



## SOFTWARE ARCHITECTURE

2024-25

Jose Emilio Labra Gayo

Pablo González

Irene Cid

Diego Martín



Escuela de  
Ingeniería  
Informática



Universidad de Oviedo

## Lab 5

React

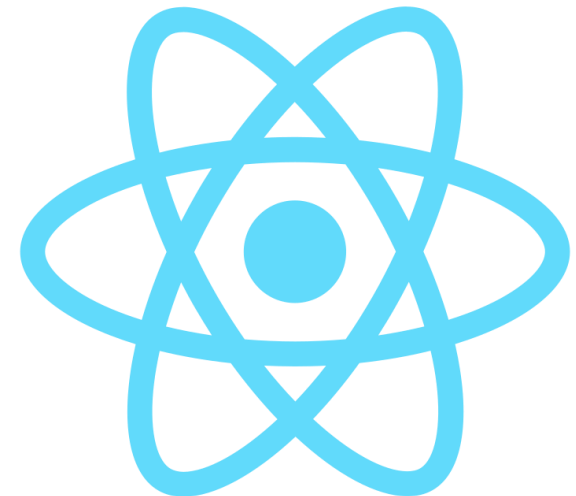
Building automation

Dependency management

# What is React.js?

React = JavaScript library for building user interfaces for the web as well as mobile applications

- Open source
- Created by Facebook (Meta)
- Based on components



# Why React?

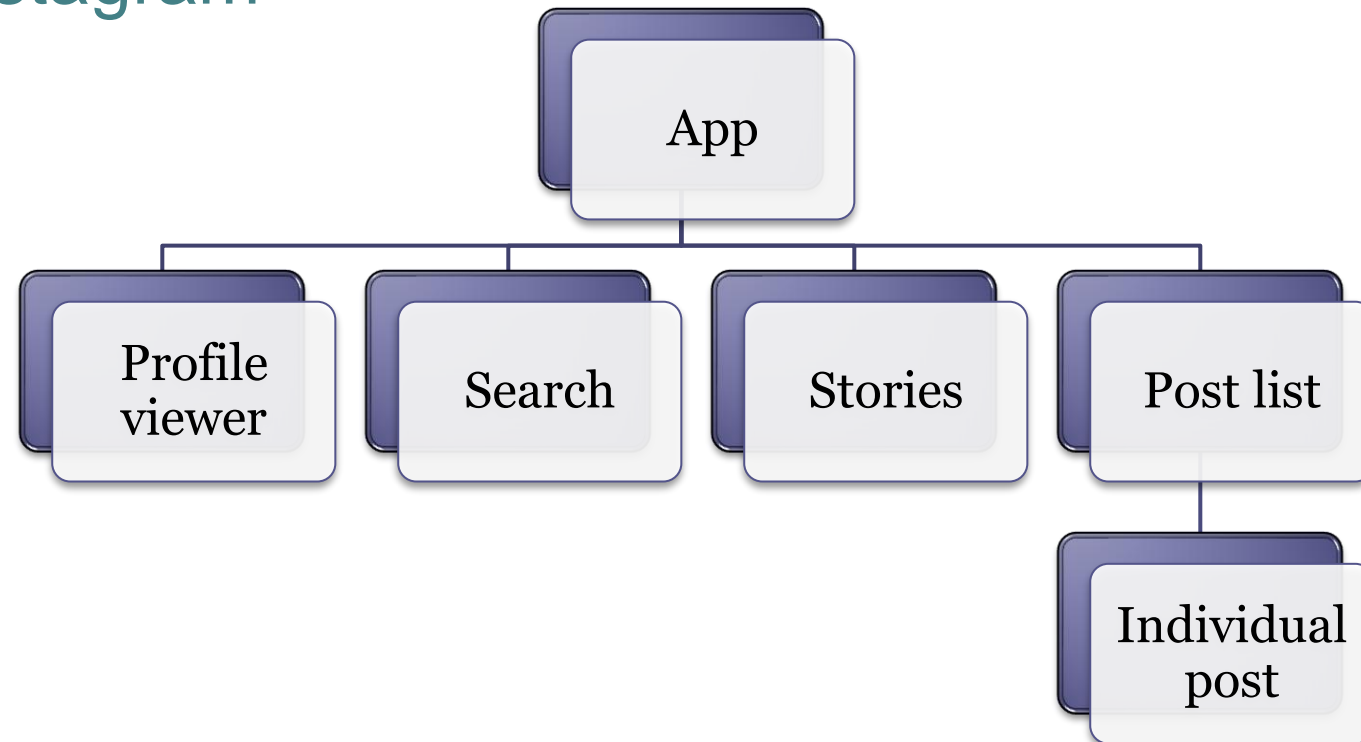
- Some reasons to use React:
  - Popularity
  - Simplicity and easy to learn
  - Reusable components
  - Native approach (React Native)
  - Lots of resources and tools for development
  - Testability

