

# Creating Vue.js Templates

---



**Chad Campbell**

INDEPENDENT SOFTWARE CONSULTANT

@chadcampbell <https://www.ecofic.com>



# Templates

**Created with HTML**

**HTML specification compliant**



# Overview



**Defining template data**

**Binding content to a template**

**Binding to HTML attributes**

**Using JavaScript expressions**



# Defining Template Data

---



```
<script type="text/javascript">  
var growler = new Vue({  
  el: '#growler'  
  
});
```

```
<html>  
  <head>  
    <title>Growler</title>  
  </head>  
  <body>  
    <div id="growler"></div>  
    <script  
      type="text/javascript"  
      src="https://unpkg.com/vue">  
    </script>  
  </body>  
</html>
```



# data Property

**At design time, represents the schema**

**At runtime, serves as the model**



```
<script type="text/javascript">
var growler = new Vue({
  el: '#growler',
  data: {
    appName: 'Growler'
  }
});
```

Plain Old  
JavaScript  
Object  
(POJO)

```
<html>
  <head>
    <title>Growler</title>
  </head>
  <body>
    <div id="growler"></div>
    <script
      type="text/javascript"
      src="https://unpkg.com/vue">
    </script>
  </body>
</html>
```



`growler.data.appName`

`growler.appName`

---

Access data Property





# Loading Data Properties

---



# Loading a Property



Making the property “reactive”



