Прив'язування класів та стилей

https://ua.vuejs.org/guide/essentials/class-and-style.html#binding-html-classes

Статичний клас (звичайний CSS клас)

Один клас

<тег **class**=" статичний_клас "> . . . . . </тег >

```
<div id="app">
  <label>
    <span class="title">Energy level</span>
    <input type="number" class="energy-input" v-model="ene
  </label>
</div>
```

```
<style>
  .title {
    background-color: green;
  }
  .energy-input {
    background-color: yellow;
  }
</style>
```

Динамічний клас (обчилюваний CSS клас)
(у задежності від значення даних використовуємо один з наперед визначених статичних CSS класів)

<тег : **class**=" змінна_модель_класу "> . . . . . </тег >

```
<div id="app">
  <label>
    Energy level
    <input type="number" :class="statusColor" v-model="energyLevel" />
  </label>
</div>
```

```
computed: {
  statusColor() {
    let currectStateClass
    if (this.energyLevel > 80) currectStateClass = 'high'
    else if (this.energyLevel > 30) currectStateClass = 'middle'
    else currectStateClass = 'low'
    return currectStateClass
  },
},
```

```
<style>
  .high {
    background-color: green;
  }
  .middle {
    background-color: yellow;
  }
  .low {
    background-color: red;
  }
</style>
```

Задання класів:

1) **Звичайні постійні статичні класи (значенням є звичайний string)**
2) **Постійні динамічні класи (у залежності від значення даних застосовувати тільки один з множини заданих класів)**
3) **Список постінийних динамічних класів(у залежності від значення даних застосовувати декілька класів (масив класів)**
4) **Умовні статичні класи (наперед визначений клас застосовувати за деякої умови (об’єкт з властивостями-класами, значеннями-умова застосування)**
5) **Умовні динамічні класи (наперед визначений клас застосовувати за деякої умови (об’єкт з властивостями-класами, значеннями-умова застосування)**
6) **Умовні і постійні динамічні класи: деякі постійні, деякі застосовуються за певної умови (масив з класами, та об’єктами з умовними класами)**
7) **Статичні і динамічні класи одночасно**

Задання класів:

1) **Постійні** (застосовуються завжди) **статичні** (не змінюються) **класи (значенням є звичайний string)**

Один клас

```
<тег class=" статичний_клас "> . . . . . </тег >
```

Декілька класів (через пробіл)

```
<тег
 class=" ст._клас1  ст._клас2  . . . "
>
  . . . . .
 </тег >
```

Задання класів:

1) **Постійні (застосовуються завжди) статичні (не змінюються) класи (значенням є звичайний string)**

Один клас

<тег **class="** статичний_клас **">** . . . . . </тег >

Декілька класів (через пробіл)

<тег
class=" ст._клас1 ст._клас2 . . . **"**
>
. . . . .
</тег >

```
<div id="app">
    <label>
        <span class="title">Energy level</span>
        <input type="number" class="energy-input" v-model="energyLevel" />
    </label>
</div>
```

Задання класів:

1) **Постійні (застосовуються завжди) статичні (не змінюються) класи (значенням є звичайний string)**

Один клас

```
<тег class=" статичний_клас "> ..... </тег >
```

Декілька класів (через пробіл)

```
<тег
 class=" ст._клас1  ст._клас2  ... "
>
  .....
 </тег >
```

1.Описуємо класи

```
<style>
  .title {
    background-color: green;
  }
  .energy-input {
    background-color: yellow;
  }
</style>
```

2.Підключаємо класи (як у звичайному HTML)

```
<div id="app">
  <label>
    <span class="title">Energy level</span>
    <input type="number" class="energy-input" v-model="energyLevel" />
  </label>
</div>
<script>
  const { createApp } = Vue

  const app = createApp({
    data() {
      return {
        energyLevel: null,
      }
    },
  }).mount('#app')
</script>
```

Energy level

Динамічне визначення класу

<u>Постійні динамічні класи</u>

2)У залежності від значення даних застосовувати один з класів

```
<тег : class=" змінна_модель_класу "> ..... </тег >
```

включаємо режим прив’язування (ставимо двокрапку перед атрибутом class)

зазначаємо вираз (а ще краще змінну-модель з data або computed), що задає необхідний клас

Динамічне визначення класу

<u>Постійні динамічні класи</u>

2)У залежності від значення даних застосовувати один з класів

<тег : **class**=" змінна_модель_класу "> . . . . . </тег >

включаємо режим прив’язування (ставимо двокрапку перед атрибутом class)

зазначаємо вираз (а ще краще змінну-модель з data або computed), що задає необхідний клас

```
<div id="app">
  <label>
    Energy level
    <input type="number" :class="statusColor" v-model="energyLevel" />
  </label>
</div>
```

Динамічне визначення класу

Постійні динамічні класи

2)У залежності від значення даних застосовувати один з класів

```
<тег : class=" змінна_модель_класу "> ..... </тег >
```

включаємо режим прив’язування (ставимо двокрапку перед атрибутом class)

зазначаємо вираз (а ще краще змінну-модель з data або computed), що задає необхідний клас

Приклад. У залежності від заряду батареї застосовувати відповідний колір фону
Заряд > 80 – зелений
30 <= Заряд <= 80 - жовтий
Заряд < 30 - червоний

Динамічне визначення класу

Постійні динамічні класи

2)У залежності від значення даних застосовувати один з класів

```
<тег : class=" змінна_модель_класу "> ..... </тег >
```

1.Описуємо класи

```
<style>
  .high {
    background-color: green;
  }
  .middle {
    background-color: yellow;
  }
  .low {
    background-color: red;
  }
</style>
```

Приклад. У залежності від заряду батареї застосовувати відповідний колір фону
Заряд > 80 – зелений
30 <= Заряд <= 80 - жовтий
Заряд < 30 - червоний

Динамічне визначення класу

Постійні динамічні класи

2)У залежності від значення даних застосовувати один з класів

```
<тег : class=" змінна_модель_класу "> ..... </тег >
```

1.Описуємо класи

```
<style>
  .high {
    background-color: green;
  }
  .middle {
    background-color: yellow;
  }
  .low {
    background-color: red;
  }
</style>
```

```
<div id="app">
  <label>
    Energy level
    <input type="number"                    v-model="energyLevel" />
  </label>
</div>
<script>
  const { createApp } = Vue

  const app = createApp({
    data() {
      return {
        energyLevel: 1,
      }
    },
    computed: {
      statusColor() {
        let currectStateClass
        if (this.energyLevel > 80) currectStateClass = 'high'
        else if (this.energyLevel > 30) currectStateClass = 'middle'
        else currectStateClass = 'low'
        return currectStateClass
      },
    },
  }).mount('#app')
</script>
```

