VUE.JS

Théorie et mise en pratique

PRÉSENTÉ PAR

Julie Metz, Adrien Lenoir, Alain Roos et Elisandre Laenens

SOMMAIRE

de la présentation

Les bases de Vue.Js

01.01

Les bases de Vue.JS

Introduction

Introduction

- Framework simple et efficace pour la construction d'interfaces utilisateur
- C'est un framework réactif, chaque action est directement mise à jour en temps réel
- Il permet de découper l'application en composants
- Créer des applications Web SPA (Single Page Applications)

01.01

Les bases de Vue.JS

Installation

Installation

- Package CDN

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

- Installation NPM pour WebPack

```
npm install vue@next
```

- Interface en ligne de commande

```
npm install -g @vue/cli
```

```
yarn global add @vue/cli
```

01.01

Les bases de Vue.JS

Le modèle MVVM

Le modèle MVVM

- Model - View - ViewModel
- **Model**
  - Les données nécessaires à l'application (data)
- **View**
  - L'interface accessible à l'utilisateur
- **ViewModel**
  - L'intermédaire bidirectionnel entre la VIEW et le MODEL

Le modèle MVVM

index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!--VIEW-->
    <div id="app"></div>
    <script>
        //VIEWMODEL
        const app = Vue.createApp({
            //MODEL
            data() {
                return {

                }
            }
        });
    </script>
</body>
</html>

```

01.01

Les bases de Vue.JS

L'affichage du MODEL dans la VIEW

L'affichage du MODEL dans la VIEW - 1

- L'interpolation de texte

Permet l'insertion d'une propriété MODEL dans l'InnerHTML d'une balise

```
<div id="app">
    <h2>{{titre}}</h2>
    <p>{{sousTitre}}</p>

</div>
<script>

    app = Vue.createApp({
        data() {
            return {
                titre: 'Ma première application Vue 3',
                sousTitre: '...en 3 minutes chrono'
            }
        }
    });
    app.mount('#app');
</script>
```

01.01

Les bases de Vue.JS

L'affichage du MODEL dans la VIEW

L'affichage du MODEL dans la VIEW - 2

- La fonction data()

  Permet la création de variables réactives utilisées dans l'application.

  Mise à jour instantanée à chaque modification d'une variable

  Utilisable grâce aux double brackets

```
app = Vue.createApp({
    data() {
        return {
            titre: 'Ma première application Vue 3',
            sousTitre: '...en 3 minutes chrono'
        }
    }
});
app.mount('#app');
```

```
<div id="app">
    <h2>{{titre}}</h2>
    <p>{{sousTitre}}</p>
```

01.01

Les bases de Vue.JS

L'affichage du MODEL dans la VIEW

L'affichage du MODEL dans la VIEW - 3

- Les propriétés calculées

Permettent de garder un code DRY et d'exécuter plusieurs méthodes en une seule

```
        app = Vue.createApp({
            data(){

                return {
                    d: new Date()
                }
            },
            computed: {
                heureFr() {
                   return this.d.toLocaleTimeString()
                },
                heureUs() {
                    return this.d.toLocaleTimeString('en-US')
                }
            }
        });
        app.mount('#app');
    </script>
</body>
```

01.01

Les bases de Vue.JS

Les directives Vue.JS

Les directives Vue.JS - 1

- Le Data binding (**v-bind**) - One way binding

  Permet d'assigner une expression à un attribut

- **v-once**

  Vue interpole l'expression une seule fois. L'expression sera ignorée pour tous les autres rafraîchissements

- **v-text**

  Pour afficher une donnée texte au innerHTML d'un élément

- **v-html**

  Pour afficher une donnée HTML au InnerHTML d'un élément

01.01

Les bases de Vue.JS

Les directives Vue.JS

Les directives Vue.JS - 2

- Le binding bidirectionnel (**v-model**) - Two way binding

  Permet au DOM d'injecter des données dans la VIEW. Cette directive n'est utilisée que dans les balises HTML de formulaire

- **v-if**, **v-else-if**, **v-else**

  Il rendent visible ou non les éléments selon la valeur TRUE ou FALSE de l'expression

- **v-show**

  Il rendent visible ou non les éléments selon la valeur FALSE de l'expression mais les éléments sont toujours présents dans le rendu

Les directives Vue.js - 2

- Le binding bidirectionnel (**v-model**) - Two way binding

```
<div id="app">
    <input type="text" placeholder="Entrez votre nom" v-model="nom">
    Bonjour {{nom}}
</div>
```

- v-if, v-else-if, v-else

```
<div v-if="montant > 100">
    Livraison gratuite!
<div>
<div v-else-if="montant > 50">
    Livraison 9.95$
</div>
<div v-else>
    Livraison 19.95$
</div>
```

- v-show

```
<div v-show="isError">
  {{ errorMessage }}
</div>
```

01.01

Les bases de Vue.JS

La gestion évènementielle

La gestion évènementielle - 1

- **V-ON** : "évènement Javascript quelconque"

  Quelques exemples :

  - v-on :keyup = réagit aux frappes de clavier
  - v-on :click = réagit aux clics de la souris
  - v-on :mousemove = réagit aux mouvements de la souris

Cette directive peut être affectée par un code Javascript sans parenthèses ou point virgule

