odoo
TRAINING

Day 1
Technical Training

Odoo JavaScript Framework

Géry Debongnie (ged)
RD Framework Team

1. Introduction
2. Practical Information
3. Odoo.sh as a development tool
4. A Primer on Odoo JS

1 Introduction

Rule #1 of customizing Odoo with Javascript:

"do it in python (or xml)"

Rule #2 of customizing Odoo with Javascript:

"do it in a different way, so you can avoid JS"

Goals

- develop an understanding on how the Odoo Javascript Framework works in general
- practical knowledge on how to solve problems in Javascript

Requirements

- intermediate knowledge of Javascript (in general)
- intermediate knowledge of Odoo
- a laptop with internet access
- basic knowledge of git

2 Practical Informations

Practical Informations

- Duration: 2 days
- Time: From 9:00 am to 5:00 pm
- Lunch and drinks included

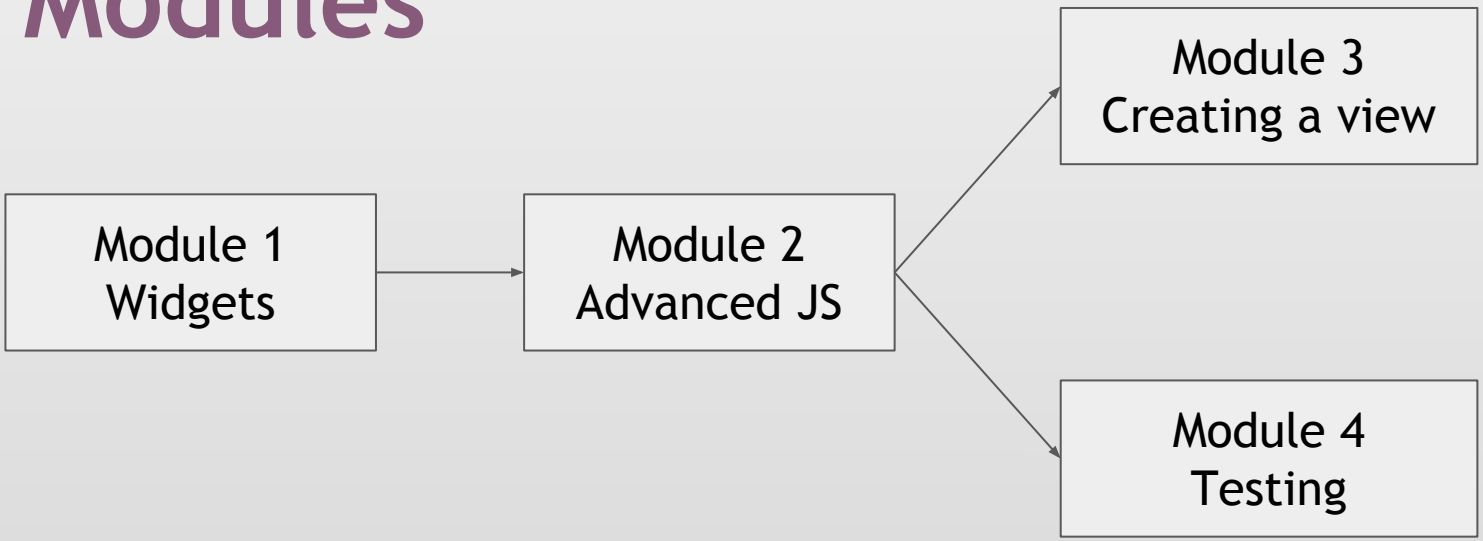
Instructors

- Aaron B. (aab)
- Géry D. (ged)
- Vincent S. (vsc)

Organization

- Odoo.sh as our tool (code editing/running odoo/testing/…)
- work in group of 2/3
- training is organized in 4 modules, each with a set of tasks