# Components

Pages are modelled using components

A component is a part of the user interface

Example: Instagram



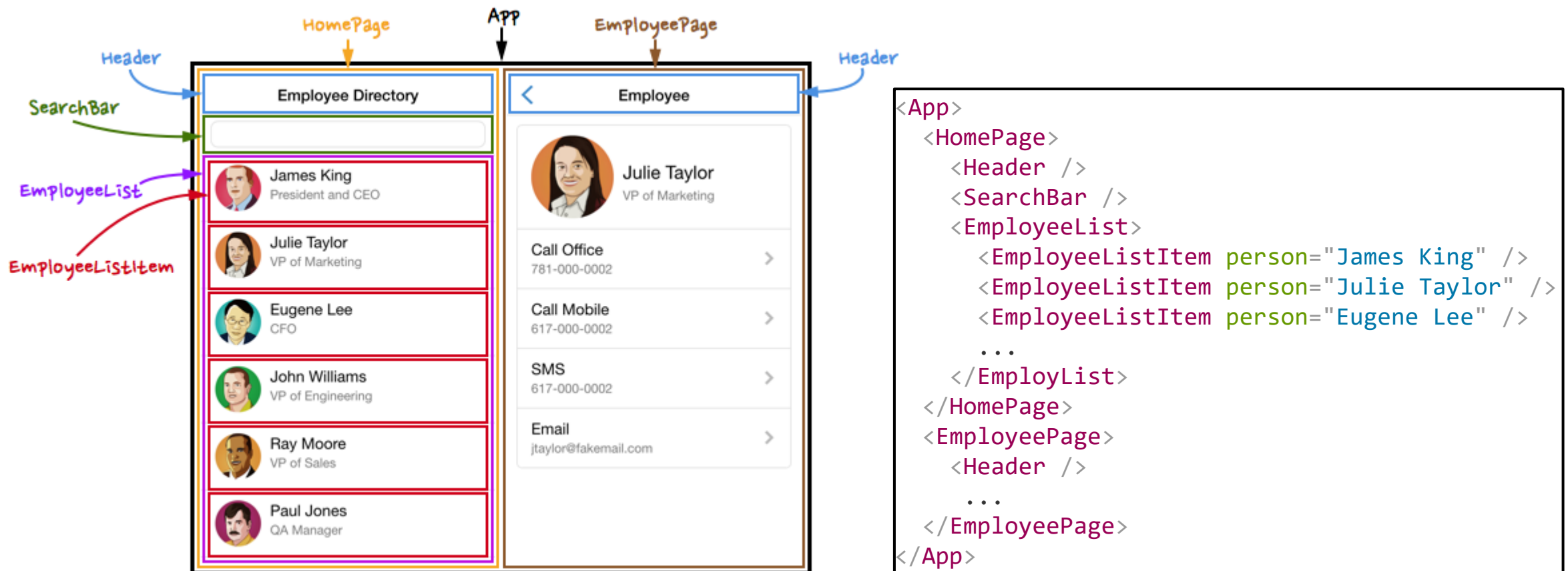
# Components

A component can be implemented as a JavaScript function (Hook)