## Data Property Caveat #1

You can only modify properties

You can't add properties at runtime

You can't remove properties at runtime



The data object serves as the schema

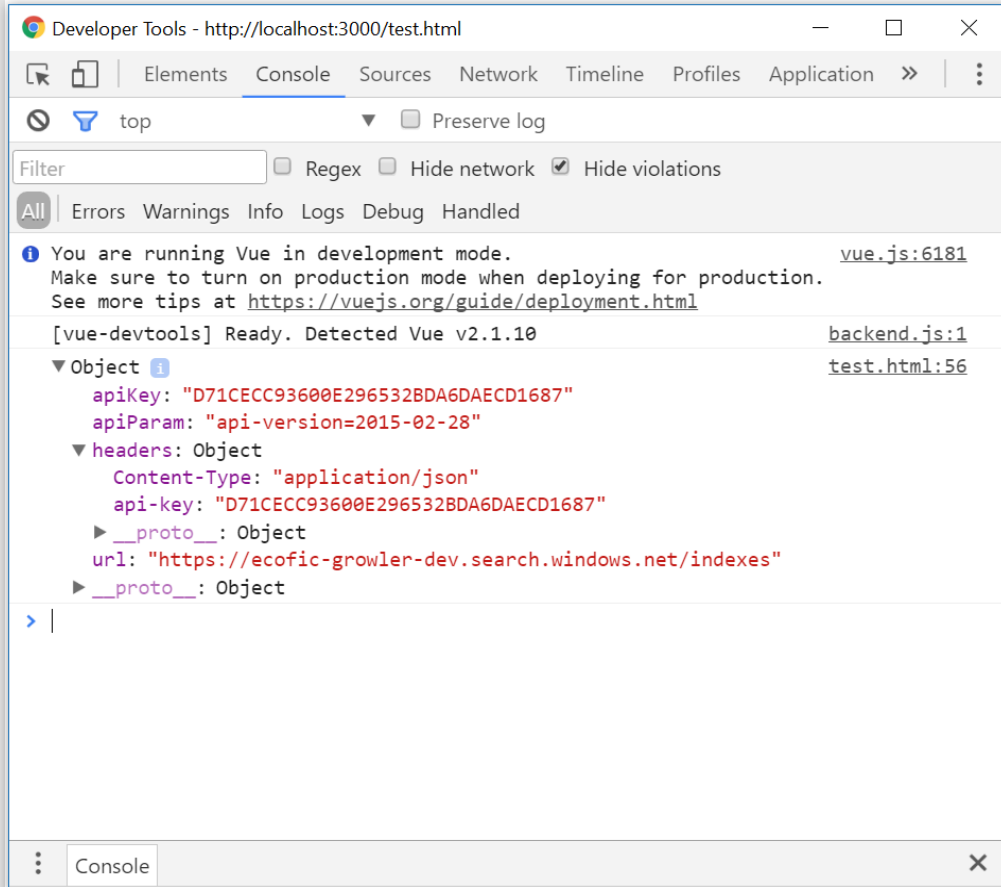


## Data Property Caveat #2

**JavaScript objects are formatted differently**

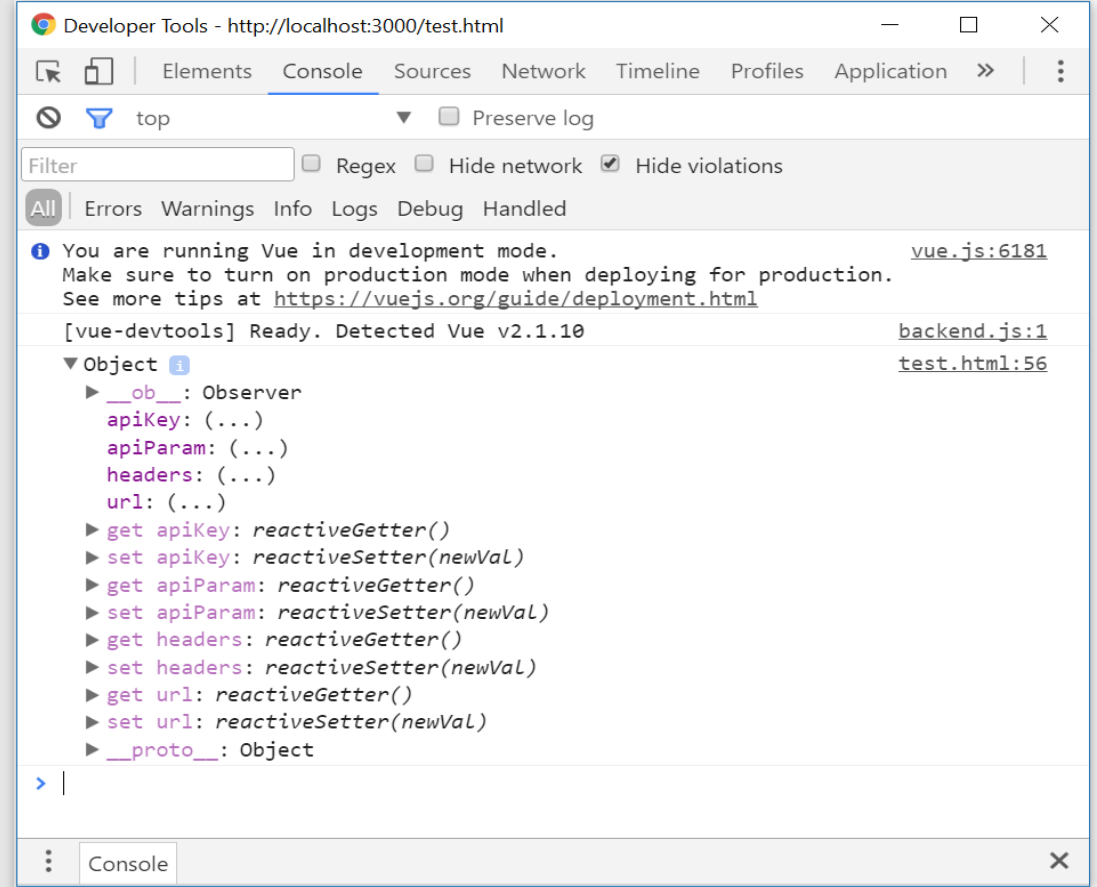


# console.log(this.searchService)



```
Developer Tools - http://localhost:3000/test.html
Elements Console Sources Network Timeline Profiles Application >>
top
Filter
[All] Errors Warnings Info Logs Debug Handled
You are running Vue in development mode. vue.js:6181
Make sure to turn on production mode when deploying for production.
See more tips at https://vuejs.org/guide/deployment.html
[vue-devtools] Ready. Detected Vue v2.1.10 backend.js:1
test.html:56
Object
  apiKey: "D71CECC93600E296532BDA6DAECD1687"
  apiParam: "api-version=2015-02-28"
  headers: Object
    Content-Type: "application/json"
    api-key: "D71CECC93600E296532BDA6DAECD1687"
  __proto__: Object
  url: "https://ecofic-growler-dev.search.windows.net/indexes"
  __proto__: Object
> |
```