Modules



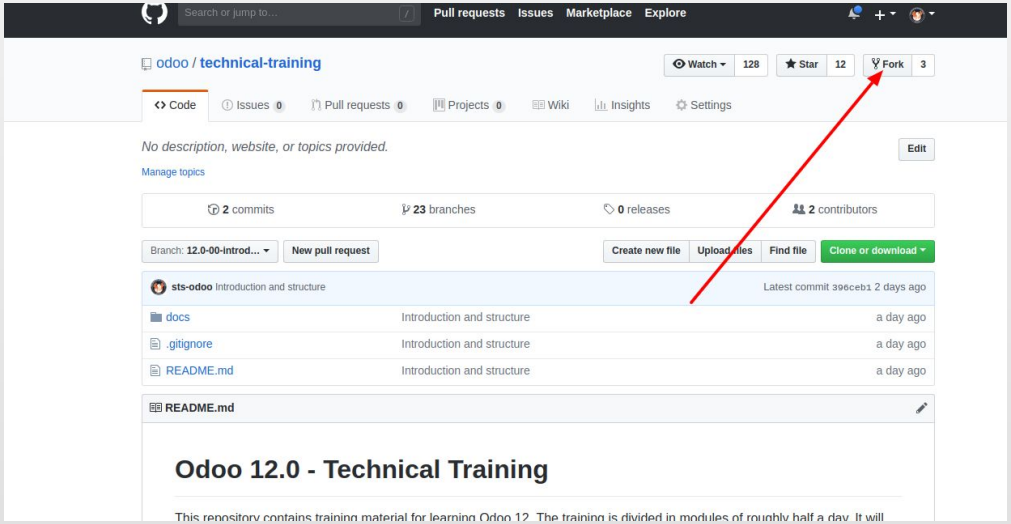
3

Odoo.sh as a development tool

Github account

Your odoo.sh is based on your github account, all the development will be hosted on github. A specific github