Universidad de Oviedo

School of Computer Science

EN
English

SOFTWARE ARCHITECTURE

Software Architecture

Lab. 12
Monitoring & profiling

2020-21

Jose Emilio Labra Gayo
Pablo González
Irene Cid
Paulino Álvarez

Monitoring and profiling

Monitoring: Observe the behaviour at runtime while software is running

- Dashboards
- Usually, after deployment

Profiling: Measure performance of a software while it is running

- Identify parts of a system that contribute to a performance problem
- Show where to concentrate the efforts
- Usually before deployment

Monitoring & profiling

Monitors an application while it is running

Records performance (CPU & memory usage)

JavaScript:

- Chrome (Timeline), Firefox Developer Edition (Performance tool)

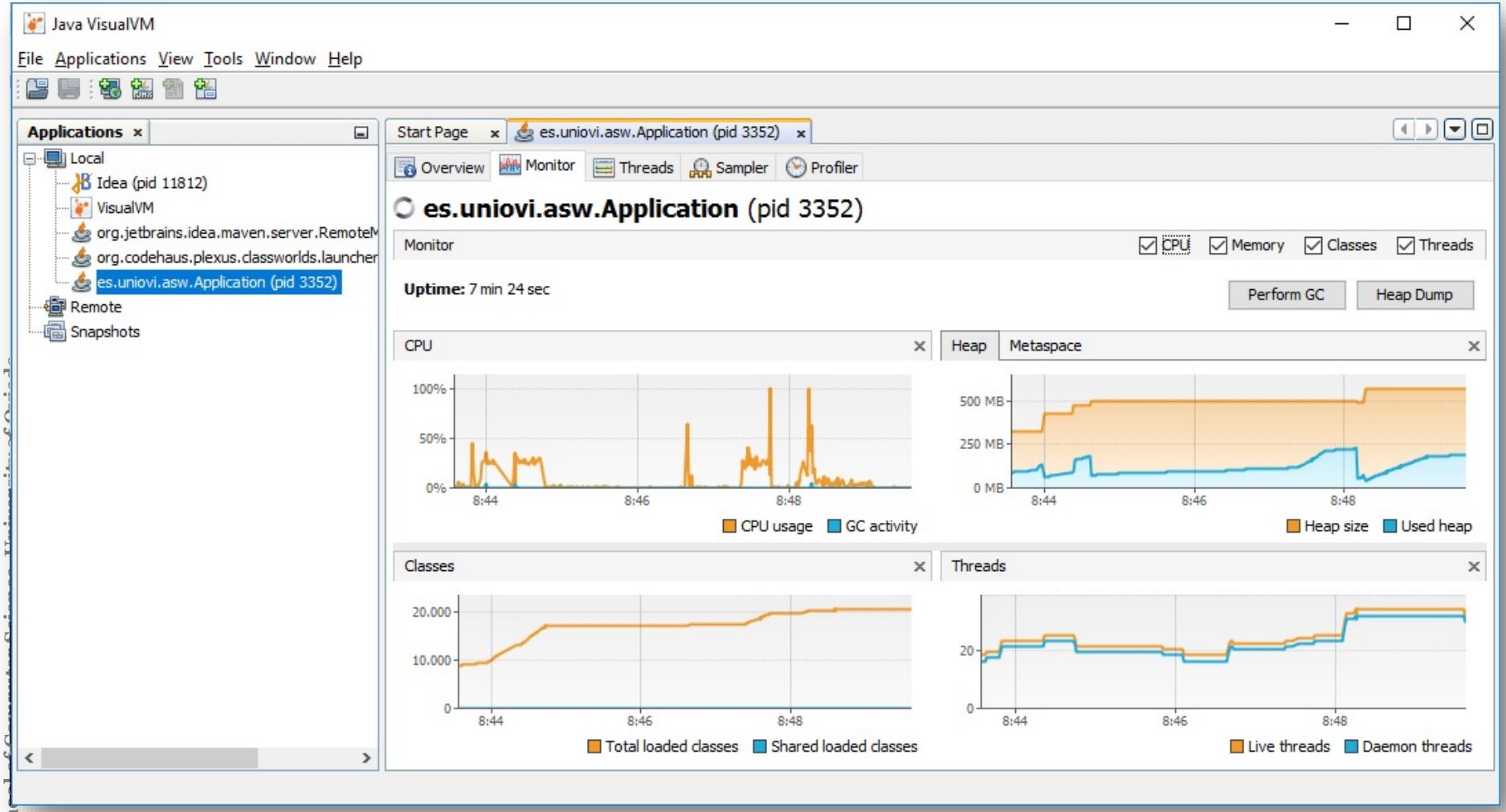
Server-side:

- JVisualVM, JProfiler, YourKit, JConsole
- Monitoring: Graphite, Datadog, Prometheus, Graphana

VisualVM

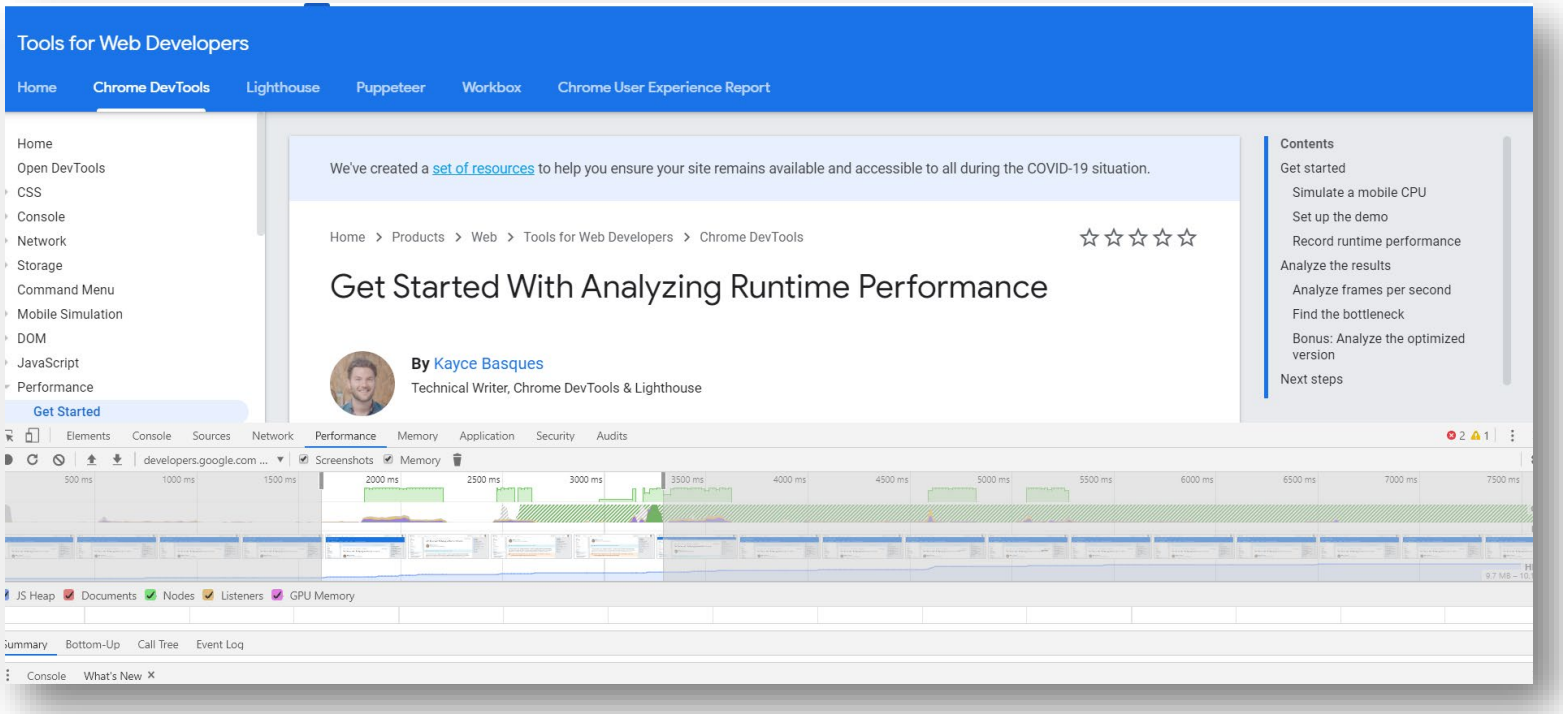
- https://visualvm.github.io/
- `jvisualvm`

Java/server JVisualVM



Browser: developer tools

Profiling/check performance



https://developers.google.com/web/tools/chrome-devtools/evaluate-performance

Example with Google Chrome

Incognito mode

At the top right, click the three dots and then New Incognito Window.