Expected

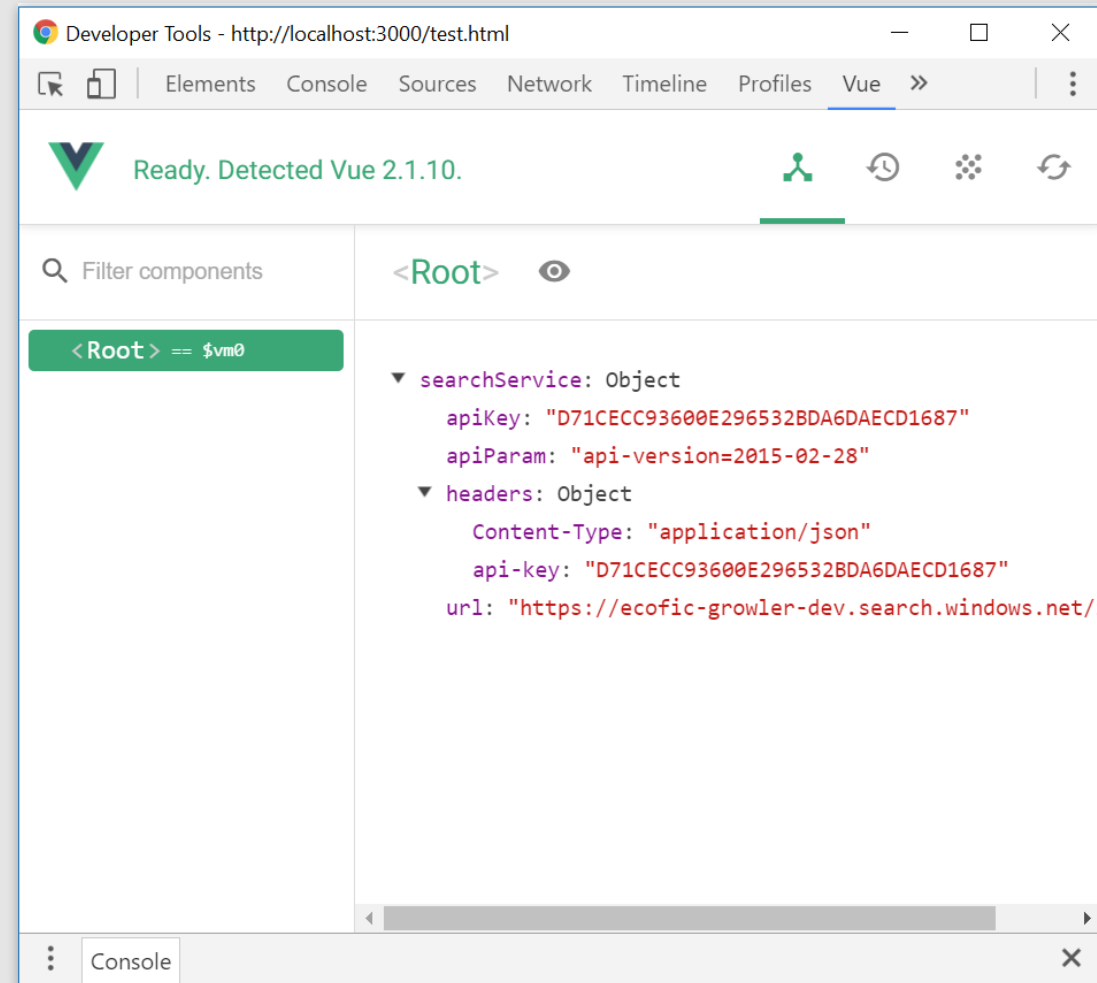


```
Developer Tools - http://localhost:3000/test.html
Elements Console Sources Network Timeline Profiles Application >>
top
Filter
[All] Errors Warnings Info Logs Debug Handled
You are running Vue in development mode. vue.js:6181
Make sure to turn on production mode when deploying for production.
See more tips at https://vuejs.org/guide/deployment.html
[vue-devtools] Ready. Detected Vue v2.1.10 backend.js:1
test.html:56
Object
  __ob__: Observer
  apiKey: (...)
  apiParam: (...)
  headers: (...)
  url: (...)
  get apiKey: reactiveGetter()
  set apiKey: reactiveSetter(newVal)
  get apiParam: reactiveGetter()
  set apiParam: reactiveSetter(newVal)
  get headers: reactiveGetter()
  set headers: reactiveSetter(newVal)
  get url: reactiveGetter()
  set url: reactiveSetter(newVal)
  __proto__: Object
> |
```

Actual



# Install vue-devtools



# Naming Properties

---



# Naming Rules

Letters, digits, dollar signs, and  
underscores

Start with a letter

Case-sensitive

Reserved words cannot be used



Should not start with  
dollar signs or underscores





# Understanding Property Values

---



# data Property Values

## Primitive values

123, "abc", "2017-02-14", [1, 2, 3]

