

Web Development Frameworks

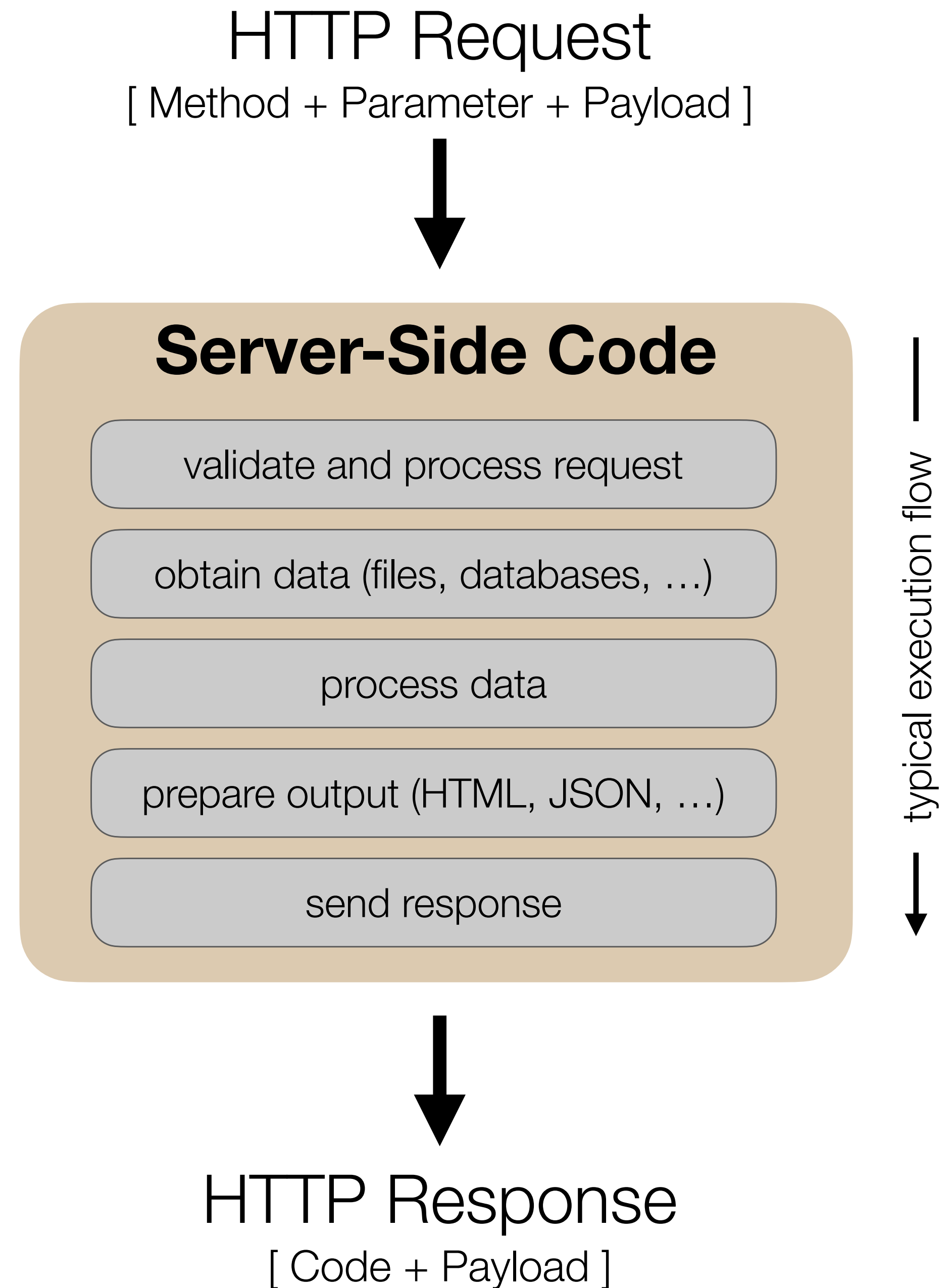
Databases and Web Applications Laboratory (LBAW)
Bachelor in Informatics Engineering and Computation (L.EIC)

