# Phần 1: Installation

## Direct <script> include

* Để đảm bảo sự ổn định qua các phiên bản cho trang web, Vue đề nghị sử dụng link kèm theo verson sử dụng như sau:

<script

src="https://cdn.jsdelivr.net/npm/vue@2.6.10/dist/vue.js">

</script>

* Đối với các ví dụ, chúng ta sẽ include trực tiếp thư viện VueJS từ CDN để khởi tạo và chạy nhanh nhất có thể.
* Trong phạm vi quyển sách này, Vue sẽ được giới thiệu thông qua CDN và thường bao gồm những file HTML riêng lẽ (index.html) để đánh dấu và các file JS đơn lẻ (main.JS) cho tất cả VUE code. File HTML sẽ được hiển thị trên brownser. Từ thư viện VUE và file main.js được lưu trữ riêng cho từng trường hợp, ví dụ sẽ sẽ được thực thi với điều kiện máy tính của bạn phải được kết nối internet.

## Code block and Context.

* Đa số code block trong quyển sách được pull từ **runnable code example.**  Sau đây là ví dụ đầu tiên.
* **upvote/app\_5/main.js**

new Vue({

el: '#app',

data: {

submissions: Seed.submissions

},

computed: {

sortedSubmissions () {

return this.submissions.sort((a, b) => {

return b.votes - a.votes

});

}

},

components: {

'submission-component': submissionComponent

}

});

* Một số ví dụ code nhất định sẽ giống với các khối xây dựng để đến một điểm nhất định và do đó có thể không phản ánh một khối code trực tiếp từ code mẫu. Nếu bạn cảm thấy như bạn thiếu ngữ cảnh đối với một ví dụ code, hãy mở tệp code đầy đủ bằng trình soạn thảo văn bản yêu thích của bạn. Cuốn sách này được viết với mong muốn rằng bạn cũng sẽ xem xét code ví dụ cùng với bản thảo.
* Lấy ví dụ, chún ta thường import thư viện để code của chúng ta có thể chạy được. Trong những chương đầu của quyển sách, chúng ta sẽ show những lệnh *import,*  bởi vì nó không thực sự chính xác khi thư viện được lấy từ một nơi khác. Tuy nhiên, trong những chương cuối, chúng ta sẽ tập trung vào các *khái niệm* chính thay vì lặp lại code mẫu đã được biểu diễn ở đầu sách.

# Khởi tạo Vue.js web Application đầu tiên.

## Building Upvote!.

* Bước đầu tiên để tiếp cận Vue, chúng ta sẽ tiến hành build một ứng dụng đơn giản Voting (ứng dụng có tên là Upvote!).
* Upvote tập trung vào việc hiển thị list những câu trả lời người dùng được phép vote. Mỗi câu trả lời sẽ hiển thị một số thông tin về câu trả lời đó, chẳng hạn như ảnh liên quan đến câu trả lời đó, mô tả. Tất cả câu trả lời được sort theo thứ tự tăng dần dựa trên số lượng votes. Up-vote icon cho mỗi câu trả lời sẽ cho phép người dùng tăng số lượng vote hoặc giảm số vote trên layout.
* Thông qua Upvote!, chúng ta sẽ hiểu được cách Vue tiếp cận việc phát triển front-end bằng cách hiểu được những nguyên tắc cơ bản liên quan đến thư viện. ở cuối của chương này, chúng ta đang trên đường xây dựng giao diện front-end năng động nhờ vào sự đơn giản của Vue!.

## Development environment setup

### Browser:

* Chúng tôi khuyến khích sử dụng goole chrome để phát triển các web với Vue, thông qua chrome chúng ta sẽ sử dụng ***chrome developer toolkit*** thông qua các hướng dẫn trong sách.
* Với chrome, Vue cung cấp một extension vô cùng tiện lợi là **Vue.js Devtolls** để gỡ lỗi các ứng dụng Vue. Extension sẽ được sử dụng trong suốt cuốn sách, vì vạy chúng tôi khuyên bạn cài đặt extension.