## Not native objects

Number, String, Date, Array



Stick with raw data



# Binding Content to a Template

---



# Binding Text

**Semantic syntax**

**Declarative syntax**

**One-time bindings**



Uses double curly  
braces `{{ ... }}`  
Often referred to as  
mustaches

## Semantic Bindings



# Creating Semantic Bindings

```
<h2>Welcome to {{ appName }}</h2>
```



# Creating Semantic Bindings

```
<h2>{{ appName }}</h2>
```



# Creating Semantic Bindings

```
<h2>{{ appName }} - {{ appVersion }}</h2>
```





```
<script type="text/javascript">
  var growler = new Vue({
    el: '#growler',
    data: {
      appName: 'Growler'
    }
  });
</script>
```

```
<h2>{{ appName }}</h2>
```



Interpolation



# Declarative Bindings

Created via directives

All baked-in directives begin with “v-”

Instead of saying “v-text”, I’ll say “text directive”



```
<h2 v-text="appName"></h2>
```

---

## v-text Directive

**Interpolates a property value as an HTML element's text.**



**If you need to bind to only part of an element, use the semantic syntax.**

# One Time Bindings

**v-once renders  
element once**

**Children only get  
rendered once**

**Helps optimize  
view performance**



# Binding to HTML

---



```
<script type="text/javascript">
var growler = new Vue({
  el: '#growler',
  data: {
    appName: '<a
href="/">Growler</a>'
  }
});
```

```
<h2>{{ appName }}</h2>
```



# Demo



v-html

Updates the innerHTML property

Can't nest bindings

Only bind to HTML you trust





# Binding HTML

```
<h2>{{ appLink }}</h2>
```

```
...
```

```
<script type="text/javascript">
```

```
var growler = new Vue({
```

```
  el: '#growler',
```

```
  data: {
```

```
    appName: 'Growler',
```

```
    appLink: '<a href="/">{{ appName }}</a>'
```

```
  }
```

```
});
```



v-html

Updates the innerHTML property

Cannot nest bindings

Only bind to HTML you trust



Do not bind to  
user-generated HTML



# Binding to HTML Attributes

---



# Introducing v-bind

**Designed to bind to HTML attributes**

**Bind data property values to HTML attributes**



```
<script type="text/javascript">
  var growler = new Vue({
    el: '#growler',
    data: {
      appName: 'Growler',
      appLogo:
        '/public/img/logo.png'
    }
  });
</script>
```

```
<div id="growler"
  
</div>
```



```
<script type="text/javascript">
  var growler = new Vue({
    el: '#growler',
    data: {
      appName: 'Growler',
      appLogo:
        '/public/img/logo.png'
    }
  });
</script>
```

```
<div id="growler"
  
</div>
```



```
<script type="text/javascript">
  var growler = new Vue({
    el: '#growler',
    data: {
      appName: 'Growler',
      appLogo:
        '/public/img/logo.png'
    }
  });
</script>
```

```
<div id="growler">
  
</div>
```



# Binding to CSS Properties

**From a JavaScript Object**

**From a JavaScript Array**





# Getting CSS Properties from an Object

**Each property name must be the name of a CSS property**

**The value can be a property name or a static value**



```
<script type="text/javascript">
  var growler = new Vue({
    el: '#growler',
    data: {
      appName: '<a
href="/">Growler</a>',
      color: '#FF6A00'
    }
  });
</script>
```

```
<div id="growler">
  <h2
    v-bind:style="{ color:color }">
    {{ appName }}
  </h2>
</div>
```



```
<h2 v-text="appName" v-bind:style="{  
  color:appNameColor,  
  fontFamily:appNameFontFamily,  
  margin:0  
}"></h2>
```

---

## font-family Binding

**Binding to CSS property names with dashes**



```
<h2 v-text="appName" v-bind:style="{  
  color:appNameColor,  
  fontFamily:appNameFontFamily,  
  margin:0  
}"></h2>
```

---

## Mixing It Up

**Mixing dynamic and static property values**



```
var growler = new Vue({  
  el: '#growler',  
  data: {  
    appName: 'Growler',  
    appStyle: {  
      color: '#FF6A00',  
      fontFamily: 'Verdana',  
      margin: 0  
    }  
  }  
});
```

```
<div id="growler">  
  <h2  
    v-text="appName"  
    v-bind:style="appStyle">  
  </h2>  
</div>
```



