Formation

Connection de la plateforme : <https://sign.m2iformation.fr/signin>

* Id : [nathan.hyppolite@dgfip.finances.gouv.fr](mailto:nathan.hyppolite@dgfip.finances.gouv.fr)
* Pin : 94953

Support de cours : <http://elmouelhia.free.fr/>

* Teaching
* Vues.js (double clic)

Il n’y a pas de norme pour le vue.js

Nommage de balise : <ma-balise> <MaBalise>

La single page évite de recharger la page ce qui permet de ne pas faire de multiple appelle

Création de projet Vue : prérequis la version de node ( voir doc )

* npm i -g vue
* npm init vue@latest

Info

**CDN**

Si 2 méthodes on le même nom le premier se fait écraser par la dernière sans message erreur.

For in permet de parcourir des liste avec le key et les autres index

Vu à plusieurs format

Le Sfc avec une extension.vue :

* js
* template(html)
* style scoped ( css )

Export default a priorité sur tous les exports et les écrase

Exemple :

```

export default function somme(a, b) {

    return a + b

};

export const produit = (a, b) => a \* b;

//export { somme, produit }

import produit from './fonctions.js'

console.log(produit(3, 5));

```

Le résultat sera 8

Vue.js utilise le lazy loding qui utilise l’asynchrone

Exemple :

```

import('./fonctions1.js')

    .then(m => console.log(m.produit(5, 6)))

    .catch(err => console.log(`Erreur : ${err}`))

    .finally(() => console.log('Fin de la promesse'))

console.log('Fin du programme');

```

**CLI**

Il y a 2 type de composant :

* Prédéfinis : v-if –on
* les composant utilisateur que l’on crée

si on ne veut plus a voir a supprimer c lib non utiliser il nous suffit aller dans le package.json

  },

    "rules": {

      "no-unused-vars":"off"

    }

On peut mettre une valeur par de faut avec un component avec slot

<template>

  <div>

    hello<slot> HUMAIN </slot>

  </div>

</template>

@ event (v-on)

: attribue ( v-bind) JavaScript -- > html

On peut rempalcer v-slot par #

Exemple :

