Vue 3.0 Updates

Evan You

VueConf TO, Nov. 2018

What's coming in Vue 3.0

- Make it faster
- Make it smaller
- Make it more maintainable
- Make it easier to target native
- Make your life easier

Make it faster

Virtual DOM implementation re-written from the ground up

Up to 100% faster mounting & patching

More compile-time hints
to reduce runtime overhead

Component fast path + Monomorphic calls + Children type detection

Template

```
<Comp></Comp>
<div>
  <span></span>
</div>
```

Compiler output

```
render() {
 const Comp = resolveComponent('Comp', this)
 return createFragment([
   createComponentVNode(Comp, null, null, 0 /* no children */),
   createElementVNode('div', null, [
     createElementVNode('span', null, null, 0 /* no children */)
   ], 2 /* single vnode child */)
 ], 8 /* multiple non-keyed children */)
}
```

- Skip unnecessary condition branches
- Easier for JavaScript engine to optimize

Optimized Slots Generation

Template

```
<Comp>
  <div>{{ hello }}</div>
</Comp>
```

Compiler output

```
render() {
 return h(Comp, null, {
   default: () => [h('div', this.hello)]
 }, 16 /* compiler generated slots */)
}
```

- Ensure dependencies are tracked by correct instance
- Avoid unnecessary parent / children re-renders

Static Tree Hoisting

Template

```
<div>
  <span class="foo">
    Static
  </span>
  <span>
    {{ dynamic }}
  </span>
</div>
```

Compiler output

```
const __static1 = h('span', {
  class: 'foo'
}, 'static')

render() {
  return h('div', [
    __static1,
    h('span', this.dynamic)
  ])
}
```

- Skip patching entire trees
- Works even with multiple occurrences

Static Props Hoisting

Template

```
<div id="foo" class="bar">
  {{ text }}
</div>
```

Compiler output

```
const __props1 = {
 id: 'foo',
 class: 'bar'
}

render() {
 return h('div', __props1, this.text)
}
```

- Skip patching the node itself, but keep patching children

Inline Handler Hoisting

Template

```
<Comp @event="count++"/>
```

Compiler output

```
import { getBoundMethod } from 'vue'

function __fn1 () {
  this.count++
}

render() {
 return h(Comp, {
   onEvent: getBoundMethod(__fn1, this)
 })
}
```

- Avoid unnecessary re-renders due to different inline function identity

Proxy-based observation mechanism with full language coverage + better perf

- Property addition / deletion
- Array index / length mutation
- Map, Set, WeakMap, WeakSet
- Classes

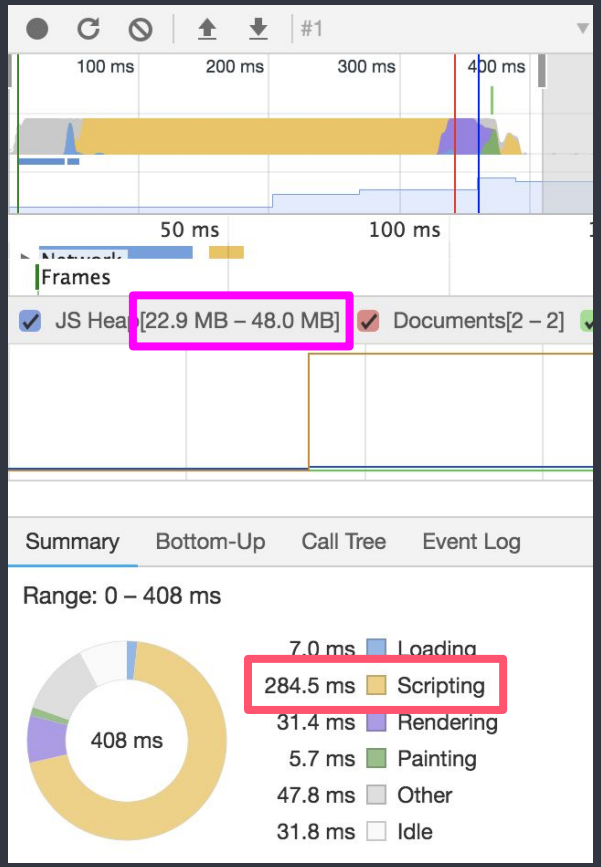
Faster instance property proxying using native Proxy

Bye Object.defineProperty!

