The Tale of Vue's End

Function-based Component API #42

Open **yyx990803** wants to merge 32 commits into `master` from `function-apis`

Conversation 715 | Commits 32 | Checks 0 | Files changed 1 | +1,190 −0

yyx990803 commented on Jun 7 • edited — Member

A proposal that consolidates upon #22 (Advanced Reactivity API), #23 (Dynamic Lifecycle Injection) and the discontinuation of #17 (Class API).

Full Rendered Proposal

High-level Q&A

Is this like Python 3 / Angular 2 / Do I have to rewrite all my code?

No. The new API is 100% compatible with current syntax and purely additive. All new additions are contained within the new `setup()` function. **Nothing is being removed or deprecated (in this RFC)**, nor do we have plan to remove / deprecate anything in the foreseeable future. **(A previous draft of this RFC indicated that there is the possibility of deprecating a number of 2.x options in a future major release, which has been redacted based on user feedback.)**

Details

Reviewers

- carlmjohnson
- mmase
- josec89
- visualfanatic
- backbone87
- ycmjason
- LinusBorg
- loilo
- Akryum
- leopiccionia
- Nandiin
- fuafa
- mrjackwills
- audiolion

yyx990803 function-based api 903f429 on Jun 7

1 contributor

634 lines (468 sloc) 21.5 KB Raw Blame History

- Start Date: 2019-05-30
- Target Major Version: 2.x / 3.x
- Reference Issues:
- Implementation PR: (leave this empty)

Summary

Expose logic-related component options via function-based APIs instead.

Basic example

- Class API (dropped)

Adoption strategy

The proposed APIs are all new additions and can theoretically be introduced in a completely backwards compatible way. However, the new APIs can replace many of the existing options and makes them unnecessary in the long run. Being able to drop some of these old options will result in considerably smaller bundle size and better performance.

Therefore we are planning to provide two builds for 3.0:

- **Compatibility build**: supports both the new function-based APIs AND all the 2.x options.
- **Standard build**: supports the new function-based APIs and only a subset of 2.x options.

In the compatibility build, `setup()` can be used alongside 2.x options. Note that `setup()` will be called before `data`, `computed` and `method` options are resolved - i.e. you can access values returned from `setup()` on `this` in these options, but not the other way around.

Current 2.x users can start with the compatibility build and progressively migrate away from deprecated options, until eventually switching to the standard build.

Preserved Options

> Preserved options work the same as 2.x and are available in both the compatibility and standard builds of 3.0. Options marked with * may receive further adjustments before 3.0 official release.

- `name`

Evan You @youyuxi · Jun 20

If you were a framework designer and want to introduce a new idea into your framework...

17% Pull an Angular 2

7% Stay the same forever

76% Evolve w/ compatibility

4,447 votes • Final results

48 24 123

Don't **Hate** *the*

Hate *the*

What's in Vue3?

Nutrition Facts

Serving Size

Amount Per Serving

Calories 0

Warnings

Do not consume ■ *While reading Hacker News*

Consume in moderation

```vue
<template>
  <div>
    Hello {{ msg }}
  </div>
</template>

<script>
export default {
  name: “HelloWorld”,
  props: {
    msg: String
  },
}
</script>
```

```
<template>
  <div>
    Hello {{ msg }}
  </div>
</template>

<script>
export default {
  name: “HelloWorld”,
  props: {
    msg: String
  },
  setup(props, context) {

  }
}
</script>
```

```vue
<template>
  <div>
    {{ msg }}
  </div>
</template>

<script>
import { value } from "vue-function-api";

export default {
  name: "HelloWorld",
  props: {
    name: String
  },
  data() {
    return {
      msg: `hello ${this.name}`;
    }
  },
};
</script>
```

```
<template>
  <div>
    {{ msg }}
  </div>
</template>

<script>
import { value } from "vue-function-api";

export default {
  name: "HelloWorld",
  props: {
    name: String
  },
  data() {
    return {
      msg: `hello ${this.name}`;
    }
  },
  setup(props) {
    …
  }
};
</script>
```

```
<template>
  <div>
    {{ msg }}
  </div>
</template>

<script>
export default {
  name: "HelloWorld",
  props: {
    name: String
  },
  setup(props) {
    return {
      msg: `hello ${props.name}!`
    }
  }
}
</script>
```