<HelloWorld >

  <template v-slot:nom> Wick</template>

  <template  v-slot:prenom>john </template>

</HelloWorld>

<HelloWorld >

  <template #nom> Wick</template>

  <template  #prenom> john </template>

</HelloWorld>

On peut mettre une variable par défaut d’une autre manière avec #default

 <HelloWorld >

  <template #default> Wick</template>

  <template  #prenom> john </template>

</HelloWorld>

Le passage de donner avec v-slot est limiter car il faut declarer chaque champs

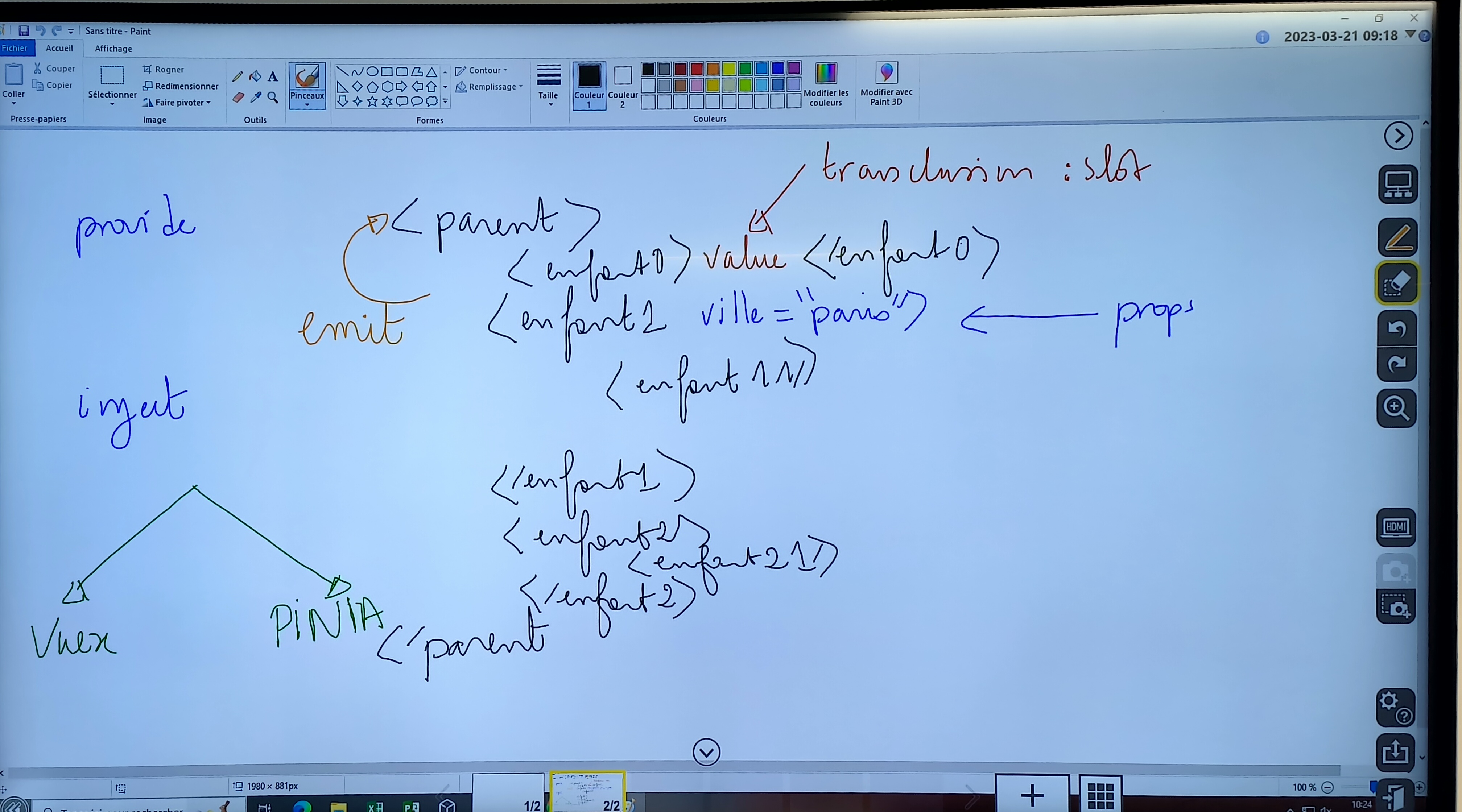
Les props permet de transmet des donner a un enfant

