



Introduction to Declarative DOM Manipulation

Using Vue.js

Hi, I'm Misa

... and the cat is Myy.

- Web developer for 5+ years.
- Working at **Knowit** since 2021.
- Actually a math and computer science teacher.
- Board games, video games and DnD.



knowit

- Nordic consultancy.
- 500 people in Finland.
- Code, data, cloud and quality.



Makers of a
sustainable future[™]

Agenda

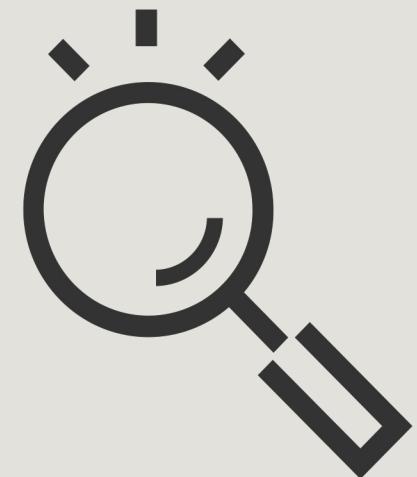
Introduction to Declarative DOM Manipulation

1. Defining Declarative
2. Vue.js
3. Workshop



Introduction to Declarative DOM Manipulation

Defining Declarative



Defining Declarative

- **Declarative** code describes what we want.
- “*I want a strawberry milkshake.*”
- **Imperative** code describes how to do things.
- “*Blend frozen strawberries and milk.
Add ice cream and blend.
Pour into a glass and serve.*”



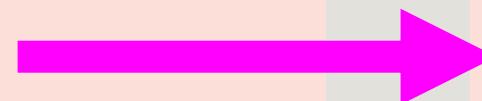
Declarative programming is a luxury made possible by tons of imperative code.

Declarative vs. Imperative Example

how 2 code

In this video I teach
you how to code.
Please like and
subscribe.

Download



User clicks button

how 2 code

In this video I teach
you how to code.
Please like and
subscribe.

Download

Imperative

```
<button id="download-button" onClick="startDownload()>
  Download
</button>
```

no usages

```
const startDownload = async () => {
  // Disable the button
  document.querySelector( {selectors: '#download-button'}
    .setAttribute( {qualifiedName: 'disabled'}, {value: ''});

  await actuallyDownloadSomeFile();

  // Enable the button
  document.querySelector( {selectors: '#download-button'}
    .removeAttribute( {qualifiedName: 'disabled'});
}
```

Declarative (Vue.js)

```
<button :disabled="isLoading" @click="startDownload">  
  Download  
</button>
```

```
const isLoading = ref(false);  
  
const startDownload = async () => {  
  isLoading.value = true;  
  await actuallyDownloadSomeFile();  
  isLoading.value = false;  
}
```



Reactive



Reactive Programming

*The declarative expression of
the relationship between values
that change over time.*

$$a = b + c$$



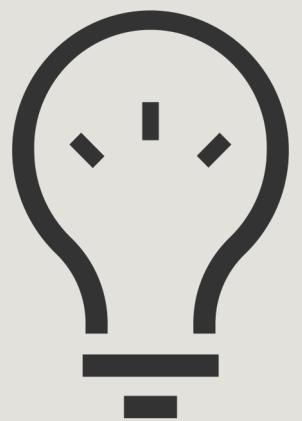
Reactive Introduction to Declarative DOM Manipulation

Using Vue.js



Introduction to Declarative DOM Manipulation

Vue.js



Vue.js

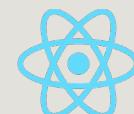


vuejs.org

- JavaScript web UI framework.
- Build reactive applications.