```
<body>

    <div id="app">
        <p>Nombres de clics sur le bouton: {{nbr}}</p>
        <button v-on:click="nbr += 1">Incrémenter le compteur</button>
    </div>
```

01.01

Les bases de Vue.JS

La gestion évènementielle

La gestion évènementielle - 2

- $event dans la gestion évènementielle

L'objet Javascript event est créé à chaque fois qu'un évènement se produit

On passe l'objet $event et non l'objet event à la fonction événementielle pour accéder aux propiétés de l'objet dans le ViewModel

```
<div id="app">
    <img src="/img//image1.jpg" alt="paysage" @mousemove="affCoord($event)">
    <div>{{xy}}</div>
</div>

<script>
    app = Vue.createApp({
        data() {
            return {
                xy: '',
            }
        },
        methods: {
            affCoord(ev) {
                this.xy = 'x=' + ev.offsetX + ', y=' + ev.offsetY;
            }
        }
```

01.01

Les bases de Vue.JS

La directive V-FOR

V-FOR pour parcourir des éléments - 1

- Avec 1 argument pour obtenir tous les éléments tour à tour

```
<div id="app">
  <p>Quels langages de programmation connaissez-vous?</p>
  <input type="text" v-model="unLang" />
  <input type="button" value="Valider" @click="ajouter" />
  <ul>
    <li v-for="lan in langages">{{lan}}</li>
  </ul>
</div>

<script>
  app = Vue.createApp({
    data() {
      return {
        unLang: "",
        langages: [],

      };
    },
    methods: {
        ajouter() {
            this.langages.push(this.unLang);
        }
    }
  });
  app.mount("#app");
</script>
</body>
</html>
```

01.01

Les bases de Vue.JS

La directive V-FOR

V-FOR pour parcourir des éléments - 2

- Avec 2 arguments pour obtenir tous les éléments tour à tour ET leur index

```
<div id="app">
    <div v-for="(elem, i) in elems">
        {{elem}}
        <button @click="suppr(i)">Supprimer</button>
    </div>
</div>

<script>

    app = Vue.createApp({
        data( ) {
            return {
                elems: ['premier', 'deuxième', 'troisième', 'quatrième']
            }
        },
        methods:{
            suppr(i) {
                this.elems.splice(i,1);
            }
        }
    });
    app.mount('#app');
</script>
```

01.01

Les bases de Vue.JS

Les arguments dynamiques

Les arguments dynamiques

- L’attribut ou l’évènement précisé respectivement après un v-on ou un v-bind peut être dynamique, c’est-à-dire issu d’une propriété définie dans le MODEL. Pour cela, on encadre l'événement ou l’attribut dynamique par des crochets.

```
<div id="app">
    <input type="radio" v-model="choix" value="click" checked><label>Clic</label>
    <input type="radio" v-model="choix" value="dblclick"><label>Double clic</label>
    <p>
        <img src="img/image1.jpg" alt="paysage" :width="taille" @[choix]="modifierTaille">
    </p>
</div>

<script>

    app = Vue.createApp({
        data() {
            return {
                choix: 'click',
                taille: 200
            }
        },
        methods: {
            modifierTaille() {
                this.taille = this.taille * 1.1;
            }
        }
    });
    app.mount('#app');
</script>
```

01.01

Les bases de Vue.JS

La directive V-FOR

La directive v-for pour parcourir un tableau d'objets JSON

- Pour cela, il suffit de la parser : nom de l'objet JSON + un point + nom de l'élément auquel on veut accéder.

```
        <ul>
            <li v-for="personne in personnes">
                {{personne.prenom}} {{personne.nom}} {{personne.age}}
            </li>
        </ul>
    </div>

    <script>
        app = Vue.createApp({
            data() {
                return {
                    personnes: [
                        { prenom: "Eric", nom: "Martin", age: "30" },
                        { prenom: "Ever", nom: "Harmaan", age: "30" },
                        { prenom: "Kevin", nom: "Martin", age: "27" },
                        { prenom: "Ai", nom: "Mienai", age: "25" },
                    ],
                };
            },
        });
        app.mount("#app");
    </script>
```

01.01

Les bases de Vue.JS

Les évènements clavier

Les évènements clavier

- Keydown : appui sur une touche
- Keyup : relâchement de la touche
- Keypress : appui suivi d'un relâchement

```
<div id="app">
    <input type="text" v-model="texte" @keyup.enter="maj" autofocus>
    <input type="button" value="Majuscule" @click="maj"></input>
</div>
```

01.01

Les bases de Vue.JS

Les classes conditionnelles

Les classes conditionnelles

- Application d'un style en fonction d'une condition du MODEL

```
<style>
  .rouge{
    color: red;

  }
  li{
    list-style-type: none;
  }
</style>
/head>
body>
<div id="app">
  <p>Quels langages de programmation connaissez-vous?</p>
  <input type="text" v-model="unLang" />
  <input type="button" value="Valider" @click="ajouter" />
  <ul>
    <li v-for="lan in langages">
      <input type="checkbox" v-model="lan.etatCase">
      <span :class="{rouge: lan.etatCase}">{{lan.lang}}</span>
    </li>
  </ul>
</div>