like data

```vue
<template>
  <div>
    {{ msg }}
  </div>
</template>

<script>
export default {
  name: "HelloWorld",
  props: {
    name: String
  },
  setup(props) {
    return {
      msg: `hello ${props.name}!`
    }
  }
}
</script>
```

only available in render context

Hello W

w

```
<template>
  <div>
    <h2>Hello {{ text }}</h2>
    <div>
      <input v-model="text">
    </div>
  </div>
</template>
```

```
<template>
  <div>
    <h2>Hello {{ text }}</h2>
    <div>
      <input v-model="text">
    </div>
  </div>
</template>

<script>
import { value } from "vue-function-api";

export default {
  name: "HelloWorld",
  setup(props) {
    const text = value(null)
    return {
      text
    }
  }
}
</script>
```

count is 0

plus one is 1

Increment

```vue
<template>
  <div>
    <span>count is {{ count }}</span>
    <br>
    <span>plus one is {{ plusOne }}</span>
    <div>
      <button @click="increment">Increment</button>
    </div>
  </div>
</template>

<script>
import { value, computed, watch } from "vue-function-api";
export default {
  name: "Counter",
  setup() {
    // reactive state
    const count = value(0);
    // computed state
    const plusOne = computed(() => count.value + 1);
    // method
    const increment = () => {
      count.value++;
    };
    // watch
    watch(
      () => count.value * 2,
```

data

computed

method

```
<script>
import { value, computed, watch } from "vue-function-api";
export default {
  name: "Counter",
  setup() {
    // reactive state
    const count = value(0);
    // computed state
    const plusOne = computed(() => count.value + 1);
    // method
    const increment = () => {
      count.value++;
    };
    // watch
    watch(
      () => count.value * 2,
      val => {
        console.log(`count * 2 is ${val}`);
      }
    );
    // expose bindings on render context
    return {
      count,
      plusOne,
      increment
    };
  }
}
```

```
<script>
import { value, computed, watch } from "vue-function-api";
export default {
  name: "Counter",
  setup() {
    // reactive state
    const count = value(0);
    // computed state
    const plusOne = computed(() => count.value + 1);
    // method
    const increment = () => {
      count.value++;
    };
    // watch
    watch(
      () => count.value * 2,
      val => {
        console.log(`count * 2 is ${val}`);
      }
    );
    // expose bindings on render context
    return {
      count,
      plusOne,
      increment
    };
  }
}
```

Compositional

Functions

Vue ~~Hooks~~

```
function useMouse() {
  const x = value(0)
  const y = value(0)
  const update = e => {
    x.value = e.pageX
    y.value = e.pageY
  }
  onMounted(() => {
    window.addEventListener('mousemove', update)
  })
  onUnmounted(() => {
    window.removeEventListener('mousemove', update)
  })
  return { x, y }
}

// in consuming component
const Component = {
  setup() {
    const { x, y } = useMouse()
    const { z } = useOtherLogic()
    return { x, y, z }
  },
  template: `<div>{{ x }} {{ y }} {{ z }}</div>`
}
```