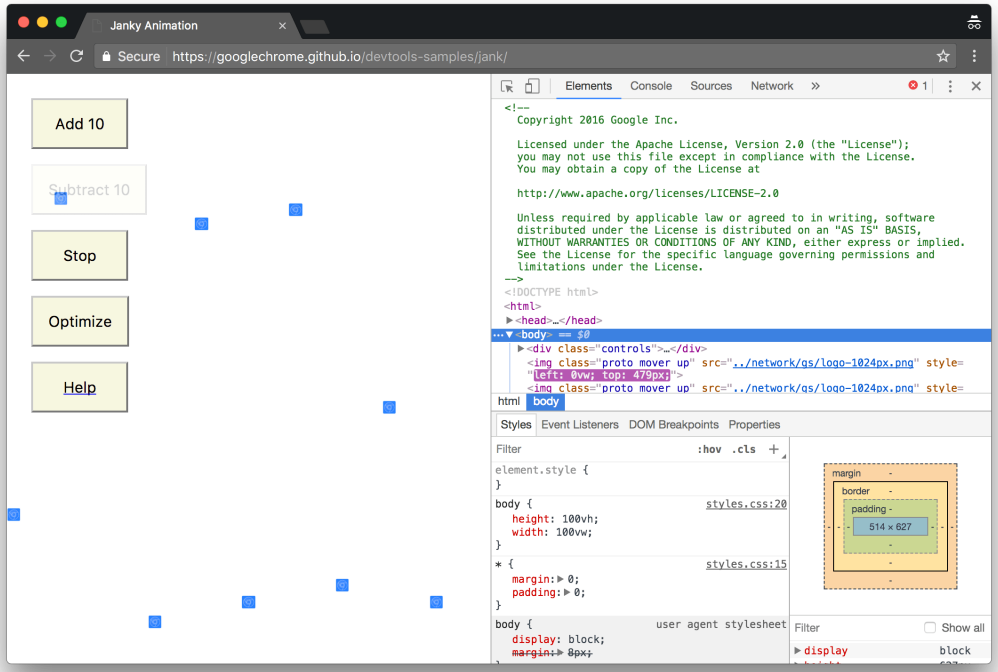
Windows, Linux, or Chrome OS: Press Ctrl + Shift + n.
Mac: Press ⌘ + Shift + n.

DevTools

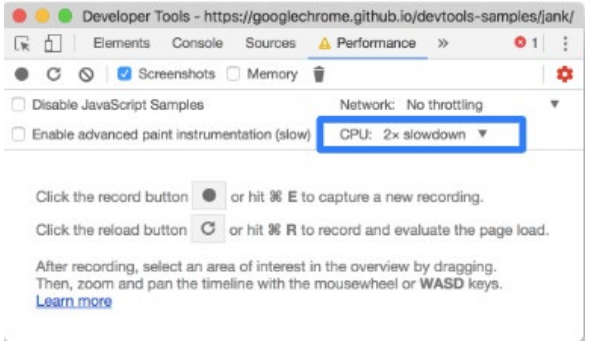
Windows, Linux: Control+Shift+I
Mac: Command+Option+I

