haute école
neuchâtel berne jura
arc gestion
neuchâtel delémont

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

64-31.1 Projet de technologie web

Vue.js

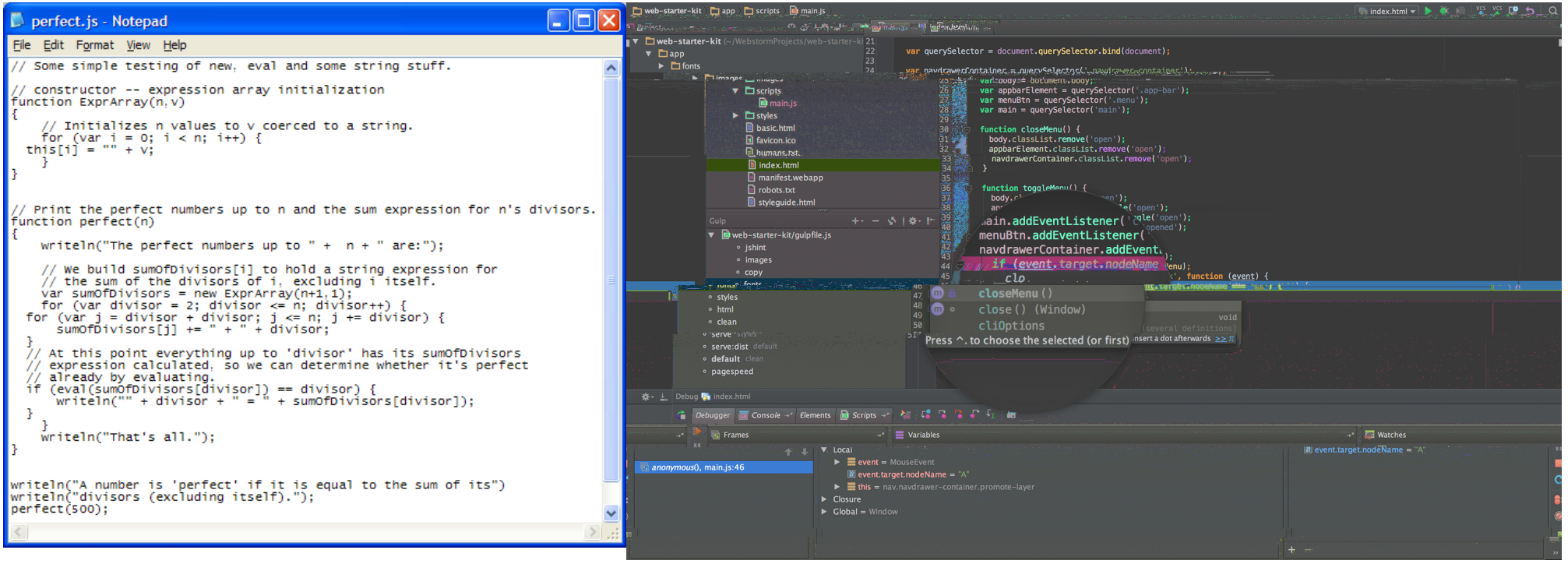
Build and deploy a Single Page Application

Boris.Fritscher@he-arc.ch

Tooling

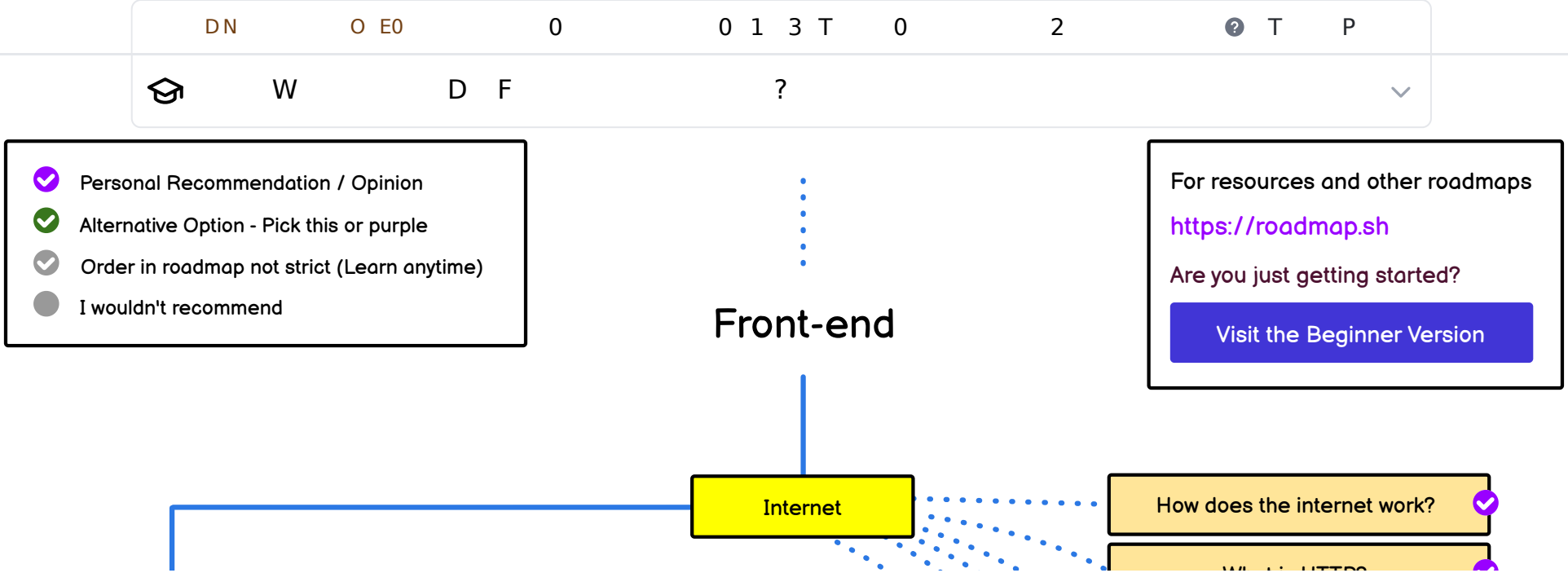
Editors

Exists in all form: from notepad.exe to full IDE: WebStorm



In between: Notepad++, Visual Studio Code, Brackets
Online editors: CodeSandbox, StackBlitz, Cloud9, Eclipse Che
Browser integrated (F12): Chrome DevTools

Frontend Developer



Preprocessors

A preprocessor is a program that processes its input data to produce output, that is used as input to another program

CSS preprocessor: Less, Sass, Stylus, ...

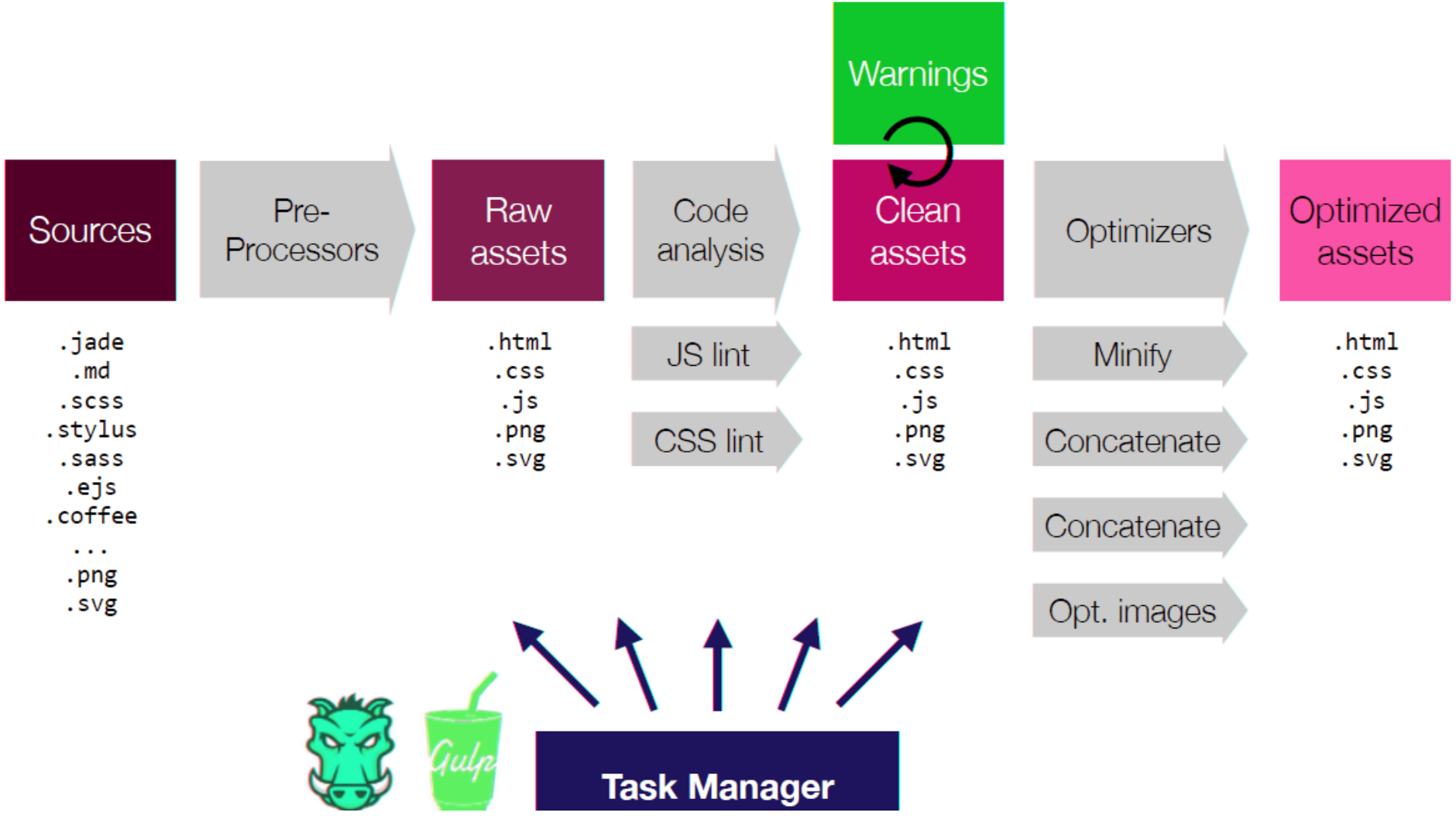
JS preprocessor: CoffeeScript, TypeScript, ECMAScript 2015 (ES6) (Traceur and Babel), ...

HTML preprocessor: Jade, Haml, Handlebars, ...

Script loader: Require, Webpack, ...

Test framework: Jasmine, Mocha, Qunit, ...


Build Pipeline


Warnings
Sources
Pre-Processors
Raw assets
Code analysis
Clean assets
Optimizers
Optimized assets
.jade
.md
.scss
.stylus
.sass
.ejs
.coffee
...
.png
.svg
.html
.css
.js
.png
.svg
JS lint
CSS lint
.html
.css
.js
.png
.svg
Minify
Concatenate
Concatenate
Opt. images
.html
.css
.js
.png
.svg
Gulp
Task Manager

Example of Pre-Processors

https://jsbin.com/mumavu/9/edit?html,css,js

Node.js & npm

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. https://nodejs.org/

It allows to run JavaScript outside of the browser

npm is a package manager for JavaScript bundled with Node.js and can run some tasks.
https://www.npmjs.com/

Vite

Is a build tool that aims to provide a faster and leaner development experience for modern web projects. It consists of two major parts:

- A dev server (Hot Module Replacement / Reload)
- A build command that bundles your code with Rollup, pre-configured to output highly optimized static assets for production.
- Project Tempaltes for vue, react, ...

Lab 1: Vue.js

Setup a local Vue.js project

Step 0: Install development environment

Download and install Node.js to get `npm`.

Download and install Git to get `git`.

Download and install Visual Studio Code to get `code`.

Check that it works from the shell:

```
$ git --version
git version X.Y.Z

$ node --version
vX.Y.Z

$ code --version
X.Y.Z
...
x64
```

Step 1: Create a new project

First change to a parent folder in which the project folder will be created.

Use vue/vite to create a new project.

```
C:\temp> npm init vue@latest
```

Step 2: Install Extensions

Change into the directory and launch vscode

```
$ cd labo-vue
$ code .
```

- install vscode Vue Volar extension
- install vscode Prettier extension
- install vscode eslint extension
- install vue chrome devtool extension

Step 3: Install dependencies

Must be done inside the project's root folder!

```
$ npm install
```

This will read `package.json` and update `node_modules` folder.

Step 4: Preview your app in the browser

Start the development server

```
npm run dev
```

edit a file and watch livereload in action.

For example add `<h1>Test vue!</h1>` to `<body>` in `./index.html`.