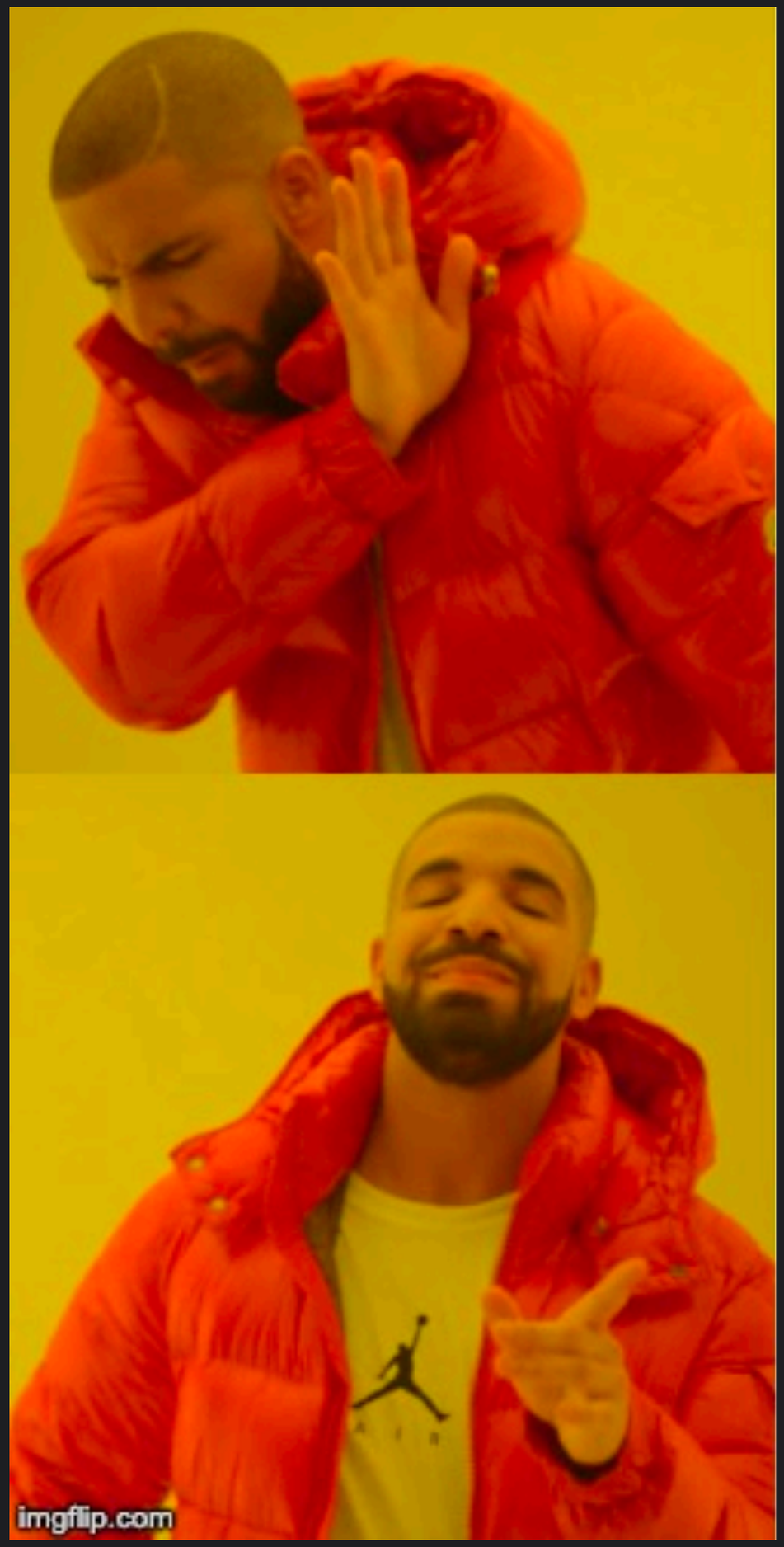
```
import { value, computed } from "vue-function-api";

export default function useCounter() {
  const count = value(0);
  const increment = () => {
    count.value = count.value + 1;
  };
  const plusOne = computed(() => count.value + 1);
  return {
    count,
    increment,
    plusOne
  };
}
```

```
<template>
  <div>
    <span>count is {{ count }}</span>
    <span>{{ plusOne }}</span>
    <div>
      <button @click="increment">Increment</button>
    </div>
  </div>
</template>
<script>
import useCounter from "../functions/useCounter";

export default {
  name: "Counter",
  setup(props, context) {
    const { count, increment, plusOne } = useCounter();

    return {
      count,
      increment,
      plusOne
    };
  }
};
</script>
```



Theoretical API

Actual API

vuejs / **vue-function-api**

Used by 37 | Watch 15 | Unstar 429 | Fork 27

Code | Issues 12 | Pull requests 1 | Projects 0 | Wiki | Security | Insights

Vue2 plugin for the function-based RFC.