## Javascript ES6/ES7

* Javascript là ngôn ngữ web. Nó có thể run khác nhau trên các trình duyệt khác nhau. Mỗi trình duyệt có cách biên dịch JavaScript code khác nhau.
* Việc áp dụng rộng rãi như ngôn ngữ kịch bản phía máy khách Internet đã dẫn đến việc hình thành một cơ quan tiêu chuẩn quản lý đặc tả của nó. Đặc điểm kỹ thuật được gọi là ECMAScript hoặc ES.
* Hiện tại đã phát triển đến ES-7. Trong sách sẽ sử dụng ES6/ES7. Đối với ES7, ES6 cũng hỗ trợ gần như hầu hết các chức năng.

## Gettting Startted.

### Tổng quan về ứng dụng

* Trong phần này sẽ cung cấp cái nhìn tổng quan về ứng dụng Upvote!.
* Dưới đây là cách tổ chức file và các thư mục cần lưu tâm.

upvote

app/

app\_1/

app\_2/

app\_3/

app\_4/

app\_5/

app\_complete/

public/

* Chúng ta sẽ include mỗi verson của app khi chúng ta xây dựng nó trong suốt chương này. Mỗi code block trong chapter này được chứa trong một thư mục riêng. Chúng ta có thể sao chép và dán code dài hơn từ các phiên bản ứng dụng này vào thư mục / ứng dụng cục bộ, khởi đầu của ứng dụng.
* Thư mục *public/ sub*  sẽ chứa toàn bộ hình ảnh và custom styles mà chúng ta sẽ sử dụng trong ứng dụng của chúng ta.
* *App\_complete* sẽ đại diện cho trạng thái hoàn thành của ứng dụng. Mở thư mục *app\_complete,* bạn sẽ thấy có 3 tập tin nằm bên trong

*App\_complete*

*Index.html*

*Main.js*

*Seed.js*

* chúng ta có thể chạy ứng dụng bằng cách run file *index.html* (chạy bằng google chrome sẽ tốt nhất.)
* Lưu ý, để các bài nộp được sắp xếp từ số phiếu cao nhất đến thấp nhất. Ứng dụng sẽ giữ các bài đăng được sắp xếp theo số phiếu, di chuyển chúng xung quanh khi số phiếu thay đổi mà không tải lại trang.

## Chuẩn bị app

* Hãy bắt đầu xây dựng ứng dụng. chúng ta sẽ hoàn toàn thao tác trên thư mục *app/* .
* Sau đây cùng xây dựng file index.html:

<!DOCTYPE html>

<html>

<head>

<link rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/bulma/0.5.3/css/bulma.css">

<link rel="stylesheet"

href="https://use.fontawesome.com/releases/v5.0.6/css/all.css">

<link rel="stylesheet"

href="../public/styles.css" />

</head>

<body>

<div id="app">

<h2 class = "title has-text-centered dividing-header">Upvote</h2>

</div>

<script src="https://unpkg.com/vue"></script>

<script src="./seed.js"></script>

<script src="./main.js"></script>

</body>

</html>

* Trong <Head> tag có 3 file stylesheet mà chúng ta đã include cho ứng dụng. Trong 3 link trên là CDN của các Framework. Trong đó có 2 framework tương đối mới đó là **Bulma** là một CSS framework, **Font Awesome**  là các icon được hỗ trợ.
* Trung tâm của ứng dụng được gói gọn trong vài dòng ở thẻ <body>.
* Thẻ *<div>* với *id*  = “*app*”là nơi các thuộc tính của Vue được load lên và đínnh kèm với template. Nói cách khác, ứng dụng Vue của chúng ta sẽ được gắn vào phần tử cụ thể này.
* Những Script được import sau sẽ đảm nhận nhiệm vụ này.

**upvote/app/index.html**<**script** src="https://unpkg.com/vue"></**script**>  
<**script** src="./seed.js"></**script**>  
<**script** src="./main.js"></**script**>

* Tag <script> đầu tiên sẽ load verson mới nhất của Vue bằng CDN. Sử dụng CDN để tải Vue là cách đơn giản và nhanh nhất để include Vue cho một ứng dụng.

