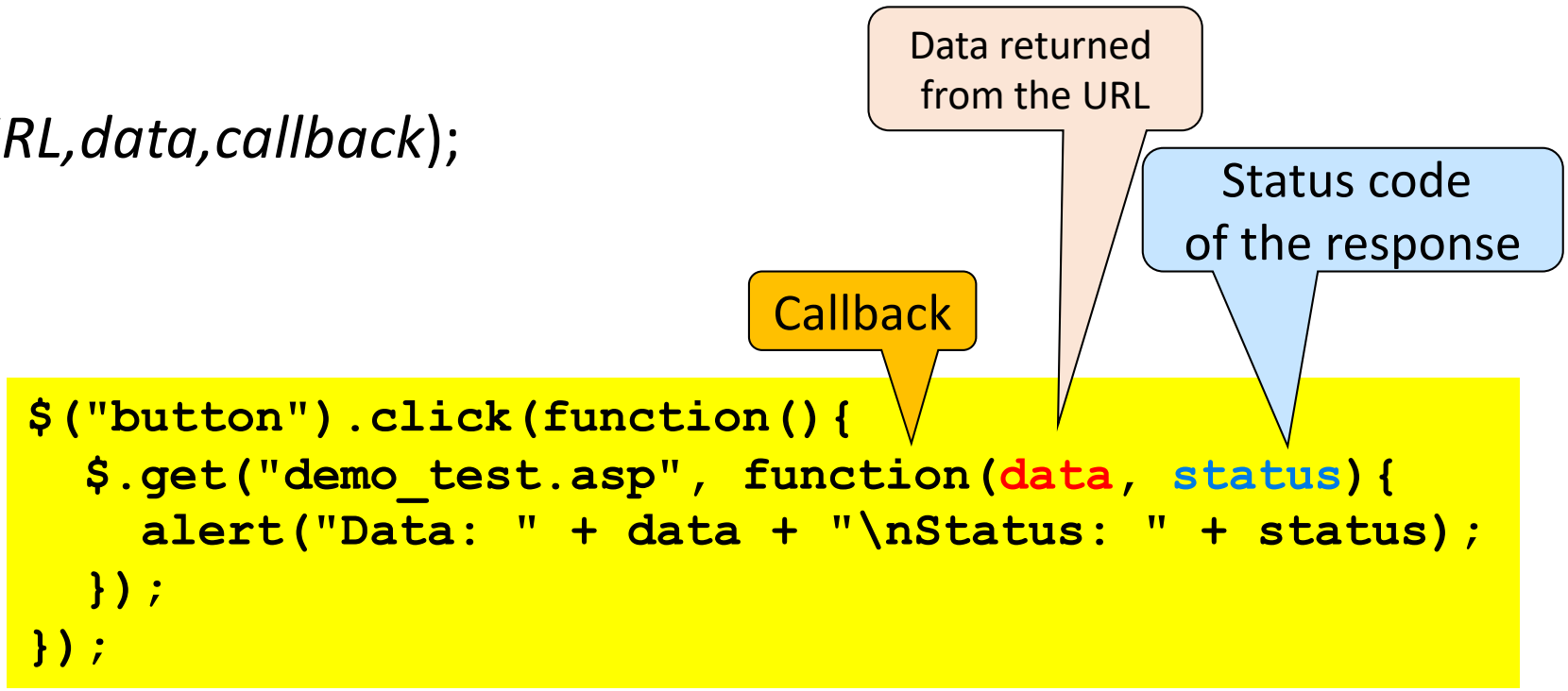
jQuery and Ajax

$(*selector*).load(*URL,data,callback*);

$.get(*URL,callback*);

$.post(*URL,data,callback*);

the callback has 2 args: "data" and "status"



noConflict

- The noConflict() method releases the hold on the $ shortcut identifier, so that other scripts can use it.
- Using full name:

```
$.noConflict();
jQuery(document).ready(function(){
  jQuery("button").click(function(){
    jQuery("p").text("jQuery is still working!");
  });
});
```

- Using Your own variable:

```
var jq = $.noConflict();
jq(document).ready(function(){
  jq("button").click(function(){
    jq("p").text("jQuery is still working!");
  });
});
```

Documentation with offline support

- https://devdocs.io/jquery/

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.