Sérgio Nunes
Dept. Informatics Engineering
FEUP · U.Porto

Outline

- Software frameworks
- History of web frameworks
- Typical components in web frameworks
- Server-side frameworks

Server-Side Web Development

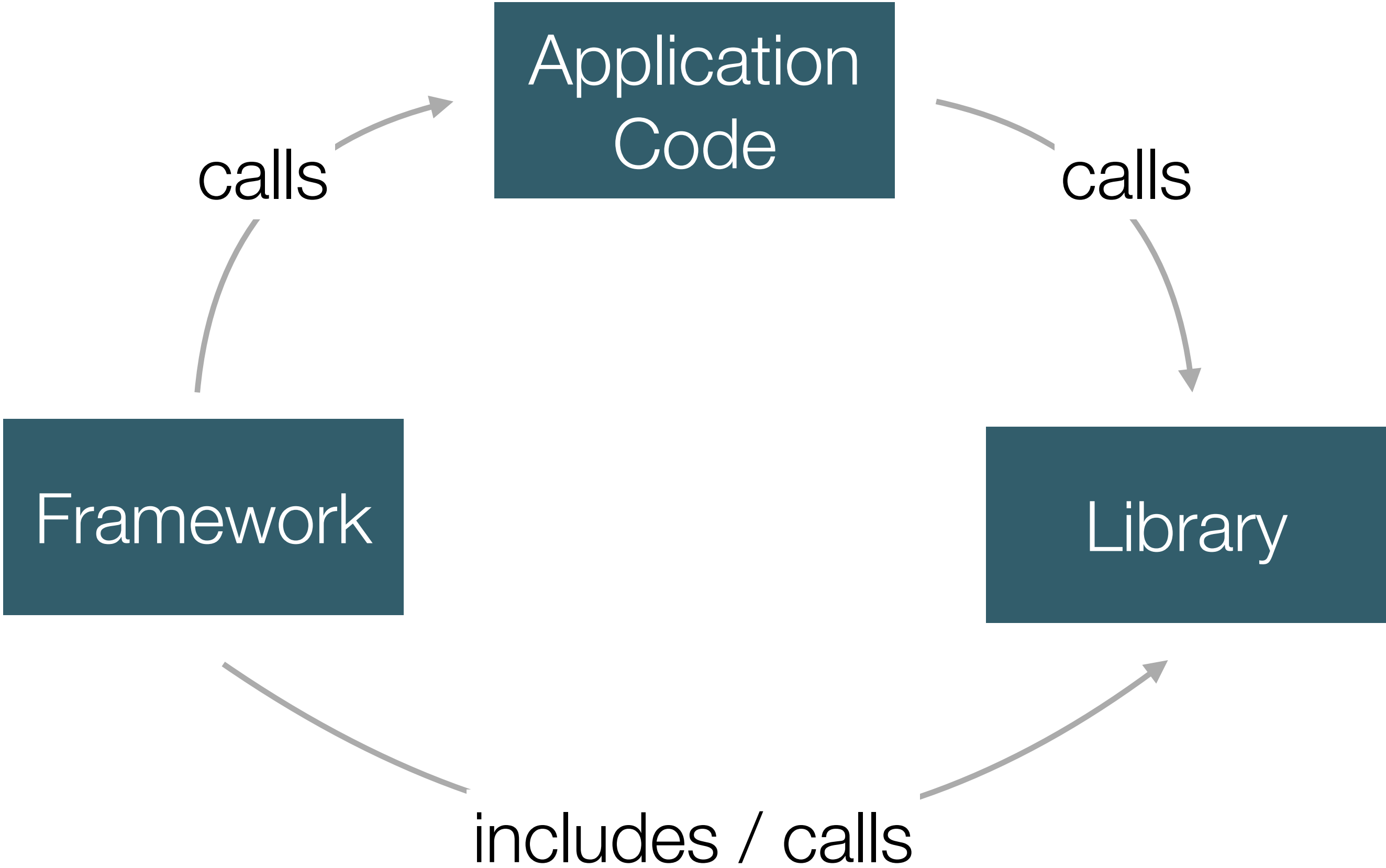


Web Development Frameworks

Software Frameworks

- Software frameworks provide a generic software foundation over which custom application-specific code can be written.
- Software frameworks often include multiple libraries, in addition to tools and rules on how to structure and use these components.
- Libraries are used by the application-specific code to support specific features.
- Frameworks control the application flow and call application-specific code.