Example with Google Chrome

https://googlechrome.github.io/devtools-samples/jank/



Performance>CPU>2 x Slowdown



Performance>Record

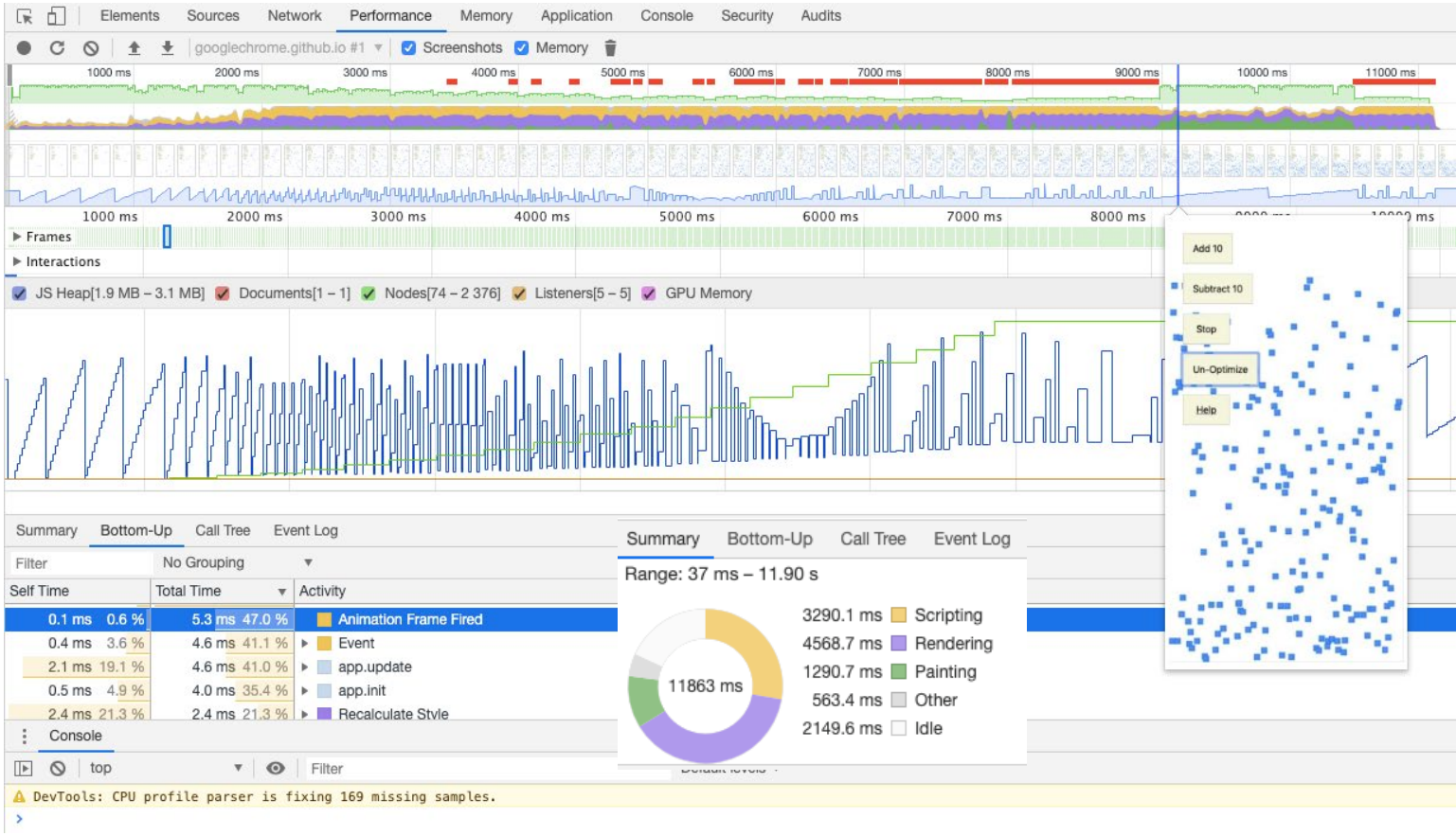
click Add 10 (20 times)

try Optimize / Un-optimize

Stop

Example with Google Chrome

Profile result:



Other tools for browser

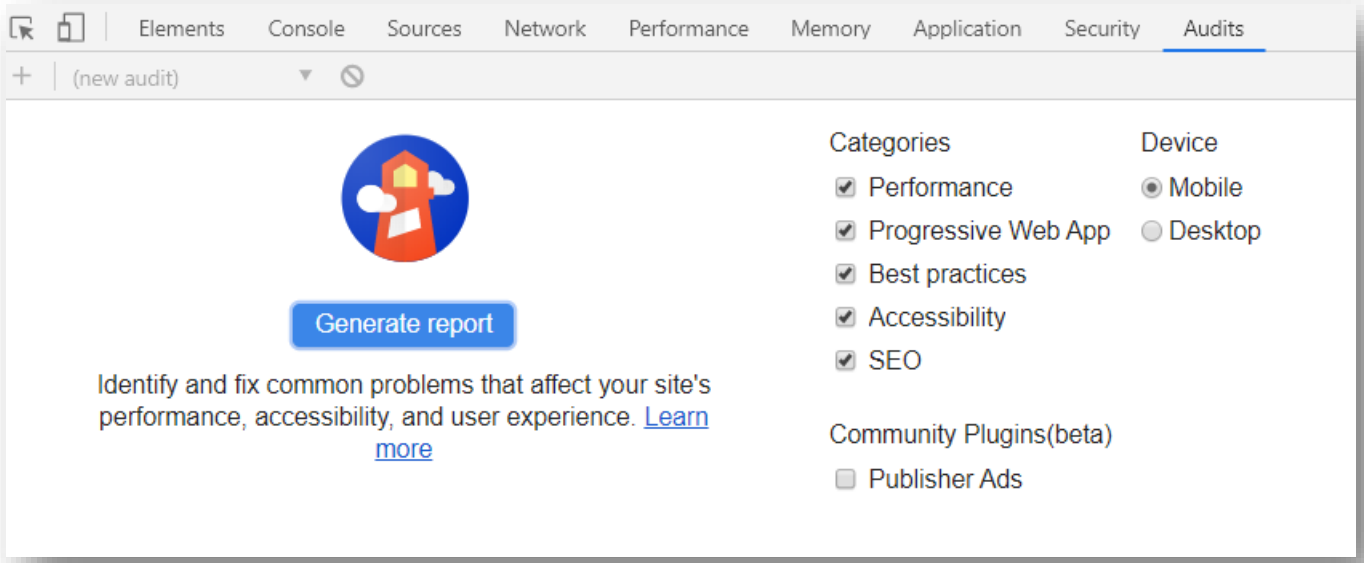
RAIL model:

Response, Animation, Idle, Load

https://developers.google.com/web/fundamentals/performance/rail

https://webpagetest.org/easy

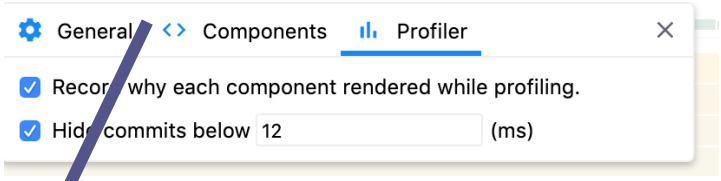
Lighthouse (with Chrome)