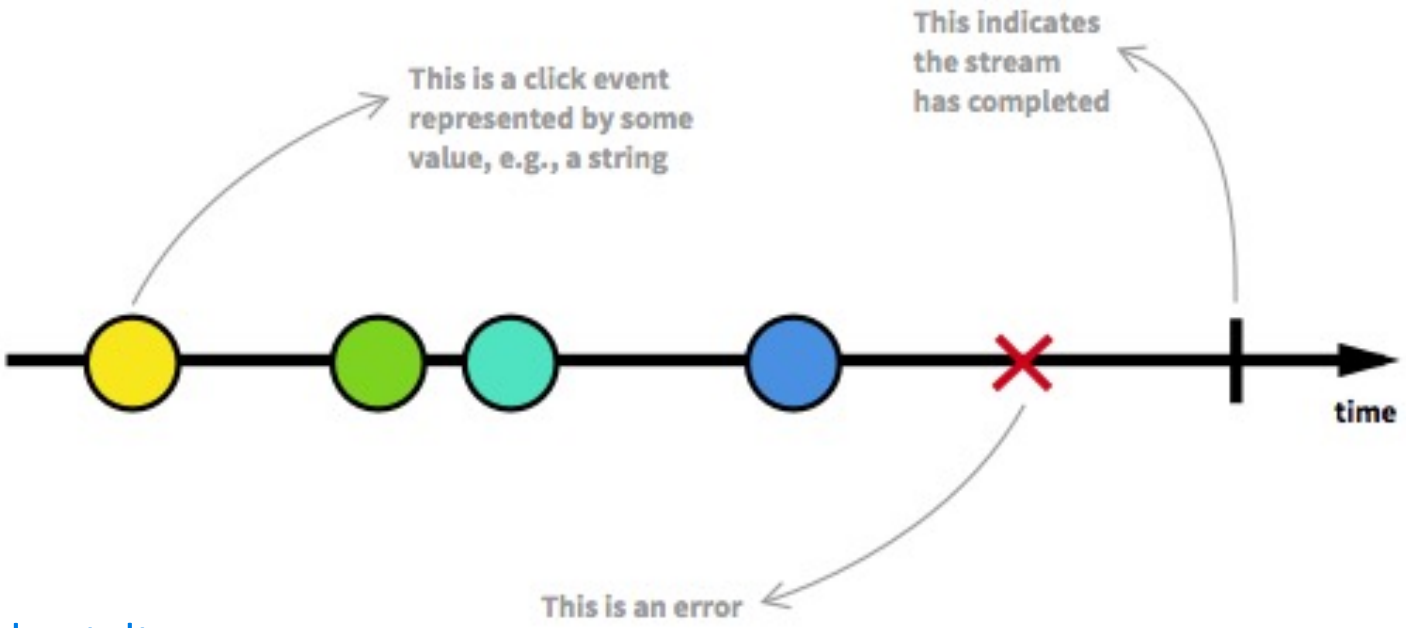


image by [andrestaltz]

