Vue Workshop:
Advanced Features from the Ground Up

Evan You

Agenda

- Intro
- Fundamentals: Reactivity
- Fundamentals: Writing Plugins
- Fundamentals: Render Functions
- --- Lunch Break ---
- State Management
- Routing
- --- Coffee Break ---
- Form Validation
- Component Patterns

Fundamentals: Reactivity

```
let a = 3
```

```
let a = 3
let b = a * 10
```

```
let a = 3
let b = a * 10
console.log(b) // 30
```

```
let a = 3
let b = a * 10
console.log(b) // 30
a = 4
console.log(b) // 30
```

```
let a = 3
let b = a * 10
console.log(b) // 30
a = 4
b = a * 10
console.log(b) // 40
```

	A	B
1	4	40 (fx = A1 * 10)

```
onAChanged(() => {
  b = a * 10
})
```

```
<span class="cell b1"></span>
```

```
<span class="cell b1"></span>


document
  .querySelector(‘.cell.b1’)
  .textContent = state.a * 10
```

```
<span class="cell b1"></span>


onStateChanged(() => {
  document
    .querySelector('.cell.b1')
    .textContent = state.a * 10
})
```

```
<span class="cell b1">
  {{ state.a * 10 }}
</span>


onStateChanged(() => {
  view = render(state)
})
```

```
onStateChanged(() => {
  view = render(state)
})
```

```
? onStateChanged(() => {
    view = render(state)
  })
```

```
let update, state
const onStateChanged = _update => {
  update = _update
}

const setState = newState => {
  state = newState
  update()
}
```

React

```
onStateChanged(() => {
  view = render(state)
})

setState({ a: 5 })
```

```
onStateChanged(() => {
  view = render(state)
})
```

? `state.a = 5`

```
autorun(() => {
  console.log(state.count)
})
```

This is the basic form of the dependency tracking systems as seen in Knockout.js, Meteor Tracker, Vue.js and MobX.

Exercise: Mini Data Observer

Goal: implement observe() and autorun() which triggers re-computation when object is mutated

observe(): make an object reactive with Object.defineProperty

autorun(): run a computation while collecting its dependencies

Fundamentals: Writing Plugins

Vue.use(plugin)

```
function (Vue, options) {
  // … plugin code
}
```

Vue.mixin(options)

$options

Exercise: Writing a Simple Plugin

Goal: write a plugin that logs a component's "bar" option if it's present.

Fundamentals: Render Functions

Initial Render

Template

-> (compiled into) Render Function

-> (returns) Virtual DOM

-> (generates) Actual DOM

Subsequent Updates

Render Function

-> (returns) New Virtual DOM

-> (diffed against Old Virtual DOM) DOM Updates

-> (applied to) Actual DOM

What is a Virtual DOM, anyway?

Actual DOM

```
document.createElement(‘div’)
```

Virtual DOM

```
vm.$createElement(‘div’)
```

Actual DOM

```
"[object HTMLDivElement]"
```

Virtual DOM

```
{ tag: ‘div’, data: { attrs: {}, … }, children: [] }
```

Actual DOM

```
"[object HTMLDivElement]"
^ Browser Native Object (expensive)
```

Virtual DOM

```
{ tag: ‘div’, data: { attrs: {}, … }, children: [] }
^ Plain JavaScript Object (cheap)
```

Virtual DOM:

(Essentially) A lightweight JavaScript data format to represent what the actual DOM should look like at a given point in time

Virtual DOM:

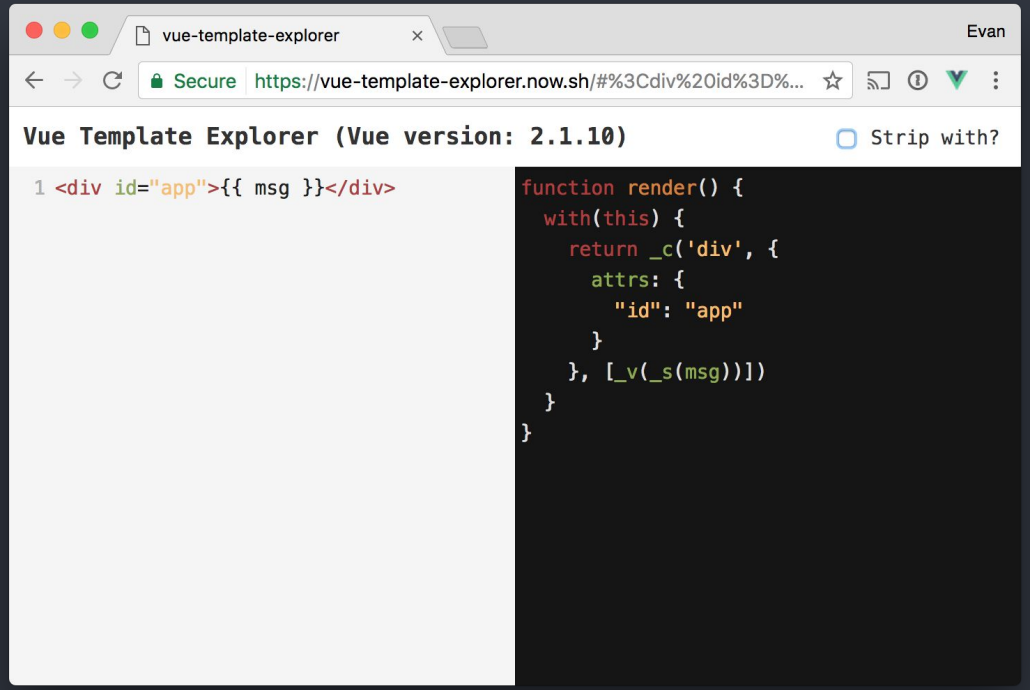
Decouples rendering logic from the actual DOM - enables rendering capabilities in non-browser environments, e.g. server-side and native mobile rendering.