Stop the server with `ctrl+c`

Step 5: Cleanup

- Delete `src/assets/base.css`
- Empty `src/assets/main.css`
- Replace `src/App.vue` with this code:

```
<template>
  <div class="container">
    <h1>Hello Bootstrap</h1>
    <div class="row">
      <div class="col-sm">
        <button class="btn btn-primary">{{ message }}</button>
      </div>
      <div class="col-sm">
        <i class="fas fa-ice-cream display-1 text-primary"></i>
      </div>
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {
```

Step 6: Setup bootstrap and fontawesome

Check package.json (before and after)

1. Add dependencies

```
$ npm install --save bootstrap @popperjs/core
$ npm install --save @fortawesome/fontawesome-free
```

2. Inside `src/main.js` add imports

```
import "bootstrap";
import "bootstrap/dist/css/bootstrap.min.css";
import "@fortawesome/fontawesome-free/css/all.min.css";
```

Step 6: Try more fontawesome

Check that the icon works:

Add another icon to the `App.vue template inside the button`

https://fontawesome.com/icons

```
<i class="fa-solid fa-cake-candles"></i>
```

Step 7: Use npm to install other packages

Lets try another bootstrap look.

```
npm install bootswatch --save
```

https://bootswatch.com/

try different CSS files from bootswatch in `main.js`

```
import "bootswatch/dist/darkly/bootstrap.min.css";
```

Test Linting Config 1

We installed two linter:

- Prettier for code formatting
- ESLint for code quality

But we need to set ower own preferences

.prettierrc.json

```
{
  "$schema": "https://json.schemastore.org/prettierrc",
  "semi": true,
  "tabWidth": 4,
  "singleQuote": false,
  "printWidth": 120,
  "trailingComma": "none"
}
```

Test Linting Config 2

But we need to set ower own preferences

In `.eslintrc.cjs` replace

```
'@vue/eslint-config-prettier/skip-formatting'
```

with

```
"@vue/eslint-config-prettier"
```

This will make lint also run format at the same time.

Run the command and check how the files are changed.

```
$ npm run lint
```

Test Linting

Copy this line into main.js and see what happens.

```
let myvar = 'Hello World'
```

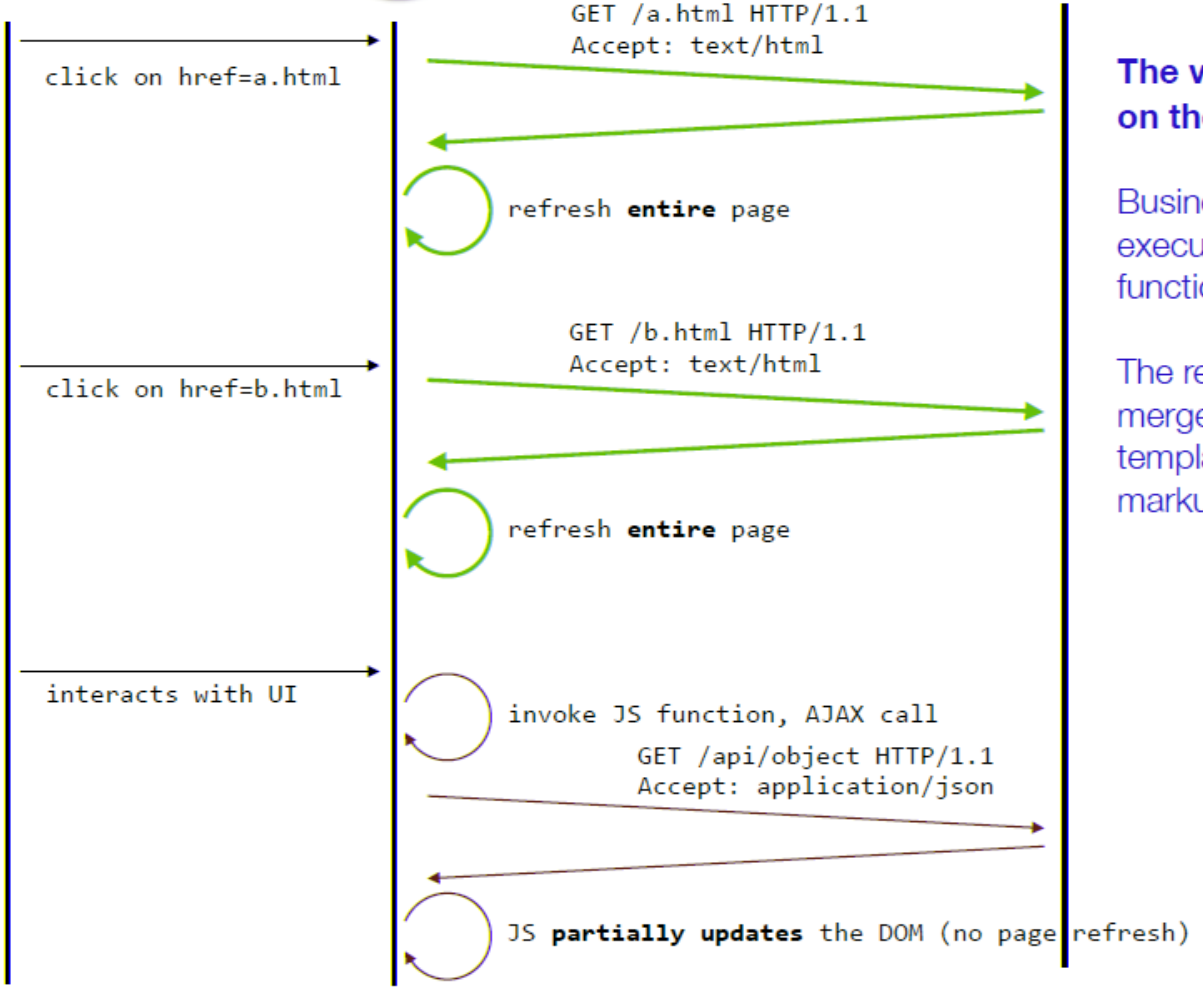
Check the PROBLEMS tab of vscode (right-click to fix problem).

Multi Page App
vs
Single Page Application (SPA)

Multi-page application architecture

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud





The view is fully built on the sever side:

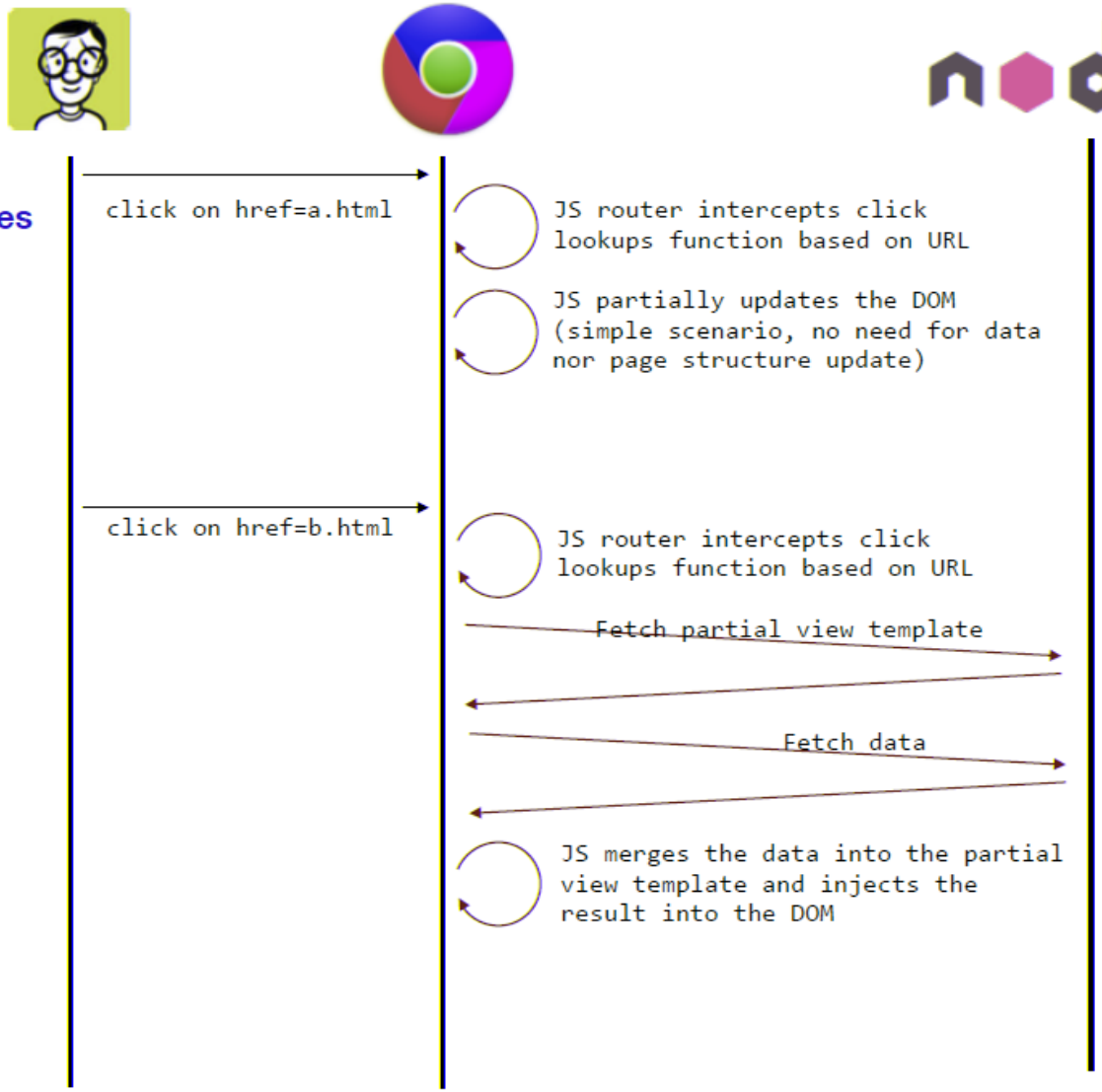
Business logic is executed in a JavaScript function.

The resulting model is merged into a view template to obtain HTML markup.

Single-page application architecture

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

The client side handles the navigation logic and builds the views.

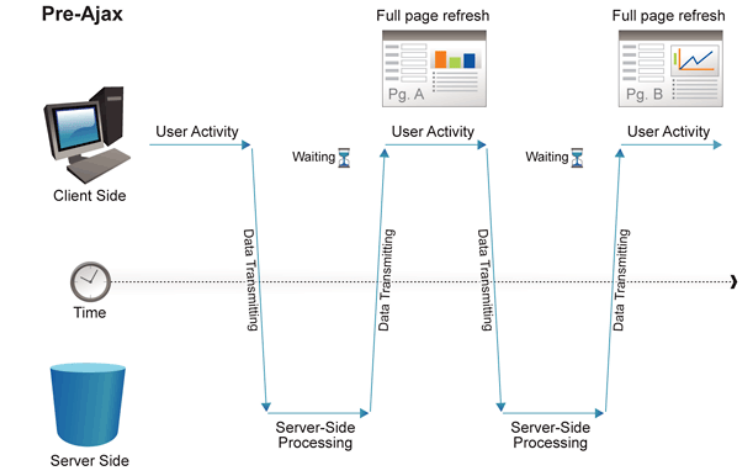


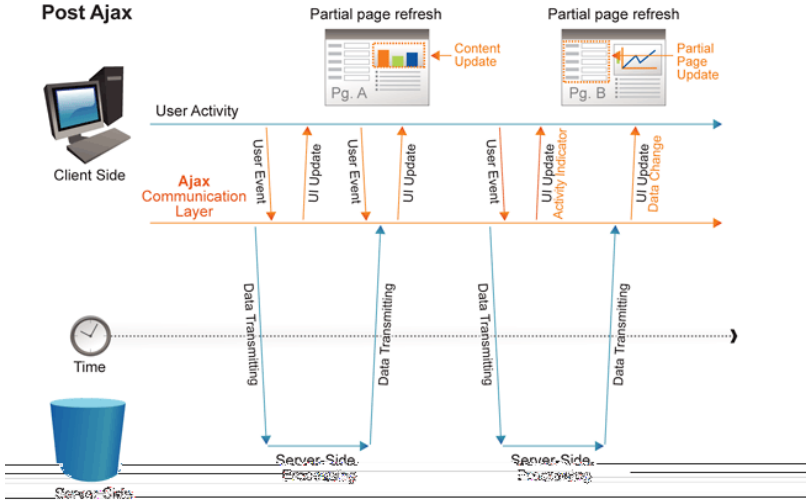
The server side is dumb.

It only serves assets (HTML, Jade, JavaScript, image files).

There is a big trend towards “single-page applications”, where some of the responsibilities are moved from the server to the client side.

- The client initially fetches a single “shell” page, which provides a rendering context and loads application modules (scripts, markup partials, stylesheets, etc.).
- When the user clicks on hyperlinks, the browser does not (immediately) send an HTTP request to fetch a new page. Instead, the event is caught and processed by a JavaScript router on the client side.
- Routing is done on the client side. The JavaScript router (typically provided by an application framework) looks at the target URL and decides which JavaScript function needs to be invoked. This function can update the DOM, sometimes in drastic manners (giving the impression that we move from an “Customers List” page to a “Customer Details” page).



