

The background features abstract green geometric shapes. On the left, a solid green triangle points downwards. On the right, a complex arrangement of overlapping, semi-transparent green triangles and polygons creates a layered, architectural effect. The text is centered between these two main graphic elements.

TP Liste et Recap

Objectifs

- ▶ Affichage nombre total de produit
- ▶ Mis en évidence des produits moins chers

Etapes (référence doc)

- ▶ Affichage nombre total de produit et mis en évidence des produits moins chers
 - ▶ Propriétés calculées
 - ▶ Liaison de classes et de styles

TP 2c

- ▶ Partir du projet sur le google drive



- ▶ Lancer le projet
- ▶ Analyse du projet
 - ▶ CSS

TP 2c

- Affichage du nombre de produit en haut

NOS PRODUITS
NOMBRE DE PRODUITS: 10

- Utiliser computed

TP 2c - solution

- Dans computed

```
computed: {  
  productsLength() {  
    return this.products.length;  
  },  
}
```

- Dans le template

```
<h1 class="text-xl p-4 text-center uppercase mb-4 bg-sky-300 text-sky-800">  
  Nos produits<br>  
  Nombre de produits: {{ productsLength }}  
</h1>
```

- Note : methods ou computed?

Computed vs Methods

Computed

- ▶ Caching , valeurs en cache.
Recalculer uniquement quand la source de données est mise à jour
- ▶ Performance
- ▶ A utiliser quand vous avez une valeur dérivée de données réactives et que vous voulez que le résultat soit recalculé uniquement lorsque les données changent.

Methods

- ▶ Pas de caching, recalculer à chaque fois
- ▶ A utiliser quand vous avez une action ou un calcul qui doit être exécuté chaque fois qu'il est invoqué, indépendamment des données réactives.

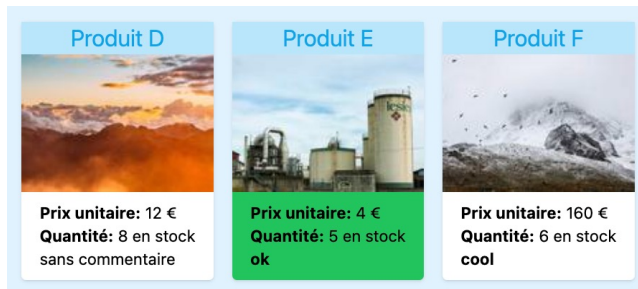
TP 2d

- ▶ Marquer le prix le moins cher
- ▶ Créer un computed pour calculer le prix le moins cher

```
cheapestPrice() {  
  return Math.min(...this.products.map(product => product.unit_price));  
}
```


TP 2d

- Marquer le prix le moins cher comme ceci



- Créer un computed pour calculer le prix le moins cher

```
cheapestPrice() {  
  return Math.min(...this.products.map(product => product.unit_price));  
}
```

- Utiliser le code suivant

```
:class="{ 'bg-green-500': product.unit_price === cheapestPrice }"
```

TP 2d

- ▶ Vérification avec Inspecter du navigateur si la classe « bg-green-500 » est bien là.
- ▶ Changer le code et supprimer bg-white de la classe par défaut.

```
:class="product.unit_price === cheapestPrice ? 'bg-green-500' : 'bg-white'"
```

Récapitulation

Confidential - Not for Public Consumption or Distribution

3 options pour les données et logique

- ▶ Dans un composant, on a 3 options
 - ▶ Data : données réactives locales
 - ▶ Computed : données calculées du composant et mise en cache
 - ▶ Methods : fonctions associées au composant
 - ▶ Utilisation de l'asynchrone

Templating

- ▶ La syntaxe de base avec les `{{ }}`
- ▶ Différentes directives vue.js comme
 - ▶ `v-html` pour interpréter 'html dans un string
 - ▶ `v-bind:attribute` ou `attribute` pour lier une donnée au template
- ▶ Les conditions classiques comme `v-if`, `v-else`, `v-else-if` et `v-show`. Attention à maîtriser la différence entre `v-if` et `v-show`
- ▶ Le looping avec `v-for` pour faire du list rendering
- ▶ Le class et style binding (Ex: `:class="{ 'active' : user.has_active_subscription }"`)

Questions

- Comparaison avec Vanilla JS