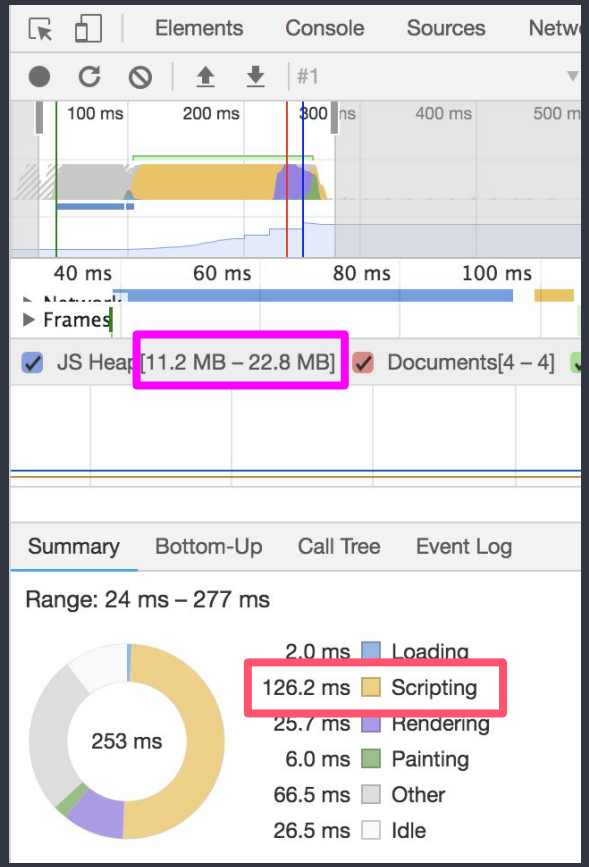
Up to 100% faster
Component instance
initialization

Double the speed
Half the memory usage

v2.5



v3.0-proto



- Rendering 3000 stateful component instances

Make it smaller

Tree-shaking Friendly

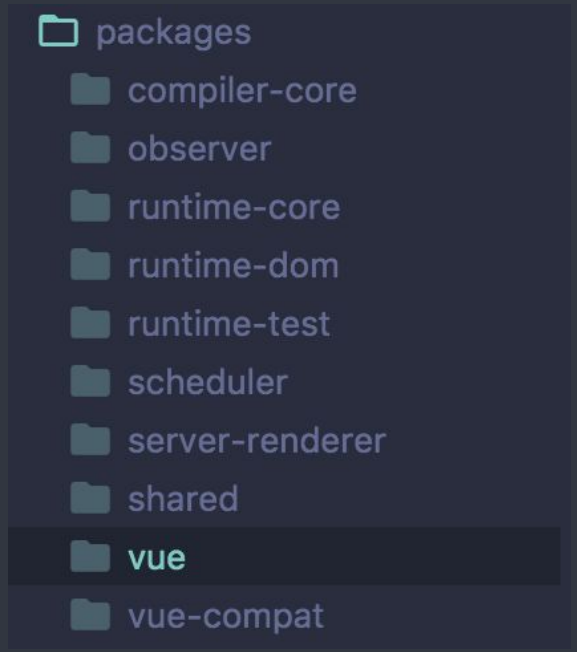
- Built-in components (keep-alive, transition...)
- Template directive runtime helpers (v-model, v-for...)
- Utility functions (asyncComponent, mixins, memoize...)

New core runtime: ~10kb gzipped

Make it more maintainable

Flow -> TypeScript

Decoupled Packages



Compiler Rewrite

- Pluggable architecture
- Parser w/ location info (source maps!)
- Serve as infrastructure for more robust IDE support

Make it easier to target native

Custom Renderer API

```
import { createRenderer } from '@vue/runtime-core'

const { render } = createRenderer({
 nodeOps,
 patchData
})
```

Make your life easier

Exposed reactivity API

```
import { observable, effect } from 'vue'

const state = observable({
 count: 0
})

effect(() => {
 console.log(`count is: ${state.count}`)
}) // count is: 0

state.count++ // count is: 1
```

Easily identify why a component is re-rendering

```
const Comp = {
  render(props) {
    return h('div', props.count)
  },
  renderTriggered(event) {
    debugger
  }
}
```

Demo

Improved TypeScript Support w/ TSX

```
interface HelloProps {
  text: string
}

class Hello extends Component<HelloProps> {
  count = 0

  render() {
    return <div>
      {this.count}
      {this.$props.text}
    </div>
  }
}
```

Demo

Better warning traces

- Now includes functional components
- Inspectable props
- Traces are available in more warning cases

Demo

Experimental Hooks API

Demo

Experimental
Time Slicing Support

Demo

But how about IE?

TL;DR: IE11 will be supported

Thank you!