<script>
  app = Vue.createApp({
    data() {
      return {
        unLang: "",
        langages: [],
      };
    },
    methods: {
      ajouter() {
        this.langages.push({
          lang: this.unLang,
          etatCase: false,
        });
      },
    },
  });
  app.mount("#app");
</script>
```

01.01

Les bases de Vue.JS

Affichage conditionnel de l'HTML

Affichage conditionnel de l'HTML

- Grâce à V-IF, V-ELSE-IF, V-ELSE

```
    </style>
</head>
<body>
    <div id="app">
        <p>Quels langages de programmation connaissez-vous?</p>
        <input type="text" v-model="texte">
        <input type="button" value="Ajouter" @click="ajouter">
        <ul>
            <li v-for="langage in langages">
                <input type="checkbox" v-model="langage.etatCase">
                <span>{{langage.lang}}</span>

                <span v-if="langage.etatCase">est coché</span>
                <span v-else>est décoché</span>
            </li>
        </ul>
    </div>
<script>
    app = Vue.createApp({
        data() {
            return {
                texte: '',
                langages: [],
            }
        },
        methods: {
            ajouter() {
                this.langages.push({
                    lang: this.texte,
                    etatCase: false
                })
            }
        }
```

Les composants de Vue.Js

01.02

Les composants de Vue.JS

Introduction

Introduction

- Ils divisent l'application
- Ils s'appellent de façon hiérarchique
- Ils peuvent être réutilisés une ou plusieurs fois dans une ou plusieurs applications
- Ils peuvent être définis de façon locale ou globale

Composant global

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Projet2-01</title>
    <script src="https://unpkg.com/vue@next"></script>
</head>
<body>

    <div id="app">
        <bonjour></bonjour>
        <bonjour></bonjour>
        <bonjour></bonjour>
        <bonjour />
    </div>

    <script>

        app = Vue.createApp({

        });
        app.component('bonjour', {
            template: '<div>Bonjour</div>'
        })
        app.mount('#app');
    </script>
</body>
</html>
```

Composant local

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Projet2-02</title>
    <script src="https://unpkg.com/vue@next"></script>
</head>
<body>

    <div id="app">
        <bonjour-loc />
    </div>

    <script>
        const BonjourLocal = {
            template: '<div>Le composant local dit bonjour</div>'
        }
        const app = Vue.createApp({
            components: {
                'bonjour-loc': BonjourLocal
            }
        });
        app.mount('#app');
    </script>
</body>
</html>
```

Les composants définis localement ne sont pas disponibles dans les sous-composants.

01.02

Les composants de Vue.JS

Manipulation

Ajout des données dans un composant

- Insérer la propriété data() dans le MODEL du composant
- Permet l'insertion de données dans un composant local ou global
- ATTENTION : cette fonction retourne un objet JSON contenant la ou les donnée(s)

01.02

Les composants de Vue.JS

Manipulation

Ajout des méthodes dans un composant

- Insérer la propriété methods dans le MODEL du composant

- Définir les méthodes du composant comme vous le feriez dans une instance de Vue

Ajout de données

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Projet2-03</title>
    <script src="https://unpkg.com/vue@next"></script>
</head>
<body>

    <div id="app">
        <rnd-loc />
    </div>

    <script>
        const RndLocal = {
            data() {
                return {
                    nombre: Math.floor(Math.random()*100 +1)
                }
            },
            template: '<div>{{nombre}}</div>'
        }
        const app = Vue.createApp({
            components: {
                'rnd-loc': RndLocal
            }
        });
        app.mount('#app');
    </script>
</body>
</html>
```

Ajout de méthode

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Projet2-06</title>
    <script src="https://unpkg.com/vue@next"></script>
</head>
<body>

    <div id="app">
        <rnd-loc />
    </div>

    <script>
        const RndLocal = {
            data() {
                return {
                    nombre: Math.floor(Math.random()*100 +1)
                }
            },
            methods: {
                nouveau() {
                    this.nombre = Math.floor(Math.random()*100 +1)
                }
            },
            template: '<div>{{nombre}}</div><button @click="nouveau">Nouveau</button>'
        }

        const app = Vue.createApp({
            components: {
                'rnd-loc': RndLocal
            }
        });
        app.mount('#app');
    </script>
</body>
</html>
```

01.02

Les composants de Vue.JS

Manipulation

Passer des données avec les props

- Les attributs dans la partie HTML d'un composant peuvent être récupérés via les propriétés (props)
- Ils sont utilisés grâce à des interpolations

```
    <div id="app">
        <taille param="grand" />
    </div>

    <script>

        app = Vue.createApp({

        });
        app.component('taille', {
            template: '<div>La taille est : {{param}}</div>',
            props: ['param']
        })
        app.mount('#app');
    </script>
</body>
```

01.02

Les composants de Vue.JS

Manipulation

Accès aux props

- Les props définies sont accessibles dans le modèle du composant
- L'objet "this" permet de les récupérer

```
<div id="app">
    <maj texte="texte à mettre en majuscule"></maj>
</div>

<script>

    app = Vue.createApp({

    });
    app.component('maj', {
        template: '<div>{{enMaj}}</div>',
        props: ['texte'],
        data() {
            return {
                enMaj: this.texte.toUpperCase()
            }
        }
    })
    app.mount('#app');
</script>
```

01.02

Les composants de Vue.JS

Manipulation

Passer des données aux composants enfants

- Pour que le composant accède aux données :
- Définir un attribut dans l'HTML du composant
- Définir une prop pour récupérer la valeur de l'attribut