67 commits | **2** branches | **1** release | **6** contributors

Branch: **master** | New pull request | Create new file | Upload files | Find File | Clone or download

liximomo feat: auto wrapping for new properties — Latest commit `69f4c8d` 4 days ago

src	feat: auto wrapping for new properties	4 days ago
test	feat: auto wrapping for new properties	4 days ago
.gitignore	chore: add vue to devDependencies and fix some typo	7 days ago
CHANGELOG.md	chore: update changlog	13 days ago
README.md	chore: update changlog	13 days ago
README.zh-CN.md	fix:modify the typos in the README.zh-CN.md	6 days ago
package.json	fix: change vue compatiable version to ^2.5.22	7 days ago
rollup.config.js	improve compatibility with rfc	21 days ago
tsconfig.json	improve compatibility with rfc	21 days ago
yarn.lock	fix: change vue compatiable version to ^2.5.22	7 days ago

README.md

```
npm install vue-function-api --save
yarn add vue-function-api
```

Framingham Center
Worcester Rd
Temple St
Sudbury Reservoir
Worcester Rd
Union Ave
C St
Arthur St
Mill St
Framingham
Winter St
Cordaville Rd
Oak St
Kendall Ln
Beaver St
Flanders Rd
Waverly St
Hollis St
Ashland
Main St
Prospect St
Cedar St
Pond St
Tri St
Green Ln
Eliot St
Whitneys
Maple St
Wood St
Hopkinton
Grove St
W Main St
Front St
S Mill St
Washington St
Chestnut St
Hayden Row
© Mapbox © O

```
<template>
  <div id="map" ref="map"></div>
</template>

<script>
  const mapboxgl = require('mapbox-gl');
  var turf = require('turf')
  export default {
    name: "RunnerMap",
    data() {
      return {
        route: null, map: null, raf: null, timestamp: null, loaded:
      }
    },
    created() {
      mapboxgl.accessToken = “API_KEY”
      fetch("https://gist.githubusercontent.com/shortdiv/XXXX.geojso
      .then(response => {
        return response.json();
      })
      .then(data => {
        this.route = data
      })
    },
    watch:{
      loaded() {
```

```
<template>
  <div id="map" ref="map"></div>
</template>

<script>
  const mapboxgl = require('mapbox-gl');
  var turf = require('turf')
  export default {
    name: "RunnerMap",
    data() {
      return {
        route: null, map: null, raf: null, timestamp: null, loaded:
      }
    },
    created() {
      mapboxgl.accessToken = “API_KEY”
      fetch("https://gist.githubusercontent.com/shortdiv/XXXX.geojs
      .then(response => {
        return response.json();
      })
      .then(data => {
        this.route = data
      })
    },
    watch:{
      loaded() {
```

```
    mounted() {
      var map = new mapboxgl.Map({
        container: this.$refs.map, // container id
        style: 'mapbox://styles/mapbox/dark-v10', //hosted style id
        center: [-71.46933442476096,42.25342191415163], // starting
        zoom: 12 // starting zoom
      });
      this.map = map
      map.on('load', () => {
        this.loaded = true
        map.addSource('point', {
          "type": "geojson",
          “data": {
            “type": “Feature",
            “properties":{},
            “geometry": {
              “type": “Point",
              “coordinates":[-71.51794373989105,42.230014854662 01]
            }
        }
      });
      map.addLayer({
        "id": "point",
        "source": "point",
        "type": "circle",
        "paint": {
          "circle-radius": 10,
```

```
mounted() {
  var map = new mapboxgl.Map({
    container: this.$refs.map, // container id
    style: 'mapbox://styles/mapbox/dark-v10', //hosted style id
    center: [-71.46933442476096,42.25342191415163], // starting
    zoom: 12 // starting zoom
  });
  this.map = map
  map.on('load', () => {
    this.loaded = true
    map.addSource('point', {
      "type": "geojson",
      “data": {
        “type": “Feature",
        “properties":{},
        “geometry": {
          “type": “Point",
          “coordinates":[-71.51794373989105,42.230014854662 01]
        }
      }
    });
    map.addLayer({
      "id": "point",
      "source": "point",
      "type": "circle",
      "paint": {
        "circle-radius": 10,
```

```
      this.map = map
      map.on('load', () => {
        this.loaded = true
        map.addSource('point', {
          "type": "geojson",
          “data": {
            “type": “Feature",
            “properties":{},
            “geometry": {
              “type": “Point",
              “coordinates":[-71.51794373989105,42.230014854662O1]