Приклад. У залежності від заряду батареї застосовувати відповідний колір фону
Заряд > 80 – зелений
30 <= Заряд <= 80 - жовтий
Заряд < 30 - червоний

Energy level 95

2.Створюємо обчислювану властивість

Динамічне визначення класу

Постійні динамічні класи

2)У залежності від значення даних застосовувати один з класів

```
<тег : class=" змінна_модель_класу "> ..... </тег >
```

Приклад. У залежності від заряду батареї застосовувати відповідний колір фону
Заряд > 80 – зелений
30 <= Заряд <= 80 - жовтий
Заряд < 30 - червоний

1.Описуємо класи

```
<style>
  .high {
    background-color: green;
  }
  .middle {
    background-color: yellow;
  }
  .low {
    background-color: red;
  }
</style>
```

```
<div id="app">
  <label>
    Energy level
    <input type="number" :class="statusColor" v-model="energyLevel" />
  </label>
</div>
<script>
  const { createApp } = Vue

  const app = createApp({
    data() {
      return {
        energyLevel: 1,
      }
    },
    computed: {
      statusColor() {
        let currectStateClass
        if (this.energyLevel > 80) currectStateClass = 'high'
        else if (this.energyLevel > 30) currectStateClass = 'middle'
        else currectStateClass = 'low'
        return currectStateClass
      },
    },
  }).mount('#app')
</script>
```

Energy level 95

3.Підключаємо обчислювану властивість

2.Створюємо обчислювану властивість

Приклад. Вводиться номер місця пацієнта у черзі.. Відображати кольром
зелений, 18 – він перший
жовтий, 14 – другий
оранжевий, 12 – третій
синій, 10 – всі інші

Динамічне визначення класів:

<u>Список постійних динамічних класів</u>

3)У залежності від значення даних застосовувати декілька класів (масив класів)

<тег **: class="[** зм._модель_класу_1 , зм._модель_класу_2 , ...**]"** > . . . . . </тег >

включаємо режим прив’язування (ставимо двокрапку перед атрибутом class)

зазначаємо вираз, що є масивом змінних-моделей, що визначають відповідні класи

Динамічне визначення класів:

<u>Список постінийних динамічних класів</u>

3)У залежності від значення даних застосовувати декілька класів (масив класів)

<тег **: class="** **[** зм._модель_класу_1 , зм._модель_класу_2 , ...**]** **"** > . . . . . </тег >

включаємо режим прив’язування (ставимо двокрапку перед атрибутом class)

зазначаємо вираз, що є масивом змінних-моделей, що визначають відповідні класи

```
<div id="app">
    <label>
        <span :class="[statusIcon, statusColor]">
            Energy level
        </span>
        <input type="number" v-model="energyLevel" />
    </label>
</div>
```

Динамічне визначення класів:

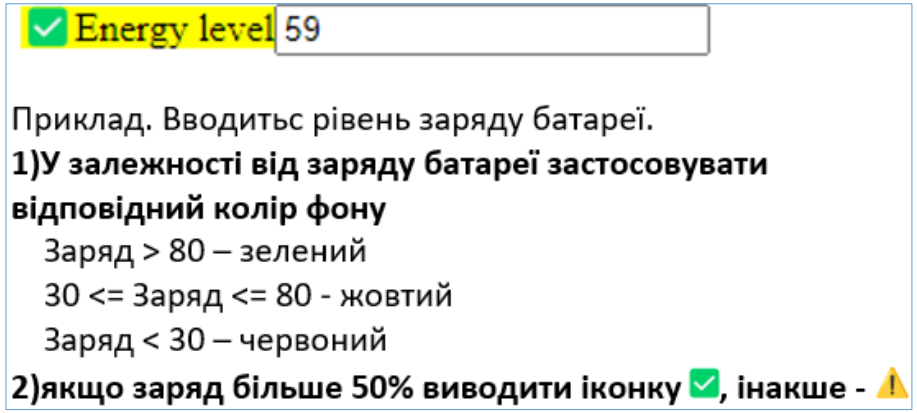
<u>Список постійних динамічних класів</u>

3)У залежності від значення даних застосовувати декілька класів (масив класів)

<тег **: class="** **[** зм._модель_класу_1 , зм._модель_класу_2 , ...**]** **"** > . . . . . </тег >

включаємо режим прив’язування (ставимо двокрапку перед атрибутом class)

зазначаємо вираз, що є масивом змінних-моделей, що визначають відповідні класи



Динамічне визначення класів:

<u>Список постінийних динамічних класів</u>

3)У залежності від значення даних застосовувати декілька класів (масив класів)

<тег : class=" [зм._модель_класу_1 , зм._модель_класу_2 , ...] "> . . . . . </тег >

1.Описуємо класи

```
<style>
  /* клалси для кольору фону */
  .high {
    background-color: green;
  }
  .middle {
    background-color: yellow;
  }
  .low {
    background-color: red;
  }
  /* класи для іконок */
  .success::before {
    content: '✅';
  }
  .warning::before {
    content: '⚠';
  }
</style>
```

✅ Energy level 59

Приклад. Вводитьс рівень заряду батареї.

1)У залежності від заряду батареї застосовувати відповідний колір фону

Заряд > 80 – зелений

30 <= Заряд <= 80 - жовтий

Заряд < 30 – червоний

2)якщо заряд більше 50% виводити іконку ✅, інакше - ⚠

Динамічне визначення класів:

Список постійних динамічних класів

3)У залежності від значення даних застосовувати декілька класів (масив класів)

<тег : class=" [зм._модель_класу_1 , зм._модель_класу_2 , ...] "> . . . . . </тег >

Energy level 59

Приклад. Вводитьс рівень заряду батареї.

