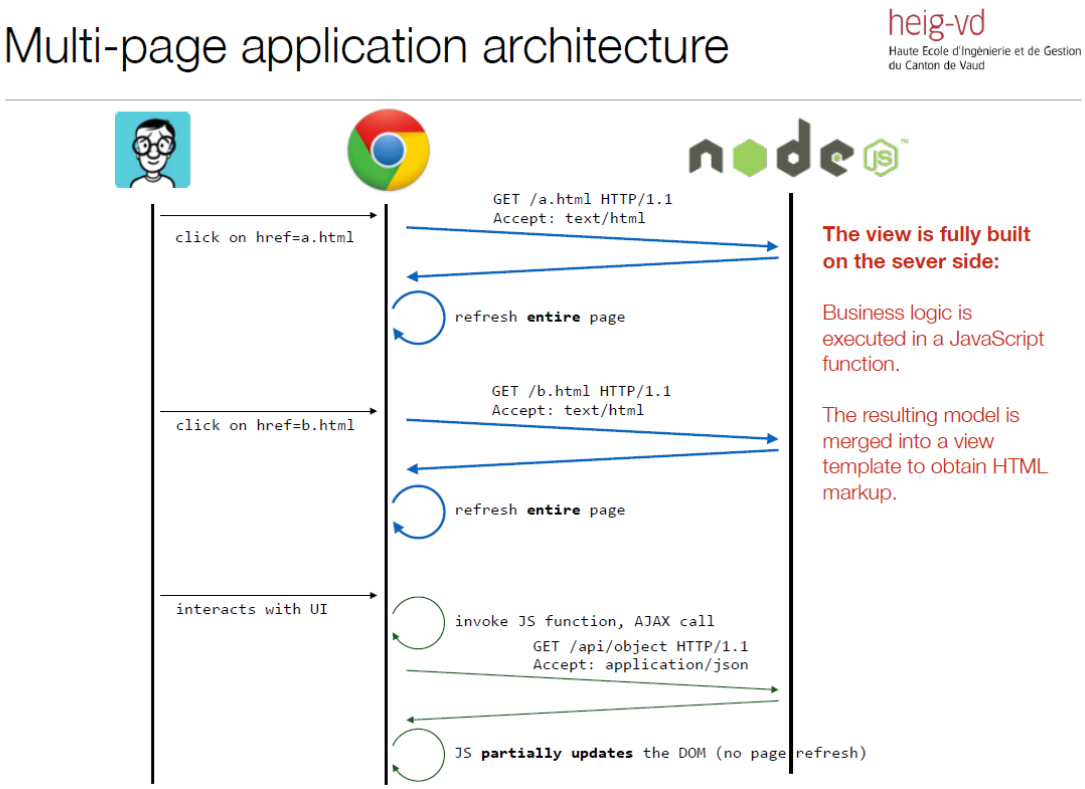
643-1-1 Projet de technologies WEB de présentation