Frameworks and Libraries



Why Frameworks

- **Advantages?**

- Implementation speed
- Tested, proven solutions
- Access to expertise and off-the-shelf solutions
- Maintenance (i.e. updates, patches)

- **Disadvantages?**

- Reduced independence
- Lower performance
- Dependence on external entities
- Technological lock-in

Choosing Frameworks

- **What to consider?**
 - Team expertise on language, libraries and framework
 - Existing code base
 - Licensing model
 - Maturity
 - Community support
 - ...

Web Development Frameworks

- Web development frameworks are designed to support the development of web applications, providing generic and integrated solutions to common use cases.
- These frameworks provide tools, libraries, and rules to address common tasks. Integration is central in contrast to the use of individual libraries.
- Currently, there is a big and diverse ecosystem of libraries and frameworks to support both backend and frontend development.
- Examples of libraries: jQuery, Bootstrap, Twig, PEAR DB.
- Examples of frameworks: ReactJS, Vue.js, Ruby on Rails, Django, Laravel.

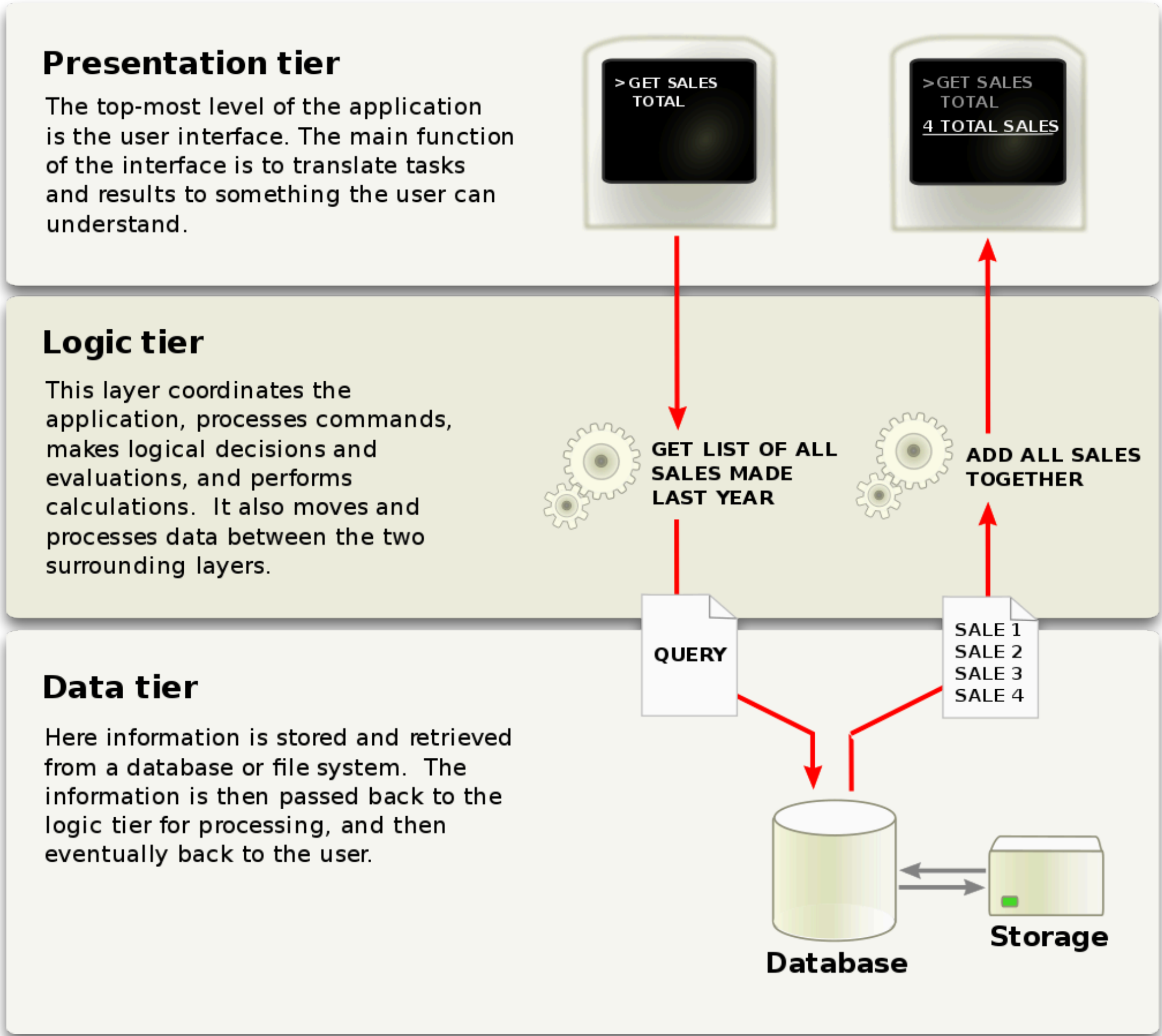
History

- In 1993, the Common Gateway Interface (CGI) was designed to enable the communication between browsers and applications, i.e. "programs as web pages". First popular web development libraries were designed to support CGI.
- During the 90s there was a strong development of libraries targeted at common use cases, e.g. outputting HTML (templating), accessing data, interface with mail, managing user input, etc.
- In the early 2000s, the first modern full-stack server-side frameworks for web development started to appear, e.g. Drupal, Ruby on Rails, Symfony, Django.
- The growth of client-side supported web applications led to the development of multiple frontend libraries, e.g. jQuery, Mustache.
- More recently, full-stack client-side frameworks for web development emerged, e.g. ReactJS, AngularJS, Vue.js.