```
    <div id="app">
        <taille v-bind:msg='message'></taille>
    </div>

    <script>

        app = Vue.createApp({
            data() {
                return {
                    message: 'Message du parent au composant taille'
                }
            }
        });
        app.component('taille', {
            template: '<div>{{msg}}</div>',
            props: ['msg']
        })
        app.mount('#app');
    </script>
</body>
</html>
```

01.02

Les composants de Vue.JS

Manipulation

Custom-events : utilisation de $emit

- Permet l'envoi de messages au parent
- $emit(ev) déclenche un évènement ev sur l'instance courante d'un composant
- Le composant peut y réagir avec **V-ON** et exécuter la méthode

```
    <div id="app">
        <bienvenue @coucou="disBonjour"></bienvenue>
    </div>

    <script>

        app = Vue.createApp({
            methods: {
                disBonjour() {
                    alert('Le parent dit bonjour');
                }
            }
        });
        app.component('bienvenue', {
            template: '<button @click="$emit(\'coucou\')">Cliquez ici</button>',
        })
        app.mount('#app');
    </script>
</body>
</html>
```

01.02

Les composants de Vue.JS

Manipulation

Lifecycle hooks

- Exécute du code à un moment du cycle de vie d'un composant
- Mise en place du hook dans la VIEW avec une méthode spécifique telle que :
  - beforeCreate
  - created
  - beforeMount
  - mounted
  - beforeUpdate
  - updated
  - beforeDestroy
  - destroyed

01.02

Les composants de Vue.JS

Manipulation

Axios

- Bibliothèque très utile pour la récupération de données d'une API Rest
- Utilisation du hook created + interpolations

```
    <div id="app">
        <table>
            <tr v-for="user in users">
                <td><img v-bind:src="user.picture.large"></td>
                <td>{{user.name.first}} {{user.name.last}}<br>
                    {{user.email}}<br>
                    {{user.phone}}
                </td>
            </tr>
        </table>
    </div>

    <script>

        app = Vue.createApp({
            data() {
                return {
                    users: [],
                }
            },
            created() {
                axios.get('https://randomuser.me/api/?results=3').then(function(reponse){
                    vm.users= reponse.data.results;
                })
            }
        });
        let vm = app.mount('#app');
    </script>
</body>
</html>
```

01.02

Les composants de Vue.JS

Manipulation

Passer des données avec des slots - 1

- Espaces réservés pour l'affichage de contenu transmis d'un composant à l'autre

- 3 types de slots :
  - Slots simples
  - Slots nommés
  - Slots avec une portée

01.02

Les composants de Vue.JS

Manipulation

Passer des données avec des slots - 2

Les slots simples

- Peuvent contenir n'importe quel code de template incluant de l'HTML

```
    <div id="app">
        <test></test>
        <test>Texte de remplacement</test>
    </div>

    <script>

        app = Vue.createApp({

        });
        app.component('test', {
            template: '<div><slot>Texte par defaut</slot></div>'
        })
        app.mount('#app');
    </script>
</body>
</html>
```

01.02

Les composants de Vue.JS

Manipulation

Passer des données avec des slots - 3

Les slots nommés

- Peuvent être nommés grâce à la directive V-SLOT de la balise template.
- **V-SLOT** peut être remplacé par #

```
    <div id="app">
        <art titre="titre de l'article">
            <template v-slot:texte>
                <p>Texte de l'article</p></art>
            </template>
    </div>

    <script>

        app = Vue.createApp({

        });
        app.component('art', {
            template: `<div>
                        <h2>{{titre}}</h2>
                        <slot name="texte"></slot>
                       </div>`,
            props: ['titre']
        })
        app.mount('#app');
    </script>
</body>
</html>
```

01.02

Les composants de Vue.JS

Manipulation

Passer des données avec des slots - 4

Les slots avec portée

- Permettent la récupération d'un contenu uniquement accessible dans un composant

- Permet de récupérer des données de l'enfant dans le parent

Slots avec portée

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Projet2-23</title>
    <script src="https://unpkg.com/vue@next"></script>
</head>
<body>

    <div id="app">
        <employes>
            <template v-slot="slotProps">
              <span><em>{{slotProps.pers.prenom}}</em></span>
               
               <span>{{slotProps.pers.nom.toUpperCase()}}</span>
            </template>
        </employes>
    </div>

    <script>
       app = Vue.createApp({

        });
        app.component('employes', {
            data() {
                return {
                    personnes: [
                        {
                            prenom: 'Jean',
                            nom: 'Segur'
                        },
                        {
                            prenom: 'Pierre',
                            nom: 'Dumanier'
                        }
                    ]
                };
            },
            template: '<div v-for="personne in personnes"><slot :pers="personne"></slot></div>'
        });
        app.mount('#app');
    </script>
</body>
</html>
```

01.02

Les composants de Vue.JS

Manipulation

Les observateurs

- Permettent la détection des changements dans une propriété définie dans le MODEL et la réaction en effectuant une action