Render Function:

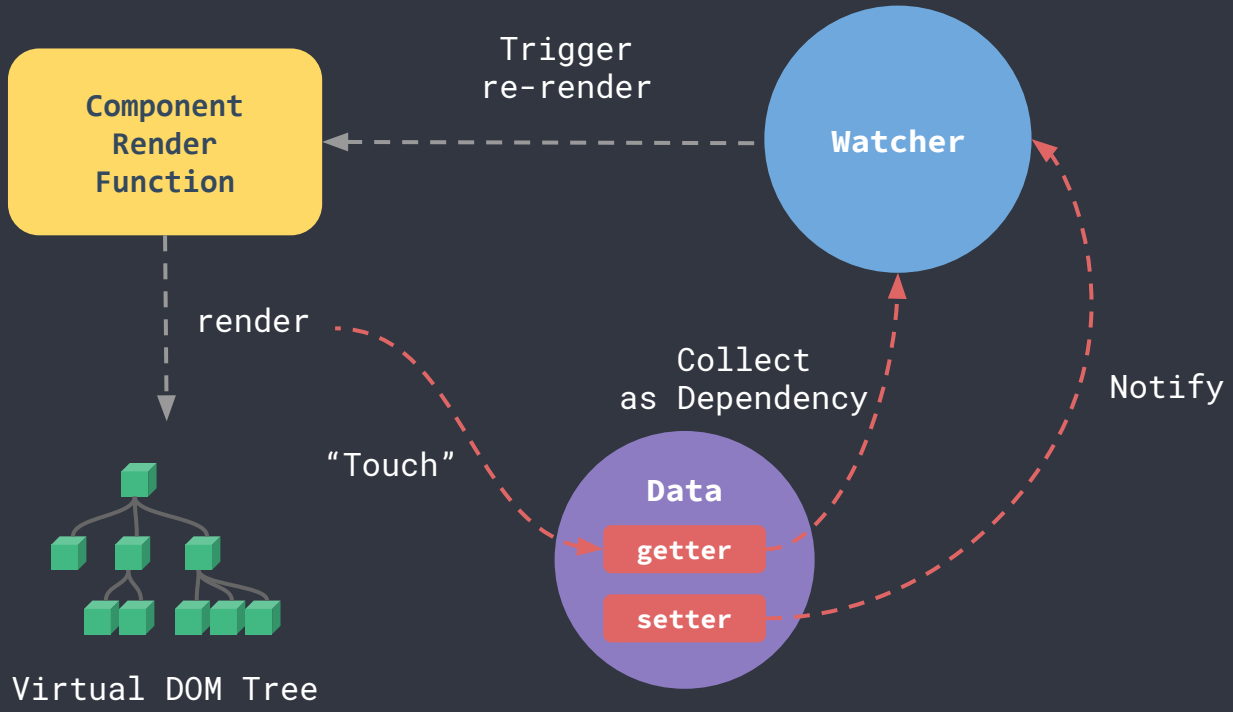
A function that returns Virtual DOM.

Template -> [Compiler] -> Render Function

https://template-explorer.vuejs.org



Putting Everything Together...


Component
Render
Function
Trigger
re-render
Watcher
render
Collect
as Dependency
Notify
“Touch”
Data
getter
setter
Virtual DOM Tree

Render Function API

```
export default {
  render (h) {
    return h('div', {}, [...])
  }
}
```

The “h” function

```
h(‘div’, ‘some text’)

h(‘div’, { class: ‘foo’ }, ‘some text’)

h(‘div’, { … }, [
  ‘some text’,
  h(‘span’, ‘bar’)
])
```

https://vuejs.org/v2/guide/render-function.html#The-Data-Object-In-Depth

h can directly render a component

```
import MyComponent from '...'

h(MyComponent, {
  props: { … }
})
```

Exercise: Using Render Functions

Goal: rendering a list of elements with different tags based on prop.

Exercise: Using Render Functions

Goal: rendering a dynamic component based on prop

Exercise: Using Render Functions

Goal: enhancing an existing component
(aka. Higher Order Component)

Exercise: Using Render Functions

Goal: enhancing a slot child component

LUNCH BREAK!

State Management

Agenda

- Shared objects as stores
- Shared Vue instances as stores
- Abstracting the mutations API
- Bonus: using a more functional interface

Exercise: Shared Object

Goal: create two Counter components that share the same state object

Exercise: Shared Vue Instance as Store

Goal: create two Counter components that share the same Vue instance as store

Exercise: Abstracting Mutations

Goal: trigger counter change with mutations

Bonus Exercise: Functional State Management

Goal: implement a redux-like state management interface

COFFEE BREAK

Routing

Agenda

- The simplest router (hashchange + <component :is>)
- Extracting a route table
- URL matching with `path-to-regexp`

Exercise: Simple Router

Goal: switch views based on hash change

Exercise: Route Table

Goal: extract route matching rules into a route table object

Exercise: Dynamic Route Segments

Goal: parsing dynamic segments from route paths

Form Validation

Different Validation Plugin Styles

- Markup-based (vee-validate)
- Model-based (vuelidate)

Exercise: A Form Validation Plugin

Goal: real time validation for a text input

Component Patterns

Higher Order Components

Example: Buttons

Abstract Components

Example: Error Boundary

Scoped Slots

Example: List, Fetch
https://jsfiddle.net/yyx990803/kyt43L2r/