Get started with [Vite](#)

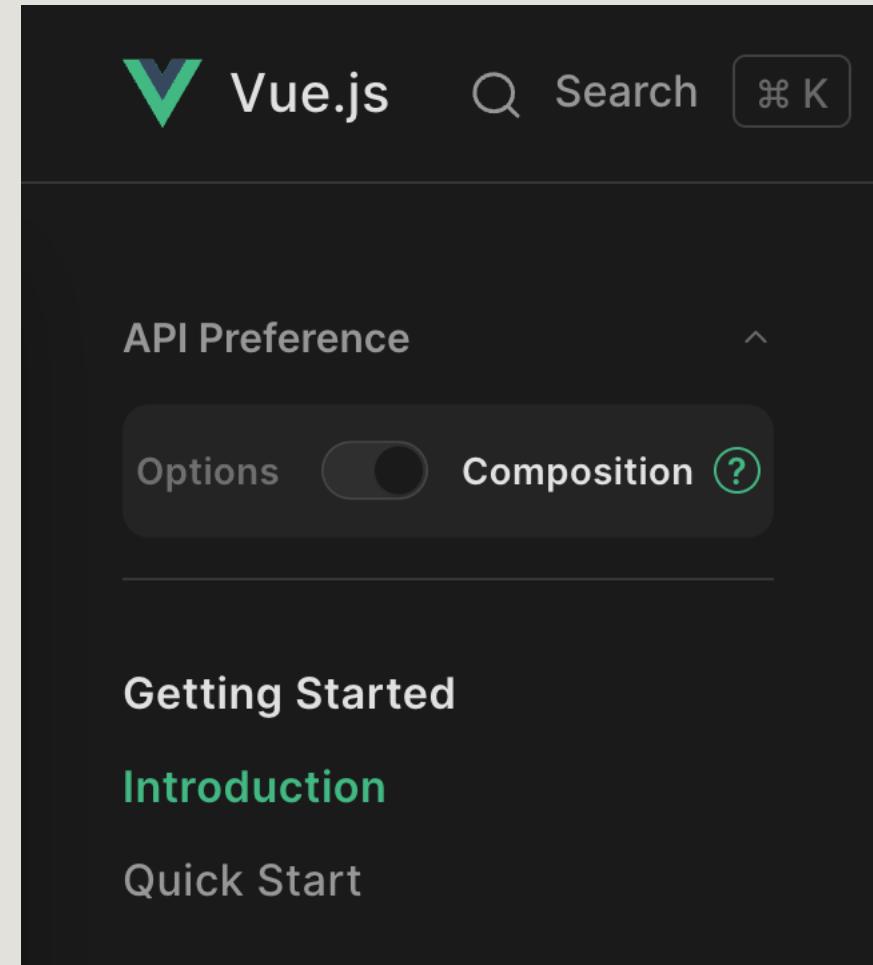


Similar to [React](#)

Today's focus:

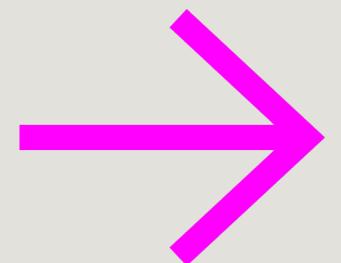
Options API

Composition API



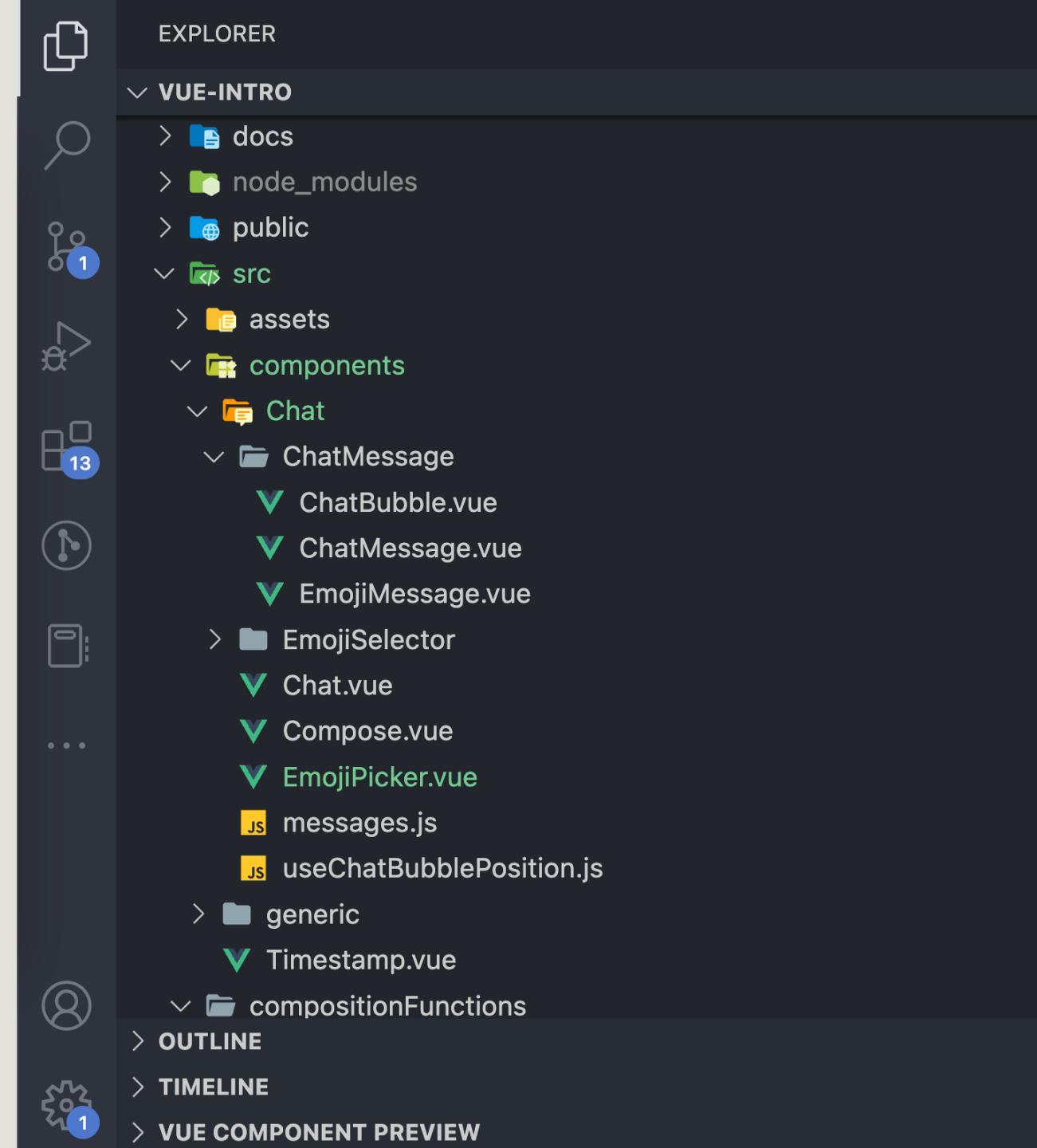
Vue Crash Course

1. Project structure
2. Components
3. Component communication
4. Directives
5. Reactivity tools



1. Vue Project Structure

- Hierarchy of **.vue** files
 - Single-file components
- Functions (helpers, utilities, etc.)
- Assets, project configuration files, etc.



2. Components

- A single **.vue** file
- Contains:
 - Script (JavaScript)
 - Template (HTML)
 - Style (CSS)

SCRIPT

TEMPLATE

STYLE

2. Components

- A single **.vue** file
- Contains:
 - Script (JavaScript)
 - Template (HTML)
 - Style (CSS)