Ajouter la propriété watch dans le MODEL de l'application ou du composant

Définir une fonction anonyme ayant le même nom que la propriété à observer

Création d'un observateur

```
<body>

    <div id="app">
        {{compteur}}
        <button @click="compteur++;">Incrémenter</button>
    </div>

    <script>

        app = Vue.createApp({
            data() {
                return {
                    compteur: 0
                }
            },
            watch: {
                compteur(value) {
                    if(value == 10) {
                        this.compteur = 0;
                    }
                }
            }
        });
```

01.02

Les composants de Vue.JS

Animations

Animations et transitions

- Transition traditionnelle en javascript et CSS

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Projet2-30</title>
    <style>
        img {
            transition: opacity 5s ease-in;
        }
    </style>
    <script>
        function cacherAfficher() {
            let ima = document.getElementById('paysage');
            let bout = document.getElementById('bouton');
            if(bout.innerHTML == "Cacher") {
                bout.innerHTML = "Afficher";
                ima.setAttribute('style', 'opacity: 0');
            } else {
                bout.innerHTML = "Cacher";
                ima.setAttribute('style', 'opacity: 1');
            }
        }
    </script>
</head>
<body>

<button id="bouton" onclick="cacherAfficher();">Cacher</button><br>
<img src="img/image1.jpg" alt="paysage" id="paysage">
</body>
</html>
```

Transition d'entrée et de sortie en Vue.JS

Utilisant l'affichage conditionnel v-if et v-else

```
    <script src="https://unpkg.com/vue@next"></script>
    <style>
        .t-enter-from, /*Etat de départ de la transition lors de l'apparition*/
        .t-leave-to {   /*Etat final de la transition lors de la disparition*/
            opacity: 0;
        }
        .t-enter-active, /*paramètres de la transition pour passer de l'état invisible à l'état visible*/
        .t-leave-active { /*paramètres de la transition pour passer de l'état visible à l'état invisible*/
            transition: opacity 5s ease-in;
        }
        /*si l'attribut name n'est pas défini dans la balise transition
        les classes à utiliser sont .v-enter-from, .v-leave-to, .v-enter-active, .v-leave-active*/
    </style>
</head>
<body>
    <div id="app">
        <button @click="visible=!visible">
            <span v-if="visible">Cacher</span>
            <span v-else="">Afficher</span>
        </button>
        <transition name="t">
            <!--Element à animer-->
            <div v-if="visible">
                <img src="img/image1.jpg" alt="paysage">
            </div>
        </transition>
    </div>

    <script>

        app = Vue.createApp({
            data() {
                return {
                    visible: true
                }
            }
        });
        app.mount('#app');
    </script>
</body>
</html>
```

01.02

Les composants de Vue.JS

Animations

Animations et transitions

- Transition d'entrée et de sortie non simultanées
- Vue propose 2 méthodes de transitions :

  IN-OUT : la transition entrante du premier élément s'effectue en premier, suivie de la transition sortante de l'élément courant

  OUT-IN : la transition sortante s'effectue en premier, suivie de la transition entrante du nouvel élément.

- Pour définir la transition, utilisez l'attribut mode dans le composant transition

Transition d'entrée et de sortie non simultanées

```
    <style>
        .t-enter-active, .t-leave-active {
            transition: opacity 1s ease;
        }
        .t-enter-from, .t-leave-to {
            opacity: 0;
        }
    </style>
</head>
<body>
    <div id="app">
        <button @click="label=!label">
            <span v-if="label">Afficher texte 2</span>
            <span v-else>Afficher texte 1</span>
        </button>
        <transition name="t" mode="out-in">
            <div v-if="label" key="1">Premier texte </div>
            <div v-else key="2">Deuxième texte</div>
        </transition>
    </div>
    <script>
        const  app = Vue.createApp({
            data() {
                return {
                    label: false
                }
            }
        });
        app.mount('#app');
    </script>
</body>
</html>
```

01.02

Les composants de Vue.JS

Animations

Animations et transitions

- Transition entre éléments

```
    <style>
        .t-enter-active, .t-leave-active {
            transition: opacity 2s ease-in;
        }
        .t-enter-from, .t-leave-to {
            opacity: 0;
        }
        h1 {
            position: absolute;
            top: 5rem;
            left: 5rem;
            font-size: 3rem;
        }
    </style>
</head>
<body>
    <div id="app">
        <button @click="anime < 4?anime++:anime=1">Animer le texte</button>
        <transition name="t">
            <h1 v-if="anime==1" key="t1">Texte 1</h1>
            <h1 v-else-if="anime==2" key="t2">Texte 2</h1>
            <h1 v-else-if="anime==3" key="t3">Texte 3</h1>
            <h1 v-else="anime>3" key="t4">Texte 4</h1>
        </transition>
    </div>
    <script>
        app = Vue.createApp({
            data() {
                return {
                    anime: 1
                }
            }
        });
        app.mount('#app');
    </script>
</body>
</html>
```

01.02

Les composants de Vue.JS

Animations

Animations et transitions

- Transition de listes

- Application d'une transition lors de l'ajout ou de la suppression des éléments dans une liste

Utilisation du composant <transition-group />

Nécessité de donner un nom à cette balise et faire porter l'animation sur ce nom

Également ajouter un attribut V-BIND :KEY unique pour chaque élément

Transition de listes

