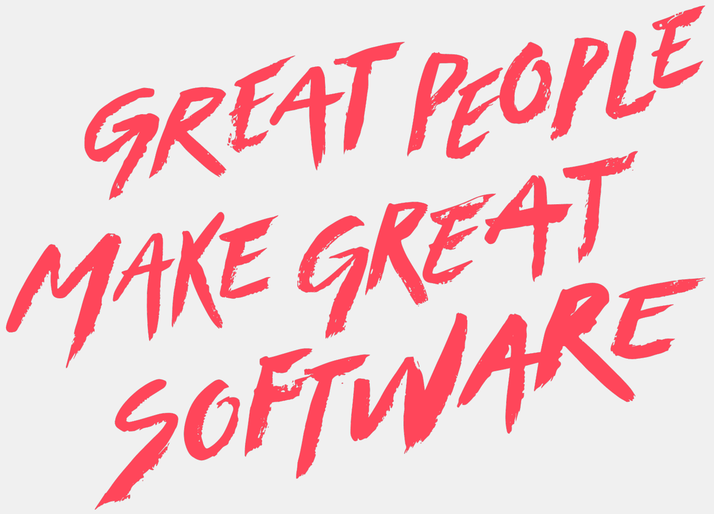
OPENSCD

EVERYTHING EVERYWHERE ALL AT ONCE



SOFTWARE ENGINEER

LEAD OF DOMAIN DEVELOPMENT

Tamás Russ

https://github.com/trusz

t.russ@sprinteins.com

AGENDA

1. What is OpenSCD
2. Brief History
3. Challenges
4. A technical approach

What is OpenSCD?

OPEN SUBSTATION COMMUNICATION DESIGNER

OpenSCD is a platform to enable fully digital substations - from design and commissioning through to maintenance and replacement.

IEC 61850 TOOL

It allows you to create IEC 61850 compatible designs.

PROGRESSIVE WEB APP

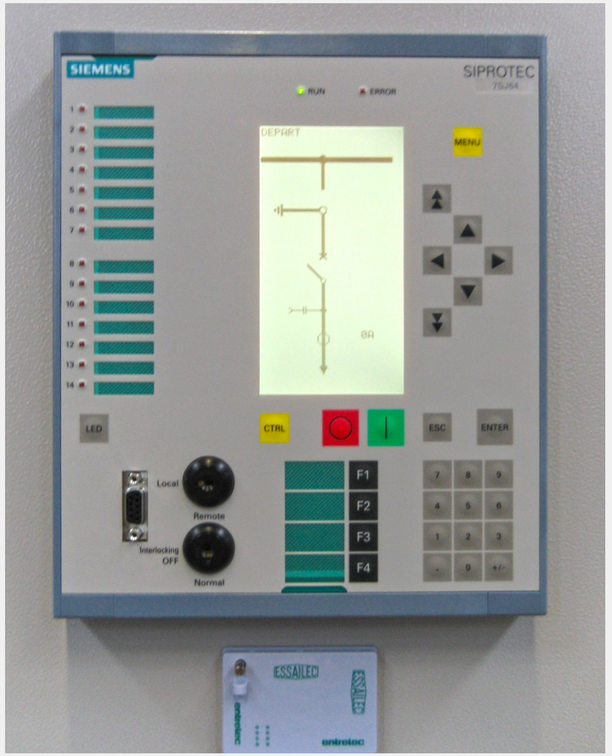
The app and all plugins are cached so after the initial the first use, everything will work offline too. No install is necessary.

PLATFORM

The plugin system enables flexibility and interchangeability.