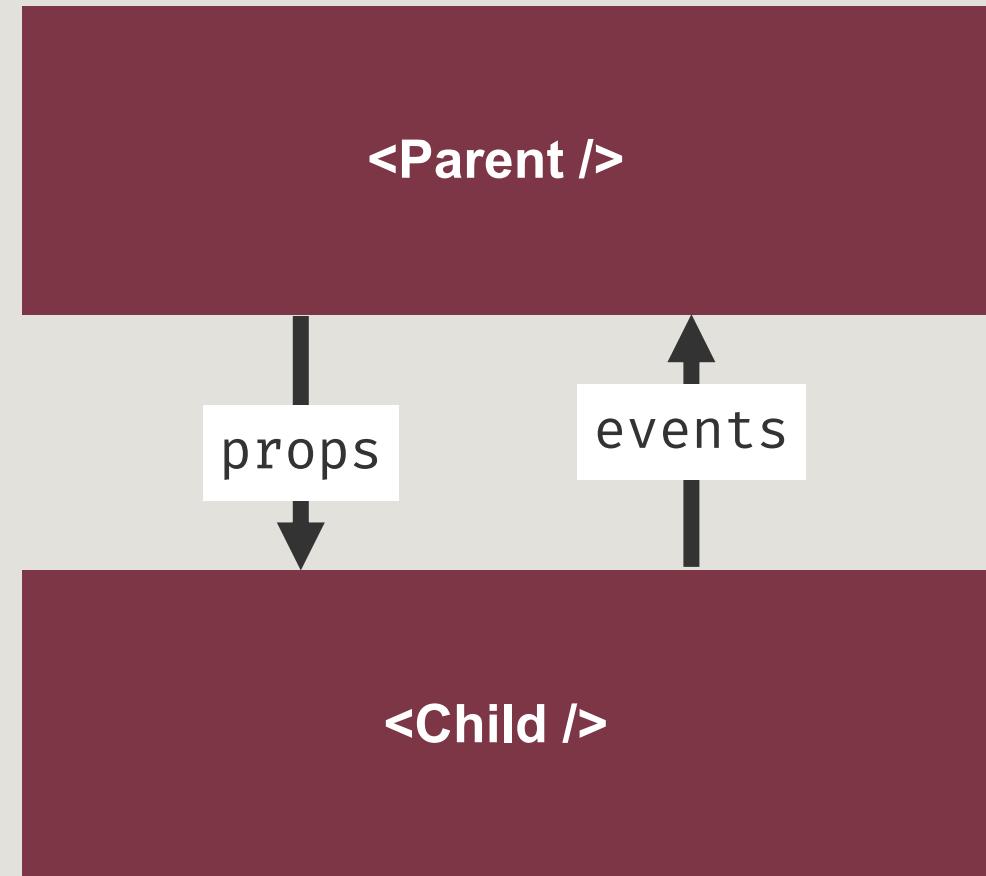
```
You, 11 months ago | 1 author (You)
1 ✓ <script setup>
2 // A chat bubble container
3 import useChatBubblePosition from "../useChatBubblePosition";
4
5 ✓ const props = defineProps({
6 ✓   direction: {
7   |   type: String,
8   |   default: "right",
9   | },
10 });
11
12 // Import CSS variables from a 'hook'
13 const { cssVars } = useChatBubblePosition(props.direction);
14 </script>
```

```
15
16 ✓ <template>
17 ✓   <div class="chat-bubble shadow-1" :style="cssVars">
18   |   0 references
19   |   <slot />
20   |   </div>
21 </template>
```

```
22 ✓ <style scoped lang="scss">
23   |   1 reference
24   ✓ .chat-bubble {
25     |   background-color: #fbfaf8;
26     |   color: #281822;
27     |   border-radius: 10px;
28     |   padding: 0.5rem 1rem;
29     |   display: inline-block;
30     |   position: relative;
31     |   align-self: var(--align);
32     |   margin: var(--margin);
```

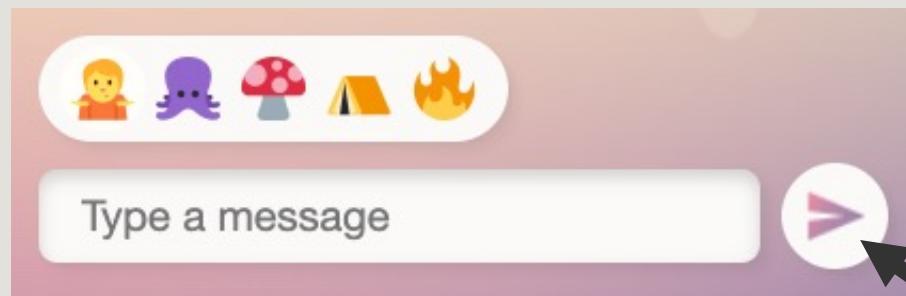
3. Component Communication

- Components pass information to each other with **props** and **events**.



3. Component Communication

- Components pass information to each other with **props** and **events**.