```
    <style>
        .fade-enter-active, .fade-leave-active {
            transition: opacity 2s ease;
        }
        .fade-enter-from, .fade-leave-to {
            opacity: 0;
        }
        input {
            margin-right: 1rem;
        }
    </style>
</head>
<body>
    <div id="app">
        <input type="text" v-model="fruit">
        <input type="button" value="Ajouter" @click="ajouter">
        <input type="button" value="Supprimer le premier" @click="supprimer"><br>
        <transition-group name="fade">
            <span v-for="element in liste" v-bind:key="element">{{element}} </span>
        </transition-group>
    </div>
    <script>
        const app = Vue.createApp({
            data() {
                return {
                    liste: ['banane', 'pomme', 'poire'],
                    fruit: ''
                }
            },
            methods: {
                ajouter() {
                    this.liste.push(this.fruit);
                },
                supprimer() {
                    this.liste.shift();
                }
            }
        });
        app.mount('#app');
    </script>
</body>
</html>
```

01.02

Les composants de Vue.JS

Animations

Animations et transitions

- Animate.CSS et Velocity.JS
- https://animate.style
- https://velocityjs.org

```
<title>Projet2-36</title>
<script src="https://unpkg.com/vue@next"></script>
<link
  rel="stylesheet"
  href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"
/>
</head>
<body>
<div id="app">
    <button @click="anime=!anime">Animer le texte</button>
    <transition enter-active-class="animate__animated animate__hinge"
                leave-active-class="animate__animated animate__bounceOutLeft">
        <h1 v-if="anime">Texte à animer</h1>
    </transition>
</div>

<script>
  app = Vue.createApp({
      data() {
          return {
              anime: false
          }
      }
  });
  app.mount("#app");
</script>
</body>
</html>
```

Single Page Applications

01.03

Les S.P.A

Introduction

Introduction aux S.P.A

- L'ensemble du site web ressemble à une application sur une seule page
- Plus besoin de charger de page
- L'URL ne pointe pas vers des pages Html mais vers des états particuliers de l’application qui ressemblent le plus souvent à des pages différentes.
- Google, Facebook, Spotify, etc

01.03

Les S.P.A

Vue-Router

S.P.A avec Vue-Router

- Plugin Vue-Router pour S.P.A
- S'intègre au noyau de Vue et facilite la création d'applications
- Chaque route = un composant responsable de l'affichage d'une page
- Pour utiliser Vue-Router : inclure le cdn correspondant dans l'en-tête de l'application

```
<script src="https://unpkg.com/vue-router@next"></script>
```

01.03

Les S.P.A

Vue-Router

S.P.A avec Vue-Router

- Les fonctionnalités de Vue-Router incluent :
- Cartographie d'itinéraire/vue imbriquée
- Configuration de routeur modulaire basée sur des composants
- Paramètres de route, requête, caractères génériques
- Afficher les effets de transition optimisés par le système de transition de Vue.js
- Liens avec les classes CSS actives automatiques
- Comportement de défilement personnalisable

S.P.A avec Vue-Router

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Squelette</title>
    <script src="https://unpkg.com/vue@next"></script>
    <script src="https://unpkg.com/vue-router@next"></script>
</head>
```

```html
<body>
    <div id="app">
        <h1>Première application SPA</h1>
        <ul>
            <li><router-link to="/">Page 1</router-link></li>
            <li><router-link to="/page2">Page 2</router-link></li>
            <li><router-link to="/page3">Page 3</router-link></li>

        </ul>
        <router-view></router-view>
    </div>
    <script>
        const page1 = {
            template: '<div>Bienvenue sur la page 1</div>'
        }
        const page2 = {
            template: '<div>Bienvenue sur la page 2</div>'
        }
        const page3 = {
            template: '<div>Bienvenue sur la page 3</div>'
        }
        const router = VueRouter.createRouter({
            history: VueRouter.createWebHistory(),
            routes: [
                {
                    path: '/',
                    component: page1
                },
                {
                    path: '/page2',
                    component: page2
                },
                {
                    path: '/page3',
                    component: page3
                }
            ]
        })
        const app = Vue.createApp({

         });
         app.use(router);
         app.mount('#app');
    </script>
</body>
```

01.03

Les S.P.A

Navigation Guards

Indicateurs de navigation – Vue Router Navigation Guards - 1

- Permettent d’exécuter du code à certain moment précis dans une application Vue.js
- Peuvent être utilisés pour, par exemple, réserver du contenu privé aux utilisateurs enregistrés
- 3 types de Navigation Guards :
  - Global
  - Par route
  - Par composants

01.03

Les S.P.A

Navigation Guards

Indicateurs de navigation – Vue Router Navigation Guards - 2

- Global : exécuté à chaque changement de route

  Ex : router.beforeEach((to, from, next) => {
  }) ;
  router.afterEach((to, from) => {
  }) ;

- Par route : exécuté sur une route particulière

  L'intercepteur de navigation beforeEnter(to, from, next) doit être défini directement dans la route concernée

