Vue Mastery

PINIA CHEAT SHEET

Initialize Pinia for your app

src/main.js

```
import { createPinia } from 'pinia'
createApp(App).use(createPinia()).mount('#app')
```

Define the store

src/stores/ProductStore.js

```
import { defineStore } from 'pinia'

export const useProductStore = defineStore('product', {
  state: () => ({
    products: [ Product , Product , Product ]
  }),
  getters: {
    productCount(state) {
      return state.products.length
    },
    productsCheaperThan(state) {

      return (price) => (
        state.products.filter(product =>
          product.price < price
        )
      )
    }
  },
  actions: {
    addProduct( Product ) {
      this.products.push( Product )
    }
  }
})t
```

*a unique **name***

*initialize the **state***

***getters** can access the state through the parameter*

*a getter can accept an **argument**, but it has to return a function instead*

*change the state with **actions***

*access the state with **this***

Use the store (Composition API)

src/App.vue

```
<script setup>
import { useProductStore } from './stores/ProductStore'

const store = useProductStore()
</script>
<template>
  <ul>
    <li v-for="product in store.products">
      ...
    </li>
  </ul>
  <p>{{ store.productCount }}</p>
  <ul>
    <li v-for="product in store.productsCheaperThan(10)">
      ...
    </li>
  </ul>
  <button @click="store.addProduct( Product )">Add</button>
```

*create a store **instance***

*access the **state** directly*

*use a **getter***

*use a getter that takes an **argument***

*use the **action***

Vue Mastery

PINIA CHEAT SHEET (PART 2)

Use the store (Options API)

```
<script>
import { useProductStore } from './stores/ProductStore'
import { mapStores } from 'pinia'

export default {
  computed: {
    ...mapStores(useProductStore)
  }
}
</script>
<template>
  <ul>
    <li v-for="product in productStore.products">
      ...
    </li>
  </ul>
  <p>{{ productStore.productCount }}</p>
  <ul>
    <li v-for="product in productStore.productsCheaperThan(10)">
      ...
    </li>
  </ul>
  <button @click="productStore.addProduct( Product )">Add</button>
```

*import the **mapStore** function*

*map the **store** as a computed property*

*this name is the combination of the store's unique name "**product**" + "**Store**". It is created by mapStores.*

Change the state without actions

```
store.x = 1
```
change one thing

```
store.$patch({ x: 1, y: 2})
```
change multiple things

```
store.$patch(state => {
  state.x = 1
  state.y = 2
})
```
alternate syntax

```
store.$state = { x: 1, y: 2, z: 3 }
```
change the entire state

```
store.$reset()
```
change the entire state back to the initial values

Subscribe to changes

```
store.$subscribe((mutation, state) => {
   ...
})
```

the state after the change

details on how the change was made