Substation

ELECTRICAL SUBSTATION



IED

INTELLIGENT ELECTRONIC DEVICE

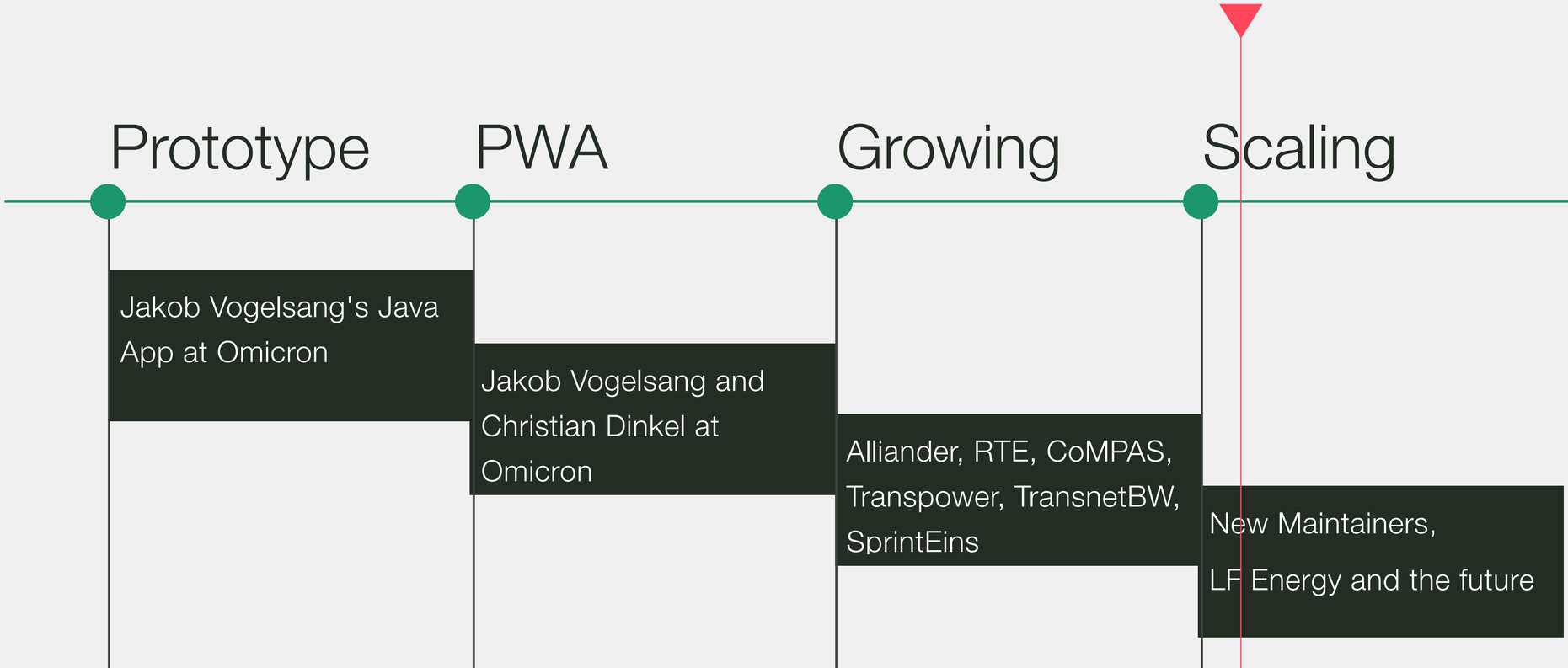
```
<SCL xmlns="http://www.iec.ch/61850/200
    <Header id="6c760f4f-3c93-445b-9793-
        <History>
                    when="2022-10-07T13
            <Hitem revision="" version="
                    when="2023-01-20T08
        </History>
    </Header>
    <Substation desc="Substation CoMPAS
        <Private type="ABBPCMInternalOb
                Private>
        <VoltageLevel desc="Voltage Lev
            <Private type="ABBPCMIntern
                    Private>
            <Voltage multiplier="k" uni
            <Bay name="BusBar A" sxy:x="
                <ConnectivityNode name="
            </Bay>
            <Bay name="BusBar B" sxy:x="
                <ConnectivityNode name="
            </Bay>
```

IEC 61850

COMMUNICATION PROTOCOL DEFINITIONS

Left: Photo by American Public Power Association on Unsplash
Middle: Zen 38, CC BY-SA 3.0, via Wikimedia Commons

Brief History



The Challenges

EVERYTHING

In order to foster a common way of designing and specifying digital substations we need to provide *all the tooling*.