- Par composants : exécuté sur certains composants spécifiques

01.03

Les S.P.A

Navigation Guards

Indicateurs de navigation – Vue Router Navigation Guards - 3

- Le Navigation Guard par composants permet de définir une interception de navigation dans un composant en utilisant des fonctions dédiées

- beforeRouteEnter : exécuté avant la confirmation de la route

- beforeRouteUpdate : exécuté quand la route change

- beforeRouteLeave : exécuté juste avant le changement de route

01.03

Les S.P.A

Routage

Routage-Transition entre les pages

- Encapsuler le composant ROUTER-VIEW dans un composant TRANSITION afin d'appliquer une transition à chacune des pages

```
<transition>
  <router-view></router-view>
</transition>
```

- Ou utiliser des transitions par route pour une transition différente en fonction des pages affichées

```
const page1 = {
template : « <transition name= ‘tr1’><div></div>
</transition> »
} ;
```

L'interface Vue-Cli

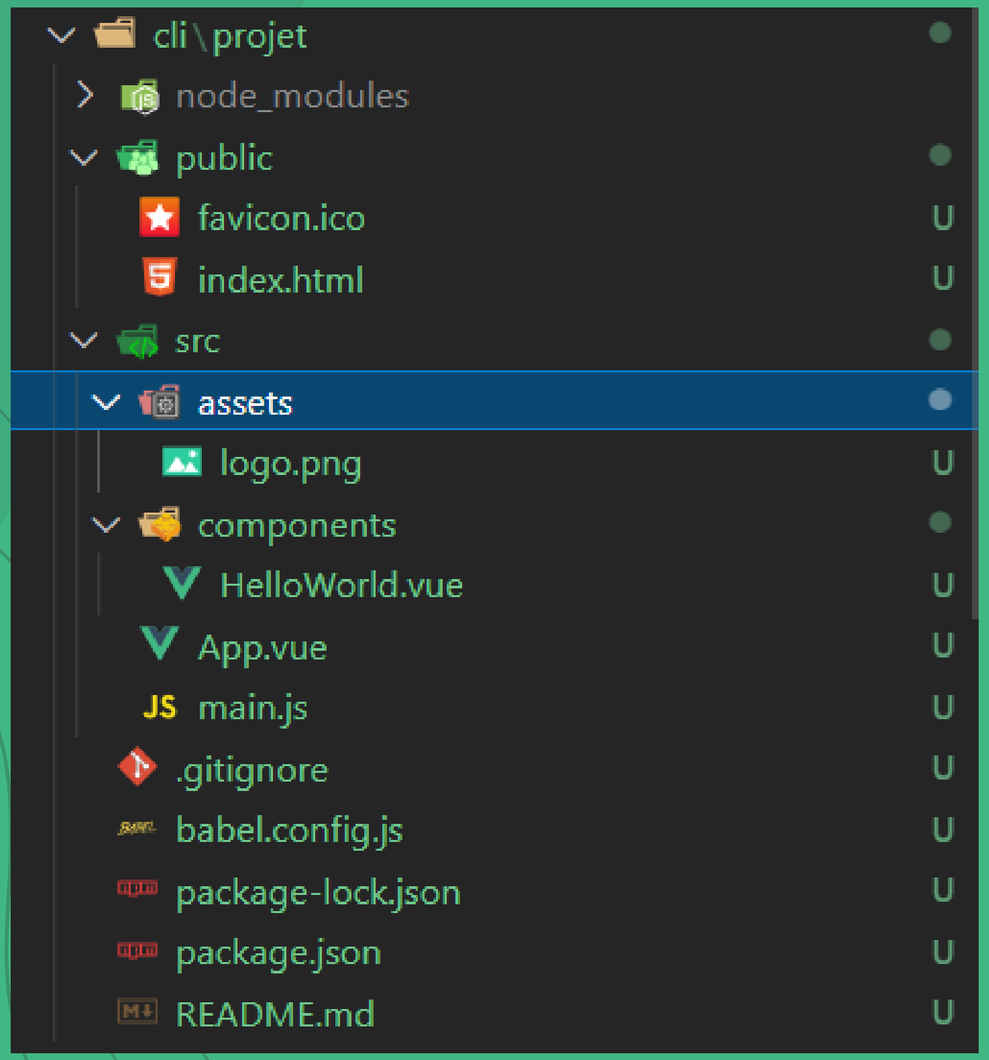
01.04

L'interface Vue-Cli
Introduction

Introduction

- Chaque composants écrit dans un fichier séparé
- Intègre tout ce dont vous avez besoin pour créer une application Vue
- Tous ces fichiers seront organisés en modules

L'architecture de l'application



01.04

L'interface Vue-Cli

Architecture

Architecture de l'application

- Un fichier src/main = point d'entrée
- Un fichier public/index.html pour afficher l'application
- De composants sous la forme de fichiers d'extension .vue

01.04

L'interface Vue-Cli

lArchitecture

Architecture de l'application

- Le fichier index.html se contente de définir le conteneur de la VUE avec une balise <div id="app>

```
<!DOCTYPE html>
<html lang="">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <link rel="icon" href="<%= BASE_URL %>favicon.ico">
    <title><%= htmlWebpackPlugin.options.title %></title>
  </head>
  <body>
    <noscript>
      <strong>We're sorry but <%= htmlWebpackPlugin.options.title %>
        doesn't work properly without JavaScript enabled. Please enable it to continue.</strong>
    </noscript>
    <div id="app"></div>
    <!-- built files will be auto injected -->
  </body>
</html>
```