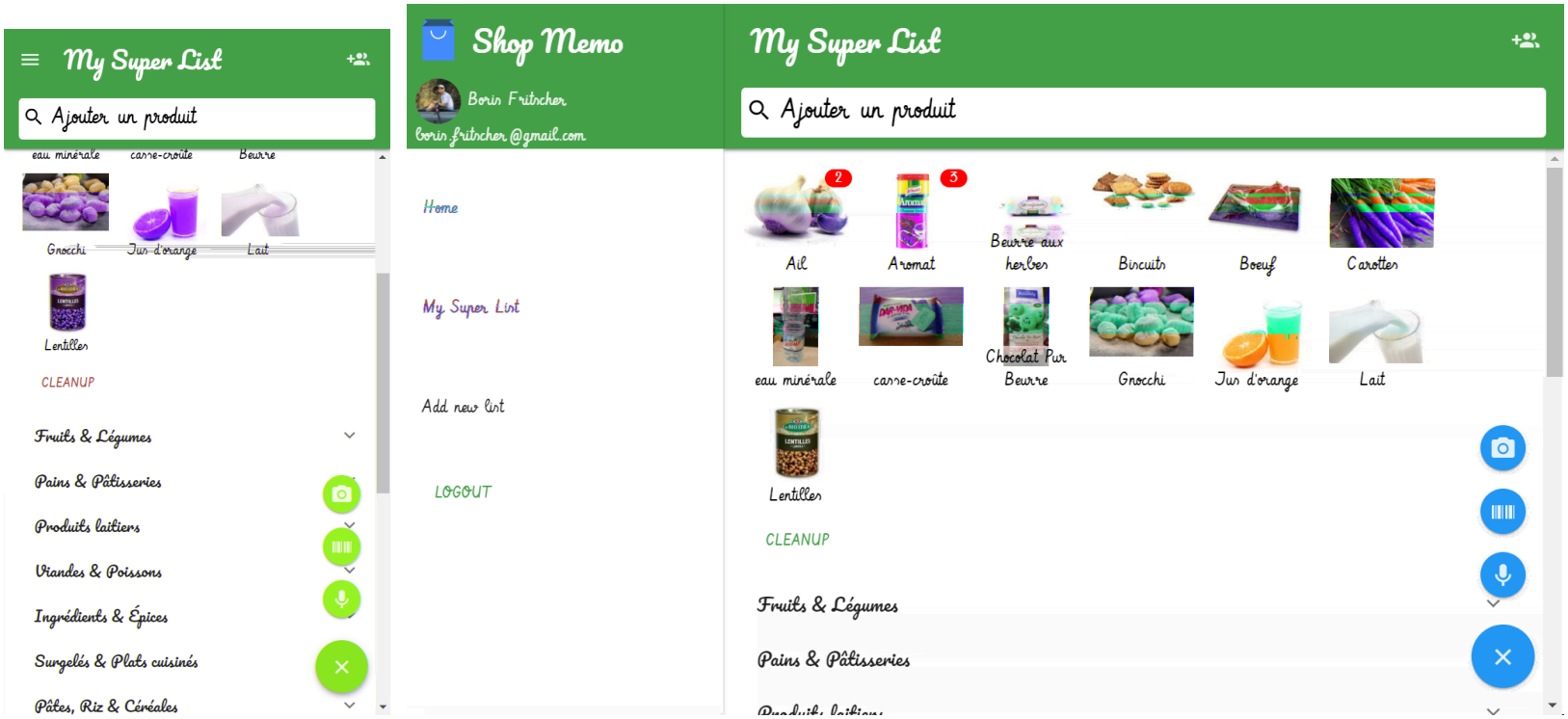


http://www.websiteoptimization.com/secrets/ajax/8-1-ajax-pattern.html

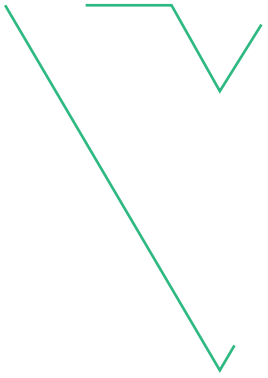
Paper to WWW

Objectifs

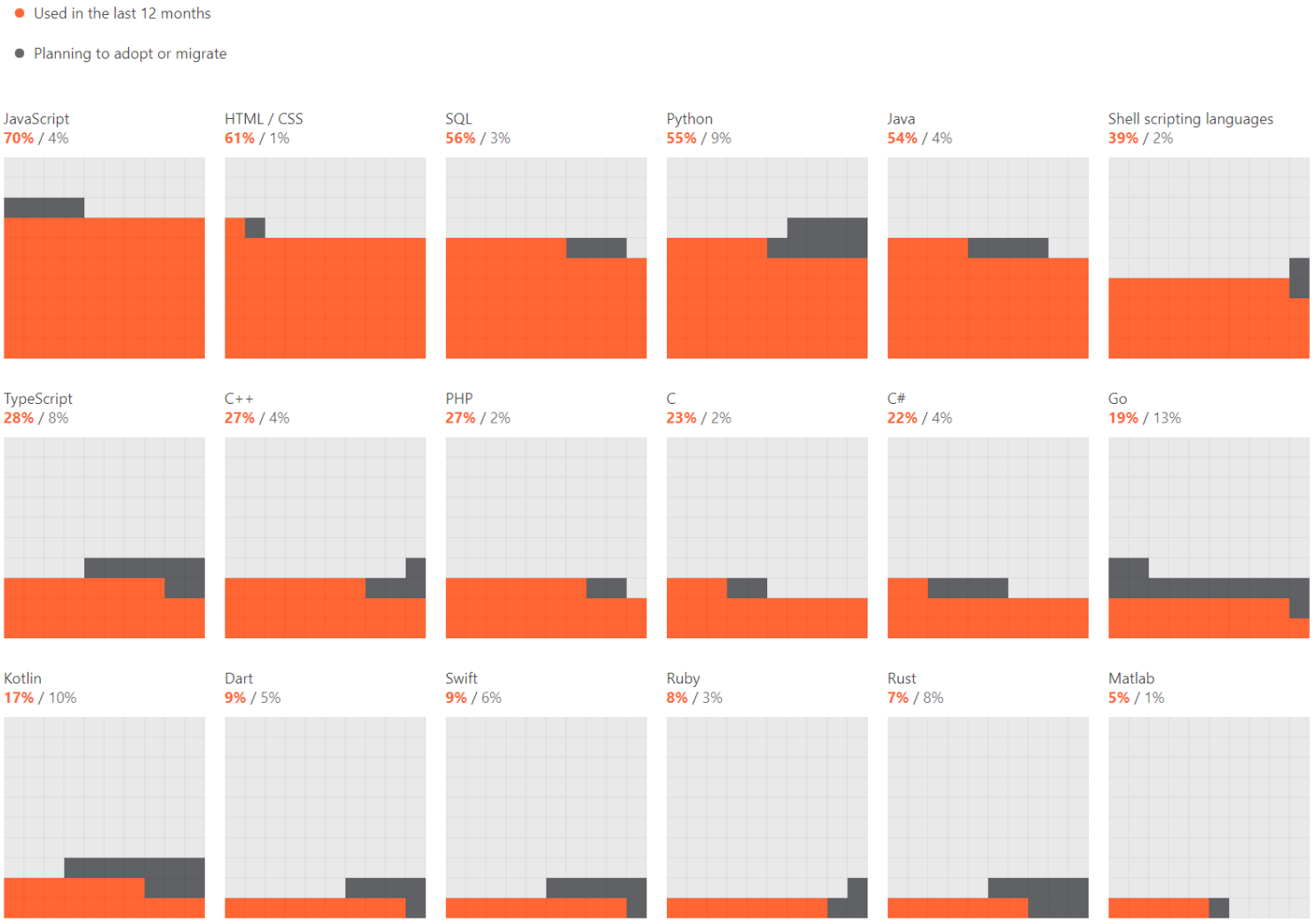
- Learn SPA with Vue.js
- Learn by Example: CatList App
- Only focus on frontent-app
- With API's and some cloud functions ("serverless")


My Super List
Ajouter un produit
eau minérale
carre-croûte
Beurre
Gnocchi
Jus d'orange
Lait
Lentilles
CLEANUP
Fruits & Légumes
Pains & Pâtisseries
Produits laitiers
Viandes & Poissons
Ingrédients & Épices
Surgelés & Plats cuisinés
Pâtes, Riz & Céréales
Shop Memo
Boris Fritscher
boris.fritscher@gmail.com
Home
My Super List
Add new list
LOGOUT
My Super List
Ajouter un produit
Ail
Aromat
Beurre aux herbes
Biscuits
Boeuf
Carottes
eau minérale
carre-croûte
Chocolat Pur Beurre
Gnocchi
Jus d'orange
Lait
Lentilles
CLEANUP
Fruits & Légumes
Pains & Pâtisseries

Why Vue.Js?

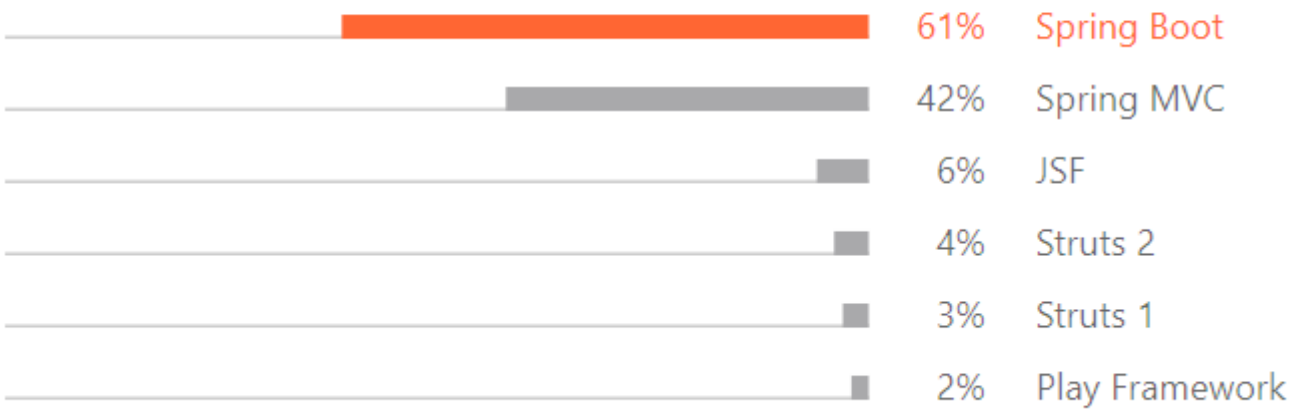


- address many of the challenges encountered in developing SPA
- large acceptation
- decouple DOM manipulation from application logic
- decouple the client side of an application from the server side
- declarative programming for user interface
- imperative programming for application business logic

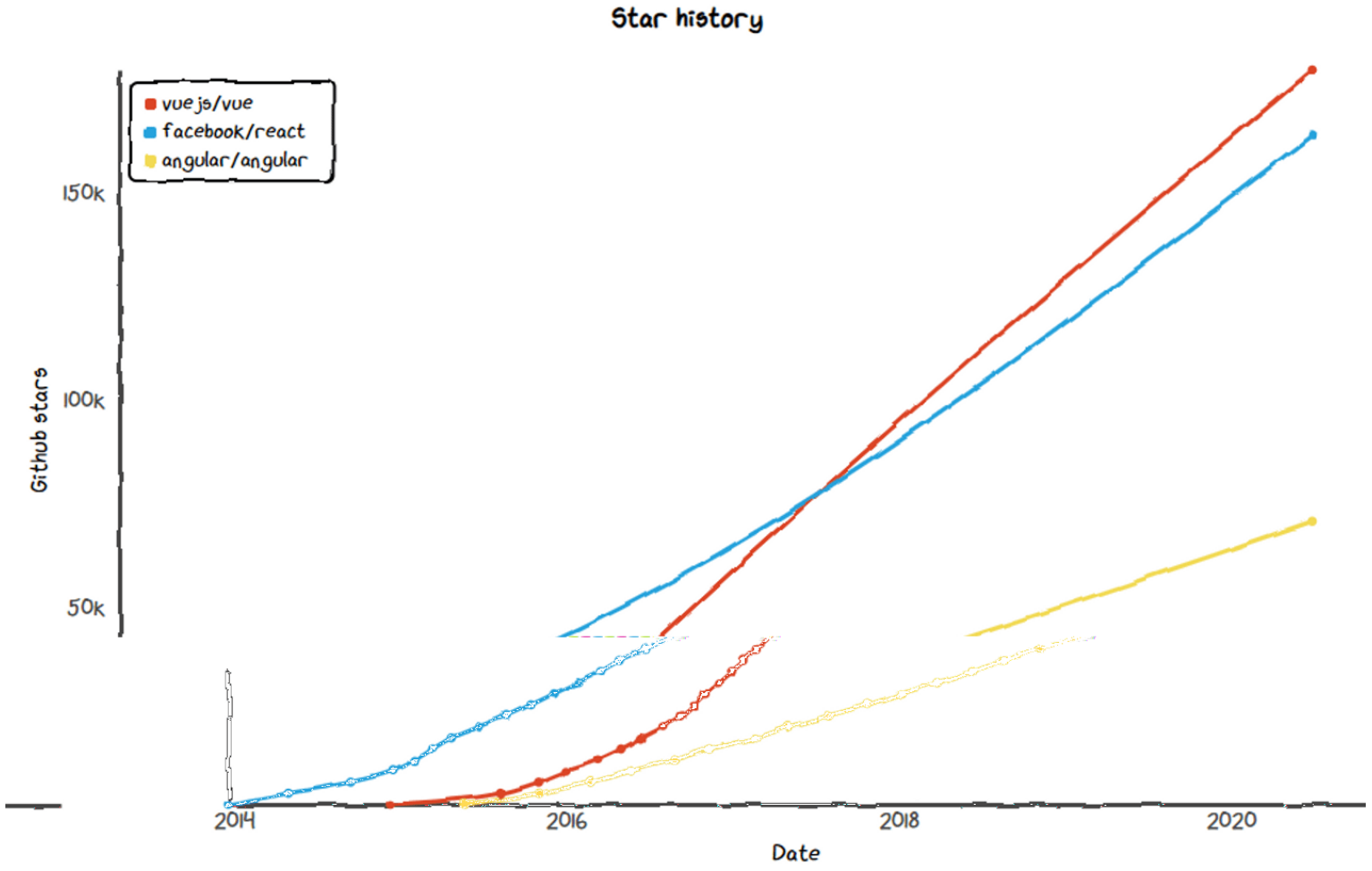


https://www.jetbrains.com/lp/devecosystem-2020/

What web frameworks do you use?