EVERYWHERE

Our collective work needs to be *available everywhere and to everybod*y. Only this way can a standard work.

ALL AT ONCE

As in every *multi-stakeholder* project, every stakeholder would like to prioritise their own need.

Our Approach

WEB STANDARDS

We depend on web standards, like custom elements, for flexibility, performance and long-term maintainability.

PLUGINS

Almost everything is a plugin. It helps to customise for many use-cases.

DISTRIBUTION

We enable distributions so you can configure your version as you deem fit.

Web Standards

PROGRESSIVE WEB APP

We offer offline usage. We cache OpenSCD and plugins so the engineers can browse and design their digital substation

CUSTOM ELEMENTS

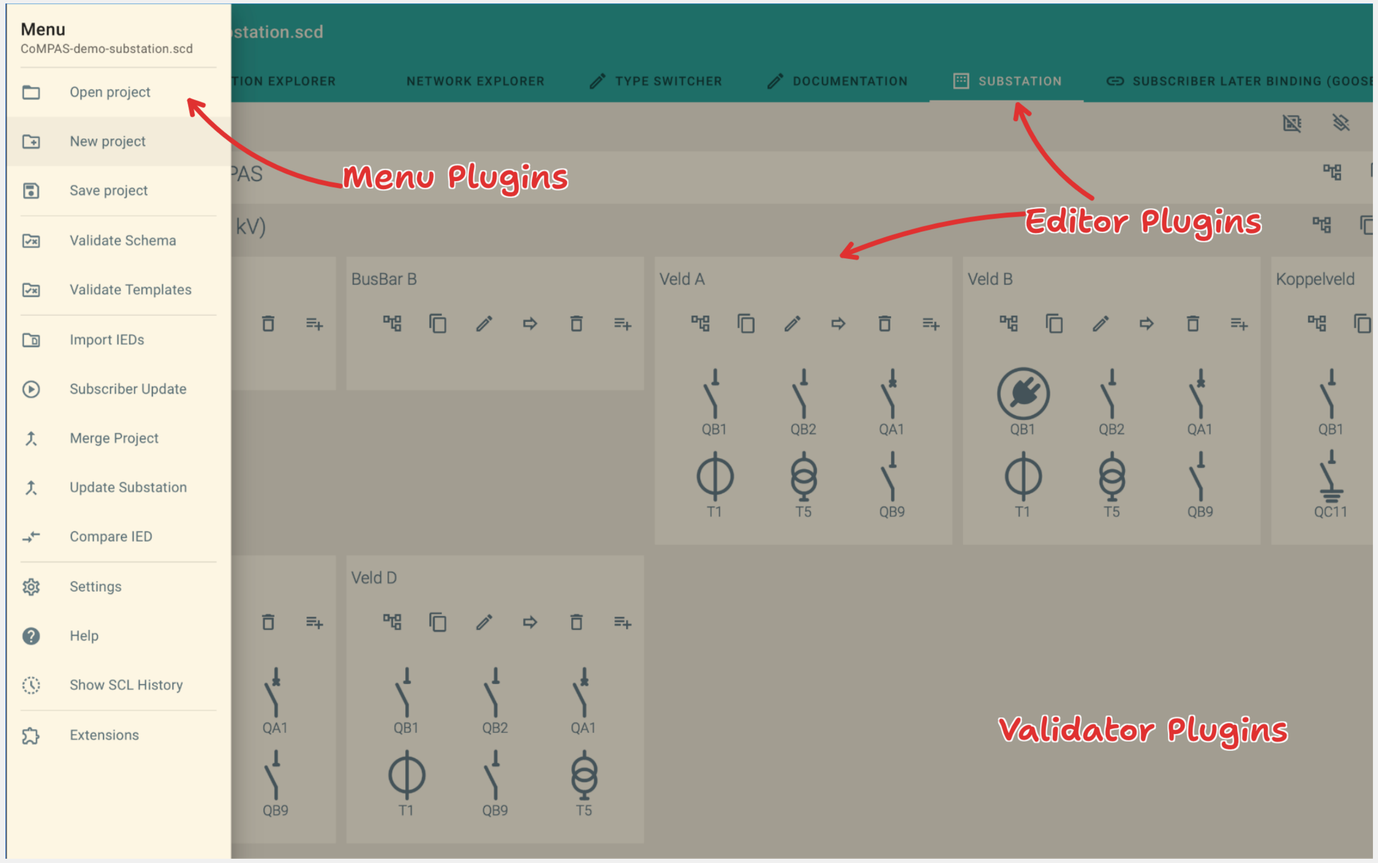
We use custom elements to provide the widest support of browsers and frontend technologies. We don't mind what you use. It only needs to be compiled to a custom element