            }
          }
        });
        map.addLayer({
          "id": "point",
          "source": "point",
          "type": "circle",
          "paint": {
            "circle-radius": 10,
            "circle-color": "#007cbf"
          }
        });
      })
    }
  }
</script>
```

```
    })
  },
  watch:{
    loaded() {
      this.timestamp = performance.now()
      this.raf = requestAnimationFrame(animateMarker.bind(this))

      function animateMarker() {
        let time = performance.now() //ms
        const duration = 6 * 1000
        let speed = 5000 / duration //in m
        const path = turf.lineString(this.route.geometry.coordinat
        const timeElapsed = time - this.timestamp

        if (timeElapsed * speed >= 5000) {
          cancelAnimationFrame(this.raf)
        } else {
        var distTravelled = (timeElapsed * speed)
        var waypoint = turf.along(path, distTravelled, 'meters')
        this.map.getSource('point').setData(waypoint)
        this.raf = requestAnimationFrame(animateMarker.bind(this))
        }
      }
    }
  },
  mounted() {
    var map = new mapboxgl.Map({
```

```
<template>
  <div class="map--wrapper">
    <div id="map" ref="map"></div>
  </div>
</template>

<script>
import { watch, value, onMounted } from "vue-function-api";
const mapboxgl = require('mapbox-gl');

import useWaypoint from "../functions/useWaypoint";

export default {
  props: {
    data: Object
  },
  setup(props, context) {
    let { movePoint } = useWaypoint(props.data);
    const mapInstance = value(null)
    const lineString = value(null)

    onCreated(() => {
      mapboxgl.accessToken = "API_KEY_HERE"
      fetch("https://gist.githubusercontent.com/shortdiv/97d74.geojs
      .then(response => {
        return response.json();
      })
```

```
<template>
  <div class="map--wrapper">
    <div id="map" ref="map"></div>
  </div>
</template>

<script>
import { watch, value, onMounted } from "vue-function-api";
const mapboxgl = require('mapbox-gl');

import useWaypoint from "../functions/useWaypoint";

export default {
  props: {
    data: Object
  },
  setup(props, context) {
    let { movePoint } = useWaypoint(props.data);
    const mapInstance = value(null)
    const lineString = value(null)

    onCreated(() => {
      mapboxgl.accessToken = "API_KEY_HERE"
      fetch("https://gist.githubusercontent.com/shortdiv/97d74.geojs
      .then(response => {
        return response.json();
      })
```

```
export default {
  props: {
    data: Object
  },
  setup(props, context) {
    let { movePoint } = useWaypoint(props.data);
    const mapInstance = value(null)
    const lineString = value(null)

    onCreated(() => {
      mapboxgl.accessToken = "API_KEY_HERE"
      fetch("https://gist.githubusercontent.com/shortdiv/97d74.geojs
      .then(response => {
        return response.json();
      })
      .then(data => {
        lineString.value = data
      })
    })
    onMounted(() => {
      var map = new mapboxgl.Map({
        container: context.refs.map,
        style: 'mapbox://styles/mapbox/dark-v10',
        center: [-71.469334424760 96,42.25342191415163],
        zoom: 12
      });
      map.on('load', () => {
```

```
})
onMounted(() => {
  var map = new mapboxgl.Map({
    container: context.refs.map,
    style: 'mapbox://styles/mapbox/dark-v10',
    center: [-71.469334424760 96,42.25342191415163],
    zoom: 12
  });
  map.on('load', () => {
  mapInstance.value = map
  map.addSource('point', {
    "type": "geojson",
    "data": {
      “type":"Feature",
      “properties":{},
      “geometry":{
        “type":"Point",
        “coordinates":[-71.51794373989105,42.230014854662 01]
      }
    }
  });
  map.addLayer({
    "id": "point",
    "source": "point",
    "type": "circle",
    "paint": {
     "circle-radius": 10,
     "circle-color": "#007cbf"
```

```
        "type": "Feature",
        "properties":{},
        "geometry":{
          "type":"Point",
          "coordinates":[-71.51794373989105,42.230014854662O1]
        }
      }
    });
    map.addLayer({
      "id": "point",
      "source": "point",
      "type": "circle",
      "paint": {
       "circle-radius": 10,
       "circle-color": "#007cbf"
      }
    });
  });
  watch(() => lineString.value, val => {
    moveMap(mapInstance.value)
  })
 }
}
</script>
```

```