## Setting up the View

* Bây giờ chúng ta có thể hiểu được những code có sẵn, chúng ta bắt đầu bằng việc phân chia và viết một vài đoạn code ngắn. Cùng bắt đầu bằng việc setup template cho việc tạo ra từng câu trả lời chúng ta muốn. Chúng ta sẽ cùng khởi tạo ra một page như <https://bulma.io/documentation/layout/media-object/> bởi vì nó như là một ví dụ tiêu biểu.
* Trong file *index.php,* chúng ta sẽ chèn một khối mẫu dưới đây chi tiêu đề *h2*:

<div class = "section">

<article class = "media" >

<figure class = "media-left">

<img class = "image is-64x64" src = "../public/images/submissions/image-yellow.png">

</figure>

<div class = "media-content">

<div class = "content">

<p>

<strong>

<a href="#" class="has-text-info">Yellow Pail</a>

<span class="tag is-small">#4</span>

</strong>

<br>

On-demand sand castle construction expertise.

<br>

<small class = "is-size-7">

Submitted by:

<img class = "image is-24x24"

src = "../public/images/avatars/daniel.jpg">

</small>

</p>

</div>

</div>

<div class = "media-right">

<span class = "icon is-small">

<i class = "fa fa-chevron-up"></i>

<storng class = "has-text-info">10</storng>

</span>

</div>

</article>

</div>

* Template trên được dựa vào Bulma’s media object (<https://bulma.io/documentation/layout/media-object/>) có một vài điều chỉnh.
* Chúng ta thêm một thẻ <div> trong thẻ <article>. Khối *<aritcle>*  là template mẫu cho việc tạo ra một câu trả lời mà user được vote và nó có 3 DOM element con:
  + Khối ***<firgure>*** với **class = *“media-left*”**  sẽ đảm nhiệm việc hiển thị ảnh chính cho subbmition đó và được căn trái.
  + Khối ***<div>*** với ***class = “media-content***” là thẻ sẽ đảm nhiệm vai trò thêm mới t submission bao gồm các thông tin cơ bản như: tiêu đề, id, mô tả, và avarta của user thực hiện câu trả lời.
  + Khối ***<div>*** với ***class = “media-right***” sẽ hiển thị một *fa-chervon-icon (*một incon được hỗ trợ sẵn*)* cùng với số lượt bình chọn cho câu trả lời đó.
* Kết quả sẽ được hiển thị khi chúng ta run file *app/index.html.*
* Thật vi diệu, chúng ta không cần viết quá nhiều HTML cho những gì chúng ta vừa thêm vào.
* Nhưng cẩn thận, hiện tại các dữ liệu vẫn đang là static. Chúng ta chỉ vừa thêm một đoạn code cứng cho tiêu đề, mô tả và các thông tin khác. Để template này thực sự có ý nghĩa, chúng ta luôn muốn thay đổi để nó có thể sử dung dynamic data.

## Making the view data-driven (đổ dữ liệu lên view)

Việc quản lý template cùng với data cho phép chúng ta có thể render tự động hiển thị chế độ xem dựa trên dữ liệu mà chúng tôi cung cấp cho nó. Hãy cùng làm quen với ứng dụng có mô hình dữ liệu.

### Data model

Trong thư mục *app*  của chúng ta, chúng ta đã include một file có tên là *seed.js.* File này bao gồm những dữ liệu cơ bản của một submission. File *seed.js* bao gồm 1 Javascriptobject được gọi là ***Seed.submission.*** Đối tượng này là một mảng các Javascript object, trong đó mỗi đối tượng đại diện cho một submission object.

