



Universidad de Oviedo



**SOFTWARE**  
**ARCHITECTURE**

# Software Architecture

Lab. 12

Monitoring & profiling

2022-23

Jose Emilio Labra Gayo  
Pablo González  
Irene Cid  
Cristian Augusto

# 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

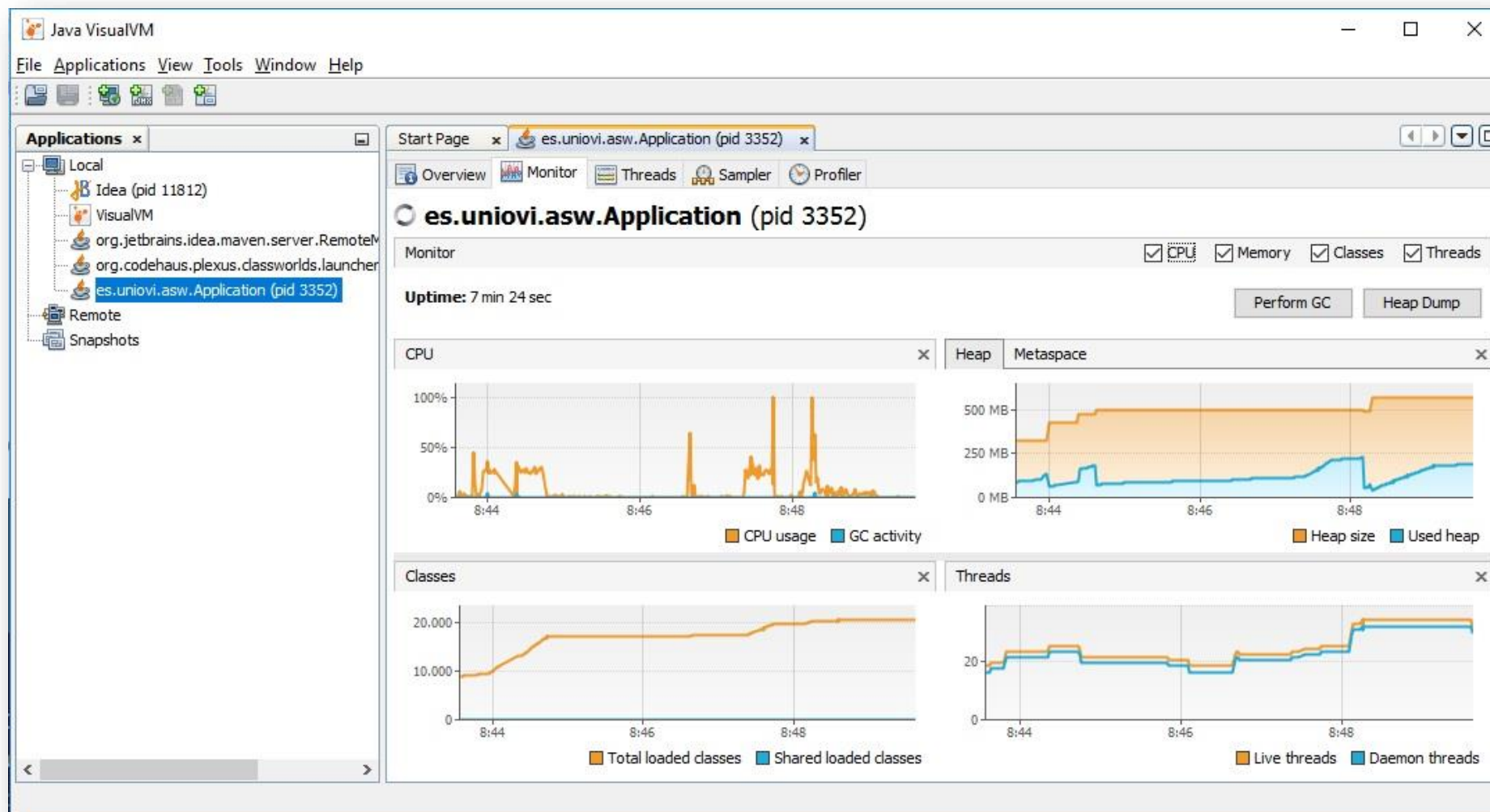
Monitoring: Graphite, Datadog, Prometheus, Graphana