Three-tier Architectures

- The Three-tier Architecture is a software architecture pattern commonly adopted in web applications.
 - Presentation tier — interface with the user; process user interactions; present views to the user.
 - Logic tier — coordinate the application; decide on the application flow; process data; move data between the two other layers.
 - Data tier — manage information; persist information; handle consistency; translate between physical models and conceptual model.

Three-tier Architecture



Source: https://en.wikipedia.org/wiki/Multitier_architecture

Common Problems in Web Development

- **Common problems in web applications development?**

- Handle access requests.
- Manage user interface components.
- Access and manipulate data.

- Session and authentication management.
- Access control management.
- Validating inputs.
- Error handling.
- Interacting with email systems.
- ...

Server-side Frameworks

- Frameworks can be grouped in three types:
 - **Micro Frameworks**, focused on routing HTTP request to a callback, commonly used to implement HTTP APIs.
 - **Full-Stack Frameworks**, feature-full frameworks that includes routing, templating, data access and mapping, plus many more packages.
 - **Component Frameworks**, collections of specialized and single-purpose libraries that can be used together to make a a micro- of full-stack framework.

Framework Components

- Core components
 - Request Routing, match incoming HTTP requests to code.
 - Template Engine, structure and separate presentation from logic.
 - Data Access, uniform data access, mapping and configuration.
- Common components
 - Security, protection against common web security attacks.
 - Sessions, session management and configuration.
 - Error Handling, capture and manage application-level errors.
 - Scaffolding, quickly generate CRUD interfaces based on data model.
 - ...