repository will be linked to a specific project on Odoo.sh.

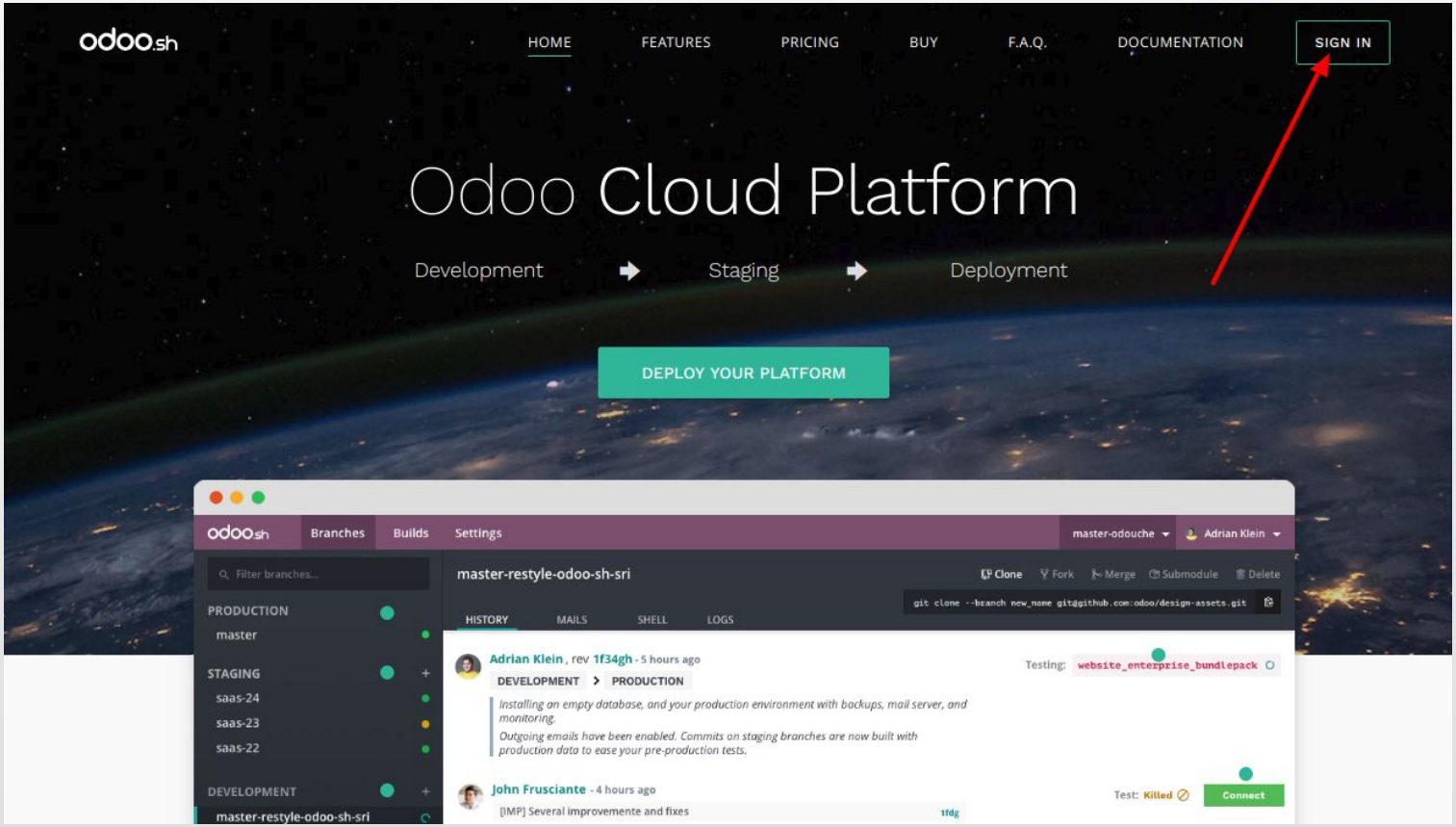
1. Create a github account if you don't have one yet
2. Fork https://github.com/odoo/technical-training