VisualVM

<https://visualvm.github.io/>

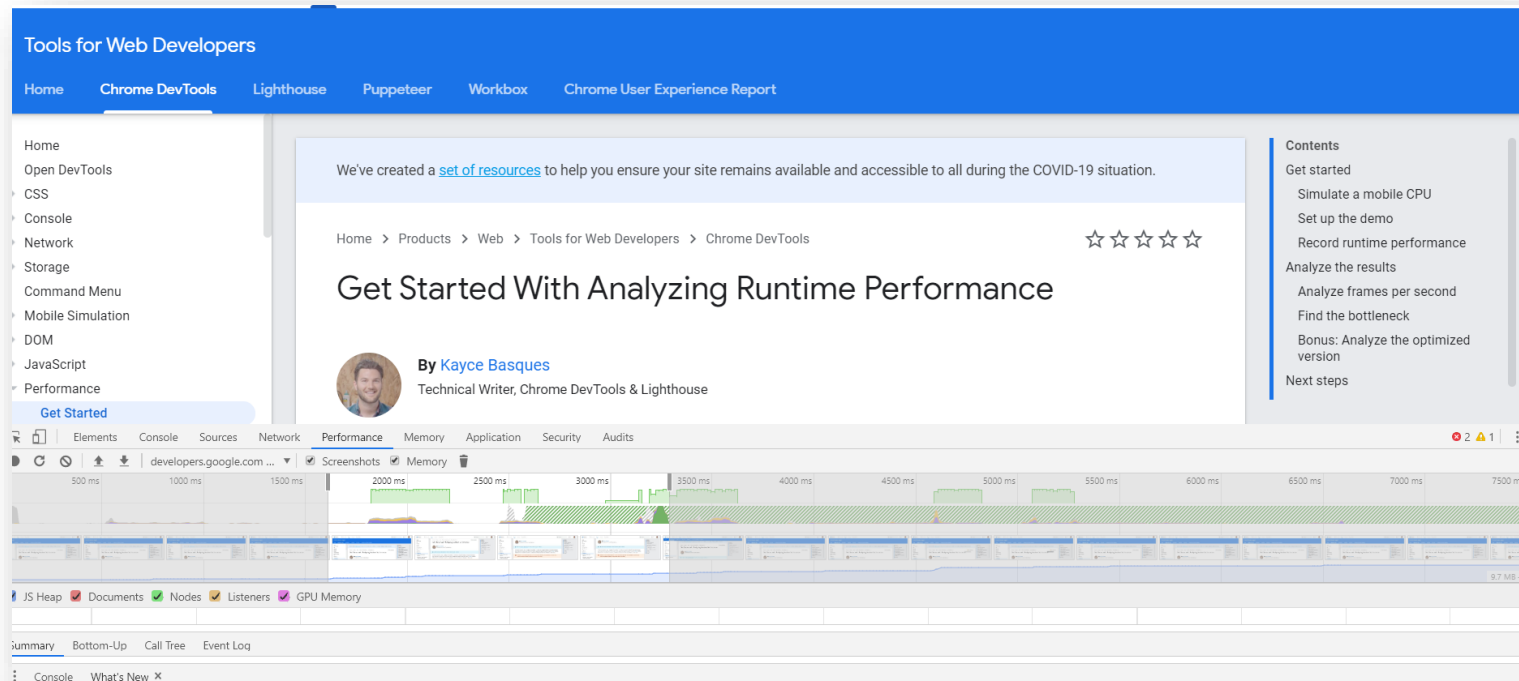
jvisualvm

# 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

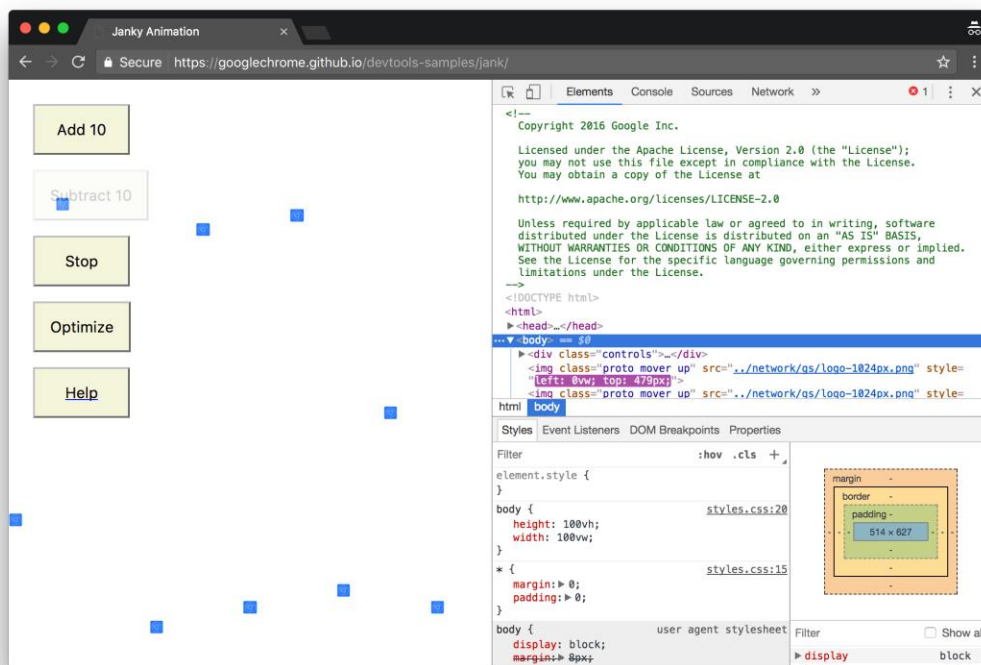
Windows, Linux: Control+Shift+I

Mac: Command+Option+I