01.04

L'interface Vue-Cli

Architecture

Architecture de l'application

- Le fichier App.vue

```
<template>
  <img alt="Vue logo" src="./assets/logo.png">
  <HelloWorld msg="Welcome to Your Vue.js App"/>
</template>
```

01.04

L'interface Vue-Cli

Architecture

Architecture de l'application

- La partie <script></script>

```
<script>
import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'App',
  components: {
    HelloWorld
  }
}
</script>
```

01.04

L'interface Vue-Cli

Architecture

Architecture de l'application

- La feuille de style

```
<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

01.04

L'interface Vue-Cli

Architecture

Architecture de l'application

- Le fichier main.js
- Importe la méthode createApp et le composant App.vue

```
import { createApp } from 'vue'
import App from './App.vue'

createApp(App).mount('#app')
```

01.04

L'interface Vue-Cli

Architecture

Fonctions de rendu (Render Functions)

- Alternative aux templates
- Chaque composant Vue implémente une fonction de rendu
- Elle retourne un DOM virtuel qui est injecté dans le DOM du navigateur pour être rendu.

Divers

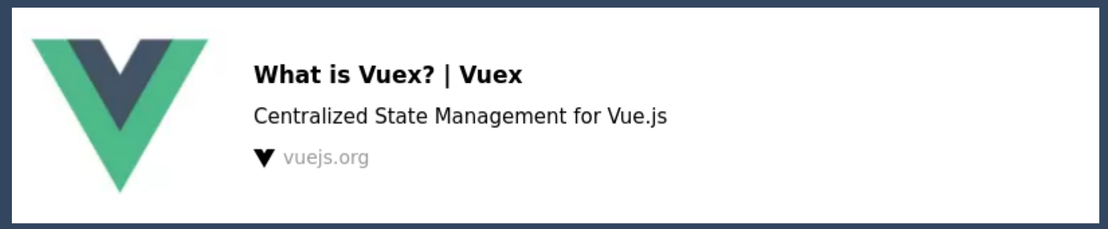
01.05

Divers

Vuex

Gestionnaire d'état Vuex

- Gestionnaire d'état + bibliothèque pour les applications Vue.js
- Zône de stockage centralisée pour tous les composants d'une application Vue.js
- Endroit de stockage = Store



01.05

Divers

Composition API

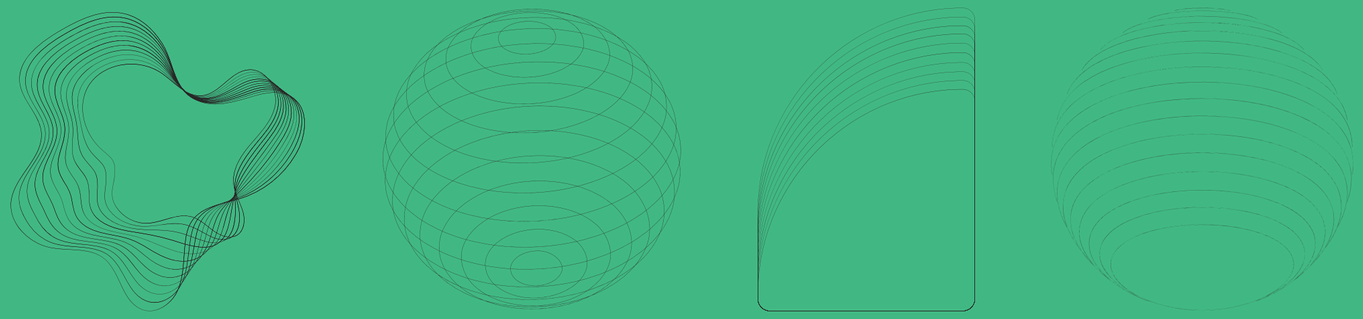
Composition API

- Alternative à l'Option API
- Permet d'écrire le code de façon regroupée sans utiliser le mot clé "this"
- Méthode stup()

Conclusion

Nous avons donc fait le tour de Vue.js. Si cela vous a intéressé, je vous conseille de vous rendre sur la page:
https://vuejs.org
La documentation est super bien faite. Il y a également pas mal de tutos qui sont pas mal faits et dans lesquels vous pourrez encore apprendre énormément de choses concernant Vue.js

Page de Ressources



https://vuejs.org
https://ichi.pro/fr/structurer-les-applications-vuejs-avec-l-api-de-composition-244740398072962
https://markus.oberlehner.net/blog/vue-3-composition-api-vs-options-api
https://www.emencia.com/news/2021/05/06/vuejs-3-presentation-de-la-composition-api
https://dev.to/ericlecodeur/apprendre-vue-js-3-jour-4-composition-api-418g
https://www.udemy.com/course/vuejs-3-la-formation-complete-pour-debutants/learn/lecture/26672046#overview
https://www.linkedin.com/learning/l-essentiel-de-vue-js-3/comprendre-les-methodes-dans-vue-js?autoAdvance=true&autoSkip=true&autoplay=true&resume=false