Using and Creating Directive

Create your own directive to use in template.

- v-text: output some text
- v-html: output some HTML content (be careful of XSS)

Custom Directives: register directive globally in **main.js**

''' Vue.directive('highlight', { // Hooks }); '''

How directives work - Hook Function

Hooks

- **bind(el, binding, vnode)**: once directive is attached
- insert(el, binding, vnode): inserted in parent node
- **update(el, binding, vnode, oldVnode)**: once component is updated (without children)
- componentUpdated(el, binding, vnode, oldVnode): once component is updated (with children)
- unbind(el, bind, vnode): once directive is removed

Creating a Simple Directive

```
// main.js
...
Vue.directive('highlight', {
  bind(el, binding, vnode) {
    el.style.backgroundColor = 'green';   // set
style
  }
});
...
```

and in the template, we can use *v-highlight* directive. No need to pass values or arguments.

```
\\ [component].vue
<template>
...
  <p v-highlight> Color this </p>
...
</template>
```

Passing Values to Custom Directives

Passing values by the **binding** argument.

```
// main.js
...
Vue.directive('highlight', {
  bind(el, binding, vnode) {
    el.style.backgroundColor = binding.value;   //
passing value
  }
});
...
```

```
\\ [component].vue
<template>
...
  <p v-highlight="'red'"> Color this </p>
...
</template>
```

Passing Arguments to Custom Directives

Passing argument names and their values by the **binding** argument.

```
// main.js
...
Vue.directive('highlight', {
  bind(el, binding, vnode) {
    if (binding.arg == 'background') {          //
checking for argument
      el.style.backgroundColor = binding.value;
    } else {
      el.style.color = bind.value;
    }
  }
});
...
```

```
\\ [component].vue
<template>
...
  <p v-highlight:background="'red'"> Color this </p>
  <p v-highlight="'red'"> Color thai </p>
...
</template>
```

Custom Directive Modifier

Adding our own modifier. For example, **delay** modifier.

```
// main.js
...
Vue.directive('highlight', {
```

```
  bind(el, binding, vnode) {
    var delay = 0;
    if (binding.modifiers['delayed']) {
      delay = 3000;
    }
    setTimeout(()=> {
      if (binding.arg == 'background') {
        el.style.backgroundColor = binding.value;
      } else {
        el.style.color = bind.value;
      }
      , delay);
  }
});
...
```

```
\\ [component].vue
<template>
...
  <p v-highlight:background.delayed="'red'"> Color
this </p>
  <p v-highlight.delayed="'red'"> Color thai </p>
...
</template>
```

Registering Directives Locally

To register directives locally in a component.

```
\\ [component].vue
<script>
  export default {
    directives: {
      'local-highlight': {    // v-local-highlight
```

```
        bind(el, binding, vnode) {
          var delay = 0;
          if (binding.modifiers['delayed']) {
            delay = 3000;
          }
          setTimeout ()=> {
            if(bindind.arg == 'background') {
              el.style.backgroundColor =
binding.value;
            } else {
              el.style.color = binding.value;
            }
          }, delay);
        }
      }
    }
  }
</script>

<template>
...
  <p v-local-highlight:background.delayed="'red'">
Color this </p>
  <p v-highlight.delayed="'red'"> Color thai </p>
...
</template>
```

Using Multiple Modifiers

```
\\ [component].vue
<script>
  export default {
    directives: {
      'local-highlight': {    // v-local-highlight
        bind(el, binding, vnode) {
          var delay = 0;
```

```
          if (binding.modifiers['delayed']) {
            delay = 3000;
          }
          if (binding.modifiers['blink']) {
            var mainColor = bind.value;
            var secondColor = 'blue';
            var currentColor = mainColor;

            setTimeout ()=> {
              setInterval(() => {
                if (currentColor == seconColor ?
currentColor = mainColor : currentColor =
secondColor;
                if(bindind.arg == 'background') {
                  el.style.backgroundColor =
currentColor;
                } else {
                  el.style.color = currentColor;
                }
              }, 1000);                 // a second

            }, delay);
          } else {
            setTimeout ()=> {
              if(bindind.arg == 'background') {
                el.style.backgroundColor =
binding.value;
              } else {
                el.style.color = binding.value
              }
            }, delay);
          }
        }
      }
    }
  }
</script>

<template>
```

```
...
  <p v-local-
highlight:background.delayed.blink="'red'"> Color
this </p>
  <p v-highlight.delayed="'red'"> Color thai </p>
...
</template>
```

Passing more Complex Values to Directives

Passing complex values. For example, passing multiple values.

```
<template>
...
  <!-- Passing Object as value -->
  <p v-local-highlight:background.delayed.blink="
{mainColor: 'red', secondColor: 'green', delay:
500}"> Color this </p>
  <p v-highlight.delayed="'red'"> Color thai </p>
...
</template>
```

```
\\ [component].vue
<script>
  export default {
    directives: {
      'local-highlight': {    // v-local-highlight
        bind(el, binding, vnode) {
          var delay = 0;
          if (binding.modifiers['delayed']) {
            delay = 3000;
          }
          if (binding.modifiers['blink']) {
            var mainColor = bind.value.mainColor;
```

```
            var secondColor = bind.value.secondColor;
            var currentColor = mainColor;

            setTimeout ()=> {
              setInterval(() => {
                if (currentColor == seconColor ?
currentColor = mainColor : currentColor =
secondColor;
                if(bindind.arg == 'background') {
                  el.style.backgroundColor =
currentColor;
                } else {
                  el.style.color = currentColor;
                }
              }, 1000);                 // a second

            }, delay);
          } else {
            setTimeout ()=> {
              if(bindind.arg == 'background') {
                el.style.backgroundColor =
binding.value.mainColor;
              } else {
                el.style.color =
binding.value.mainColor;
              }
            }, delay);
          }
        }
      }
    }
  }
</script>
```