3. A copy of the technical training will be on: https://github.com/YOURGITHUBUSER/technical-training

Sign in

Sign in on odoo.sh with your github credentials

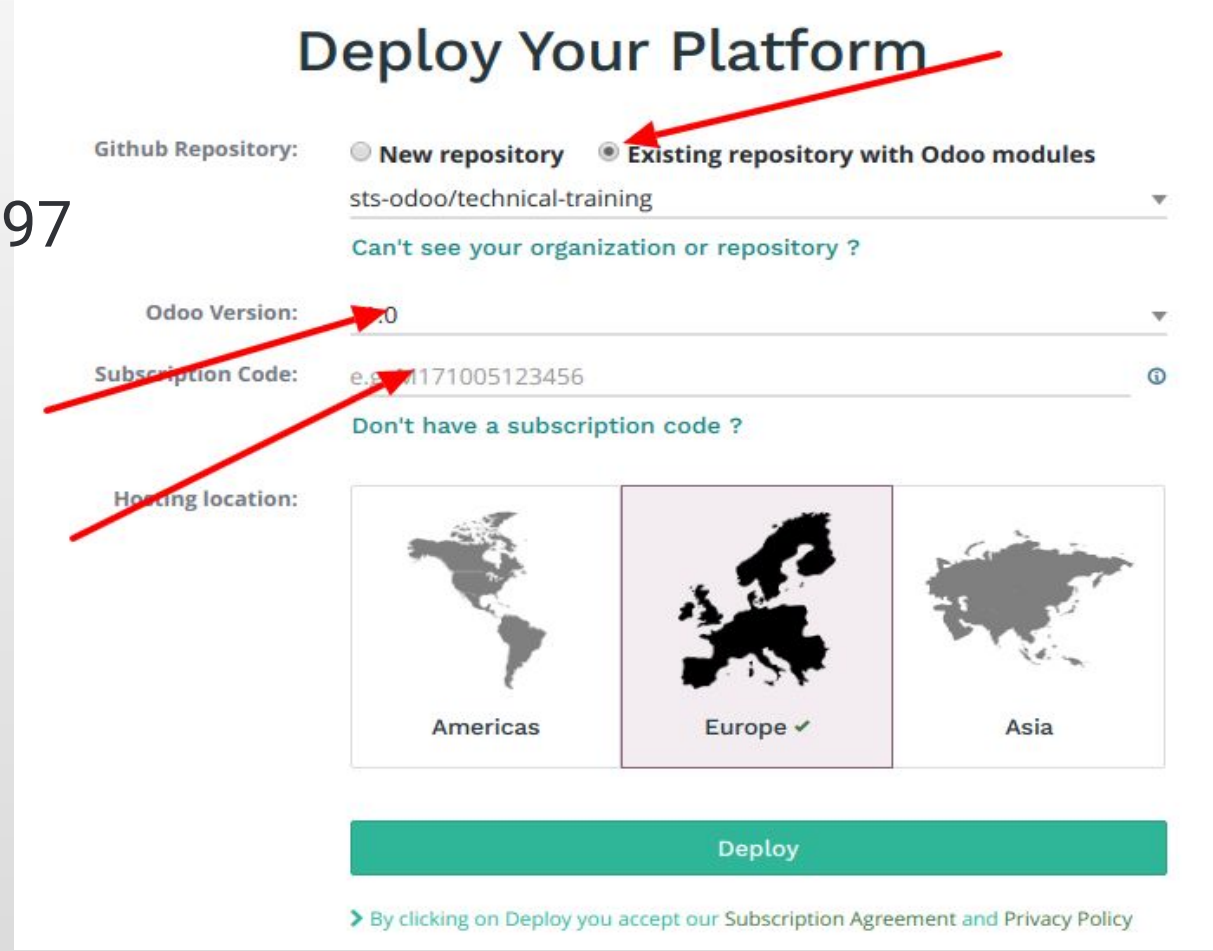


Create a project

- Create a project on Odoo.sh based on your own technical-training repository
- Version 12
- Subscription Code:

OXP1809108524197

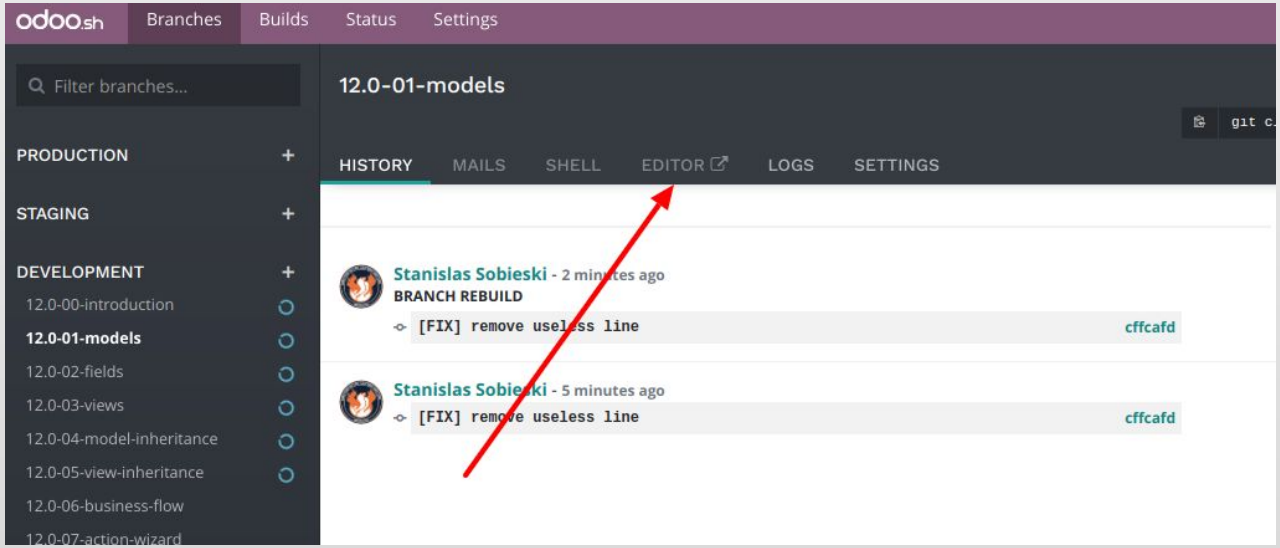
Temporary access for this training



Unmute your branches

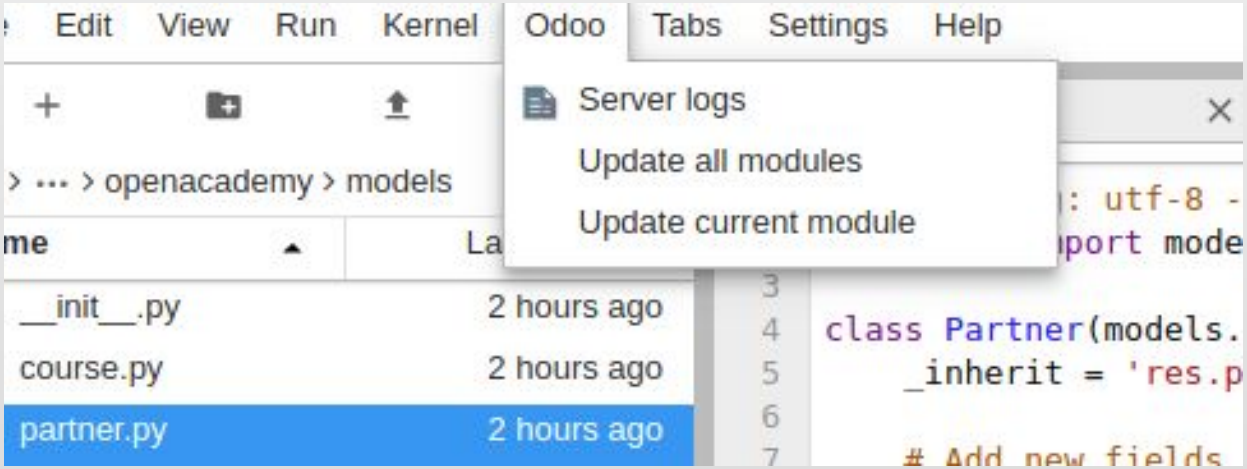
- On the project page, go on Builds, unmute and rebuild branches:
  - **12.0-19-javascript-training**

- Go on Branches and select 12.0-19-javascript-training and launch the editor (Jupyterlab) and let's get to work



Code Editor

- On development branches, the build is launched with the --dev=reload parameter, that means any python code changes will trigger a reload
- If changes are made to the data structure: fields and models or on actual data (records), an update of the module is required and is possible with the menu Odoo > Update current module within the editor



Save your changes on your own repository

- Open a shell
- Go on src/user and use regular git command to commit your changes
- Use git push https HEAD:12.0-01-models to push your changes on your own repository. It will ask for your github credentials before actually pushing.
- Note that your new commit will trigger a new build on the same branch on odoo.sh

4

A Primer on Odoo Javascript

https://www.odoo.com/documentation/12.0/

(section on Javascript Reference)