Compose.vue

```
<Button icon="send" @click="send" />
```

props

events

Button.vue

```
const props = defineProps({  
  icon: {  
    type: String,  
    default: null,  
  },  
});
```

```
const emit = defineEmits(["click"]);  
emit("click", someParameter);
```



4. Directives

- Built-in reusable code or logic.
- Allow developers to manipulate the DOM in many ways.
- E.g. conditional rendering and looping.
- [See the docs!](#)

Directives

v-text

v-html

v-show

v-if

v-else

v-else-if

v-for

v-on

v-bind

v-model

v-slot

v-pre

v-once

v-memo

v-cloak

Components

<Transition>

<TransitionGroup>

<KeepAlive>

<Teleport>

<Suspense>

Special Elements

<component>

<slot>

<template>

/ 4. Directives

v-if

- Conditionally render an element.
- See also: *v-else* and *v-else-if*.

```
<marquee v-if="year < 2010">
| Hello from Myspace
</marquee>
```

/ 4. Directives

v-for

- Loop through a list.
- Render each element.

```
<Game  
  v-for="game in games"  
  :title="game.name"  
  :description="game.description"  
/>
```



/ 4. Directives

v-bind

- Binds a variable to a prop or DOM attribute.
- The "value" will be the variable's value, not the literal text entered.
- **v-bind:title**
can be shortened to **:title**.

```
<Timestamp v-bind:date="props.message.timestamp" />
```

```
<Timestamp :date="props.message.timestamp" />
```

24.1.2023 14:22

props.message.timestamp

/ 4. Directives

v-model

- Binds a variable to a prop or DOM attribute.
- Also adds a **handler** to change that value.
- Works automatically on most elements!

```
<Input v-model="value" />  
  
// The above is short for  
<Input  
  :value="value"  
  @input="  
    /*function that updates the value*/  
  "  
/>
```

/ 4. Directives

v-on

- Event handler.
- “*When an event with this name is emitted, do this.*”
- v-on:click
can be shortened to @click

```
<Button v-on:click="send" />
```

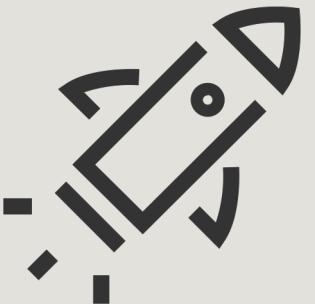
```
<Button @click="send" />
```



```
function send() { ... }
```

5. Reactivity Tools

- Make your app reactive.
- [See the docs!](#)



`ref()`
`computed()`
`watch(), watchEffect()`
`provide(), inject()`

/ 5. Reactivity Tools

ref()

- Declare a reactive variable.
- Use `reactive()` for a reactive object.

```
const age = ref(0);
```

```
function growOlder() {  
  age.value = age.value + 1;  
}
```

```
<p>  
  I am {{ age.value }} years old.  
</p>
```

/ 5. Reactivity Tools

computed()

- Store a reactive computed value.

```
const area = computed(() =>
  width * height
);
```

/ 5. Reactivity Tools

watch(), watchEffect()

- Watch one or more variables.
- Do something when their values change.

```
// Watch the query variable  
  
watch(query, () => {  
  fetch(`/items?search=${query}`)  
});  
  
// Automatically watch all reactive  
// variables  
  
watchEffect(() => {  
  fetch(`/items?search=${query}`)  
});
```

/ 5. Reactivity Tools

provide(), inject()

- Provide data to all components in the component tree.
- Inject some provided data into a component.
- No need to drill the data as props through multiple components.

```
// Somewhere high up in the
// component tree

const allTheData = { ... };
provide('data', allTheData);

// Some small component way down
// in the component tree

const allTheData = inject('data');
```

/ 5. Reactivity Tools (not really)

Slots

- Display content inside a component.
- Slots can be named.