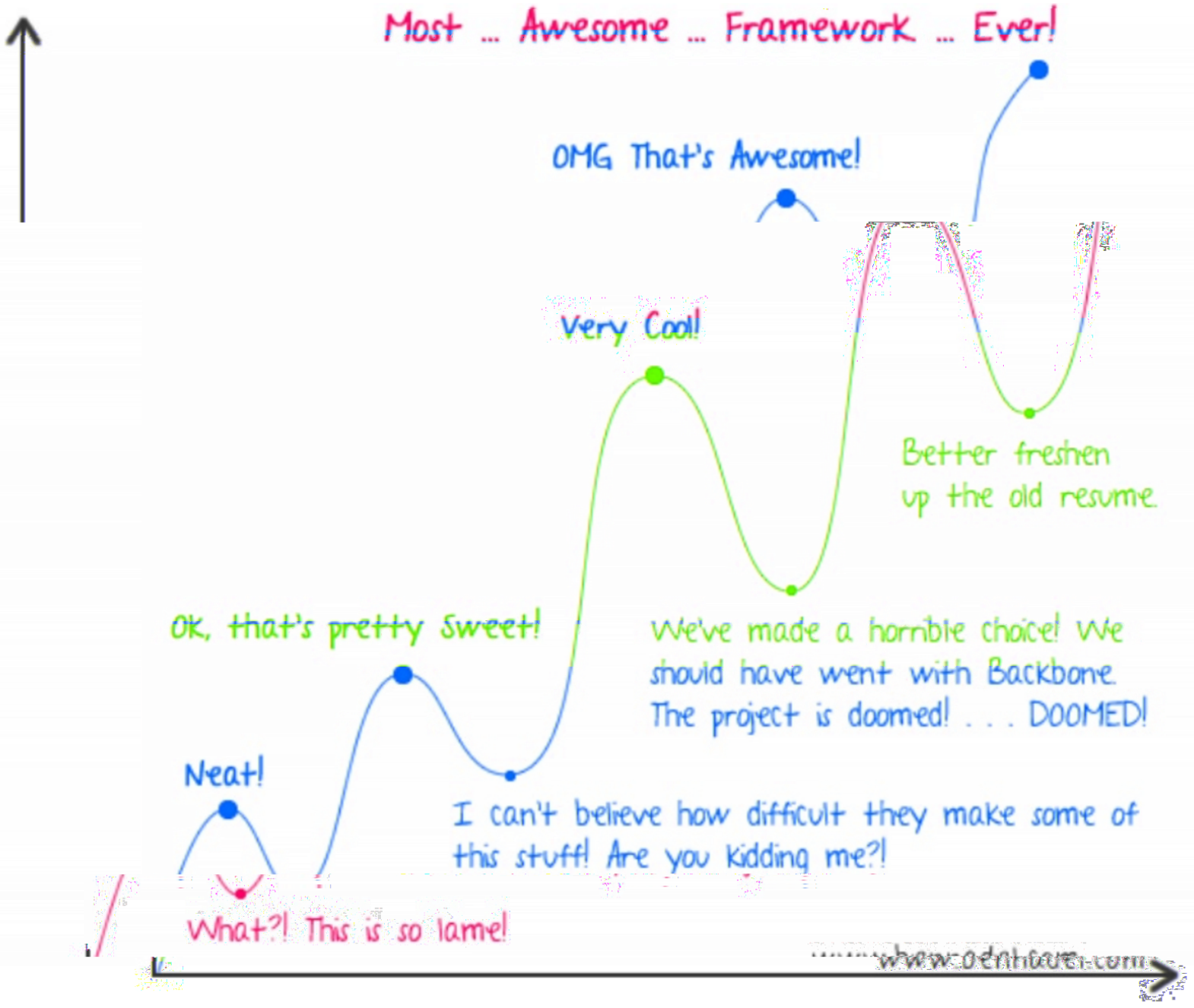


https://www.jetbrains.com/lp/devecosystem-2020/java/



https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

https://www.valuecoders.com/blog/technology-and-apps/vue-js-comparison-angular-react/


Most ... Awesome ... Framework ... Ever!
OMG That's Awesome!
Very Cool!
Better freshen up the old resume
Ok, that's pretty Sweet!
We've made a horrible choice! We should have went with Backbone. The project is doomed! . . . DOOMED!
Neat!
I can't believe how difficult they make some of this stuff! Are you kidding me?!
What?! This is so lame!

Vue.js a Model View ViewModel (MVVM) framework

Guide Vue.js

Vue constructor and text interpolation

HTML

```
<div id="app">
  {{ message }}
</div>


<script
src="https://unpkg.com/vue@3">
</script>
```

Javascript

```
Vue.createApp({
    data() {
        return {
        message: 'Hello Vue!'
        };
    }
}).mount('#app');
```

Text interpolation

Vue.js expressions are JavaScript-like code snippets that are usually placed in bindings such as `{{ expression }}`.

```
<div>{{ message.toUpperCase() }}</div>       <!-- HELLO VUE.JS! -->
<div>{{ message.slice(0,5) }}</div>          <!-- Hello         -->
<div>You say {{ ok ? 'YES' : 'NO' }}</div>   <!-- You say NO    -->
```

> *Template expressions are sandboxed and only have access to a whitelist of globals such as `Math` and `Date`. You should not attempt to access user defined globals in template expressions.*

https://vuejs.org/guide/essentials/template-syntax.html

Some Directives

Directive	Description
v-model	Create a two-way binding on a form input element or a component.
v-bind	Dynamically bind one or more attributes, or a component prop to an expression.
v-if, v-else-if, v-else	Conditionally render the element based on the truthy-ness of the expression value.
v-for	Render the element or template block multiple times based on the source data.
v-on:click	Attaches an event listener to the element.

https://vuejs.org/api/built-in-directives.html#built-in-directives

Directive v-model data-bind input elements

```
<div id="app">
  <p><label>Nom: <input v-model="name"></label></p>
  <p>Hello {{name}}</p>
</div>

<script src="https://unpkg.com/vue@3"></script>
```

```
Vue.createApp({
  data() {
    return {
      name: ''  // Important attribute must exist on data object!
    };
  }
}).mount('#app');
```

Directive v-model modifiers

.trim - trim input
.lazy - sync after change event

```
<div id="app">
  <p><label>Nom: <input v-model.trim.lazy="name"></label></p>
  <p>Hello {{name}}</p>
  <pre>{{name}}</pre>
</div>

<script src="https://unpkg.com/vue@3"></script>
```

```
Vue.createApp({
  data() {
    return {
      name: ''
    };
  },
}).mount('#app');
```

https://vuejs.org/guide/essentials/forms.html#modifiers

Directive v-model modifiers

.number - cast input string to numbers

```
<div id="app">
  <p>
    <label>Num1: <input v-model.number="num1" type="number"></label> +
    <label>Num2: <input v-model.number="num2" type="number"></label>
    = {{ num1 + num2 }}
  </p>
</div>

<script src="https://unpkg.com/vue@3"></script>
```

```
Vue.createApp({
  data() {
    return {
      num1: 4,
      num2: 2
    };
  },
}).mount('#app');
```

https://vuejs.org/guide/essentials/forms.html#number

Directive show / hide elements

Conditionally render the element based on the truthy-ness of the expression value.

```
<div v-if="type === 'A'"> A </div>
```

Toggle’s the element’s display CSS property based on the truthy-ness of the expression value.

```
<div v-show="type === 'B'"> B </div>
```

```
<div id="app">
  <p><label>Nom: <input v-model="type"></label></p>
  <div v-if="type === 'A'"> A </div>
  <div v-show="type === 'B'"> B </div>
</div>

<script src="https://unpkg.com/vue@3"></script>
```

```
Vue.createApp({
  data() {
    return {
      type: 'A'
    };
  },
}).mount('#app');
```

Directive conditional

```
<div v-if="type === 'A'">
  A
</div>

<div v-else-if="type === 'B'">
  B
</div>

<div v-else-if="type === 'C'">
  C
</div>

<div v-else>
  Not A/B/C
</div>
```

Directive repeat elements

```
<div v-for="item in items">
  {{ item.text }}
</div>
<div v-for="(val, index) in array"></div>
<div v-for="(val, key) in object"></div>
<div v-for="(val, key, index) in object"></div>
```

`v-bind:key="item.id"` needed on custom elements or for move ordering purpose

```
<div id="app">
  <ul>
    <li v-for="name in names">{{ name }}</li>
  </ul>
</div>

<script src="https://unpkg.com/vue@3"></script>
```

```
Vue.createApp({
  data() {
    return {
      names: ['Fred','Alice', 'Bob']
    };
  },
}).mount('#app');
```

Vue Instance Methods

- Runs whenever an update occurs
- Not cached
- Getter/setter
- Typically invoked from v-on/@, but flexible

```
<div id="app">
  <button v-on:click="addOne">Click Me!</button> {{count}}
</div>

<script src="https://unpkg.com/vue@3"></script>
```

```
Vue.createApp({
  data() {
    return {
      count: 0
    };
  },
  methods: {
    addOne() {
      this.add(1);
    },
    add(nb) {
      this.count += nb;
    }
  }
}).mount('#app');
```

Methods and Events

Methods used to handle events can access event object through $event

For common event manipulation there are helpers

.stop	call event.stopPropagation().
.prevent	call event.preventDefault().
.{keyAlias}	only trigger handler on certain keys.
.right	only trigger handler for right button mouse events.
.left	only trigger handler for left button mouse events.
.middle	only trigger handler for middle button mouse events.

https://vuejs.org/guide/essentials/event-handling.html#event-modifiers

```html
<div id="app">
  <form v-on:submit.prevent="onSubmit"></form>

  <!-- chain modifiers -->
  <button @click.stop.prevent="showCoordinates($event)">Where did you click?</bu

  <!-- key modifier using keyAlias -->
  <input @keyup.enter="onEnter">
</div>

<script src="https://unpkg.com/vue@3"></script>
```

```javascript
Vue.createApp({
  data() {
    return {
      count: 0
    };
  },
  methods: {
    showCoordinates(evt) {
      console.log(evt.clientX, evt.clientY);
    },
    onEnter() {window.alert('hello');},
    onSubmit() {}
  }
}).mount('#app');
```

Exercice Vue.js: model, if, for

Editer App.vue pour créer la page suivante :

- Cliquer sur « Ajouter » ajoute le montant qui se trouve dans l'input à une liste en dessous.
- Si le montant n'est pas supérieur à 0 il faut indiquer un message d'erreur, autrement le message est caché.
- On peut effacer un élément de la liste avec le bouton « X »

Lab 2: deploy to github pages

Step 0: Install

Create a github.com account and install git-scm.com to have a version of `git`.

Step 0: Use LF also on windows

Create `.gitattributes` with content:

```
# Force all line endings to be \n
* text eol=lf

###############################################################
# git can corrupt binary files if they're not set to binary.
###############################################################

# Apple office documents are actually folders, so treat them as binary.
*.numbers binary
*.pages binary
*.keynote binary

# Image files
*.png binary
*.jpg binary
*.jpeg binary
*.gif binary
*.webp binary
*.ico binary

# Movie and audio files
```

Step 1: Git Main

- Create github repository labo-xyz
- Init a git repository
- Add and Commit your code.
- Add remote
- Push your code to Github.