Reactive programming

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

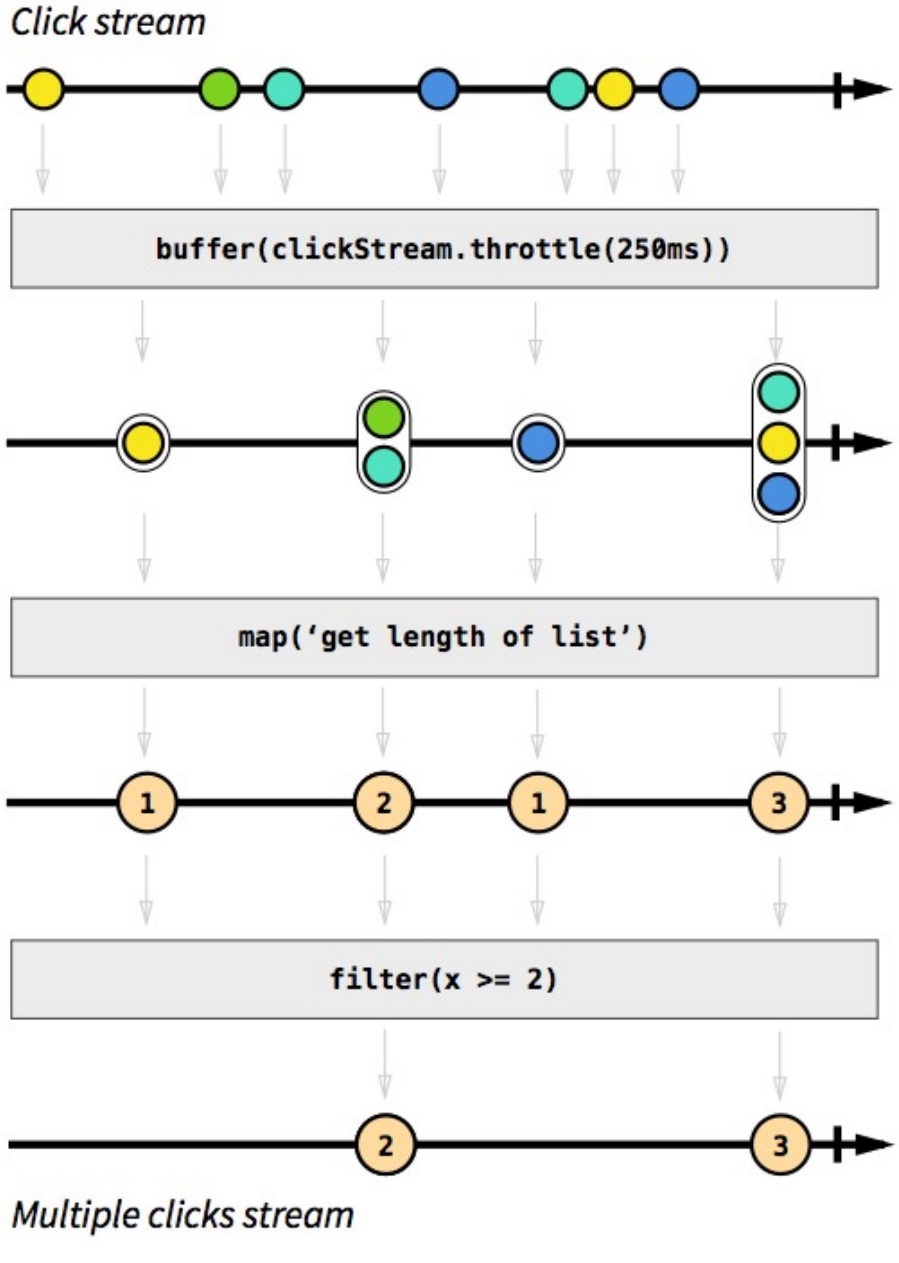
- when a value is emitted,
- when an error is emitted,
- when 'completed' is emitted.