**const** submissions = [  
 {  
 id: 1,  
 title: 'Yellow Pail',  
 description: 'On-demand sand castle construction expertise.',  
 url: '#',  
 votes: 16,  
 avatar: '../public/images/avatars/daniel.jpg',  
 submissionImage: '../public/images/submissions/image-yellow.png',  
 },  
 *// ...*]

mỗi submission có một id duy nhất và một series các thông tin của nó như *tiêu đề, mô tả, votes, …*

vì chúng ta chỉ hiển thị từng submission đơn lẽ lên view, nên đầu tiên chúng ta phải tập trung vào việc lất data từ từng submission object riêng biệt (ví dụ *submission[0]*) lên template.

## The Vue instance (Cá thể Vue)

Cá thể Vue này là điểm khởi đầu cho tất cả các ứng dụng Vue khác. **Instances Vue(cá thể Vue)** này cho phép một option object được phép bao gồm chi tiết của nó chẳng hạn như template, data, phưog thức (method)… Cấp bậc đơn giản nhất của instance cho phép chúng ta tham chiếu đến DOM mà các cá thể được đính kèm theo nó.

Cùng tham khảo ví dụ của nó bằng cách một Vue instances cho ứng dụng của chúng ta. Chúng ta sẽ viết tất cả Vue code của phần còn lại trong chương này bên trong file *main.js.* Hãy cùng mở file *main.js* và khởi tạo một Vue instances sử dụng Vue function:

new Vue({

el: '#app'

});

Chúng ta vừa chỉ định cho một elemnt HTML với *id = “app”*  là điểm gắn kết của ứng dụng Vue

Của chúng ta, bằng cách sử dụng bằng cách sử dụng option ***el*** và cung cấp một chuỗi giá trị của *#app.* Bất cứ đâu trong element bây giờ có thể sử dụng Vue Javascript code này.

**Vue instances** cũng có thể trả về tất cả dữ liệu cần được xử lý ở View (trả về data ở page update chẳng hạn). Dữ liệu này phải được chỉ định như là một *data*  object của ***Instance.*** Đây là cách chúng ta sắp xếp kết nối giữa dữ liệu trong file seed.js của chúng ta và template VIew. Hãy để cập nhật instance bằng cách chỉ định một đối tượng dữ liệu mới. Trong đối tượng, chúng ta sẽ bao gồm một submission khóa sẽ có cùng giá trị với mảng Seed.submissions:

**upvote/app\_2/main.js**

new Vue({

el: '#app',

data: {

submissions: Seed.submissions

}

});

Trong HTML template, bây giờ chúng ta có thể thao khảo đến dữ liệu của tất cả submission bằng cách truy cập đến *submission*.

Với Vue instance mà chúng ta đã khởi tạo và bao gồm dữ liệu của submission bây giờ chúng ta có thể làm việc theo hướng đồng bộ hóa dữ liệu trong model với view. Nói cách khác, bây giờ dữ liệu của chúng ta có thể liên kết dữ liệu đối tượng dữ liệu với DOM.

## Data Binding

Hình thức ràng buộc dữ liệu đơn giản nhất trong Vue là sử dụng cú pháp “Mustache” (râu ria) được biểu thị bằng dấu { {} }. Chúng ta sẽ áp dụng cú pháp này để liên kết tất cả các văn bản trong HTML của chúng ta (ví dụ như tiêu đề, mô tả,..)

Tuy nhiên, cú pháp “Mustache” không thể được sử dụng trong một số thuộc tính như *href, id, src,…* Vue cung cấp thuộc tính tự nhiên như **v-bind** (được hiểu như là một chỉ định của Vue) để ràng buộc các thuộc tính của HTML. Chúng ta sẽ sử dụng chỉ định này cho việc cập nhật các thuộc tính *src* trong template của chúng ta.