Step 2: Production version test

Create a built, mimified version of your page with your .

```
npm run build
```

Notice that you have a dist folder with this new content.

Test production version with:

```
npm run preview
```

Step 3: Deploy static site on Github

Github offers to serve static web pages
https://pages.github.com/

There are two options:

- User site: http://usernameABC.github.io by creating a special repository with the name: usernameABC.github.io
- Projet site: http://usernameABC.github.io/repositoryYXZ by creating a special branch gh-pages inside the repositoryXYZ

Setup Github and gh-pages via Github Actions

- configure github Actions to build and deploy to gh-pages
- configure vue to support /labo-xyz/ in production
- commit and push

Github Action Config

- create folder `.github\workflows`
- add this file `build_deploy.yml`

```yaml
name: Build and Deploy to GH-Pages

on:
  push:
    branches:
      - master
      - main

permissions:
  contents: write

jobs:
  build_deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v2

      - name: Setup Node
        uses: actions/setup-node@v2
```

Configure Vite publicPath

Edit `vite.config.js` file to handle subfolder deployment.

```
export default defineConfig({
  ...,
  base: process.env.NODE_ENV === "production" ? "/labo-test/" : "/",
});
```

Try to deploy

After a successful `npm run build` commit all changes and push:

```
git add . --all
git commit
git push
```

The site can be accessed at: https://heg-web.github.io/labo-test/ when action workflow is finished.

You did it!

Your First Single Page Web Application is deployed in Production!

Directive dynamique attributes

Dynamically bind one or more attributes, or a component prop to an expression.

```
<div v-bind:attributes=""></div>

<img v-bind:src="'/base/' + n + '.jpg'">
```

A short form exists

```
<div :attributes=""></div>

<img :src="'/base/' + n + '.jpg'">
```

Directive bind multiples attributes

```
<div :attr1="" :attr2=""></div>

<div v-bind="{attr1: '', attr2: ''}"></div>
```

Directives bind dynamic css classes

```
<div v-bind:class="{ active: isActive }"></div>
```

string	:class="'classString'"
variable	:class="classNameVariable"
array	:class="[classVarA, 'classNameB']"
object	:class="{className: someBooleann}"
ternary	:class="bool ? 'active' : 'inactive'"
method	:class="classNameReturningFunction()"

https://speakerdeck.com/bhawkes/introduction-to-vue-js?slide=27

Computed Properties

- Runs only when a dependency has changed
- Cached
- Should be used as a property, in place of data

```
<div id="app">
  <p>{{date1}} {{date2()}} {{count}}</p>
  <button @click="count+=1">render</button>
</div>

<script src="https://unpkg.com/vue@3"></script>
```

```
Vue.createApp({
  data() { return { count: 0 };
  },
  computed: {
    date1() { return new Date(); }
  },
  methods: {
    date2() { return new Date(); }
  }
}).mount('#app');
```

Computed Properties Setter

Computed properties are by default getter-only, but you can also provide a setter when you need it:

```
// ...
computed: {
  fullName: {
    // getter
    get() {
      return this.firstName + ' ' + this.lastName;
    },
    // setter
    set(newValue) {
      const names = newValue.split(' ');
      this.firstName = names[0];
      this.lastName = names[names.length - 1];
    }
  }
}
// ...
```

Vue Concepts Summary

Vue.js Concepts	Description
ViewModel	the data shown to the user in the view and with which the user interacts
View	what the user sees (the DOM)
Template	HTML with additional markup
Directives	extend HTML with custom attributes and elements
Components	a special kind of tag used for component-based application structure
Expressions	access variables and functions from the ViewModel
Computed	Computed properties are cached, and only re-computed on reactive dependency changes
Methods	Methods to be mixed into the Vue instance

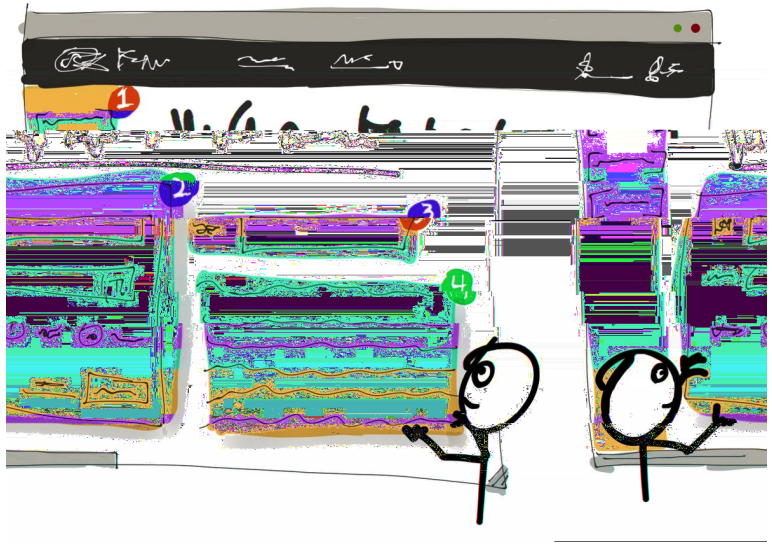
Exercice Vue.js: computed and methods

Editer App.vue pour créer la page suivante (avec des computed et des methods):

Component Based Applications

Components provide a way to write small parts with a consistent API that can easily be orchestrated as part of a larger screen, application or system.

An encapsulated set of behaviors or process and logic, with a well-known interface or API to access that component’s functionality.

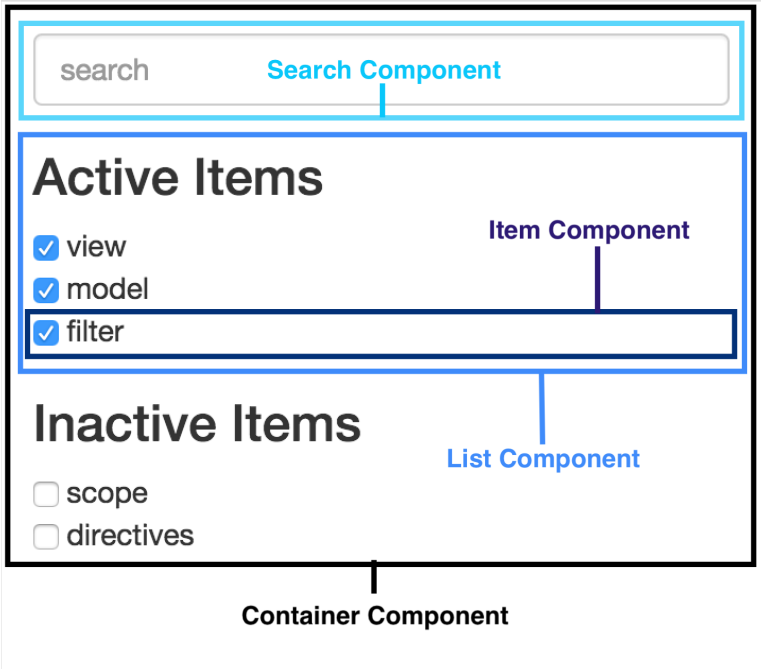


https://derickbailey.com/2015/08/26/building-a-component-based-web-ui-with-modern-javascript-frameworks/

Advantages of Components

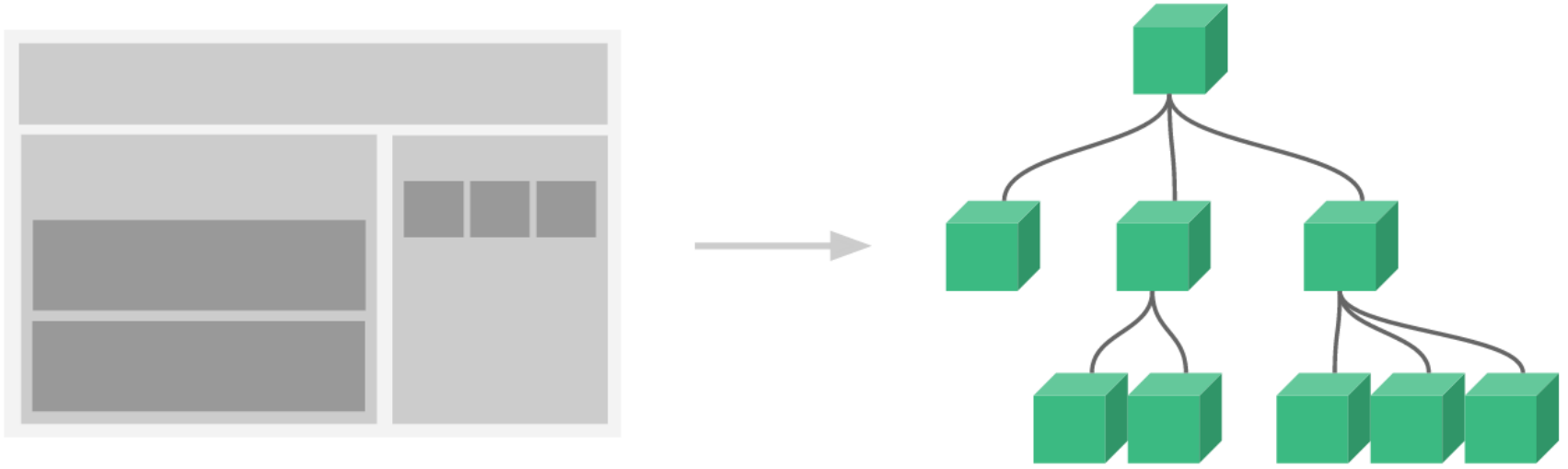
A component is a small, potentially re-usable set of logic, behaviors and interface elements

- Reusable
- Data flow boundaries in/out
- Isolated scope
- Simple or with state easier to predict
- Testable



http://busypeoples.github.io/post/thinking-in-components-angular-js/

Components in Vue.js



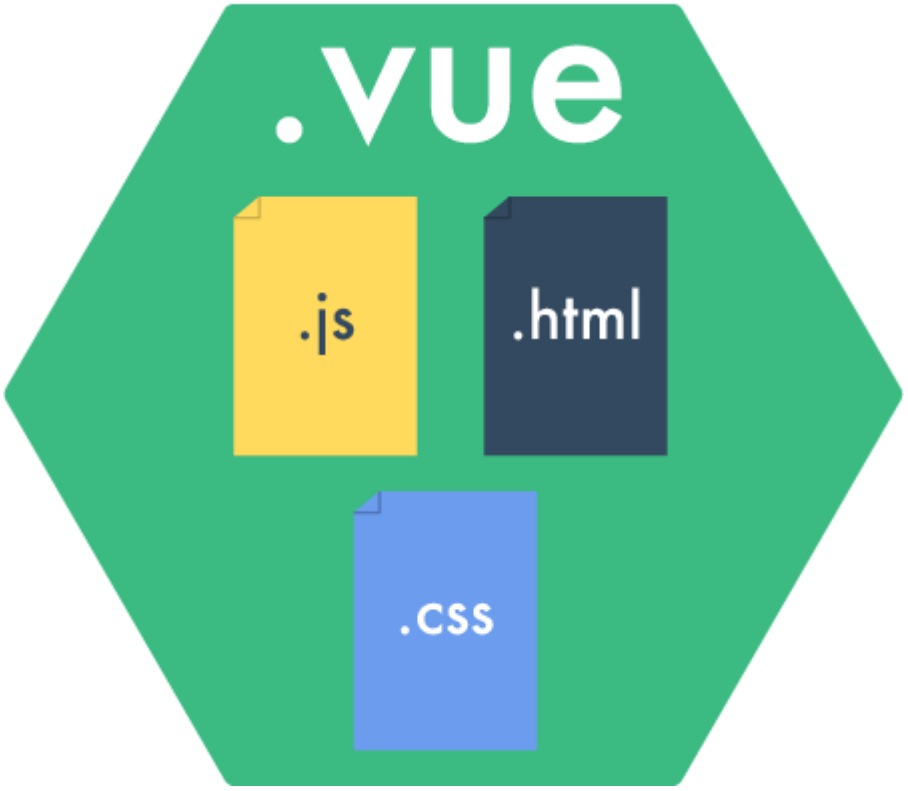
```html
<div id="app">
  <my-tag></my-tag>
</div>

<script src="https://unpkg.com/vue@3"></script>
```

```
app = Vue.createApp({});
app.component('my-tag', {
  template: `<div>{{ text }}<div>`,
  data() {
    return {
      text: 'Hello From My Tag!'
    }
  }
});
app.mount('#app');
```

> *Note that Vue does not enforce the W3C rules for custom tag names (all-lowercase, must contain a hyphen) though following this convention is considered good practice.*