Request Routing

- Request routing maps HTTP access requests to specific functions.
- URL design is handled independently from application code using request routing.
- Clean URLs (aka "friendly URLs") are an important part of a web application: usability, SEO, technology independence, etc.
- https://sigarra.up.pt/feup/pt/cur_geral.cur_view?pv_curso_id=742
- <https://sigarra.up.pt/feup/pt/curso/mieic>

Laravel Example

```
Route::get('posts/{post}/comments/{comment}', function ($postId, $commentId) {  
    //  
});
```

```
Route::get('user/{name}', function ($name) {  
    //  
})->where('name', '[A-Za-z]+');
```

```
Route::get('user/{id}', function ($id) {  
    //  
})->where('id', '[0-9]+');
```

```
Route::get('user/{id}', function ($id) {  
    return 'User '.$id;  
});
```

Template Engines

- Many of the first libraries for web development were template engines.
- Focused on the separation between presentation code and logic code.
- There are many independent libraries. Frameworks either use existing libraries or develop their own system.
- Notable solutions: Smarty (PHP), Blade (PHP), Jinja (Python), mustache (*).

Laravel Example (Blade)

```
<html>
  <head>
    <title>App Name - @yield('title')</title>
  </head>
  <body>
    @section('sidebar')
      This is the master sidebar.
    @show

    <div class="container">
      @yield('content')
    </div>
  </body>
</html>
```

```
@if (count($records) === 1)
    I have one record!
@endif
@elseif (count($records) > 1)
    I have multiple records!
@else
    I don't have any records!
@endif
```

```
Route::get('greeting', function () {
    return view('welcome', ['name' => 'Samantha']);
});
```

Data Access

- The data access layer can be managed with different levels of automatism and control.
- Data layer independence can be achieved using libraries that provide a uniform access to different technologies. Example: PHP PDO.
- A higher level of coupling can be achieved by providing access to data through a mapping between the underlying data structures and an object layer (ORM). Example: ActiveRecord (Laravel, RoR).
- This coupling between the data layer and the application's data model can imply database migrations.

Laravel Example (Eloquent)

```
$user = DB::table('users')->where('name', 'John')->first();  
  
echo $user->name;
```

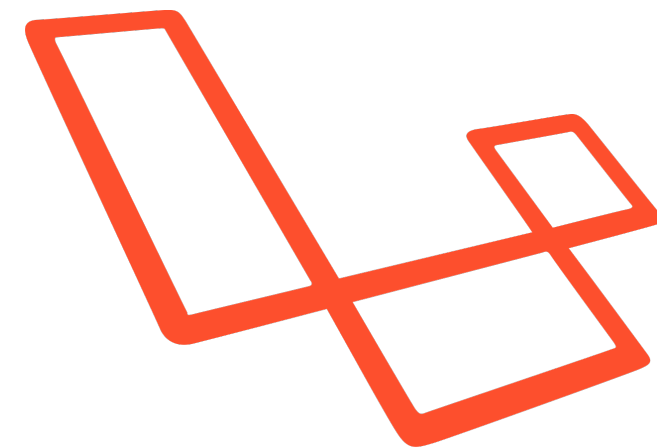
```
$users = DB::table('users')->count();  
  
$price = DB::table('orders')->max('price');
```

```
$users = DB::table('users')  
    ->select(DB::raw('count(*) as user_count, status'))  
    ->where('status', '<>', 1)  
    ->groupBy('status')  
    ->get();
```