1)У залежності від заряду батареї застосовувати відповідний колір фону

Заряд > 80 – зелений

30 <= Заряд <= 80 - жовтий

Заряд < 30 – червоний

2)якщо заряд більше 50% виводити іконку ✅, інакше - ⚠️

1.Описуємо класи

```
<style>
  /* клалси для кольору фону */
  .high {
    background-color: green;
  }
  .middle {
    background-color: yellow;
  }
  .low {
    background-color: red;
  }
  /* класи для іконок */
  .success::before {
    content: '✅';
  }
  .warning::before {
    content: '⚠️';
  }
</style>
```

```
computed: {
  statusColor() {
    let currectStateClass = null
    if (this.energyLevel) {
      if (this.energyLevel > 80) currectStateClass = 'high'
      else if (this.energyLevel > 30) currectStateClass = 'middle'
      else currectStateClass = 'low'
    }
    return currectStateClass
  },
  statusIcon() {
    let iconClass = null
    if (this.energyLevel) {
      iconClass = this.energyLevel >= 50 ? 'success' : 'warning'
    }
    return iconClass
  },
},
```

2.Створюємо обчислювані властивість

Динамічне визначення класів: ***Список постінийних динамічних класів***

3)У залежності від значення даних застосовувати декілька класів (масив класів)

<тег : class=" [зм._модель_класу_1 , зм._модель_класу_2 , ...] "> . . . . . </тег >

Energy level 59

Приклад. Вводитьс рівень заряду батареї.

1)У залежності від заряду батареї застосовувати відповідний колір фону

Заряд > 80 – зелений

30 <= Заряд <= 80 - жовтий

Заряд < 30 – червоний

2)якщо заряд більше 50% виводити іконку ✅, інакше - ⚠️

1.Описуємо класи

```
<style>
  /* клалси для кольору фону */
  .high {
    background-color: green;
  }
  .middle {
    background-color: yellow;
  }
  .low {
    background-color: red;
  }
  /* класи для іконок */
  .success::before {
    content: '✅';
  }
  .warning::before {
    content: '⚠️';
  }
</style>
```

```
<div id="app">
  <label>
    <span :class="[statusIcon, statusColor]">
      Energy level
    </span>
    <input type="number" v-model="energyLevel" />
  </label>
</div>
```

3.Підключаємо обчислювану властивість

```
computed: {
  statusColor() {
    let currectStateClass = null
    if (this.energyLevel) {
      if (this.energyLevel > 80) currectStateClass = 'high'
      else if (this.energyLevel > 30) currectStateClass = 'middle'
      else currectStateClass = 'low'
    }
    return currectStateClass
  },
  statusIcon() {
    let iconClass = null
    if (this.energyLevel) {
      iconClass = this.energyLevel >= 50 ? 'success' : 'warning'
    }
    return iconClass
  },
},
```

2.Створюємо обчислювані властивість

Приклад.

Динамічне визначення класів: ***<u>Умовні статичні класи</u>***

4)Наперед визначений клас застосовувати за деякої умови (об'єкт з властивостями-класами, значеннями-умова застосування)

```
<div
  :class="{
    клас_1 : умова_застосування_1,
    клас_2 : умова_застосування_2,
    . . . . . .
  }"
>
 . . . . . .
</div>
```

вказуємо клас, що застосовуємо у залежності від умови

умова застосуванян класу
(логічний вираз:
true – застосовуємо,
false – не застосовуємо)

включаємо
режим
прив'язування
(ставимо двокрапку
перед атрибутом class)

Динамічне визначення класів: ***<u>Умовні статичні класи</u>***

4)Наперед визначений клас застосовувати за деякої умови (об'єкт з властивостями-класами, значеннями-умова застосування)

```
<div
  :class="{
    клас_1 : умова_застосування_1,
    клас_2 : умова_застосування_2,
    ..........
  }"
>
 ..............
</div>
```

вказуємо клас, що застосовуємо у залежності від умови

умова застосуванян класу
(логічний вираз:
true – застосовуємо,
false – не застосовуємо)

включаємо
режим
прив'язування
(ставимо двокрапку
перед атрибутом class)

```
<input
    type="number"
    :class="{
        'allowed':isWorkerAgeValid,
        'empty':isAgeEmpty
    }"
    v-model="personAge"
/>
```

```
<style>
  .allowed {
    background-color: green;
  }
  .empty {
    background-color: yellow;
  }
</style>
```

Динамічне визначення класів: ***<u>Умовні статичні класи</u>***

4)Наперед визначений клас застосовувати за деякої умови (об’єкт з властивостями-класами, значеннями-умова застосування)

```
<div
   :class="{
     клас_1 : умова_застосування_1,
     клас_2 : умова_застосування_2,
     . . . . . .
    }"
>
 . . . . . .
</div>
```

Приклад. Вводимо вік працівника.

1) Якщо вік працівника більше за 18, то колір фону – зелений

2) Якщо поле порожне – колір фону жовтий

Динамічне визначення класів: ***<u>Умовні статичні класи</u>***

4)Наперед визначений клас застосовувати за деякої умови (об'єкт з властивостями-класами, значеннями-умова застосування)

```
<div
   :class="{
     клас_1 : умова_застосування_1,
     клас_2 : умова_застосування_2,
     ..........
   }"
>
 ..............
</div>
```

Приклад. Вводимо вік працівника.

1) Якщо вік працівника більше за 18, то колір фону – зелений

2) Якщо поле порожнє – колір фону жовтий

1.Описуємо класи

```
<style>
  .allowed {
    background-color: green;
  }
  .empty {
    background-color: yellow;
  }
</style>
```

Динамічне визначення класів: ***<u>Умовні статичні класи</u>***

4)Наперед визначений клас застосовувати за деякої умови (об'єкт з властивостями-класами, значеннями-умова застосування)

```
<div
   :class="{
     клас_1 : умова_застосування_1,
     клас_2 : умова_застосування_2,
     ..........
   }"
>
 ..............
</div>
```

Приклад. Вводимо вік працівника.
1) Якщо вік працівника більше за 18, то колір фону – зелений
2) Якщо поле порожне – колір фону жовтий

1.Описуємо класи

```
<style>
  .allowed {
    background-color: green;
  }
  .empty {
    background-color: yellow;
  }
</style>
```

2.Створюємо обчислювані властивості, що відображають умови застосування класів

```
computed: {
  isWorkerAgeValid() {
    return this.personAge && this.personAge >= 18
  },
  isAgeEmpty() {
    return !this.personAge
  },
},
```

Динамічне визначення класів: ***<u>Умовні статичні класи</u>***

4)Наперед визначений клас застосовувати за деякої умови (об'єкт з властивостями-класами, значеннями-умова застосування)

```
<div
   :class="{
      клас_1 : умова_застосування_1,
      клас_2 : умова_застосування_2,
      ..........
   }"
>
 ..............
</div>
```

1.Описуємо класи

```
<style>
  .allowed {
    background-color: green;
  }
  .empty {
    background-color: yellow;
  }
</style>
```

3.Підключаємо обчислювану властивість

```
<input
    type="number"
    :class="{
        'allowed':isWorkerAgeValid,
        'empty':isAgeEmpty
    }"
    v-model="personAge"
/>
```

Приклад. Вводимо вік працівника.