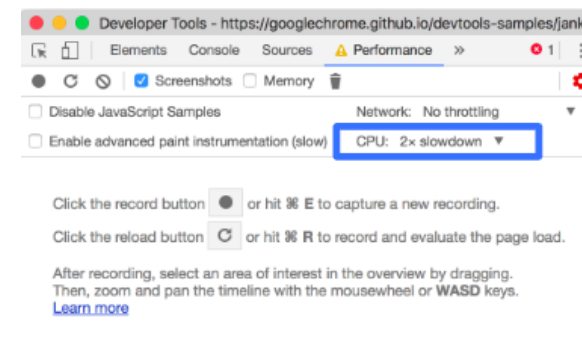


# Example with Google Chrome

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



Performance > CPU > 2 x Slowdown

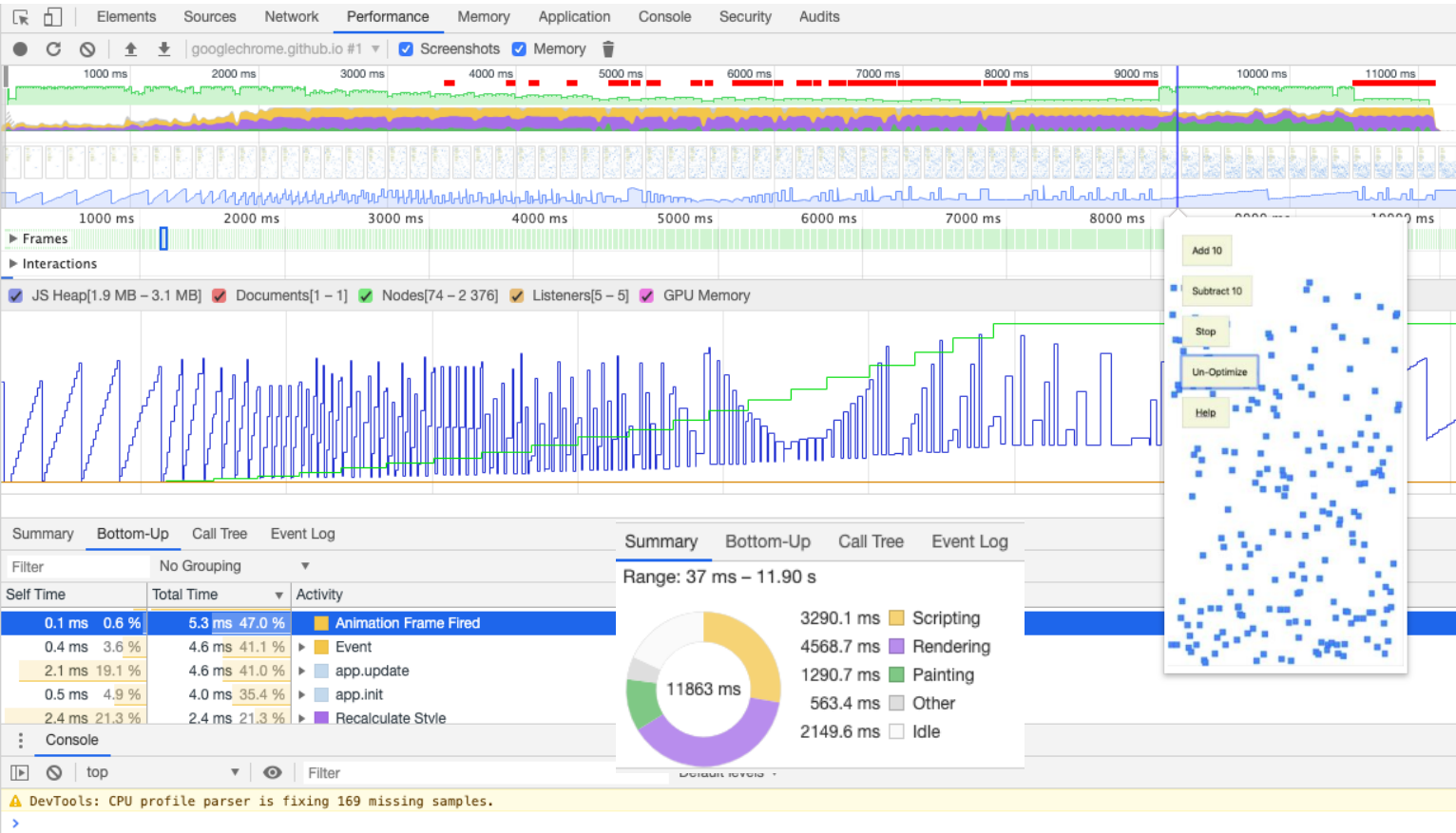


Performance > Record  