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

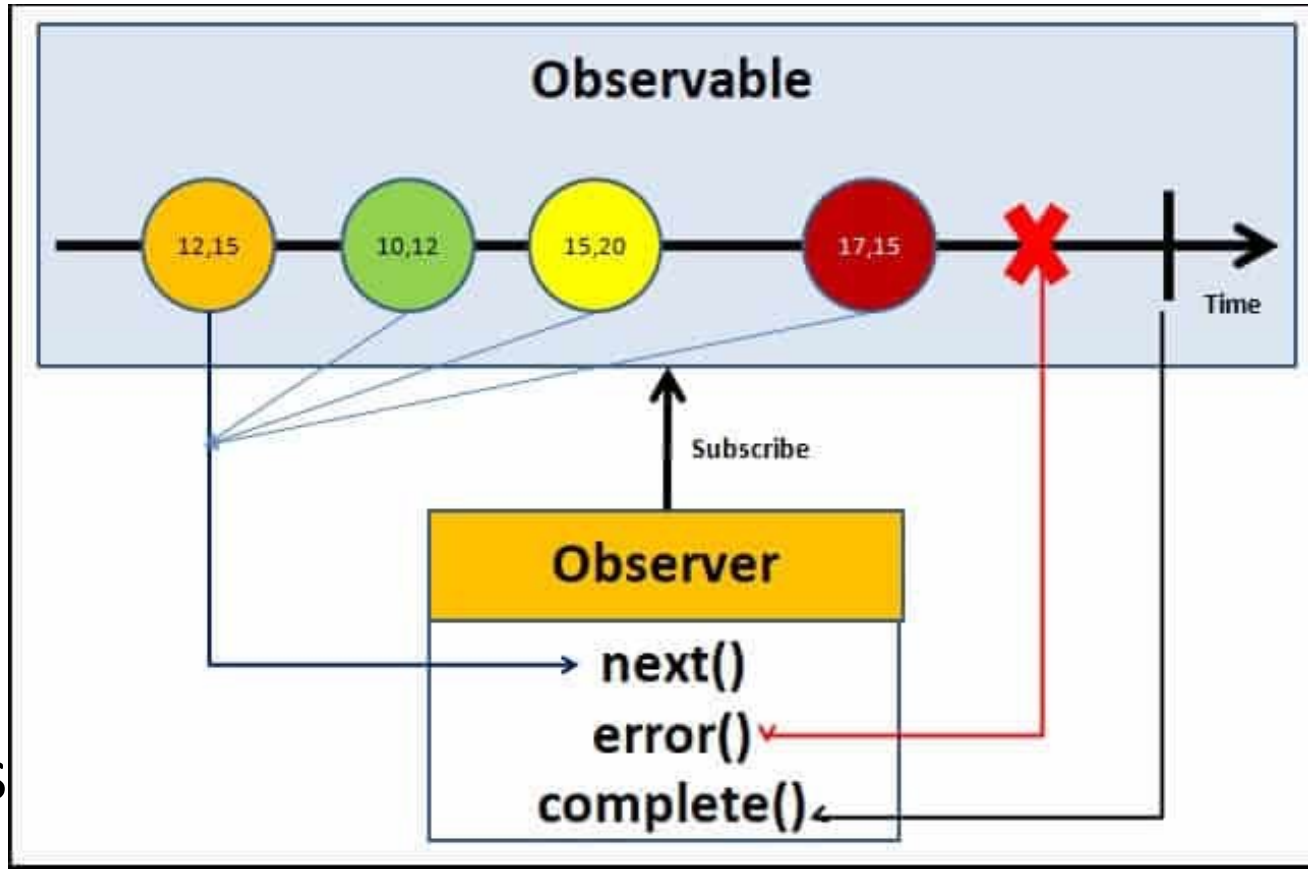
AREA	OPERATORS
Creation	`from`, `fromEvent`, `of`
Combination	`combineLatest`, `concat`, `merge`, `startWith` , `withLatestFrom`, `zip`
Filtering	`debounceTime`, `distinctUntilChanged`, `filter`, `take`, `takeUntil`
Transformation	`bufferTime`, `concatMap`, `map`, `mergeMap`, `scan`, `switchMap`
Utility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

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

RxJS

The observable:

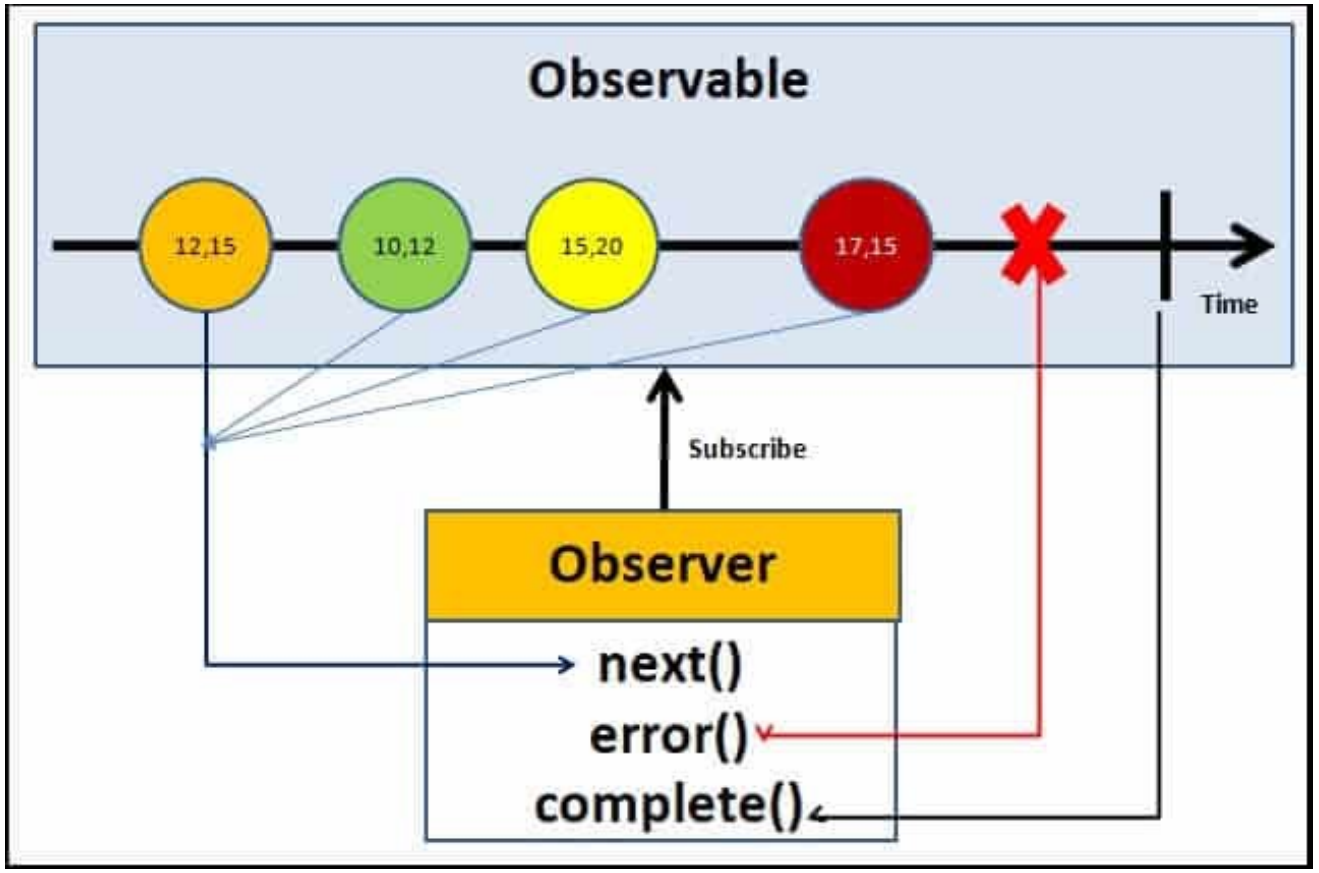
- invokes the next() callback whenever a value arrives in the stream. It passes the value as the argument to the next callback.