1) Якщо вік працівника більше за 18, то колір фону – зелений

2) Якщо поле порожне – колір фону жовтий

2.Створюємо обчислювані властивості, що відображають умови застосування класів

```
computed: {
    isWorkerAgeValid() {
        return this.personAge && this.personAge >= 18
    },
    isAgeEmpty() {
        return !this.personAge
    },
},
```

Динамічне визначення класів: ***<u>Умовні статичні класи</u>***

4)Наперед визначений клас застосовувати за деякої умови (об'єкт з властивостями-класами, значеннями-умова застосування)

```
<div
   :class="{
     клас_1 : умова_застосування_1,
     клас_2 : умова_застосування_2,
     ..........
   }"
>
 ..............
</div>
```

1.Описуємо класи

```
<style>
  .allowed {
    background-color: green;
  }
  .empty {
    background-color: yellow;
  }
</style>
```

3.Підключаємо обчислювану властивість

```
<input
    type="number"
    :class="{
        'allowed':isWorkerAgeValid,
        'empty':isAgeEmpty
    }"
    v-model="personAge"
/>
```

Приклад. Вводимо вік працівника.

1) Якщо вік працівника більше за 18, то колір фону – зелений

2) Якщо поле порожне – колір фону жовтий

2.Створюємо обчислювані властивості

```
computed: {
    isWorkerAgeValid() {
        return this.personAge && this.personAge >= 18
    },
    isAgeEmpty() {
        return !this.personAge
    },
},
```

Приклад.

Динамічне визначення класів: ***<u>Умовні динамічні класи</u>***

5)Наперед визначений клас застосовувати за деякої умови

(об’єкт з властивостями-моделями класів (*<u>у квадратних дужках!!!</u>*), значеннями-умова застосування)

```
<div
  :class="{
    [ зм._подель_класу_1 ] : умова_застосування_1,
    [ зм._подель_класу_2 ] :  умова_застосування_2,
    . . . . . .
  }"
>
. . . . . .
</div>
```

вказуємо клас, що застосовуємо у залежності від умови

умова застосуванян класу (логічний вираз:
true – застосовуємо,
false – не застосовуємо)

включаємо режим прив’язування (ставимо двокрапку перед атрибутом class)

Динамічне визначення класів: ***Умовні динамічні класи***

5)Наперед визначений клас застосовувати за деякої умови

(об'єкт з властивостями-моделями класів (*у квадратних дужках!!!*), значеннями-умова застосування)

```
<div
  :class="{
    [ зм._подель_класу_1 ] : умова_застосування_1,
    [ зм._подель_класу_2 ] : умова_застосування_2,
    ..........
  }"
>
 ..............
</div>
```

вказуємо клас, що застосовуємо у залежності від умови

умова застосуванян класу (логічний вираз: true – застосовуємо, false – не застосовуємо)

включаємо режим прив'язування (ставимо двокрапку перед атрибутом class)

```
<div id="app">
  <label> Age <input type="number" v-model="personAge" /> </label>
  <button
    :class="{
      [ statusClass ] : isWorkerAgeValid,
    }"
    :disabled="!isWorkerAgeValid"
  >
    Зареєструватись </button>
</div>
```

```
computed: {
  isWorkerAgeValid() {
    return this.personAge && this.personAge >= 18
  },
  statusClass() {
    return this.personAge < 45 ? 'young' : 'senior'
  },
},
```

```
<style>
  .young {
    background-color: green;
  }
  .senior {
    background-color: blue;
  }
</style>
```

Динамічне визначення класів: ***Умовні динамічні класи***

5)Наперед визначений клас застосовувати за деякої умови (властивості-моделі класів, значення-умови застосування)

```
<div
   :class="{

     [ зм._подель_класу_1 ] : умова_застосування_1,

     [ зм._подель_класу_2 ] : умова_застосування_2,
     ..........
    }"
>
 ..............
</div>
```

Приклад. Користувач вводить вік працівника.
Якщо вік користувача некоректний (<18), то кнопка реєстрації заблокована.
Якщо коректний вік користувача < 45, то кнопка зелена
Якщо коректний вік користувача >= 45, то кнопка синя

1.Описуємо класи

```
<style>
  .young {
    background-color: green;
  }
  .senior {
    background-color: blue;
  }
</style>
```

Динамічне визначення класів: ***<u>Умовні динамічні класи</u>***

5)Наперед визначений клас застосовувати за деякої умови (властивості-моделі класів, значення-умови застосування)

```
<div
   :class="{

     [ зм._подель_класу_1 ] : умова_застосування_1,

     [ зм._подель_класу_2 ] : умова_застосування_2,
     ..........
   }"
>
 ..............
</div>
```

Приклад. Користувач вводить вік працівника.
Якщо вік користувача некоректний (<18), то кнопка реєстрації заблокована.
Якщо коректний вік користувача < 45, то кнопка зелена
Якщо коректний вік користувача >= 45, то кнопка синя

1.Описуємо класи

```
<style>
  .young {
    background-color: green;
  }
  .senior {
    background-color: blue;
  }
</style>
```

2.Створюємо обчислювані властивості для динамічних класів та умов застосування

```
computed: {
  isWorkerAgeValid() {
    return this.personAge && this.personAge >= 18
  },
  statusClass() {
    return this.personAge < 45 ? 'young' : 'senior'
  },
},
```

Динамічне визначення класів: ***Умовні динамічні класи***

5)Наперед визначений клас застосовувати за деякої умови (властивості-моделі класів, значення-умови застосування)

```
<div
  :class="{
    [ зм._подель_класу_1 ] : умова_застосування_1,
    [ зм._подель_класу_2 ] : умова_застосування_2,
    ..........
  }"
>
 ..............
</div>
```

3.Підключаємо обчислювану властивість

Приклад. Користувач вводить вік працівника.
Якщо вік користувача некоректний (<18), то кнопка реєстрації заблокована.
Якщо коректний вік користувача < 45, то кнопка зелена
Якщо коректний вік користувача >= 45, то кнопка синя

```
<div id="app">
  <label> Age <input type="number" v-model="personAge" /> </label>
  <button
    :class="{
      [ statusClass ] : isWorkerAgeValid,
    }"
    :disabled="!isWorkerAgeValid"
  >
    Зареєструватись </button>
</div>
```

1.Описуємо класи

```
<style>
  .young {
    background-color: green;
  }
  .senior {
    background-color: blue;
  }
</style>
```

2.Створюємо обчислювані властивості для динамічних класів та умов застосування

```
computed: {
  isWorkerAgeValid() {
    return this.personAge && this.personAge >= 18
  },
  statusClass() {
    return this.personAge < 45 ? 'young' : 'senior'
  },
},
```

Приклад.