import { value, state } from "vue-function-api";
var turf = require("turf");

export default function useWaypoint(route) {
  const movePoint = map => {
    const waypointVal = value({
      type: "Feature",
      geometry: {
        type: "Point",
        coordinates: route.geometry.coordinates[0]
      }
    });
    const waypointState = state({
      timestamp: performance.now(),
      raf: null
    });
    const animateMarker = () => {
      let time = performance.now(); //ms
      const duration = 6 * 1000; // 6000ms or 6s
      let speed = 5000 / duration; //in m
      const path = turf.lineString(route.geometry.coordinates);
      const timeElapsed = time - waypointState.timestamp;
      if (timeElapsed * speed >= 5000) {
        cancelAnimationFrame(waypointState.raf);
      } else {
        var distTravelled = timeElapsed * speed;
```

```
import { value, state } from "vue-function-api";
var turf = require("turf");

export default function useWaypoint(route) {
  const movePoint = map => {
    const waypointVal = value({
      type: "Feature",
      geometry: {
        type: "Point",
        coordinates: route.geometry.coordinates[0]
      }
    });
  const timeStamp = value(performance.now());
  const raf = value(null);
  const animateMarker = () => {
    let time = performance.now(); //ms
    const duration = 6 * 1000; // 6000ms or 6s
    let speed = 5000 / duration; //in m
    const path = turf.lineString(route.geometry.coordinates);
    const timeElapsed = time - timestamp.value;
    if (timeElapsed * speed >= 5000) {
      cancelAnimationFrame(waypointState.raf);
    } else {
      var distTravelled = timeElapsed * speed;
      waypointVal.value = turf.along(path, distTravelled, "meters"
```

```
  const timeStamp = value(performance.now());
  const raf = value(null);
  const animateMarker = () => {
    let time = performance.now(); //ms
    const duration = 6 * 1000; // 6000ms or 6s
    let speed = 5000 / duration; //in m
    const path = turf.lineString(route.geometry.coordinates);
    const timeElapsed = time - timestamp.value;
    if (timeElapsed * speed >= 5000) {
      cancelAnimationFrame(waypointState.raf);
    } else {
      var distTravelled = timeElapsed * speed;
      waypointVal.value = turf.along(path, distTravelled, "meters"
      map.getSource('point').setData(waypointVal.value)
      raf.value = requestAnimationFrame(
        animateMarker.bind(this)
      );
    }
  };
  waypointState.raf = requestAnimationFrame(
    animateMarker.bind(this)
  );
 }
 return { movePoint }
}
```

```
<template>
  <div class="map--wrapper">
    <div id="map" ref="map"></div>
  </div>
</template>

<script>
import { watch, value, onMounted } from "vue-function-api";
const mapboxgl = require('mapbox-gl');

import useWaypoint from "../functions/useWaypoint";

export default {
  props: {
    data: Object
  },
  setup(props, context) {
    let { movePoint } = useWaypoint(props.data);
    const mapInstance = value(null)
    const lineString = value(null)

    onCreated(() => {
      mapboxgl.accessToken = "API_KEY_HERE"
      fetch("https://gist.githubusercontent.com/shortdiv/97d74.geojs
      .then(response => {
        return response.json();
      })
```

```
<template>
  <div class="map--wrapper">
    <div id="map" ref="map"></div>
  </div>
</template>

<script>
import { watch, value, onMounted } from "vue-function-api";
const mapboxgl = require('mapbox-gl');

import useWaypoint from "../functions/useWaypoint";

export default {
  props: {
    data: Object
  },
  setup(props, context) {
    let { movePoint } = useWaypoint(props.data);
    const mapInstance = value(null)
    const lineString = value(null)

    onCreated(() => {
      mapboxgl.accessToken = "API_KEY_HERE"
      fetch("https://gist.githubusercontent.com/shortdiv/97d74.geojs
      .then(response => {
        return response.json();
      })
```

```
            “coordinates":[-71.51794373989105,42.23001485466201]
          }
        }
      });
      map.addLayer({
        "id": "point",
        "source": "point",
        "type": "circle",
        "paint": {
         "circle-radius": 10,
         "circle-color": "#007cbf"
        }
      });
    });
    watch(() => lineString.value, val => {
      moveMap(mapInstance.value)
    })
  }
}
</script>
```

Will stuff **change** in Vue3**?**

Yes, Most Certainly


Vue 3 is coming for you