- 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()

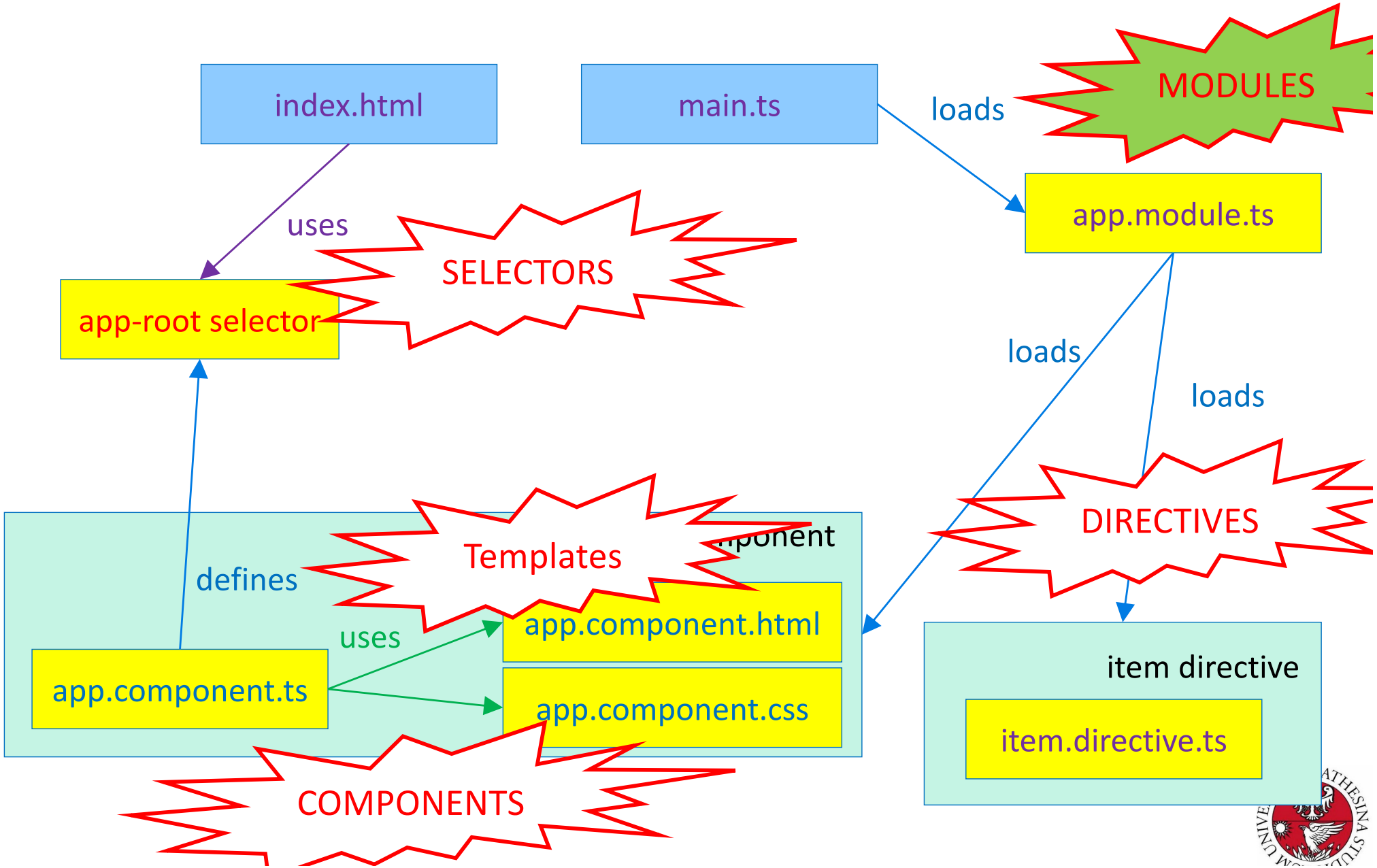
All three callbacks are optional.

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

Angular

Angular: the structure



- How components communicate?
  - (parent to child, child to parent, component to service to component)
- How component communicate with HTTP Servers?
- What is the role of Services?
- What is Routing?

Vue

A progressive framework.

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