Hãy chuyển đổi giữa những code cứng và dữ liệu trong template tham chiếu đến nội dung của đối tượng đầu tiên trong mảng *submission, submission[0].* Điều này sẽ làm cho khối mẫu mới được thêm hiện tại trong như bên dưới đây:

<div class = "section">

<article class = "media" >

<figure class = "media-left">

<img class = "image is-64x64"

v-bind:src="submissions[0].submissionsImage">

</figure>

<div class = "media-content">

<div class = "content">

<p>

<strong>

<a v-bind:href="submissions[0].url" class="has-text-info">

{{ submissions[0].title }}

</a>

<span class="tag is-small"># {{ submissions[0].id }}</span>

</strong>

<br>

{{ submissions[0].description }}

<br>

<!---is-size-7 là định dạnh font chữ của bulma

is-24x24 là hỗ trợ định dạng kích thức của bulma-->

<small class = "is-size-7">

Submitted by:

<img class = "image is-24x24"

v-bind:src="submissions[0].avatar">

</small>

</p>

</div>

</div>

<div class = "media-right">

<span class = "icon is-small">

<i class = "fa fa-chevron-up"></i>

<storng class = "has-text-info">{{ submissions[0].votes }}</storng>

</span>

</div>

</article>

</div>

Nếu chúng ta kết hợp mọi thứ một cách thích hợp, chúng ta sẽ không thấy sự thay đổi trong view information

## List Rendering

Chúng ta đã khởi tạo thành công một đối tượng Vue và ràng buộc cho từng submission object trong view của chúng ta. Vấn đề tiếp theo của chúng ta là render tất cả submission object cho khung View của chúng ta bằng cách hiển thị mỗi đối tượng submisssion dưới dạng một khối riêng biệt.

Vì chúng ta sẽ render một danh sách các đối tượng submission. Chúng ta sẽ sử dụng câu lệnh **v-for** của Vue.

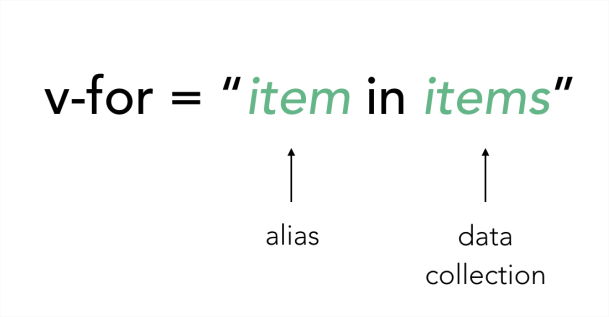
## Lệnh *v-for*

Lệnh v-for được sử dụng để hiển thị danh sách các mục dựa trên nguồn dữ liệu. Trong trường hợp của chúng ta, chúng ta muốn hiển thị một submission cho mỗi đối tượng gửi trong mảng *seed.submission*.

Thẻ <*article*> trong file *index.html* là một yếu tố HTML tiêu chuẩn, hiện đang hiển thị cho từng bài post submission

<**article** class="media">  
*<!-- Rest of the submission template -->*</**article**>

Lệnh v-for yêu cầu một cú pháp cụ thể dọc theo các dòng của mục trong các mục, trong đó các mục là một tập hợp dữ liệu và mục là bí danh cho mọi phần tử được lặp lại theo (tương tự như vòng lặp foreach trong PHP) :



Trong template của chúng ta, từ *submissions* là một tập hợp dữ liệu chúng ta sẽ lặp lại nó. Biến *Submission* lúc này được xem như là một bí danh thích hợp biểu thị từng đối tượng con mà ta muốn sử dụng. Chúng ta sẽ thêm câu lệnh ***v-for*** vào khối <*article*> như sau:

### Key:

Vue chỉ định một thuộc tính khóa cho mọi phần tử được lặp trong một v-for được truy từ xuất danh sách. Vue sử dụng thuộc tính khóa để tạo các ràng buộc duy nhất cho mỗi nút Nhận dạng của nút.

Để chỉ định tính duy nhất này cho từng mục trong danh sách, chúng ta sẽ gán một khóa(Key) cho mỗi lần thực hiện lặp. Chúng ta sẽ sử dụng id của một submisson vì một ***submission.id*** sẽ không bao giờ bằng với một ***submission.id*** khác. Bởi vì chúng ta sử dụng các giá trị động, chúng ta sẽ cần sử dụng v-bind để liên kết khóa của chúng ta với ***submission.id:***