Notable Web Frameworks

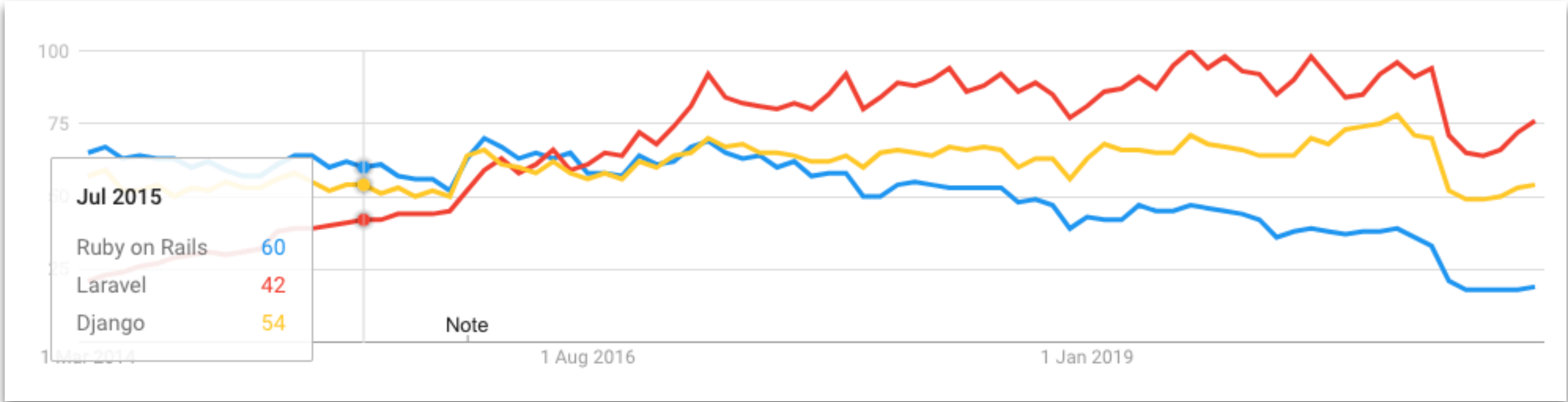


django



https://en.wikipedia.org/wiki/Comparison_of_web_frameworks

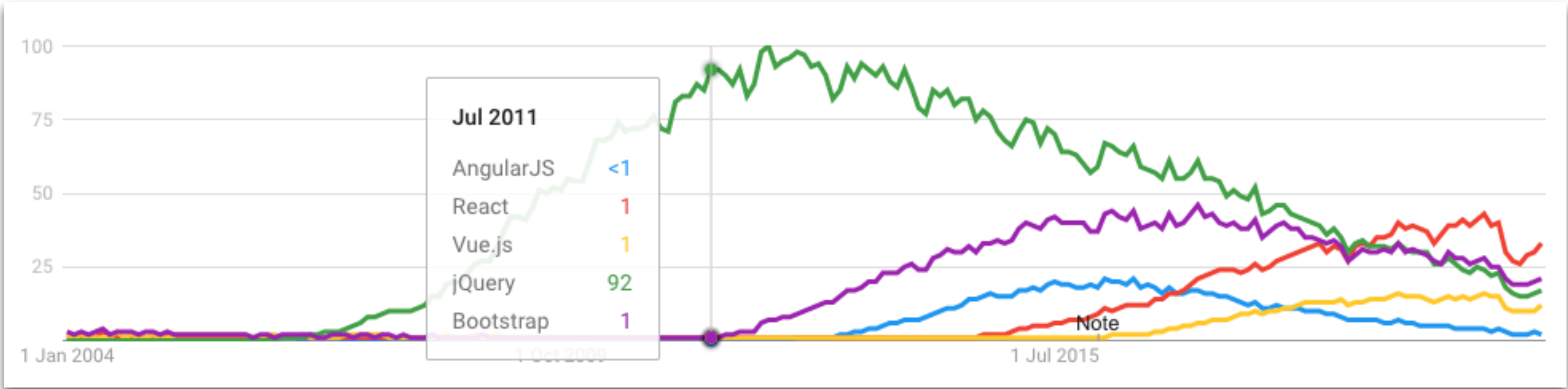
Trends



Client-side Frameworks

- Client-based web applications, rely on frontend technologies to manage both the Presentation and the Logic layers. In a nutshell, the current web page is dynamically manipulated instead of loading a new page.
- Single-page web applications (SPA) are an extreme example of this architecture. All application views are mapped to a single document.
- Frontend web frameworks typically adopt an MVC pattern (or variants) to support this architectural style, including: data-bindings, templates, routing.
- Most notable solutions: AngularJS (Google), React (Facebook), Vue.js.

Trends



Other Topics

- Content Management Systems, e.g. WordPress.
- Static-site generators.
- ...

References

- MDN, Server-side web frameworks, developer.mozilla.org/docs/Learn/Server-side/First_steps/Web_frameworks
- MDN, Understanding client-side JavaScript frameworks, developer.mozilla.org/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks
- Laravel, laravel.com