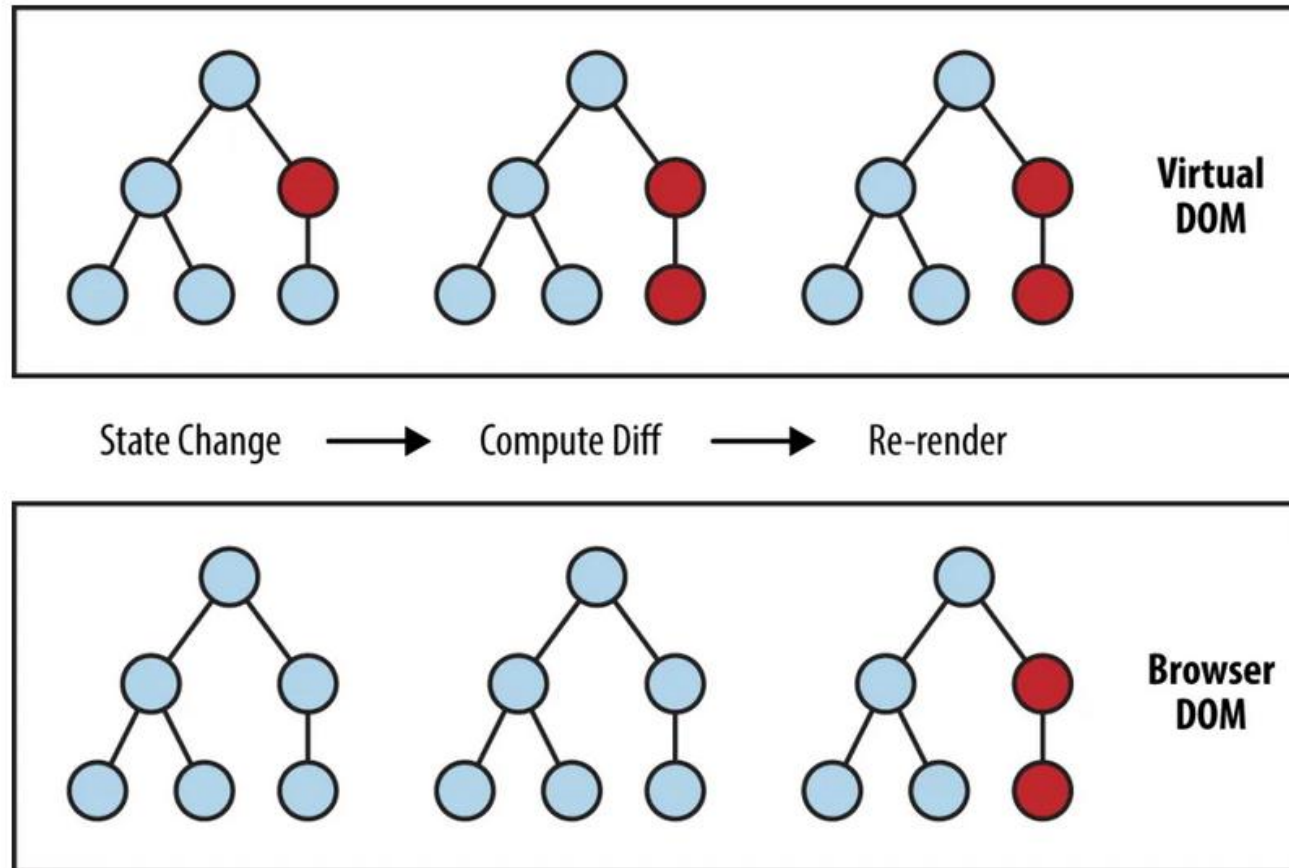
- It has a state
- And a render method that controls what is displayed in UI
- When the state changes, react updates the element and its child's in memory
- This element representation in memory is called Virtual DOM

React **reacts**  
to changes

# Example of components



# Virtual DOM



# JSX (Javascript XML)

JSX is a syntax extension to Javascript

Looks like HTML

JSX files are compiled to Javascript

JSX

```
const element = (  
  <h1 className="greeting">  
    Hello, world!  
  </h1>  
);
```



Javascript

```
const element = React.createElement(  
  'h1',  
  {className: 'greeting'},  
  'Hello, world!'  
);
```



# Components defined with Hooks

Hooks introduced in version 16, replace classes by functions

- In the following example, we use the `useState` hook to handle the name changes in the app
- Once the button is clicked, the state is changed, the virtual DOM updated, and the page is automatically refreshed

```
const App = () => {
  const [name, setName] = useState('World');
  return (
    <div className="App">
      <h1>Hello, {name}!</h1>
      <button onClick={() => setName('James')}>
        Click me to change the name
      </button>
    </div>
  );}
```

# Additional documentation

## Exercises about React state (in Spanish)

[Ej1](#) Create a counter

[Ej2](#) Complex states(objects)