Emit permet de transmettre des données a un parent

Prodivide permet de transmettre les donner a enfant et petit enfant

Et il les reçoivent avec inject

Pina permet le transmission inter component et la gestion etat



Deux manière de construire

-api Option

- export default

- data

* Méthode
* Watch

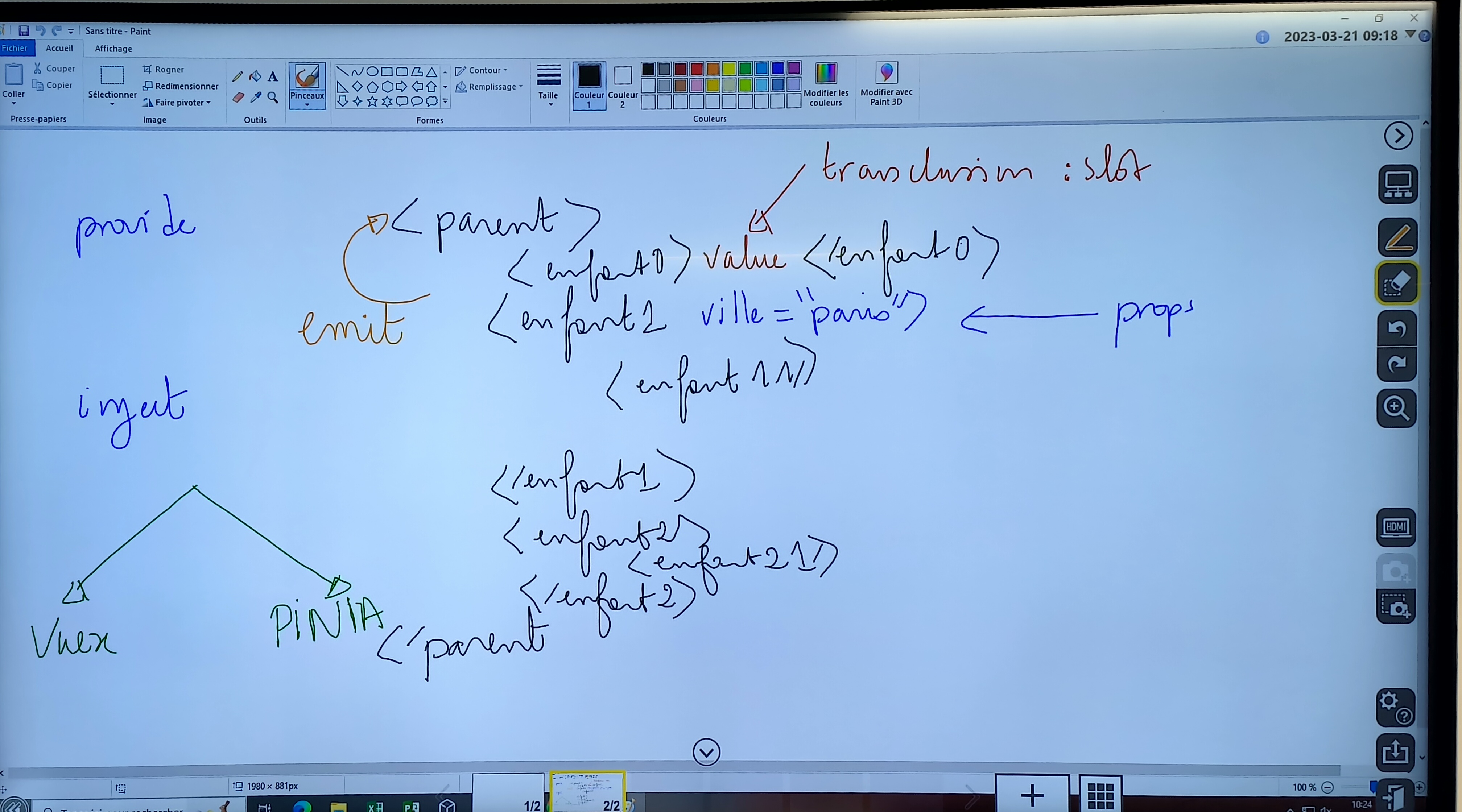
-api Composition

-

-

-

**Emit exemple**



**Fichier enfant**

**Template**

<div>

    <input type="text" v-model="nom" placeholder="nom">

    <button @click="$emit('envoyer', this.nom)">Envoyer au parent</button>

  </div>

**Script**

 emits : ['envoyer'],

**Fichier parent**

**Template**

 <p> Nom recu de l'enfant {{ nom }}</p>

 <HelloWorld @envoyer="afficherNom">

  <template #default> Wick</template>

  <template  #prenom> john </template>

</HelloWorld >

**Script**

   data(){

      return {

        nom :null

      }

    },

    methods : {

      afficherNom (data) {

        this.nom = data

      }

    }

**Pour recuperer les value dans un url classqiue on utilise « pa**

**O**n peut avoir des alias avec le router

 {

    path: '/',

    name: 'home',

    component: HomeView,

    alias: ['/home', '/accueil']

  },

Mais aussi rediriger

// redirection

  {

    path: '/person',

    redirect: '/personne'

  },