v-bind:атрибут=«назва змінної»
(замість «v-bind:» можна писати просто «:») – додаємо перед довільним атрибутом, щоб передавати в нього значення з JS.

v-on:click= «назва функції»
(замість «v-on:» можна писати просто «@») – обробник події кліку мишкою

v-on:keypress.enter= «назва функції» – обробник події натисненої клавіші на клавіатурі

v-if= «умова» - задаємо умову в тезі, і все, що описано/вкладено в ньому відображатиметься лише, якщо умова буде true

v-for = «змінnна in назва масиву» - змінна1 по черзі матиме значення кожного елементу

v-for = «(змінnна1, змінна2) in назва масиву» - змінна1 по черзі матиме значення кожного елементу, змінна2 – міститиме в собі індекс цього елементу.

v-cloak – директива, яка додається елементам поки Vue ще не прогрузилося. Якщо поєднати її використання із css можна уникнути «стрибків» сторінки під час завантаження.

CSS:
```
[v-cloak] { display: none;  }
```

v-model="назва змінної" – директива, яка дозволяє зв'язати зміну надпису в input і змінної – двохстороннє звязування. (Як ми робили функцію для нотаток, щоб value зчитувалося у змінну).

v-show – приховує елементи, як v-if, але v-if повністю видаляє їх з DOM елементу, а v-show приховує за допомогою display:none.
Синтаксис: v-show=«умова» – показується, якщо умова true.

Створення елементу Vue:

```
let myAplication = {

    data() {
        return {
            ключ: значення,
          ключ2: [значення може бути масивом/об'єктом]
        }
    },
    methods: {
        назва функції(параметр) {
          тіло функції
        },
     computed(){
     функції(Тут потрібно створювати функції, які будуть залежні він зміни якоїсь зміної.)

    }
}

Vue.createApp(myAplication).mount('#myApp')
```

LocalStorage

Існує три варіанти локального збереження даних у браузері:

-Cookies
-LocalStorage
- Session Storage

```
// Для запису потрібно використати setItem:
localStorage.setItem('a', 51);
// Щоб "витягнути" дані назад
// і записати в якусь змінну:
let b = localStorage.getItem('a')
```

Всі дані, які ми записуємо – перетворюються в тип string і коли ми їх «виймаємо» назад вони залишаються в такому ж типі. Для правильного перетворення «туди-назад» використовуються наступні методи:

```
// Для запису потрібно використати setItem:
localStorage.setItem('a', JSON.stringify(51) );
// Щоб "витягнути" дані назад
// і записати в якусь змінну:
let b = JSON.parse(localStorage.getItem('a'))
```

Для перевірки, чи вже є якийсь запис з таким ключем:

```
if (localStorage.getItem('hello') != null){
    // якщо такий ключ існує (не дорівнює порожній множині)
}
```

Очищення localStorage:

```
// Видаляємо ключ
localStorage.removeItem("Ключ")
// Очищаємо все сховище
localStorage.clear()
```

LocalStorage

Існує три варіанти локального збереження даних у браузері:

-Cookies
-LocalStorage
- Session Storage

```
// Для запису потрібно використати setItem:
localStorage.setItem('a', 51);
// Щоб "витягнути" дані назад
// і записати в якусь змінну:
let b = localStorage.getItem('a')
```

Всі дані, які ми записуємо – перетворюються в тип string і коли ми їх «виймаємо» назад вони залишаються в такому ж типі. Для правильного перетворення «туди-назад» використовуються наступні методи:

```
// Для запису потрібно використати setItem:
localStorage.setItem('a', JSON.stringify(51) );
// Щоб "витягнути" дані назад
// і записати в якусь змінну:
let b = JSON.parse(localStorage.getItem('a'))
```

Для перевірки, чи вже є якийсь запис з таким ключем:

```
if (localStorage.getItem('hello') != null){
    // якщо такий ключ існує (не дорівнює порожній множині)
}
```

Очищення localStorage:

```
// Видаляємо ключ
localStorage.removeItem("Ключ")
// Очищаємо все сховище
localStorage.clear()
```