Динамічне визначення класів:

5)Декілька динамічних класів: деякі постійно, деякі застосовувати за певної умови (масив з класами, та об'єктами з умовними класами)

```
<div
  : class=" [
      зм._модель_класу_1 ,
      зм._модель_класу_2 ,
      . . . . . .
    {
      стат._клас_1 : умова_застосування_1,
      стат._клас_2 : умова_застосування_2,
      . . . . . .
      [ дин._клас_1 ] : ум._застост._дин._1,
      [ дин._клас_2 ] : ум._застост._дин._2,
      . . . . . .
    }
  ]"
>
. . . . . .
</div>
```

динамічні постійні класи (класи можуть бути різні з деякої мноижни, але завжди один з них застосовується)

Класи застосовуються тільки при виконанні умови
Уумова застосування - логічний вираз:
true – застосовуємо,
false – не застосовуємо

включаємо режим прив'язування (ставимо двокрапку перед атрибутом class)

Динамічне визначення класів:

5)Декілька динамічних класів: деякі постійно, деякі застосовувати за певної умови (масив з класами, та об'єктами з умовними класами)

```
<div
  : class=" [
      зм._модель_класу_1 ,
      зм._модель_класу_2 ,
       . . . . . .
     {
       стат._клас_1 : умова_застосування_1,
       стат._клас_2 : умова_застосування_2,
       . . . . . .
       [ дин._клас_1 ] : ум._застост._дин._1,
       [ дин._клас_2 ] : ум._застост._дин._2,
       . . . . . .
     }
   ]"
>
  . . . . . .
</div>
```

Приклад. Вводимо кількість грошей користувача та вартість товару.

//---------- ***Динамічний постійний***

Колір кнопки «Купити»:

- не введено якесь значення – сіра,
- недостатньо грошей - червона,
- достатньо грошей – зелена

//---------- ***Динамічні статичні***

Якщо вартість покупки більше 1000 грн (елітний) – додати жовту рамку

Якщо вартість покупки більье 100 000 (супер елітний) – додати тінь

//---------- ***Динамічні умовні***

Шрифт тексту кнопки **застосовувати якщо введено дані**:

- якщо вартість і ціна співпадають - червоний
- інкше - оранжевий

Динамічне визначення класів:

5)Декілька класів (масив класів): деякі постійно, деякі застосовувати за певної умови (масив з класами, та об'єктами з умовними класами)

```
<div
  : class=" [
      зм._модель_класу_1 ,
      зм._модель_класу_2 ,
       . . . . . .
     {
       стат._клас_1 : умова_застосування_1,
       стат._клас_2 : умова_застосування_2,
       . . . . . .
       [ дин._клас_1 ] : ум._застост._дин._1,
       [ дин._клас_2 ] : ум._застост._дин._2,
       . . . . . .
     }
   ]"
>
 . . . . . .
</div>
```

```
<button
  :class="[
    statusColor,
    {
      'elite':isElite,
      'super-elite':isSuperElite,
      [titleFont] : isCompleted
    }
]"
>
  Купити
</button>
```

Приклад. Вводимо кількість грошей користувача та вартість товару.

//---------- ***Динамічний постійний***

Колір кнопки «Купити»:
- не введено якесь значення – сіра,
- недостатньо грошей - червона,
- достатньо грошей – зелена

//---------- ***Динамічні статичні***

Якщо вартість покупки більше 1000 грн (елітний) – додати жовту рамку

Якщо вартість покупки більше 100 000 (супер елітний) – додати тінь

//---------- ***Динамічні умовні***

Шрифт тексту кнопки **застосовувати якщо введено дані**:
- якщо вартість і ціна співпадають - червоний
- інкше - оранжевий

Динамічне визначення класів:

5)Декілька класів (масив класів): деякі постійно, деякі застосовувати за певної умови (масив з класами, та об'єктами з умовними класами)

```
<div
  : class=" [
      зм._модель_класу_1 ,
      зм._модель_класу_2 ,
       ..........
      {
        стат._клас_1 : умова_застосування_1,
        стат._клас_2 : умова_застосування_2,
        ..........
        [ дин._клас_1 ] : ум._застост._дин._1,
        [ дин._клас_2 ] : ум._застост._дин._2,
        ..........
      }
   ]"
>
 ..........
</div>
```

```
<style>
  .empty {
    background-color: grey;
  }
  .allowed {
    background-color: green;
  }
  .not-allowed {
    background-color: red;
  }

  .elite {
    border: 3px solid yellow;
  }
  .super-elite {
    box-shadow: -5px -5px 20px 20px blue;
  }

  .equal-font {
    color: red;
  }
  .not-equal-font {
    color: orange;
  }
</style>
```

```
<div id="app">
  <label>
    Price
    <input type="number" v-model="productPrice" />
  </label>
  <label>
    User money
    <input type="number" v-model="userMoney" />
  </label>
  <button
    :class="[

      statusColor,

      {

        'elite':isElite,
        'super-elite':isSuperElite,

        [titleFont] : isCompleted

      }
    ]"
    >
    Купити
  </button>
</div>
```

```
computed: {
  isCompleted() {
    return this.userMoney && this.productPrice
  },
  statusColor() {
    let currectStateClass
    if (!this.isCompleted) currectStateClass = 'empty'
    else if (this.userMoney >= this.productPrice)
      currectStateClass = 'allowed'
    else currectStateClass = 'not-allowed'
    return currectStateClass
  },
  titleFont() {
    return this.userMoney === this.productPrice
      ? 'equal-font'
      : 'not-equal-font'
  },
  isElite() {
    return this.isCompleted && this.productPrice > 1000
  },
  isSuperElite() {
    return this.isCompleted && this.productPrice > 100000
  },
},
```

Приклад.