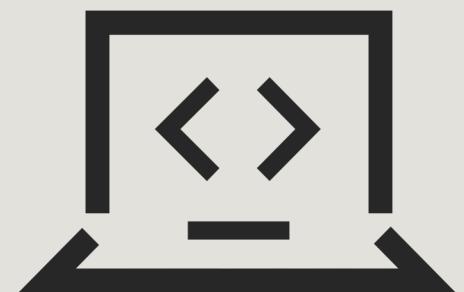
```
// MyContainer.vue
<div>
  <slot />
  <p>Check out these links!</p>
  <slot name="links" />
</div>

// App.vue
<MyContainer>
  <p>Contact us via pigeon!</p>
  <template #links>
    <a href="...">Rent a pigeon</a>
    <a href="...">Buy an organic pigeon</a>
  </template>
</MyContainer>
```

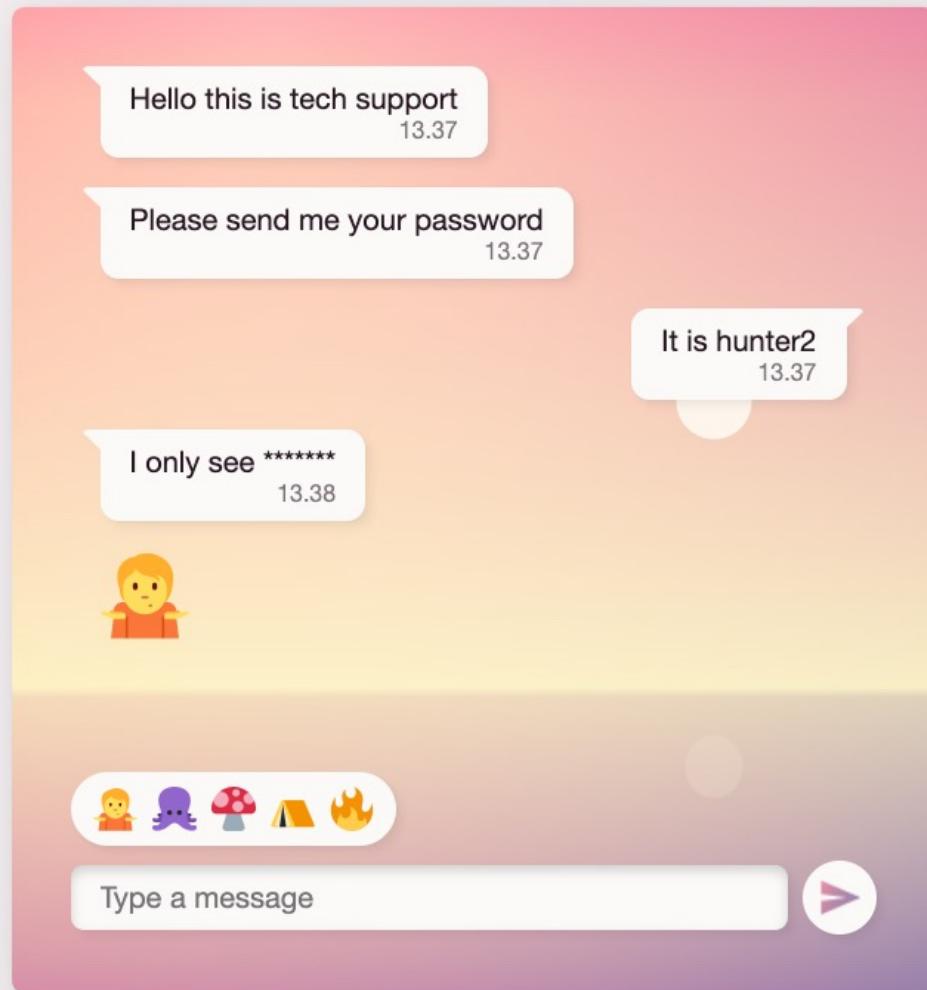


Introduction to Declarative DOM Manipulation

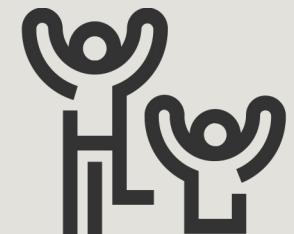
Workshop



Finish a chat app



1. Get the code:
 - shorturl.at/DJLP3
2. Read the README for instructions.
3. Get coding!
4. Help a friend and ask questions.



Thanks