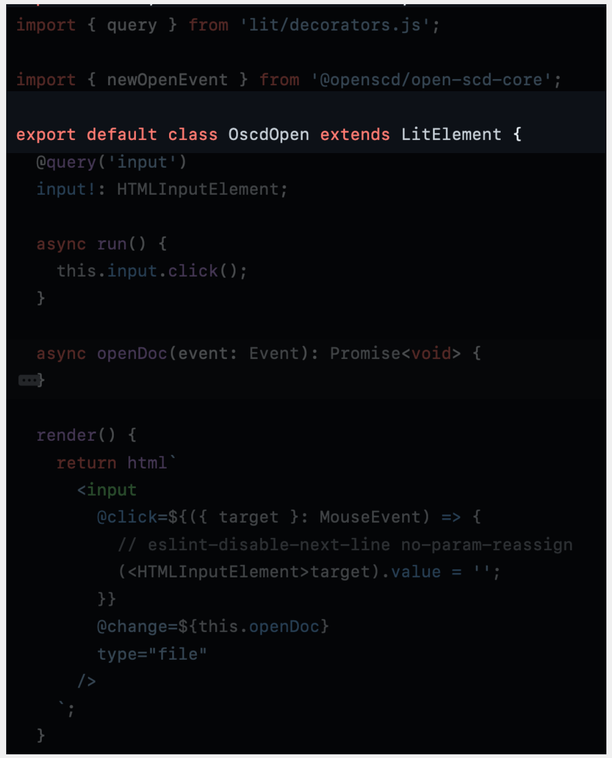
TECHNOLOGY INDEPENDENT

To reach a wider audience of developers. Quickly start with whatever you want.

The Plugin System

- Almost everything is a plugin
- Menu points, Editors, Validators
- We load and cache the source code of the plugins dynamically



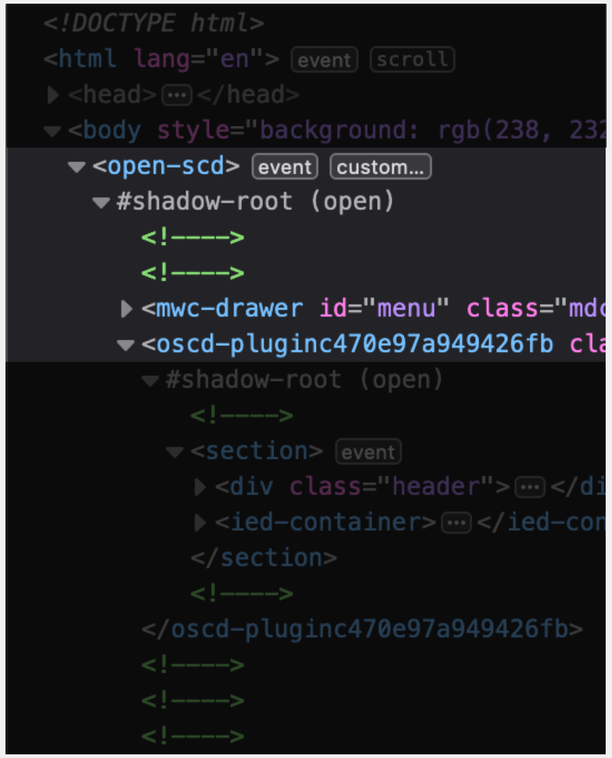


```
import { query } from 'lit/decorators.js';

import { newOpenEvent } from '@openscd/open-scd-core';

export default class OscdOpen extends LitElement {
  @query('input')
  input!: HTMLInputElement;

  async run() {
    this.input.click();
  }

  async openDoc(event: Event): Promise<void> {
...}

  render() {
    return html`
      <input
        @click=${({ target }: MouseEvent) => {
          // eslint-disable-next-line no-param-reassign
          (<HTMLInputElement>target).value = '';
        }}
        @change=${this.openDoc}
        type="file"
      />
    `;
  }
```