Динамічне визначення класів:

5)Декілька динамічних класів: деякі постійно, деякі застосовувати за певної умови (масив з класами, та об’єктами з умовними класами)

```
<div
  : class=" [
      зм._модель_класу_1 ,
      зм._модель_класу_2 ,
       . . . . . .
     {
       стат._клас_1 : умова_застосування_1,
       стат._клас_2 : умова_застосування_2,
       . . . . . .
       [ дин._клас_1 ] : ум._застост._дин._1,
       [ дин._клас_2 ] : ум._застост._дин._2,
       . . . . . .
     }
   ]"
>
 . . . . . .
</div>
```

Динамічне визначення класів:

5)Декілька динамічних класів: деякі постійно, деякі застосовувати за певної умови (масив з класами, та об’єктами з умовними класами)

```
<div
  : class=" [
      зм._модель_класу_1 ,
      зм._модель_класу_2 ,
      ..........
    {
      стат._клас_1 : умова_застосування_1,
      стат._клас_2 : умова_застосування_2,
      ..........
      [ дин._клас_1 ] : ум._застост._дин._1,
      [ дин._клас_2 ] : ум._застост._дин._2,
      ..........
    }
  ]"
>
 ..............
</div>
```

```
<style>
  .empty {
    background-color: grey;
  }
  .allowed {
    background-color: green;
  }
  .not-allowed {
    background-color: red;
  }

  .elite {
    border: 3px solid yellow;
  }
  .super-elite {
    box-shadow: -5px -5px 20px 20px blue;
  }

  .equal-font {
    color: red;
  }
  .not-equal-font {
    color: orange;
  }
</style>
```

Динамічне визначення класів:

5)Декілька динамічних класів: деякі постійно, деякі застосовувати за певної умови (масив з класами, та об'єктами з умовними класами)

```
<div
  : class=" [
      зм._модель_класу_1 ,
      зм._модель_класу_2 ,
      ..........
    {
      стат._клас_1 : умова_застосування_1,
      стат._клас_2 : умова_застосування_2,
      ..........
      [ дин._клас_1 ] : ум._застост._дин._1,
      [ дин._клас_2 ] : ум._застост._дин._2,
      ..........
    }
  ]"
>
 ..............
</div>
```

```
<style>
  .empty {
    background-color: grey;
  }
  .allowed {
    background-color: green;
  }
  .not-allowed {
    background-color: red;
  }

  .elite {
    border: 3px solid yellow;
  }
  .super-elite {
    box-shadow: -5px -5px 20px 20px blue;
  }

  .equal-font {
    color: red;
  }
  .not-equal-font {
    color: orange;
  }
</style>
```

```
computed: {
  isCompleted() {
    return this.userMoney && this.productPrice
  },
  statusColor() {
    let currectStateClass
    if (!this.isCompleted) currectStateClass = 'empty'
    else if (this.userMoney >= this.productPrice)
      currectStateClass = 'allowed'
    else currectStateClass = 'not-allowed'
    return currectStateClass
  },
  titleFont() {
    return this.userMoney === this.productPrice
      ? 'equal-font'
      : 'not-equal-font'
  },
  isElite() {
    return this.isCompleted && this.productPrice > 1000
  },
  isSuperElite() {
    return this.isCompleted && this.productPrice > 100000
  },
},
```

Динамічне визначення класів:

5)Декілька динамічних класів: деякі постійно, деякі застосовувати за певної умови (масив з класами, та об'єктами з умовними класами)

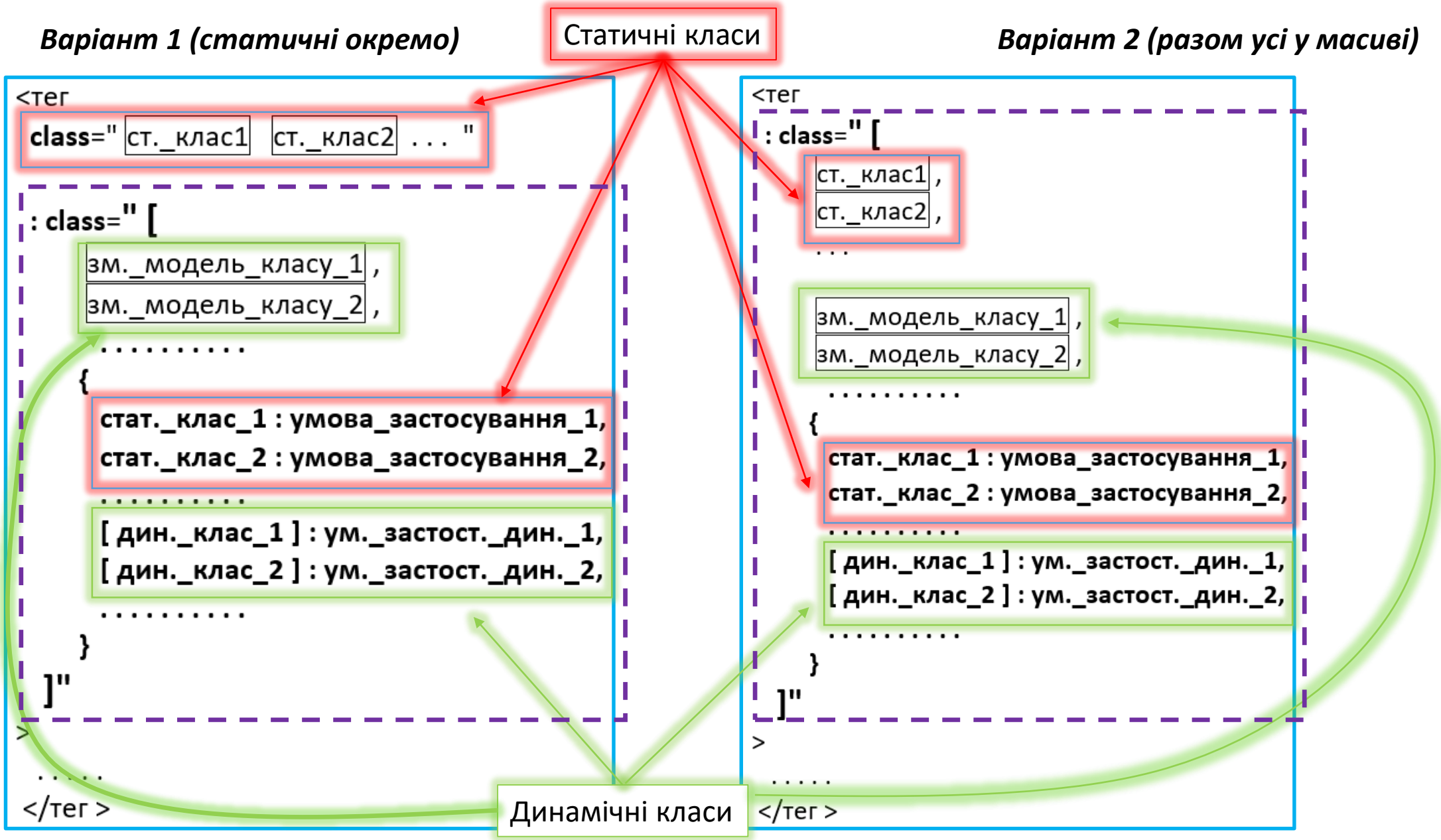
```
<div
  : class=" [
      зм._модель_класу_1 ,
      зм._модель_класу_2 ,
      ..........
    {
      стат._клас_1 : умова_застосування_1,
      стат._клас_2 : умова_застосування_2,
      ..........
      [ дин._клас_1 ] : ум._застост._дин._1,
      [ дин._клас_2 ] : ум._застост._дин._2,
      ..........
    }
  ]"
>
 ..............
</div>
```

```
<style>
  .empty {
    background-color: grey;
  }
  .allowed {
    background-color: green;
  }
  .not-allowed {
    background-color: red;
  }

  .elite {
    border: 3px solid yellow;
  }
  .super-elite {
    box-shadow: -5px -5px 20px 20px blue;
  }

  .equal-font {
    color: red;
  }
  .not-equal-font {
    color: orange;
  }
</style>
```

```
computed: {
  isCompleted() {
    return this.userMoney && this.productPrice
  },
  statusColor() {
    let currectStateClass
    if (!this.isCompleted) currectStateClass = 'empty'
    else if (this.userMoney >= this.productPrice)
      currectStateClass = 'allowed'
    else currectStateClass = 'not-allowed'
    return currectStateClass
  },
  titleFont() {
    return this.userMoney === this.productPrice
      ? 'equal-font'
      : 'not-equal-font'
  },
  isElite() {
    return this.isCompleted && this.productPrice > 1000
  },
  isSuperElite() {
    return this.isCompleted && this.productPrice > 100000
  },
},
```