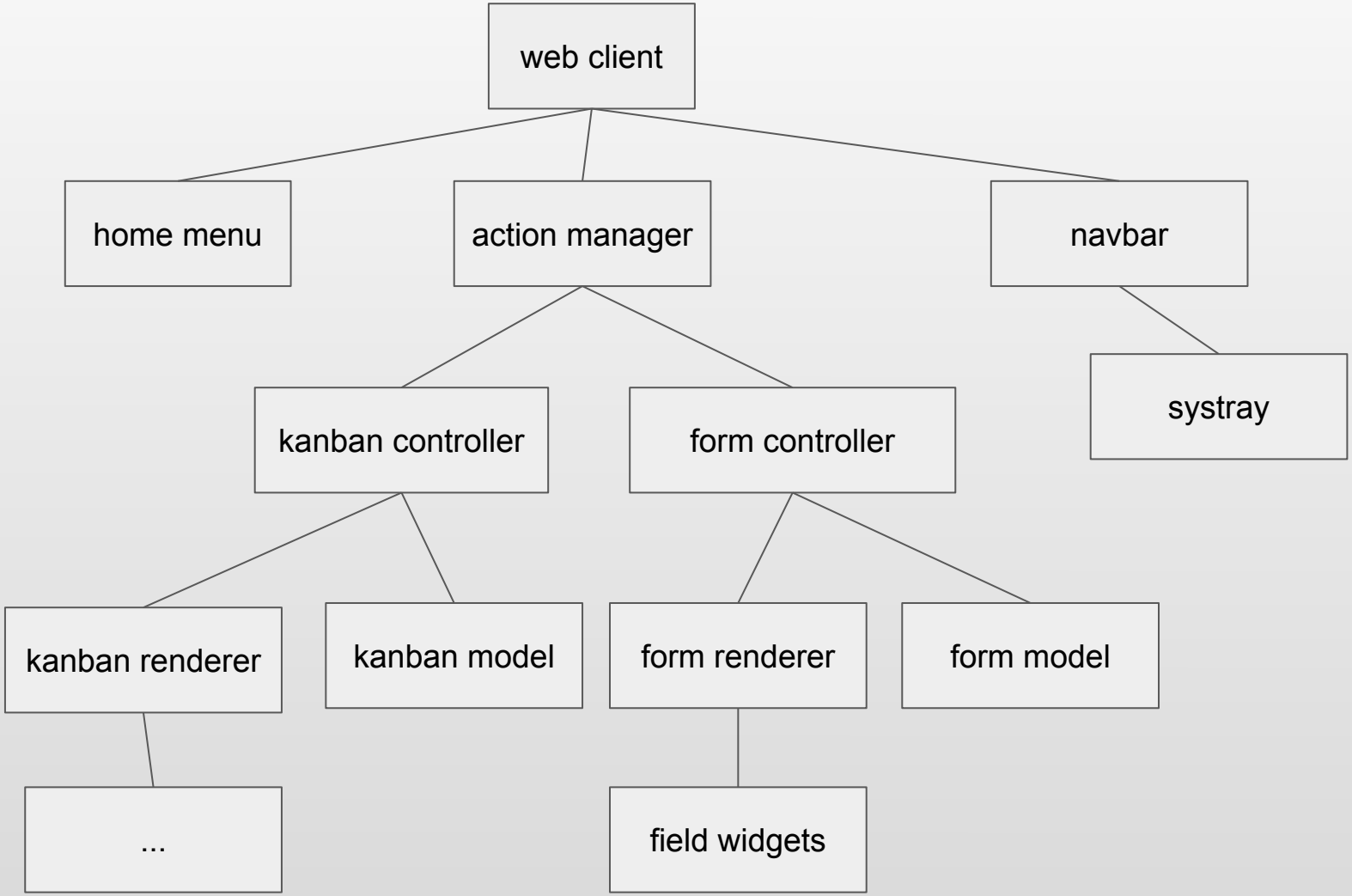
Web Client

- a SPA (single page application)
- made with our custom framework
- use QWeb as template engine
- extensible
- 35k/45k lines of JS code/tests