Components using .vue



https://speakerdeck.com/bhawkes/introduction-to-vue-js

Global vs Local Registrations

Global

```
app.component('my-component', {
  // options
})
```

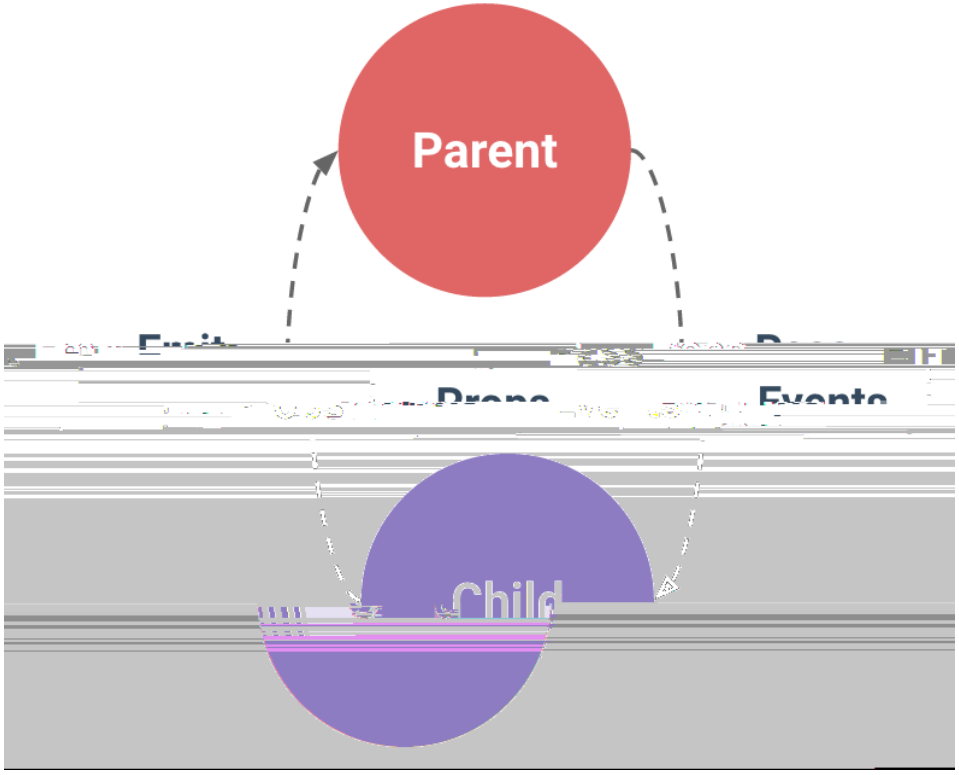
Local

```
import MyTag from './components/MyTag';
const Child = {
  template: '<div>A custom component!</div>'
}
Vue.createApp({
  // ...
  components: {
    // <my-component> will only be available in parent's template
    'my-component': Child,
    'my-tag': MyTag // shorthand MyTag possible
  }
}).mount('#app');
```

Components Communication in Vue.js

Component receive data through attributes binding by exposing properties.

Component send changes up to the parent by emiting events, to avoid mutations!



Components Properties in Vue.js

In javascript use **camelCasing**, which will be converted to **kebab-case** in HTML.

In compoonent expose properties

```
{
  //...
  props: ['myprop', 'myProp2'],
  // ...
  mounted(){
    this.myProp2;
  }
}
```

```
<my-comp v-bind:myprop="" :my-prop2></my-comp>
```

Warning: changing an attribute of a bound object mutates it's state outside the scope of the component.

Components Emitting Events

Define

```
app.component('myCheckbox', {
  emits: ['hello'],
  methodes: {
    change() {
      this.$emit('hello', 'world');
    }
  },
  template: `<label>{{name}} <span @click="change"> </span></label>`
});
```

Use

```
<my-checkbox @hello="doSomethingWithWorld"></my-checkbox>
```

Complet Component Example

```
<div id="app">
  <my-countdown v-bind:start="count" v-on:zero="alarm()"></my-countdown>
</div>
<script src="https://unpkg.com/vue@3"></script>
```

```
// Vue component
MyCountdown = {
  template: `<div v-on:click="countDown()">{{ count }}</div>`,
  props: ['start'],
  emits: ['zero'],
  data() {
    return {
      count: this.start
    };
  },
  methods: {
    countDown() {
      if (this.count <= 0) return;
      this.count = this.count -1;
      if(this.count === 0) {
        this.$emit('zero');
      }
    }
  }
```

Components Properties Validation in Vue.js

Define

```
app.component('myCheckbox', {
  props: {
    isOk: {
      type: Boolean,
      required: true,
      default: false
    },
    name: {
      type: String,
      required: true,
      default: 'Bob'
    }
  },
  template: `<label>{{name}} <span v-if="isOk"> </span></label>`
});
```

Use

```
<my-checkbox :is-ok="booleanValue"></my-checkbox>
```

Components Properties Validation GOTCHA

Objects and arrays need their defaults to be returned from a function:

```
text: {
  type: Object,
  default: function () {
    return { message: 'hello mr. magoo' }
  }
}
```

Example

To listen for a native event on the root element of component, .native has to be added:

```
<my-checkbox @click.native="doSomething"></my-checkbox>
```

By default, v-model on a component uses **value** as the prop and **input** as the event

Custom Events

Events can also be used to communicated between components. Vue.js events do not *bubble up* or *trickle down* the tree.

We attach and emit on a common object for example the **$root**.

```
// in component A
const clickHandler = clickCount => {
  console.log(`Oh, that's nice. It's gotten ${clickCount} clicks! :)`)
}
this.$root.$on('i-got-clicked', clickHandler);

// in component B
this.$root.$emit('i-got-clicked', this.clickCount);

// in component A to stop listening
// Stop listening.
this.$root.$off('i-got-clicked', clickHandler);
```

Transitions

```
<transition name="fade">
  <p v-if="show">hello</p>
</transition>

<transition-group name="flip-list" tag="ul">
  <!-- multiple elements / move animations -->
</transition-group>
```

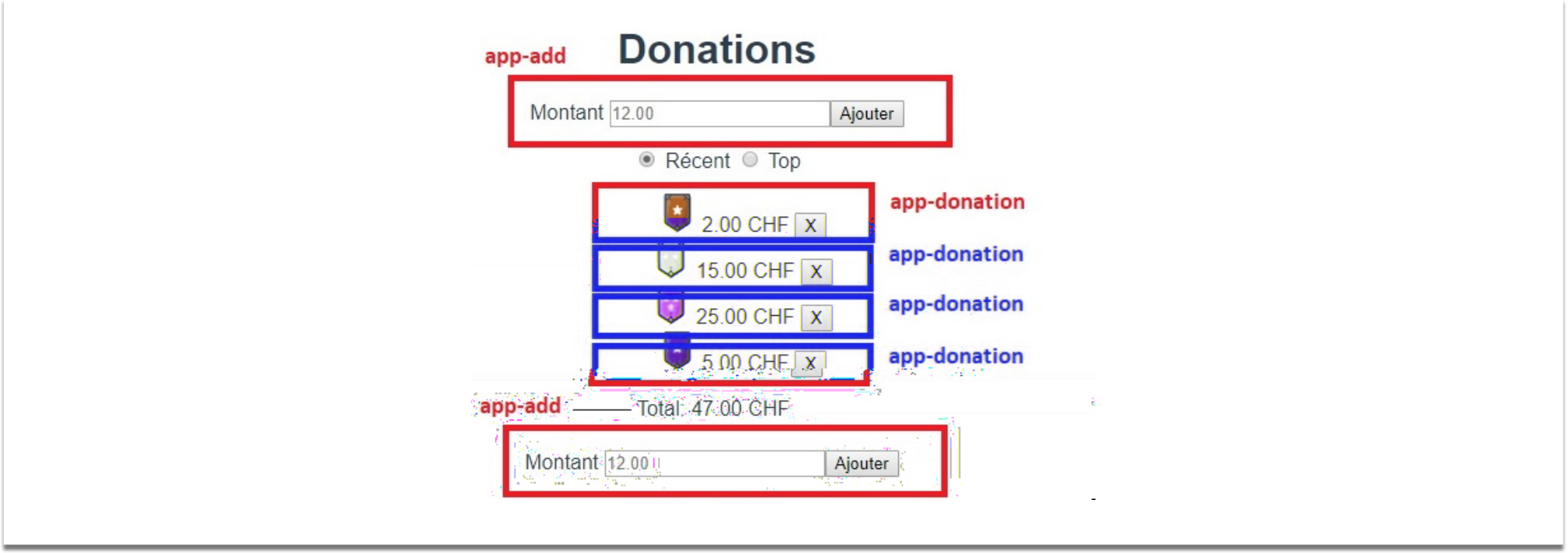
```
.fade-enter-active, .fade-leave-active {
  transition: opacity 1s
}
.fade-enter-from, .fade-leave-to {
  opacity: 0
}
.flip-list-move, .flip-list-enter-active, .flip-list-leave-active {
  transition: transform 1s;
}
.flip-list-enter-from, .flip-list-leave-to {
  transform: scale(0);
}
```

https://vuejs.org/guide/built-ins/transition.html#css-based-transitions

https://vuejs.org/guide/built-ins/transition-group.html#move-transitions

Exercice Vue.js: components

Transformer (Refactor) l'application en composant selon:



- Créer une fonction reutilisable toChf() qui retourne un nombre formaté en chf.
- Ajouter, transformer le code, pour que l'application fonctionne encore de la même façon.

Exercice Vue.js: transitions

Ajouter des effets de transitions au message et à la liste:

JSON

JavaScript Object Notation is a lightweight data-interchange format. It is easy for *humans* to **read and write**. It is easy for *machines* to **parse and generate**. It is based on a subset of the JavaScript Programming Language

```
{
  "key_string": "hello",
  "key_number": 3,
  "key_array": ["some text", 34]
  "key_object": {
    "other_key": "value"
    "key_boolean": true,
    "null possible": null
  }

}
```

http://json.org/

JSON API in JavaScript

JSON.stringify(*object*)	create a JSON_string
JSON.parse(*JSON_string*)	create an object from a string

```
//optional formatter and indentation spacing for pretty-print
JSON.stringify( {hello: {text: 'world'}}, null, 2 )
//results in the following string
'{
  "hello": {
    "text": "world"
  }
}'
```

localStorage

Interface of the Web Storage API provides access to storage for a particular domain.

localStorage.length	Returns an integer representing the number of data items stored in the Storage object.
localStorage.key(number)	will return the name of the nth key in the storage.
localStorage.getItem(key)	will return that key's value.
localStorage.setItem(key, value)	will add that key to the storage, or update that key's value if it already exists.
localStorage.removeItem(key)	will remove that key from the storage.
localStorage.clear()	will empty all keys out of the storage.

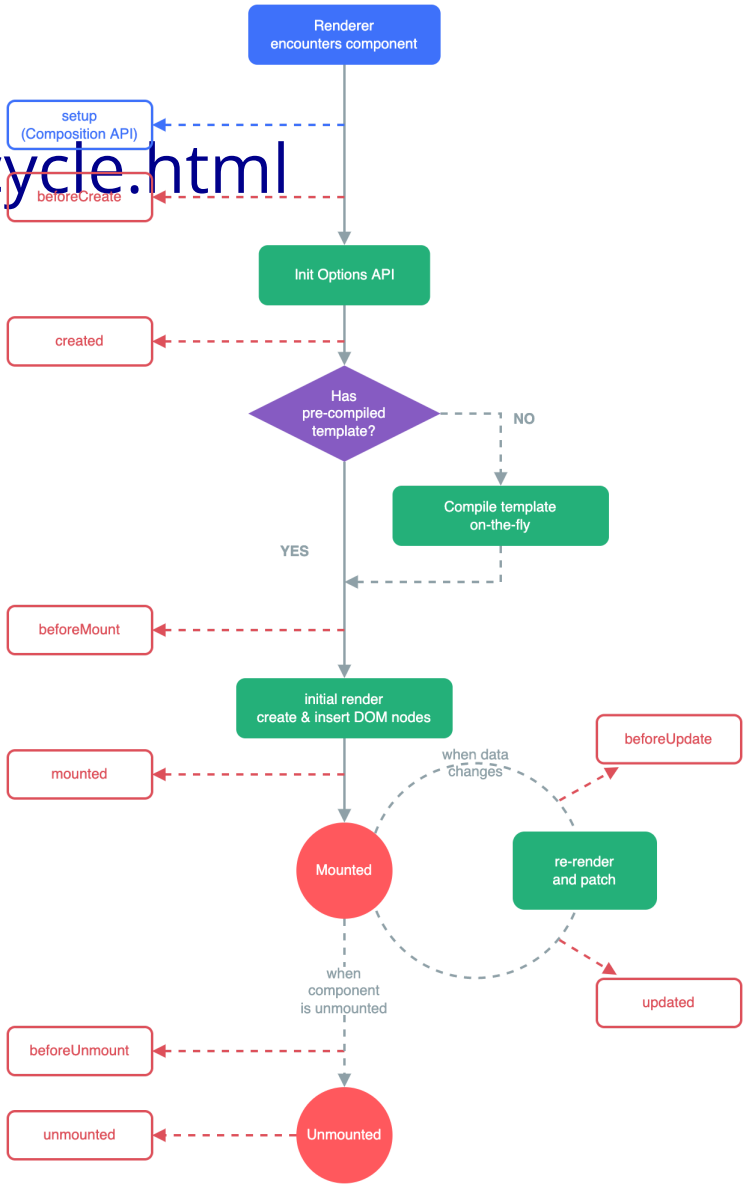
localStorage content can be viewed in chrome developer tools resource tab

https://developer.mozilla.org/en-US/docs/Web/API/Storage

Lifecycle Hooks

https://vuejs.org/guide/essentials/lifecycle.html

created, mounted, unmounted, ...