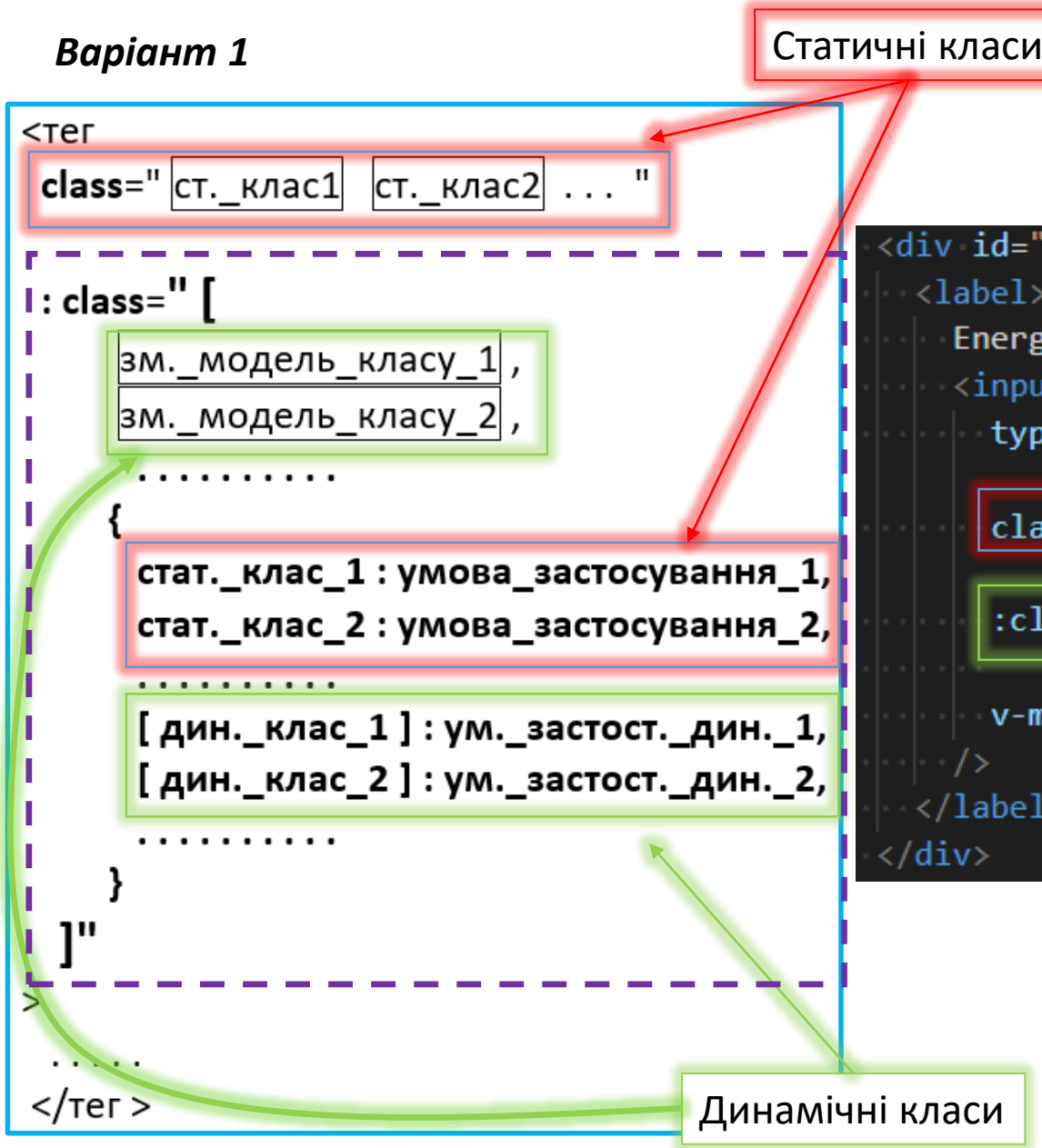
Приклад

Задання класів:
7)Статичні і динамічні класи одночасно



Задання класів:
7)Статичні і динамічні класи одночасно

Варіант 1



Статичні класи

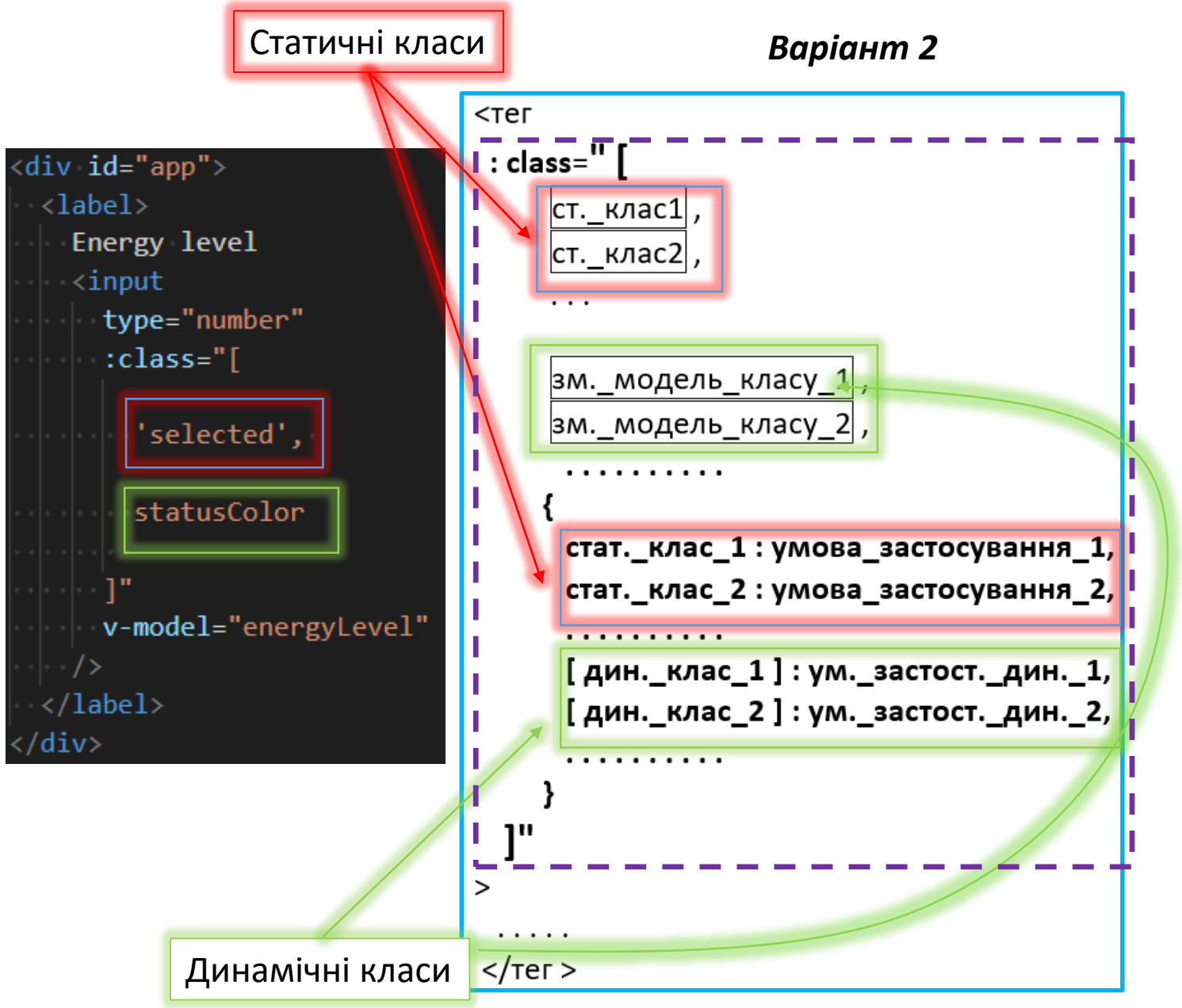
Динамічні класи

```
<div id="app">
  <label>
    Energy level
    <input
      type="number"
      class="selected"
      :class="statusColor"
      v-model="energyLevel"
    />
  </label>
</div>
```

```
const app = createApp({
  data() {
    return {
      energyLevel: 1,
    }
  },
  computed: {
    statusColor() {
      let currectStateClass
      if (this.energyLevel > 80)
        currectStateClass = 'high'
      else if (this.energyLevel > 30)
        currectStateClass = 'middle'
      else currectStateClass = 'low'
      return currectStateClass
    },
  },
}).mount('#app')
```

```
<style>
  .high {
    background-color: green;
  }
  .middle {
    background-color: yellow;
  }
  .low {
    background-color: red;
  }

  .selected {
    box-shadow: -5px -5px 20px 20px blue;
  }
</style>
```

Задання класів:
7)Статичні і динамічні класи одночасно



Задання класів:
7)Статичні і динамічні класи одночасно

```
const app = createApp({
  data() {
    return {
      energyLevel: 1,
    }
  },
  computed: {
    statusColor() {
      let currectStateClass
      if (this.energyLevel > 80)
        currectStateClass = 'high'
      else if (this.energyLevel > 30)
        currectStateClass = 'middle'
      else currectStateClass = 'low'
      return currectStateClass
    },
  },
}).mount('#app')
```

```
<style>
  .high {
    background-color: green;
  }
  .middle {
    background-color: yellow;
  }
  .low {
    background-color: red;
  }

  .selected {