[Ej3](#) Different handlers()

[Ej4](#) Adding elements to an array

[Ej5](#) Change a component's behaviour (background color)

# Additional documentation

## Exercises rendering in React

[Ej1](#) Array rendering

[Ej2](#) Refactoring

[Ej3](#) Adding elements to the array

[Ej4](#) Adding elements from a form

# Additional documentation

## Asynchronous programming

[Ej1](#) Fetch() -> Do an API request

[Ej2](#) useEffect()

[Ej3](#) Conditional rendering

[Ej4](#) One Refactoring

[Ej5](#) Requests using axios library

# Additional documentation

Exercises using Typescript + React

[Ej1](#) Counter with typescript

[Ej2](#) 2nd exercise

[Ej3](#) Example of an interface

# Additional documentation

## More links

Course [Bootcamp Fullstack](#)

[First Node.js conference](#) by Ryan Dahl

Vite (<https://vitejs.dev/>)

# Software builders

## Tasks

### Compilation

From source code to binary code

### Packaging

Dependency management and integration

Also called linking

### Test execution

### Deployment

Documentation creation / *release notes*

# Building automation

Automatize building tasks

Objectives:

- Avoid errors (minimize “*bad buildings*”)

- Eliminate redundant and repetitive tasks

- Manage complexity

- Improve the product quality

- Store a building and release history

- Continuous integration

- Save time and money



# Automation tools

- Makefile (C)
- Ant (Java)
- Maven (Java)
- Npm (Node.js)
- SBT (Scala, JVM languages)
- Gradle (Groovy, JVM languages)
- rake (Ruby)
- cargo (Rust)
- etc.

# npm

## Node.js Package Manager

Initially created by Isaac Schlueter

Later became Npm inc.

3 things:

1. Website (<https://www.npmjs.com/>)  
User and organization management
2. Software registry  
Public/private packages
3. CLI application  
Dependency and task management  
Configuration file: package.json

# npm configuration: package.json

- Configuration file: package.json
  - npm init creates a simple skeleton
- Fields:

```
{
  "name":           "...mandatory...",
  "version":        "...mandatory...",
  "description":    "...optional...",
  "keywords":       "...",
  "repository":     {... },
  "author":         "...",
  "license":        "...",
  "bugs":           {...},
  "homepage":       "http://. . .",
  "main":           "index.js",
  "devDependencies": { ... },
  "dependencies":   { ... }
  "scripts":        { "test": " ... " },
  "bin":            {...},
}
```



Note: Yeoman provides fully featured scaffolding

# npm packages

Registry: <http://npmjs.org>

Installing packages:

2 options:

Local

```
npm install <packageName> --save (--save-dev)
```

Downloads <packageName> contents to node\_modules folder

Global

```
npm install -g <packageName>
```

Store the dependency in the package.json

Only for development



# npm dependencies

## Dependency management

Local packages are cached at `node_modules` folder

Access to modules through: `require('...')`

Global packages (installed with `--global` option)

Saved in `/usr/local/npm` (Linux OS)

Scoped packages marked by `@`

Referencing a module inside our project

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



# npm commands and scripts

npm contains lots of commands

start -> node service.js

test -> jest

ls lists installed packages

...

Custom scripts:

run <name>

More complex tasks in NodeJs

Gulp, Grunt



# npm packages

- Dependencies: Stored in package.json
- Package: Identified by name and version
- Rule for names:
  - Less than or equal to 214 characters.
  - Can't start with a dot or an underscore.
  - New packages must not have uppercase letters in the name.
  - The name ends up being part of a URL, an argument on the command line, and a folder name. Therefore, the name can't contain any non-URL-safe characters.



# npm semantic versioning

- Version of the package: Semantic versioning
  - Must be parseable by [node-semver](#)
- Ranges: Comparators which specify versions that satisfy the range
  - For example, the comparator `>=1.2.7` would match the versions 1.2.7, 1.2.8, 2.5.3, and 1.3.9, but not the versions 1.2.6 or 1.1.0.
  - More at <https://docs.npmjs.com/misc/semver>





# npm package.json fields

Reference: <https://docs.npmjs.com/files/package.json>

Fields:

- description
- keywords
- homepage: URL to Project homepage
- bugs: URL of project's issue tracker and/or the email address to which issues should be reported
- people fields: author, contributors.
  - The “author” is one person. “contributors” is an array of people. A “person” is an object with a “name” field and optionally “url” and “email”



# npm package.json fields

- **files:** An array of file patterns that describes the entries to be included when your package is installed as a dependency
- **file patterns follow a similar syntax to .gitignore, but reversed:**
  - Including a file, directory, or glob pattern (\*, \*\*/\*, and such) will make it so that file is included in the tarball when it's packed.
  - Omitting the field will make it default to ["\*"], which means it will include all files.