Exercice Vue.js: localStorage

Persister les données localement

- Sauvegarder dans le localstorage la liste à chaque fois que celle-ci change.
- Au chargement de la page, s'il y a une liste sauvegardé dans le localstorage, alors récupérer cette valeur.

Multiples Views and Router

A SPA has to support multiple virtual views to simulate pages.

This can be achieved with a router, routes and components.

```
const router = createRouter({
  routes: [
    // dynamic segments start with a colon
    { path: '/user/:id', name: 'user', component: User }
  ]
});
```

```
<!-- will host the component corresponding to the route -->
<router-view></router-view>

<!-- Vue instance has a special property with route params -->
<div>{{ $route.params.id }}</div>

<!-- create links by lookup of the route -->
<router-link :to="{ name: 'user', params: { id: 123 }}">User</router-link>
```

```
// changing route in code.
this.$router.push({ name: 'user', params: { id: 123 }})
```

Example Flow of Components and Router

$route.params vs component props

Using $route in your component creates a tight coupling with the route (views not reusable components).

$route.params are visible in the URL and string only!

Props are parameters of a component to pass down variables of any type.

It is possible to Pass Props to Route Components to decouple them.

Lifecycle Hooks

Router also has some hooks

https://router.vuejs.org/guide/essentials/dynamic-matching.html#reacting-to-params-changes

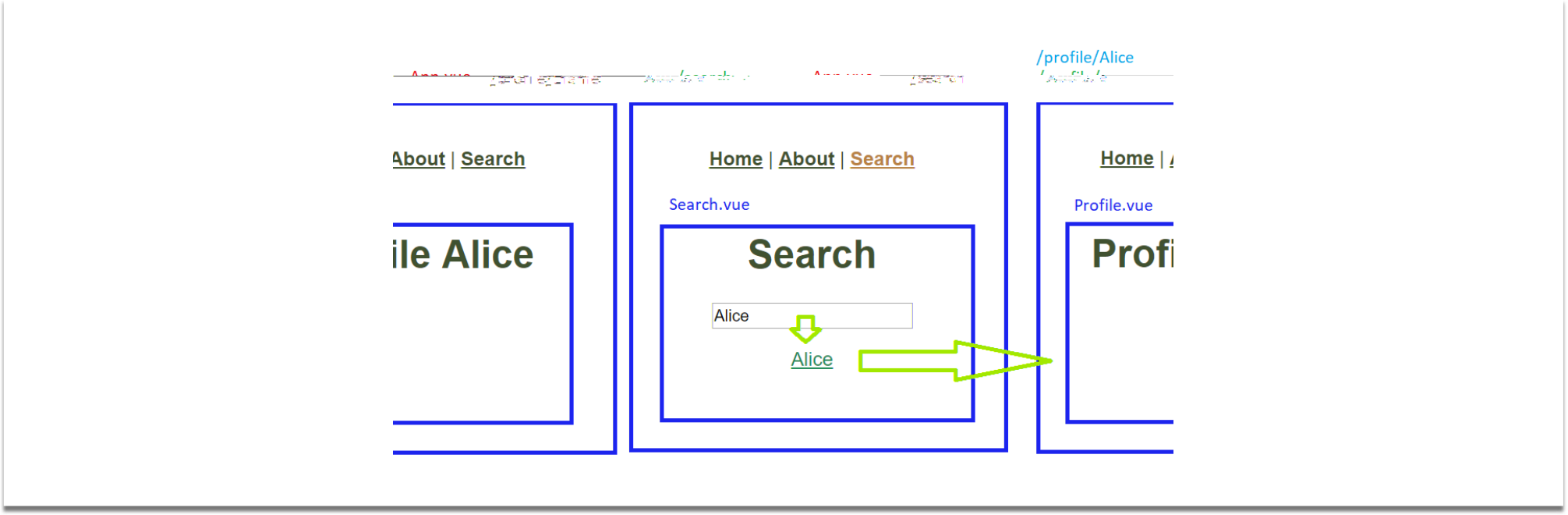
https://router.vuejs.org/guide/advanced/navigation-guards.html#global-before-guards

```
beforeRouteUpdate (to, from, next) {
  // react to route changes...
  // don't forget to call next()
}
```

Exercice Vue.js: router

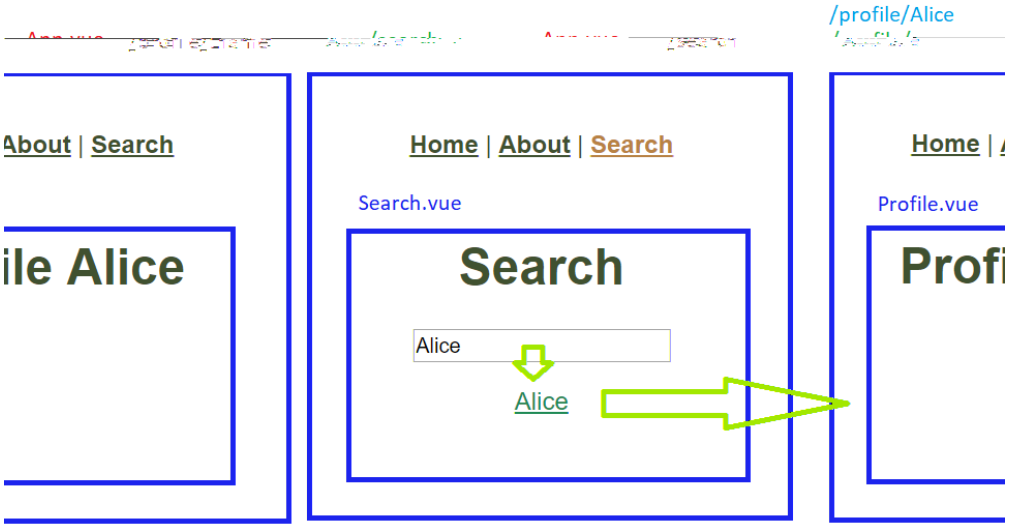
Remplacer `createWebHistory` avec `createWebHashHistory` dans `src/router/index.js`

Ajouter deux pages supplémentaires pour créer ceci:



Exercice Vue.js: router

- Un lien principal Search,
- Une page Search qui génère dynamiquement l'attribut `to` d'un `<router-link>` selon le texte d'un input.
- Une page Profile qui affiche le nom du paramètre de l'URL.



Vue.js Advanced Stuff

Access DOM through Refs

ref is used to register a reference to an element or a child component. The reference will be registered under the parent component's $refs object.

```
<div id="app">
  <div ref="someID"></div>
</div>

<script src="https://unpkg.com/vue@2"></script>
```

```
Vue.createApp({
  mounted() {
    this.$refs.someID.innerText = 'DOM Direct Manipulation is BAD!';
  }
}).mount('#app');
```

Watches

Vue.js more

- Composition API
- Mixins
- Custom Directives
- Slots
- Vue.nextTick
- Plugins
- Server-Side Rendering
- Code Splitting
- Route lazy loading
- State Management: Pina

Resources

- https://vuejs.org/guide/introduction.html
- https://vuejs.org/api/
- https://www.grafikart.fr/formations/vuejs (attention Vue.js 2)

JavaScript Asynchronous Programming

Asynchronous programming techniques

JavaScript relies on asynchronous programming:

- The JS engine is single-threaded. For this reason, IO operations have to be non-blocking.
- An event loop is used both in the browser and on the server (node.js):
  - As the program executes, events are added to a queue. Every event has an associate callback function.
  - A dispatcher takes the next event in the queue and invokes the callback function (on the single thread).
  - When the callback function returns, the dispatcher takes the next event in the queue, and continues forever (it’s an event loop).

Callback

```
setTimeout( function() {
    console.log("the callback has been invoked");
}, 2000);
```

An event will be added to the queue in 2000 ms. In other words, the function passed as the first argument will be invoked in 2 seconds or more (the thread might be busy when the event is posted...).

```
$(document).mousemove( function(event) {
    $("span").text(event.pageX + ", " + event.pageY);
});
```

An event will be added to the queue whenever the mouse moves. In each case, the callback function has access to the event attributes (coordinates, key states, etc.).

```
$.get( "ajax/test.html",
```

An event will be added when the AJAX request has been processed, i.e. when a response has been received. The callback function has access to the payload.

Beyond simple callbacks…

- The principle of passing a callback function when invoking an asynchronous operation is pretty straightforward.
- Things get more tricky as soon as you want to coordinate multiple tasks. Consider this simple example…

Do this first…

… when done, do this.

A first attempt…

```
var milkAvailable = false;

function milkCow() {
  console.log("Starting to milk cow...");
  setTimeout(function() {
    console.log("Milk is available.");
    milkAvailable = true;
  },  2000);
}

milkCow();
console.log("Can I drink my milk? (" + milkAvailable + ")");
```

A first attempt…

```
var milkAvailable = false;

function milkCow() {
  console.log("Starting to milk cow...");
  setTimeout(function() {
    console.log("Milk is available.");
    milkAvailable = true;
  },  2000);
}

milkCow();
console.log("Can I drink my milk? (" + milkAvailable + ")");
```

FAIL

Fixing the issue with a callback...

```
var milkAvailable = false;

function milkCow(done) {
  console.log("Starting to milk cow...");
  setTimeout(function() {
    console.log("Milk is available.");
    milkAvailable = true;
    done();
  }, 2000);
}

milkCow( function() {
  console.log("Can I drink my milk? (" + milkAvailable + ")");
});
```

Fixing the issue with a callback...

```
var milkAvailable = false;

function milkCow(done) {
  console.log("Starting to milk cow...");
  setTimeout(function() {
    console.log("Milk is available.");
    milkAvailable = true;
    done();
  }, 2000);
}

milkCow( function() {
  console.log("Can I drink my milk? (" + milkAvailable + ")");
});
```

SUCCESS

Beyond simple callbacks...

- Ok... but what happens when I have more than 2 tasks that I want to execute in sequence?
- Let’s say we want to have the sequence B, C, D, X, Y, Z, E, F, where X, Y and Z are asynchronous tasks.

Beyond simple callbacks...

- Ok... but what happens when I have more than 2 tasks that I want to execute in sequence?
- Let's say we want to have the sequence B, C, D, X, Y, Z, E, F, where X, Y and Z are asynchronous tasks.

```
function f() {
  syncB();
  syncC();
  syncD();
  asyncX();
  asyncY();
  asyncZ();
  syncE();
  syncF();
}
```

```
B  result  available
C  result  available
D  result  available
E  result  available
Z  result  available
Y  result  available
F  result  available
X  result  available
```

Sequence with callbacks

Sequence with callbacks

```
function f() {
  syncB();
  syncC();
  syncD();
  asyncX(function() {
    asyncY(function() {
      asyncZ(function() {
        syncE();
        syncF();
      });
    });
  });
}
```

```
B  result  available
C  result  available
D  result  available
X  result  available
Y  result  available
Z  result  available
E  result  available
F  result  available
```

Sequence with callbacks

```
function f() {
  syncB();
  syncC();
  syncD();
  asyncX(function() {
    asyncY(function() {
      asyncZ(function() {
        syncE();
        syncF();
      });
    });
  });
}
```

```
B  result  available
C  result  available
D  result  available
X  result  available
Y  result  available
Z  result  available
E  result  available
F  result  available
```

But welcome to the **"callback hell"** aka **"callback pyramid"**

Callback parallel tasks

- Now, let's imagine that we have 3 asynchronous tasks. We want to invoke them in parallel and wait until all of them complete.
- Typical use case: you want to send several AJAX requests (to get different data models) and update your DOM once you have received all responses.

```
function f( done ) {
 async1( function( r1 ) {
    reportResult( r1 );
  });
  async2( function( r2 ) {
    reportResult(r2);
  });
  async3( function( r3 ) {
    reportResult( r3 );
  })
  done();
}
```

Callback parallel tasks

- Now, let's imagine that we have 3 asynchronous tasks. We want to invoke them in parallel and wait until all of them complete.
- Typical use case: you want to send several AJAX requests (to get different data models) and update your DOM once you have received all responses.

```
function f( done ) {
 async1( function( r1 ) {
    reportResult( r1 );
  });
  async2( function( r2 ) {
    reportResult(r2);
  });
  async3( function( r3 ) {
    reportResult( r3 );
  })
  done();
}
```

Double fail: not only is done() invoked to early, but also there is no result to send back...