URL: /web/

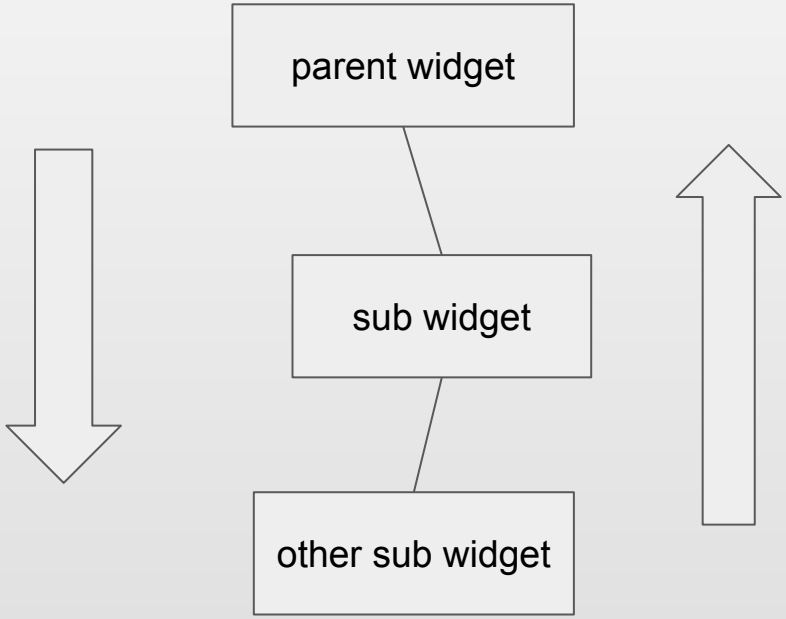
Code: addons/web/static/src

Web Client Component Tree (partial)



Communication between components

Downstream:
- constructor
- method call



Upstream:
events (bubbling up)

Last resort: events on a bus (to avoid if possible)

Assets Management

Asset bundles (css/js)

- assets_backend: web client
- assets_frontend: website
- assets_common: both

Adding a file to a bundle

- add a *assets.xml* file in the *views/* folder
- add the string ‘views/assets.xml’ in the ‘data’ key in the manifest file
- create an inherited view of the desired bundle, and add the file(s) with an xpath expression

```
<template id="assets_backend" name="helpdesk assets" inherit_id="web.assets_backend">
    <xpath expr="//script[last()]" position="after">
        <link rel="stylesheet" type="text/scss" href="/helpdesk/static/src/....scss"/>
        <script type="text/javascript" src="/helpdesk/static/src/js/....js"></script>
    </xpath>
</template>
```

https://www.odoo.com/documentation/12.0/reference/javascript_reference.html#assets-management

Odoo Javascript Modules

JS module resolution: at runtime

```
// in file a.js
odoo.define('module.A', function (require) {
    "use strict";

    var A = ...;

    return A;
});

// in file b.js
odoo.define('module.B', function (require) {
    "use strict";

    var A = require('module.A');

    var B = ...; // something that involves A

    return B;
});
```

https://www.odoo.com/documentation/12.0/reference/javascript_reference.html#javascript-module-system

Widget: the building block for UI

Widget lifecycle

- **init** (constructor)
- **willStart:** (async), before dom is ready
- [template rendering]
- **start:** widget dom is ready
- **destroy:** destructor

https://www.odoo.com/documentation/12.0/reference/javascript_reference.html#widgets

4 simple rules for your components

- Do not depend on your parent…
- Separate public/private/handlers
- Document your code
- Test your component

Example (except doc)

```
var MyCounter = Widget.extend({
      events: {
            click: '_onClick'
      },
      init: function (parent, value) {
            this._super(parent);
            this.value = value;
      },
      start: function () {
            this._render();
      },
      //-------------------------------
      // Public
      //-------------------------------
      increment: function () {
            this.value++;
            this._render();
      },
```

```
      //-------------------------------
      // Private
      //-------------------------------
      _render: function () {
            this.$el.html(
                  $('<span>').text(this.value)
            );
      },
      //-------------------------------
      // Handlers
      //-------------------------------
      _onClick: function () {
            this.increment();
      },
});
```

odoo
TRAINING

Let’s get to work.