    box-shadow: -5px -5px 20px 20px blue;
  }
</style>
```

```
<div id="app">
  <label>
    Energy level
    <input
      type="number"
      :class="[

        'selected',

        statusColor

      ]"
      v-model="energyLevel"
    />
  </label>
</div>
```

Статичні класи

Варіант 2

```
<тег
: class=" [
    ст._клас1 ,
    ст._клас2 ,
    ...

    зм._модель_класу_1 ,
    зм._модель_класу_2 ,
    ..........
  {
    стат._клас_1 : умова_застосування_1,
    стат._клас_2 : умова_застосування_2,
    ..........
    [ дин._клас_1 ] : ум._застост._дин._1,
    [ дин._клас_2 ] : ум._застост._дин._2,
    ..........
  }
 ]"
>
  .....
</тег >
```

Динамічні класи

Приклад

Задання стилів:

1) **Звичайні статичні стилі (значенням є звичайний string). Краще використати клас!!!!**
2) **Стилі з динамічним значенням властивостей (об'єкт, де ключі -- це CSS властивості записані у CamelCase, значення – значення CSS властивості)**

```
<input
  type="number"

  style="font-size: 2em"

  :style="
    {
      scale:scaleValue,
      backgroundColor:'red'
    }
  "


  v-model="energyLevel"
/>
```

Задання стилів:

1) **Звичайні статичні стилі (значенням є звичайний string). Краще використати клас!!!!**
2) **Стилі з динамічним значенням властивостей (об'єкт, де ключі -- це CSS властивості записані у CamelCase, значення – значення CSS властивості)**

```
<input
  type="number"

  style="font-size: 2em"

  :style="
    {
      scale:scaleValue,
      backgroundColor:'red'
    }
  "

  v-model="energyLevel"
/>
```

```
const app = createApp({
  data() {
    return {
      energyLevel: 100,
    }
  },
  computed: {
    scaleValue() {
      return this.energyLevel ? this.energyLevel / 100 : 1
    },
  },
}).mount('#app')
```

Приклад