Callback parallel tasks with counter

```
function f( done ) {
  <span class="fragment highlight-current-re
  var results = [];
  <span class="fragment highlight-current-re
  function reportResult( result ) {
    result.push( result );
    numberOfPendingTasks  = 1;
    if ( numberOfPendingTasks === 0 ) {
      done( null, results );
    }
  }
  </span><span class="fragment highlight-cur
  async1( function( r1 ) {
    reportResult( r1 );
  });
  async2( function( r2 ) {
    reportResult( r2 );
  });
  async3( function( r3 ) {
    reportResult( r3 );
  });</span>
```

When this reaches 0, I know that all the tasks have completed. I can invoke the "done" callback function that I received from the client. I can pass the array of results to the function.

When a task completes, it invokes this function and passes its result. The result is added to the array and the number of pending tasks is decremented.

The three tasks are asynchronous, so they pass their own callback functions and receive a result when the operation completes.

Async libs to the rescue: Promise

A **promise** must be in **one of three states**: *pending*, *fulfilled*, or *rejected*.

When *pending*, a promise:

- may transition to either the *fulfilled* or *rejected* state.

When *fulfilled*, a promise:

- **must not transition** to any other state.
- must have a **value**, which must not change.

When *rejected*, a promise:

- **must not transition** to any other state.
- must have a **reason**, which must not change.

https://github.com/promises-aplus/promises-spec

A promise must provide a then method to access its current or eventual value or reason. A promise's `then` method accepts two arguments:

- `promise.then( onFullfilled,  onRejected )`
- If `onFulfilled` is a function:
  - it must be called after promise is *fulfilled*, with promise's value as its first argument.
  - it must not be called before promise is *fulfilled*.
  - it must not be called more than once.
- If `onRejected` is a function,
  - it must be called after promise is *rejected*, with promise's reason as its first argument.
  - it must not be called before promise is *rejected*.
  - it must not be called more than once

https://github.com/promises-aplus/promises-spec

then must return a promise.

```
promise2  = promise1.then(onFulfilled,
onRejected);
```

- If either onFulfilled or onRejected returns a value x, run the Promise Resolution Procedure Resolve(promise2, x).
- If either onFulfilled or onRejected throws an exception e, promise2 must be rejected with e as the reason.
- If onFulfilled is not a function and promise1 is fulfilled, promise2 must be fulfilled with the same value as promise1.
- If onRejected is not a function and promise1 is rejected, promise2 must be rejected with the same reason as promise1.

https://github.com/promises-aplus/promises-spec

Promise in ECMAScript 2015

```
const promise = new Promise(function(resolve, reject) {
  // do a thing, possibly async, then…

  if (/* everything turned out fine */) {
    resolve("Stuff worked!");
  }
  else {
    reject(Error("It broke"));
  }
});

promise.then(function(result) {
  console.log(result); // "Stuff worked!"
}, function(err) {
  console.log(err); // Error: "It broke"
});
```

https://developers.google.com/web/fundamentals/getting-started/primers/promises

Chaining Transforming values

```
const promise = new Promise(function(resolve, reject) {
  resolve(1);
});

promise.then(function(val) {
  console.log(val); // 1
  return val + 2;
}).then(function(val) {
  console.log(val); // 3
})
```

Wait for all

```
Promise.all(arrayOfPromises).then(function(arrayOfResults) {
  //...
})
```

https://developers.google.com/web/fundamentals/getting-started/primers/promises

API and Remote Data

Example api: https://api.thecatapi.com/v1/images/search?limit=5&page=10&order=Desc

Chrome DevTools allows to view Network Traffic

Testing API advanced requests

Most API require additional header in addition of other request than GET.

Insomnia a tool to test apis

Getting JSON content with Fetch

```
fetch('./api/some.json')
  .then(response => {
      return response.json();
    }
  )
  .then(data => {
      console.log(data);
  })
  .catch(function(err) {
    console.log('Fetch Error :-S', err);
  });
```

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Getting JSON content with axios

```
npm install axios --save
```

```
import axios from 'axios';

axios.get('https://api.thecatapi.com/v1/images/search', {
          headers: {
            'x-api-key': 'DEMO-API-KEY'
          }
        })
  .then(response => {
    console.log(response.headers);
    console.log(response.data);
  })
  .catch(error => {
    console.log(error);
  });
```

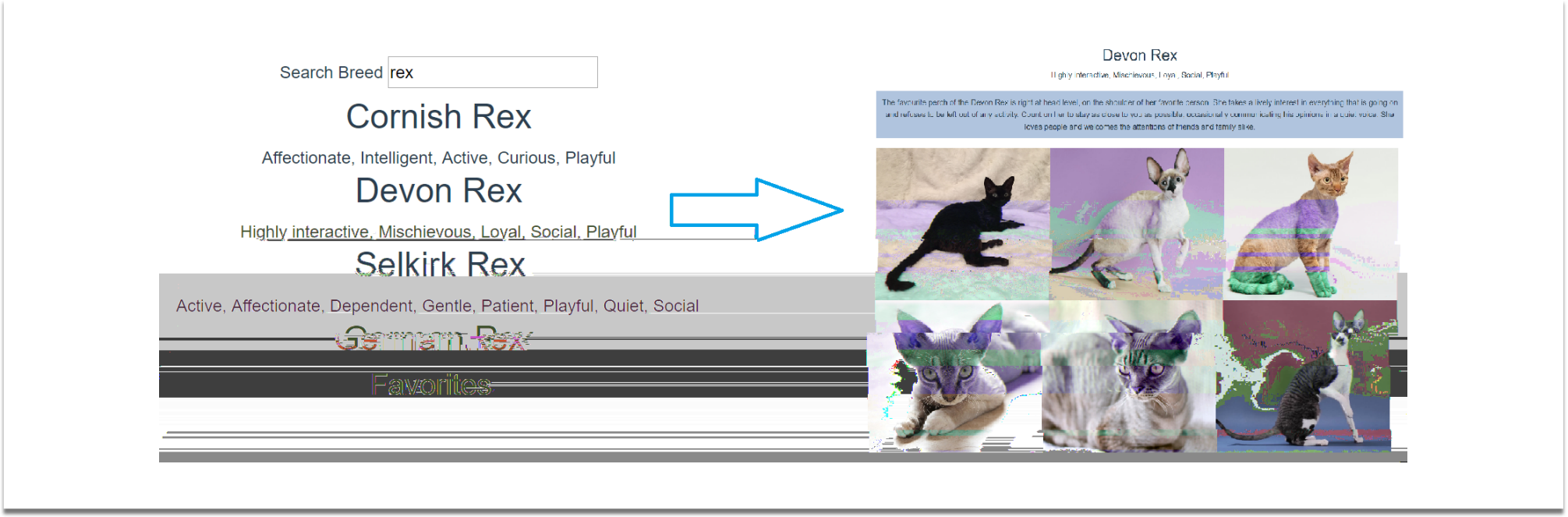
Load Data in Vue.js

```
import { createApp } from "vue";
import axios from "axios";

createApp({
  data() {
    return {
      dataLoaded: false,
      apiReply: {}
    };
  },
  methods: {
    loadData: function () {
      axios
        .get("https://api.thecatapi.com/v1/images/search")
        .then((response) => {
          this.apiReply = response.data;
          this.dataLoaded = true;
        });
    }
  },
  created() {
```

Exercice Vue.js: axios

Transformer le projet précédent pour utiliser le site thecatapi.com:



Exercice Vue.js: axios

- Entrer du texte dans la boîte de recherche affiche une liste de Breeds
- Cliquer sur un élément de la liste amène à une page détaille.
- La page détaille utilise le code id de la Breed récupérer en tant que paramètre d'URL pour rechercher 6 photos de cette race.

API with real data

- https://countapi.xyz/
- https://docs.thedogapi.com/
- https://docs.thecatapi.com/
- https://transport.opendata.ch/docs.html
- http://api.themoviedb.org/3/ //need proxy for api key or cors
- https://www.themealdb.com/api.php
- https://www.thecocktaildb.com/api.php

Google Cusom Search API

https://developers.google.com/custom-search/json-api/v1/reference/cse/list

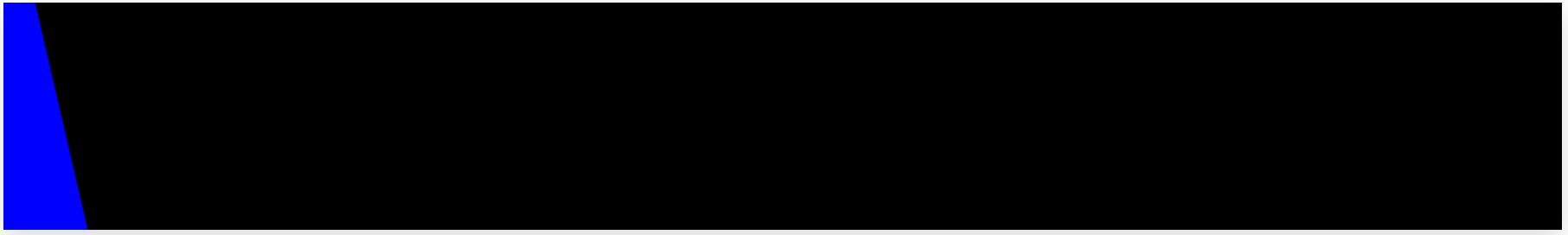
```
`https://www.googleapis.com/customsearch/v1?cx=011288001747608865807:a7rxzv4srri
```

Async / Await

```
async function myFirstAsyncFunction() {
  try {
    const fulfilledValue = await promise;
  }
  catch (rejectedValue) {
    // …
  }
}
```

```
function logFetch(url) {
  return fetch(url)
    .then(response => response.text())
    .then(text => {
      console.log(text);
    }).catch(err => {
      console.error('fetch failed', err);
    });
}

async function logFetch(url) {
  try {
    const response = await fetch(url);
    console.log(await response.text());
  }
  catch (err) {
    console.log('fetch failed', err);
  }
}
```



Using Material instead of bootstrap

https://quasar.dev/start/vite-plugin

Read the docs, copy examples, ...

Web App Manifest

Web App Manifests are one of the key pieces to making your web app look and feel like a native app

https://tomitm.github.io/appmanifest/

- Add to Homescreen
- Fullscreen
- Notifications
- Meta viewport
- Colors
- Zoom, touch interactions

Synchronised persistent datastorage

Firebase Database

Install Firebase

```
$ npm install -g firebase-tools
```

Available commands: login, init, serve, deploy

Integration with Vue.js

```
$ npm install firebase --save
```

To use login UI for firebase authentification

```
$ npm install firebaseui --save
```

https://github.com/firebase/FirebaseUI-Web

Documentation

- https://firebase.google.com/
- https://firebase.google.com/docs/web/setup#add-sdks-initialize
- https://firebase.google.com/docs/auth/web/github-auth

```
// plugins/firebase.js
import Vue from "vue";
import firebase from "firebase/compat/app";
import "firebase/compat/auth";
import "firebase/compat/database";
// import "firebase/compat/storage";
// import "firebase/compat/firestore";

// Initialize Firebase
// Copy from google firebase console (Authentication>Web Setup)
const config = {
  apiKey: "AIzaSyDZt98CIUYUBPeW32wvtA5hWOFlSLp03C0",
  authDomain: "ptw.firebaseapp.com",
  databaseURL: "https://ptw.firebaseio.com",
  projectId: "firebase-ptw",
  storageBucket: "firebase-ptw.appspot.com",
  messagingSenderId: "281865054216",
  appId: "1:281865054216:web:e2d79f491c1cc7d1"
};
export default firebase.initializeApp(config);
```

Lab: Firebase Messages

Lab: Firebase Game

https://gist.github.com/bfritscher/f15258ad2161eda24a3215963

Setup Firebase User Security

```
{
    "rules": {
       "users": {
      "$uid": {
        // grants write access to the owner of this user account whose uid must
        ".write": "auth !== null && auth.uid === $uid",
        ".read": "auth !== null && auth.uid === $uid"
        }
      }
    }
}
```

https://firebase.google.com/docs/database/security/quickstart

Firebase Login:

https://gist.github.com/bfritscher/dea68fd13dbd172647eb60ebe

Serverless: Firebase Cloud Functions

https://github.com/firebase/functions-samples/tree/master/exif-images

https://firebase.google.com/docs/reference/functions/functions.s

https://cloud.google.com/vision/docs/reference/libraries#client-libraries-install-nodejs

https://firebase.google.com/docs/functions/config-env

Firebase Functions: Keeping Secrets

```
$ firebase functions:config:set service.name="value"
firebase functions:config:get
```

```
functions.config().someservice.id

exports.groupA = {
  function1: functions.https.onRequest(...);
  function2: functions.database.ref('\path').onWrite(...);
}
exports.groupB = require('./groupB');
```

Deploy to Firebase

```
$ firebase deploy --only functions
$ firebase deploy --only hosting
$ firebase deploy --only functions:function1,function2
```

Possible next steps after deploy

- Analytics & SPA
  - Virutal page views
  - Events
- A/B testing your site!


To infinity
and beyond...