# npm files included

- Certain files are always included, regardless of settings:
  - package.json
  - README
  - CHANGES / CHANGELOG / HISTORY
  - LICENSE / LICENCE
  - NOTICE
  - The file in the “main” field



# npm package.json fields

- **main:** module ID that is the primary entry point to your program
  - This should be a module ID relative to the root of your package folder.
  - For most modules, it makes the most sense to have a main script and often not much else.
- **browser:** If the module is meant to be used client-side the browser field should be used instead of the main field.
  - This is helpful to hint users that it might rely on primitives that aren't available in Node.js modules (eg a window).



# npm package.json fields

- repository: the place where the code lives.

```
"repository": {  
  "type" : "git",  
  "url" : "https://github.com/npm/cli.git"  
}  
  
"repository": {  
  "type" : "svn",  
  "url" : "https://v8.googlecode.com/svn/trunk/"  
}
```



# npm package.json fields

- `config`: Used to set configuration parameters used in package scripts that persist across upgrades.

```
{  
  "name" : "foo" ,  
  "config" : { "port" : "8080" }  
}
```



# npm package.json fields

- dependencies: Dependencies are specified in a simple object that maps a package name to a version range.
  - The version range is a string which has one or more space-separated descriptors.
  - Version ranges based on semantic versioning:
    - See <https://docs.npmjs.com/misc/semver>



# npm package.json fields

- **devDependencies:** Dependencies required to develop the application such as unit tests.
- **URL dependencies:**
  - You may specify a tarball URL in place of a version range.
  - This tarball will be downloaded and installed locally to your package at install time.

```
<protocol>://[<user>[:<password>]@]<hostname>[:<port>][:][/]<path>[#<commit-ish> | #semver:<semver>]
```





# npm

- GIT URLs: Following form:

```
<protocol>://[<user>[:<password>]@]<hostname>[:<port>][:][/]<path>[#<commit-ish>|#semver:<semver>]
```

- Example

```
git+ssh://git@github.com:npm/cli.git#v1.0.27  
git+ssh://git@github.com:npm/cli#semver:^5.0  
git+https://isaacs@github.com/npm/cli.git  
git://github.com/npm/cli.git#v1.0.27
```



# Task Execution : Grup and Gulp

Execute JavaScript tasks:

- Compress images
- Package modules (webpack)
- Minimize js and css files
- Run tests
- Transcompile – babel.js

These tasks can be directly run with npm scripts or with Gulp and/or Grunt



# Task Execution : Grup y Gulp

- Grup:

- Module fs
- Installation:

```
npm install -g grunt
npm install -g grunt-cli
```

- package.json configuration

```
{  "name": "ASW",
  "version": "0.1.0",
  "devDependencies": {
    "grunt-contrib-jshint": "~0.10.0",
    "grunt-contrib-nodeunit": "~0.4.1",
    "grunt-contrib-uglify": "~0.5.0"
  }
}
```

- Gulp:

- Module stream
- Installation:

```
npm install --save-dev gulp
npm install -g gulp-cli
```

- gulpfile.js configuration

```
function defaultTask(cb) {
  // tasks
  cb();
}
exports.default = defaultTask
```



# Examples

Wrapper

```
module.exports = function(grunt) {  
  // CONFIGURE GRUNT  
  grunt.initConfig({  
    (pkg.name)  
    pkg: grunt.file.readJSON('package.json'),  
  });  
  grunt.loadNpmTasks('grunt-contrib-uglify');  
  grunt.registerTask('default', ['uglify']);  
};
```

Wrapper

```
gulp.task('jpgs', function()  
{ return gulp.src('src/images/*.jpg')  
  .pipe(imagemin({ progressive: true }))  
  .pipe(gulp.dest('optimized_images')); });
```



End