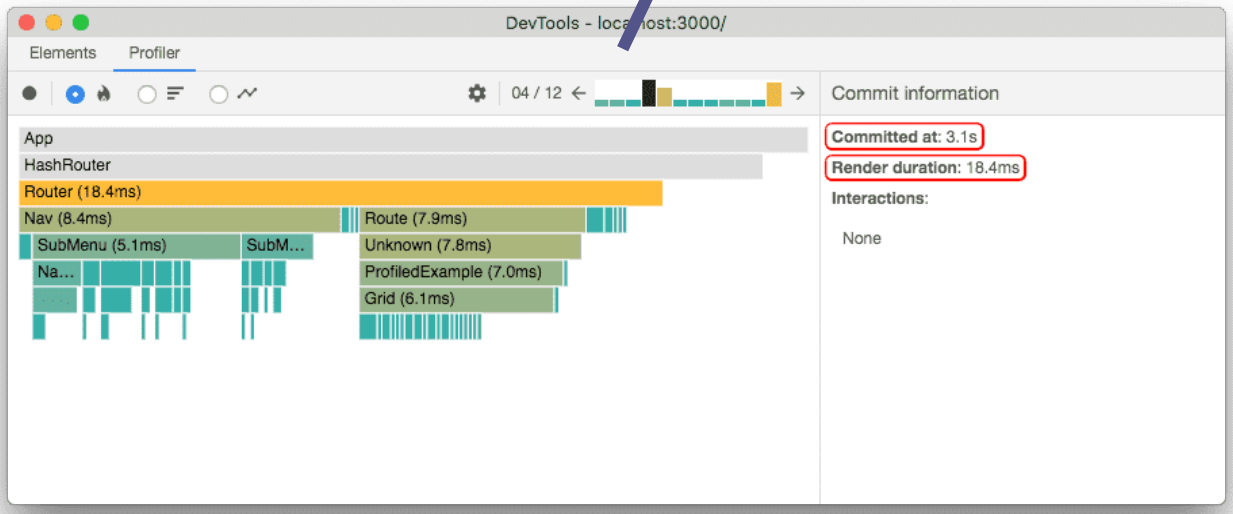


React Developer Tools

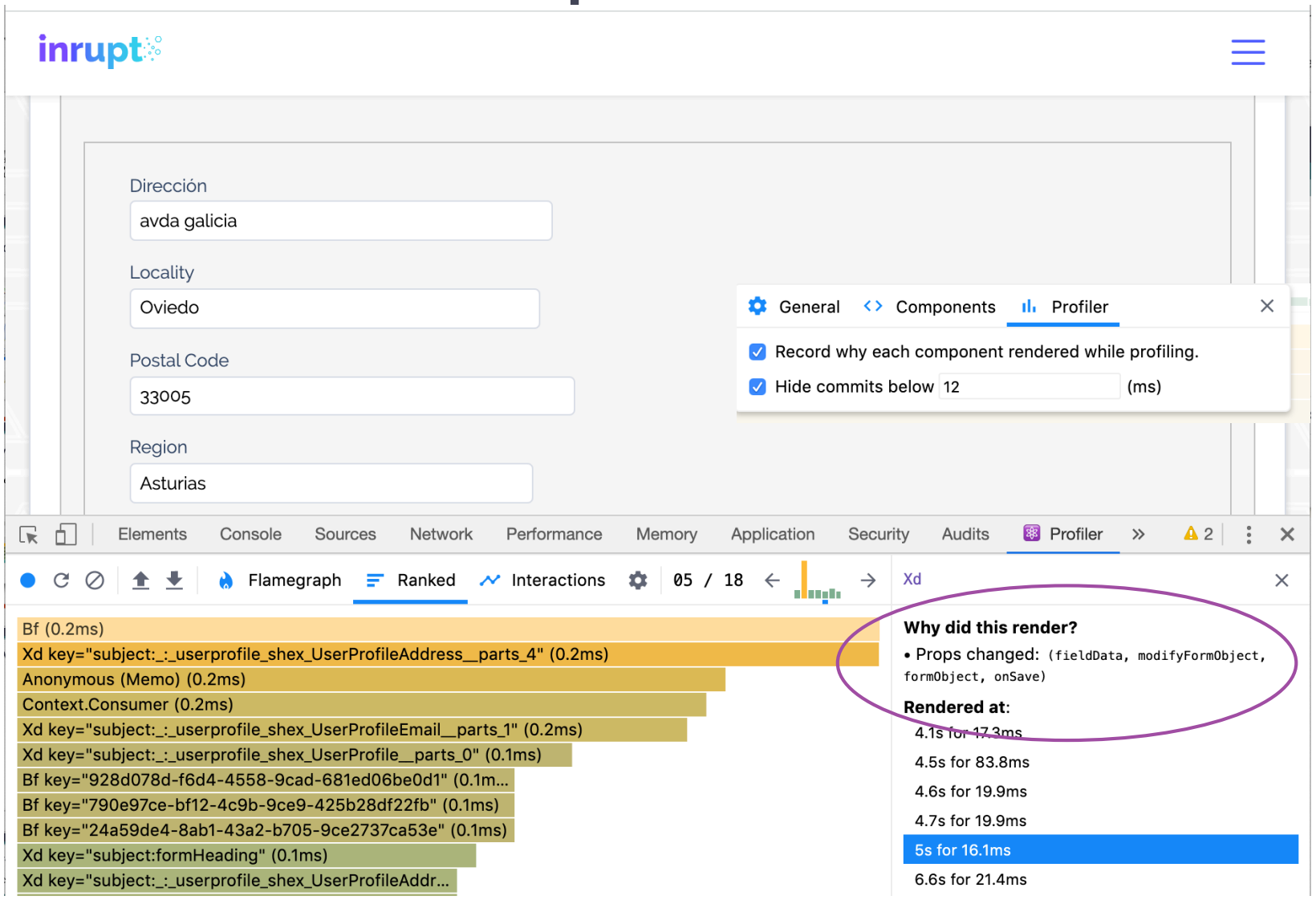
React works in two stages:

- Render
- Commit

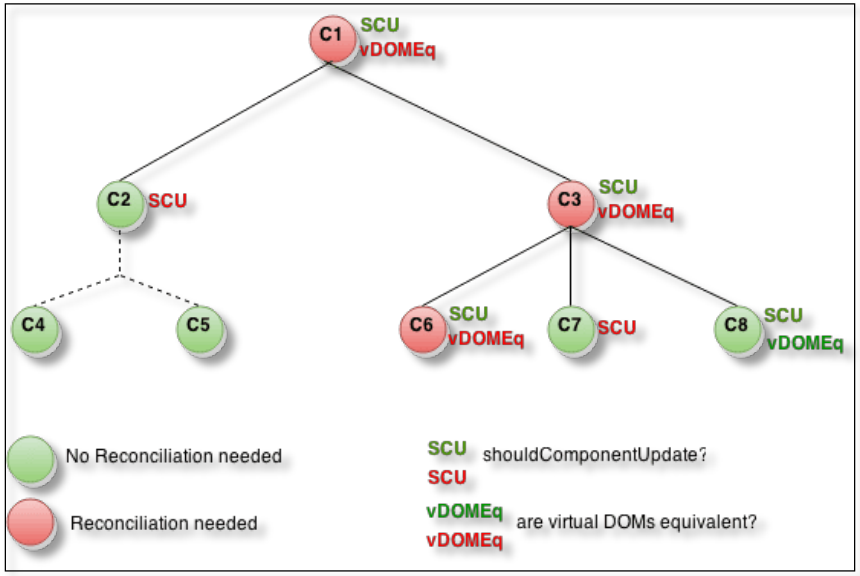




React Developer Tools



React DOM – Virtual DOM




```
class CounterButton extends React.PureComponent {
  constructor(props) {
    super(props);
    this.state = {count: 1};
  }

  render() {
    return (
      <button
        color={this.props.color}
        onClick={() => this.setState(state => ({count: state.count
+ 1}))}>
        Count: {this.state.count}
      </button>
    );
  }
}
```

```
  shouldComponentUpdate(nextProps, nextState) {
    if (this.props.color !== nextProps.color) {
      return true;
    }
    if (this.state.count !== nextState.count) {
      return true;
    }
    return false;
  }
```

Source: https://es.reactjs.org/docs/optimizing-performance.html

Server side monitoring

- Cloud platforms like Heroku provide monitoring solutions
  - Also available in Google Cloud, Amazon AWS.
  - In the case of Heroku, this solution is not free
- There is also the option to set up our own monitoring solution
- Which software to use: *Prometheus* and *Graphana*
- Guide: https://github.com/arquisoft/radarin_0/tree/master/restapi#monitoring-prometheus-and-grafana

Server side monitoring

- We need a library that can extract some metrics from our restapi
  - *npm install prom-client express-prom-bundle*

```
//Monitoring middleware
const metricsMiddleware = promBundle({includeMethod: true});
app.use(metricsMiddleware);
```

  - If we launch the restapi, in */metrics* we will be able to see some row data that would be used by Graphana to plot nice charts
  - We can choose which metrics to measure [doc]

Server side monitoring

- Graphana cannot use this data directly, we need Prometheus

  - Prometheus will retrieve the data exposed by the restapi and store it so it can be consumed by Graphana
  - We will work with a docker image [prom/prometheus] that can be configured through a single file

restapi > monitoring > prometheus > ! prometheus.yml

```yaml
global:
  scrape_interval: 5s
scrape_configs:
  - job_name: "example-nodejs-app"
    static_configs:
      - targets: ["restapi:5000"]
```

Server side monitoring

- How to configure Graphana
  - Graphana will use Prometheus as data source
  - We also have a docker image for running it [grafana/grafana]
  - We need to configure the datasource and the dashboard (which charts to plot)

Links

Monitoring & Profiling

Get Started With Analyzing Runtime Performance
https://developers.google.com/web/tools/chrome-devtools/evaluate-performance/

How to Use the Timeline Tool
https://developers.google.com/web/tools/chrome-devtools/evaluate-performance timeline-tool#profile-js

Presentation

Presentation Zen Garr Reynolds
https://www.presentationzen.com/
https://www.amazon.com/gp/product/0321811984