Unregistered Custom Element

```
private addContent(plugin: Omit<Plugin, 'content'>): Plugin {
  const tag = pluginTag(plugin.src);
  if (!loadedPlugins.has(tag)) {
    loadedPlugins.add(tag);
    import(plugin.src)
      .then(mod => customElements.define(tag, mod.default));
  }
  return {
    ...plugin,
    content: staticTagHtml`<${tag}
        .doc=${this.doc}
        .docName=${this.docName}
        .editCount=${this.editCount}
        .docId=${this.docId}
        .pluginId=${plugin.src}
        .nsdoc=${this.nsdoc}
        class="${classMap({
          plugin: true,
          menu: plugin.kind === 'menu',
          validator: plugin.kind === 'validator',
          editor: plugin.kind === 'editor',
        })}"
      ></${tag}>`,
  };
}
```

Dynamic Loading



```
<!DOCTYPE html>
<html lang="en"> event scroll
▶ <head>…</head>
▼ <body style="background: rgb(238, 232
  ▼ <open-scd> event custom…
    ▼ #shadow-root (open)
        <!---->
        <!---->
      ▶ <mwc-drawer id="menu" class="mdc
      ▼ <oscd-pluginc470e97a949426fb cla
        ▼ #shadow-root (open)
            <!---->
          ▼ <section> event
            ▶ <div class="header">…</di
            ▶ <ied-container>…</ied-cor
            </section>
            <!---->
        </oscd-pluginc470e97a949426fb>
        <!---->
        <!---->
        <!---->
```

Result

A plugin written in Svelte

```ts
import Plugin from './plugin.svelte'
import * as pkg from "../package.json";

export default class NewPlugin extends HTMLElement {

    private plugin: Plugin

    connectedCallback() {
        this.attachShadow({ mode: "open" });
        this.plugin = new Plugin({
            target: this.shadowRoot,
            props: {
                doc: this._doc
            }
        });

        const style = document.createElement("style");
        style.innerHTML = globalThis.pluginStyle[pkg.name];
        this.shadowRoot.appendChild(style);
    }

    private _doc: XMLDocument
    public set doc(newDoc: XMLDocument){
        this._doc = newDoc
        if(!this.plugin) { return }

        this.plugin.$set({doc: newDoc})
    }

}
```

```svelte
<svelte:options tag={null} />

<script lang="ts">
  import { NetworkExplorer } from "@oscd-plugins/uilib/src/lib/plugins/network-explorer";
  import * as pckg from "../package.json";

  // Inputs
  export let doc: XMLDocument;
</script>

{#if doc}
  <NetworkExplorer root={doc?.documentElement} />
{/if}

<input type="hidden" name="package-name" value={pckg.name} />
<input type="hidden" name="package-version" value={pckg.version} />
```

Distributions

YOUR OWN PLATFORM

You can deploy OpenSCD yourself in your own infrastructure.

YOUR OWN ADDONS

We provide the building blocks to create and customise your own distribution.

YOUR OWN PLUGINS

You can create your own plugins and use them in multiple distribution

Thank you

OPENSCD ORGANISATION

GitHub

#OPEN-SCD @LFENERGY

Slack

OPENSCD.ORG

Website

OPENSCD.GITHUB.IO

OpenSCD