# Getting CSS Properties from an Array

**Styles are often shared across multiple elements**

**For multiple style definitions**



```
var growler = new Vue({  
  el: '#growler',  
  data: {  
    appName: 'Growler',  
    accentColor: {  
      color: '#FF6A00',  
    },  
    headers: {  
      fontFamily: 'Verdana',  
      margin: 0  
    }  
  }  
});
```

```
<div id="growler">  
  <h2  
    v-text="appName"  
    v-bind:style=  
      "[accentColor, headers]">  
  </h2>  
</div>
```



# Binding to CSS Classes

**Helps you maintain separation between  
your design and data**





```
var growler = new Vue({  
  el: '#growler',  
  data: {  
    appName: 'Growler',  
    accentColor: {  
      color: '#FF6A00',  
    },  
    headers: {  
      fontFamily: 'Verdana',  
      margin: 0  
    }  
  }  
});
```

```
<div id="growler">  
  <h2  
    v-text="appName"  
    v-bind:style=  
      "[accentColor, headers]">  
  </h2>  
</div>
```



```
var growler = new Vue({  
  el: '#growler',  
  data: {  
    appName: 'Growler',  
    accentColor: 'accent-color',  
    headers: 'headers'  
  }  
});
```

```
<style>  
  .headers {  
    font-family: 'Verdana';  
    margin: 0;  
  }  
  
  .accent-color {  
    color: #FF6A00;  
  }  
</style>  
  
<h2  
  v-text="appName"  
  v-bind:class=  
    "[accentColor, headers]">  
</h2>
```



```
var growler = new Vue({  
  el: '#growler',  
  data: {  
    appName: 'Growler',  
    headerStyles: [  
      'accent-color',  
      'headers'  
    ]  
  }  
});
```

```
<style>  
  .headers {  
    font-family: 'Verdana',  
    margin: 0  
  }  
  
  .accent-color {  
    color: #FF6A00;  
  }  
</style>  
  
<h2  
  v-text="appName"  
  v-bind:class="headerStyles">  
</h2>
```



```
var growler = new Vue({  
  el: '#growler',  
  data: {  
    appName: 'Growler'  
  }  
});
```

```
<style>  
  .headers {  
    font-family: 'Verdana',  
    margin: 0  
  }  
  
  .accent-color {  
    color: #FF6A00;  
  }  
</style>  
  
<h2  
  v-text="appName"  
  v-bind:class="{  
    'headers': true  
  }">  
</h2>
```



```
var growler = new Vue({  
  el: '#growler',  
  data: {  
    appName: 'Growler',  
    isOnline: false  
  }  
});
```

```
<style>  
  .headers {  
    font-family: 'Verdana',  
    margin: 0  
  }  
  
  .accent-color {  
    color: #FF6A00;  
  }  
</style>  
  
<h2  
  v-text="appName"  
  v-bind:class="{  
    'headers': true,  
    'accent-color': isOnline  
  }">  
</h2>
```



# Review

Get CSS Classes from an Object

Get CSS Classes from an Array

Bind to specific CSS properties

Directives



# Using JavaScript Expressions

---



```
var isLocal = location.host.includes('localhost');
```

---

JavaScript Statement





```
location.host.includes('localhost')
```

---

## JavaScript Expression

**A line of code that produces a value**



Use {{ ... }}

Using JavaScript  
Expressions in a Template



# Expressions

**Evaluated within the context of a view**



```
var growler = new Vue({  
  el: '#growler',  
  data: {  
    appName: 'Growler',  
    isOnline: false  
  }  
});
```

```
<h2  
  v-text="appName"  
  v-bind:style=  
    "{ color:isOnline? '#FF6A00':'#000' }">  
</h2>
```



# The Vue Sandbox

An isolated environment



# Whitelisted Globals

## Properties

Infinity, undefined, NaN

## Functions

parseFloat, parseInt, isNaN, isFinite, decodeURI, decodeURIComponent, encodeURI, encodeURIComponent

## Objects

Math, Number, Date, Array, Object, Boolean, String, RegExp, Map, Set, JSON, Intl

## Webpack / Browerify

require



# Global Variables

That are not whitelisted can't  
be used in expressions

The popular \$ used in  
jQuery can't be used in  
Vue expressions



# Summary



**Use JavaScript Expressions**

**Bind to HTML Attributes**

**Bind Content to a Template**

**Setup the data**