click Add 10 (20 times)  
try Optimize / Un-optimize  
Stop

# Example with Google Chrome

Profile result:

Frames per Second →  
CPU →



Bottleneck →



# Other tools for browser

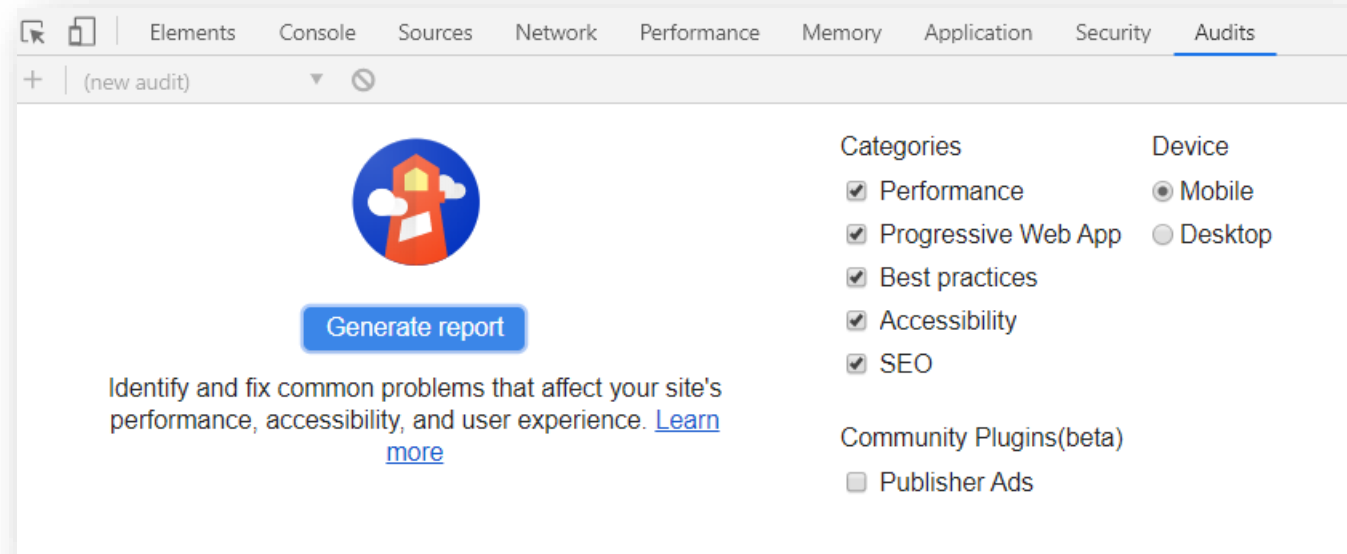
## RAIL model:

Response, Animation, Idle, Load

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

<https://webpagetest.org/easy>

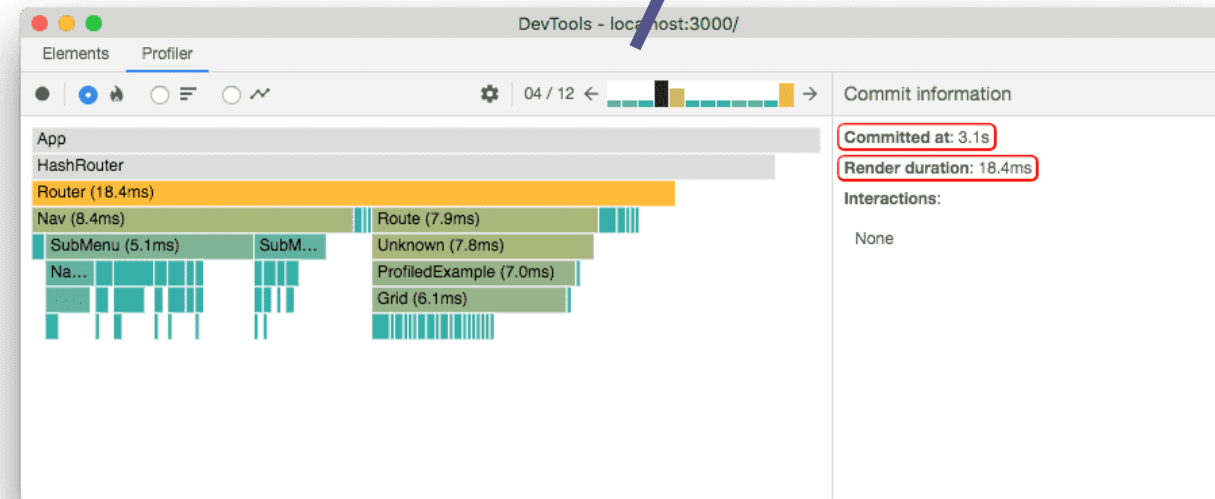
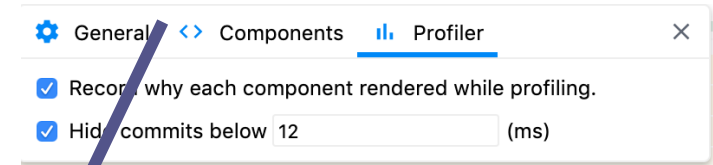
## Lighthouse (with Chrome)



# React Developer Tools

React works in two stages:

- Render
- Commit



The screenshot displays the React DevTools interface. The top section shows a form with fields for "Dirección" (avda galicia), "Locality" (Oviedo), "Postal Code" (33005), and "Region" (Asturias). Below the form, the "Profiler" tab is active, displaying a list of components and their render times. A purple circle highlights the "Why did this render?" section for the selected component, which lists props changes and render times.

Dirección  
avda galicia

Locality  
Oviedo

Postal Code  
33005

Region  
Asturias

General <> Components **Profiler**

☒ Record why each component rendered while profiling.  
☒ Hide commits below 12 (ms)

Elements Console Sources Network Performance Memory Application Security Audits **Profiler** » ⚠️ 2

Flamegraph Ranked Interactions 05 / 18

Xd

Bf (0.2ms)  
Xd key="subject:.\_UserProfileAddress\_\_parts\_4" (0.2ms)  
Anonymous (Memo) (0.2ms)  
Context.Consumer (0.2ms)  
Xd key="subject:.\_UserProfileEmail\_\_parts\_1" (0.2ms)  
Xd key="subject:.\_UserProfile\_\_parts\_0" (0.1ms)  
Bf key="928d078d-f6d4-4558-9cad-681ed06be0d1" (0.1m...  
Bf key="790e97ce-bf12-4c9b-9ce9-425b28df22fb" (0.1ms)  
Bf key="24a59de4-8ab1-43a2-b705-9ce2737ca53e" (0.1ms)  
Xd key="subject:formHeading" (0.1ms)  
Xd key="subject:.\_UserProfileAddr..."

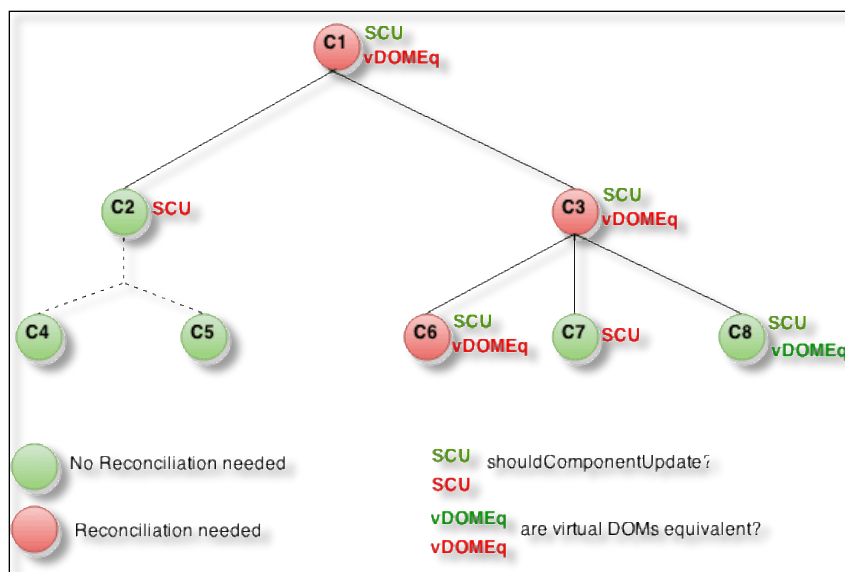
**Why did this render?**

- Props changed: (fieldData, modifyFormObject, formObject, onSave)

**Rendered at:**

- 4.1s for 17.3ms
- 4.5s for 83.8ms
- 4.6s for 19.9ms
- 4.7s for 19.9ms
- 5s for 16.1ms**
- 6.6s for 21.4ms

# 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;
}
  
```

# 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 *Grafana*
- Guide: [https://github.com/arquisoft/lomap\\_0/tree/master/restapi#monitoring-prometheus-and-grafana](https://github.com/arquisoft/lomap_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*

```
const metricsMiddleware:RequestHandler = 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](#)]



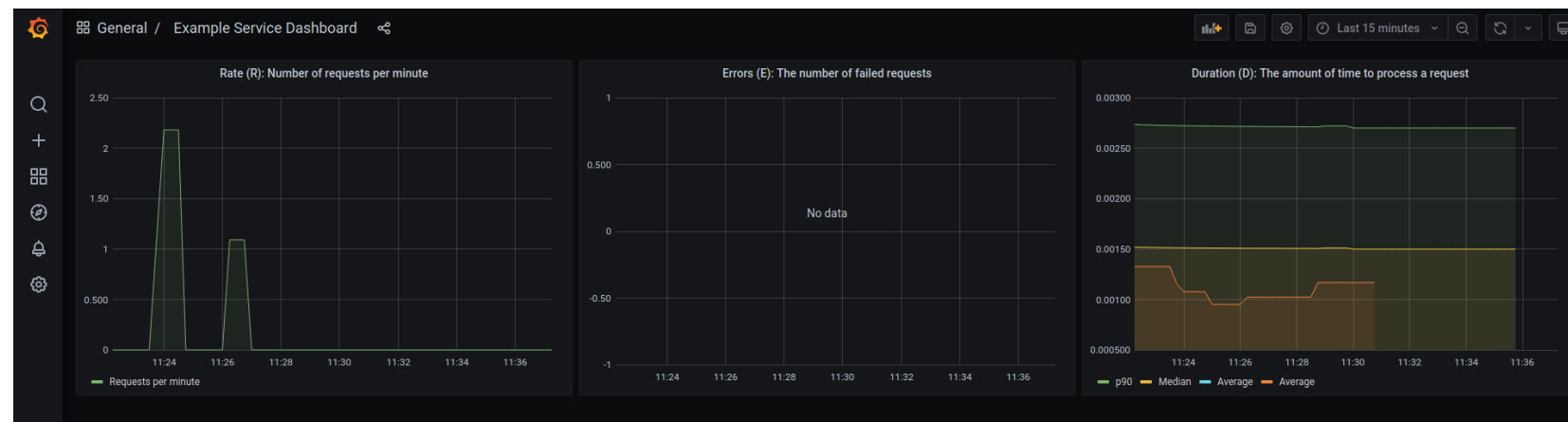
# Server side monitoring

- 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
1  global:
2    scrape_interval: 5s
3  scrape_configs:
4    - job_name: "example-nodejs-app"
5      static_configs:
6        - targets: ["restapi:5000"]
```

# Server side monitoring

- How to configure Grafana
  - Grafana 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](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>