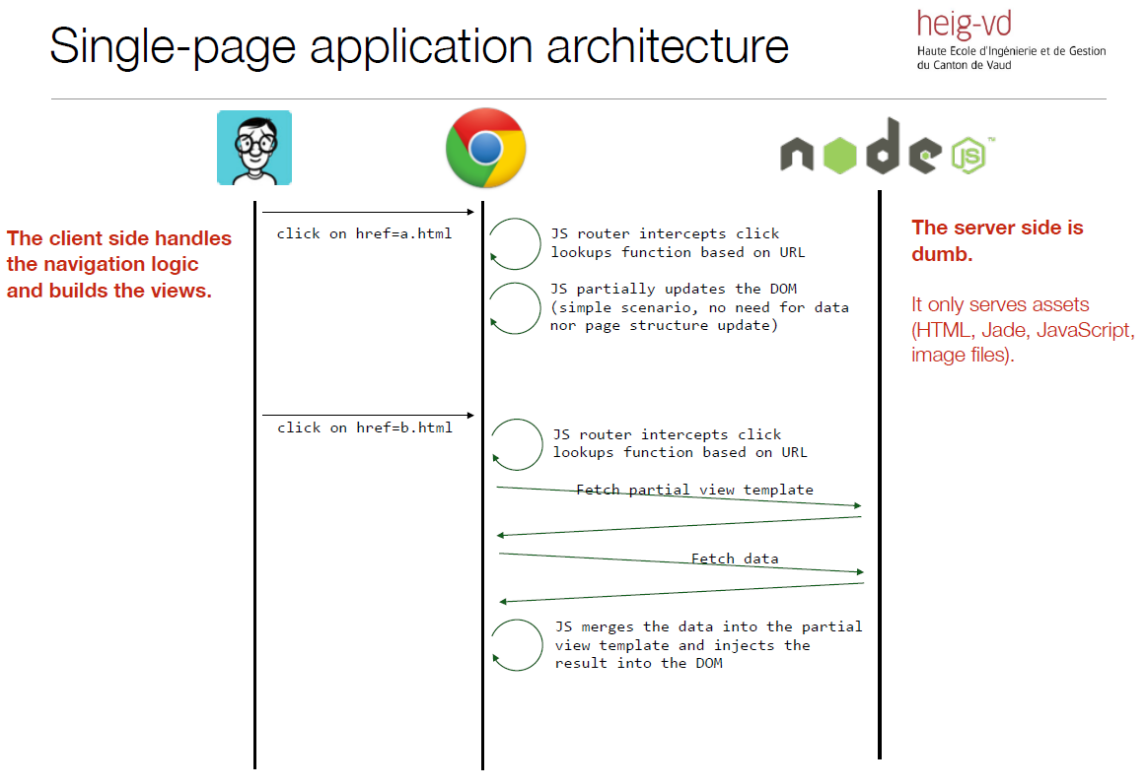
Zero to Hero

Part 2: Build and deploy a Single Page Application

Vue.js

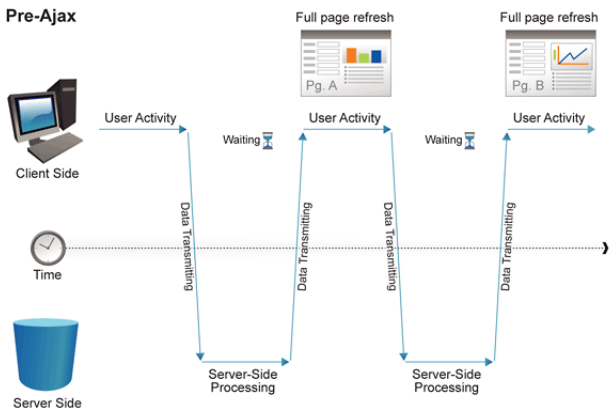
Multi Page App
vs
Single Page Application (SPA)

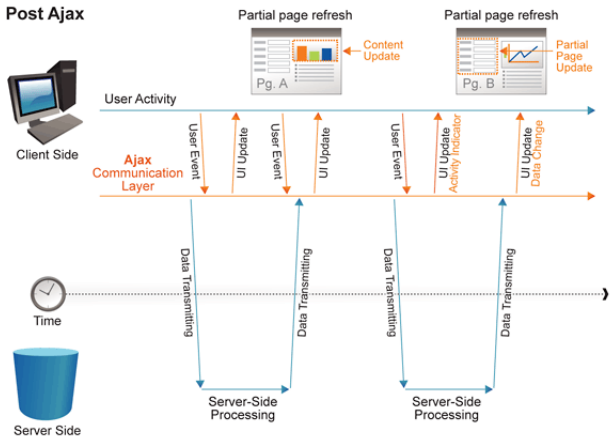




There is a big trend towards "single-page applications", where some of the responsibilities are moved from the server to the client side.

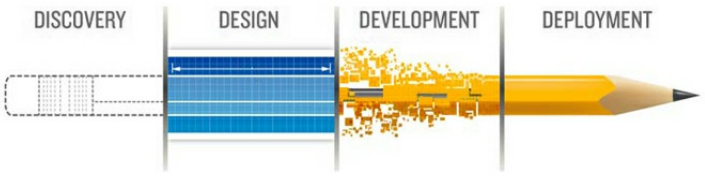
- The client initially fetches a single "shell" page, which provides a rendering context and loads application modules (scripts, markup partials, stylesheets, etc.).
- When the user clicks on hyperlinks, the browser does not (immediately) send an HTTP request to fetch a new page. Instead, the event is caught and processed by a JavaScript router on the client side.
- Routing is done on the client side. The JavaScript router (typically provided by an application framework) looks at the target URL and decides which JavaScript function needs to be invoked. This function can update the DOM, sometimes in drastic manners (giving the impression that we move from an "Customers List" page to a "Customer Details" page).





http://www.websiteoptimization.com/secrets/ajax/8-1-ajax-pattern.html

Paper to WWW



Note:

Mobile application developement can also be done wiht HTML/CSS/JavaScript, with this kind of application and then be packaged into a native app with a webview.

Who is guilty?

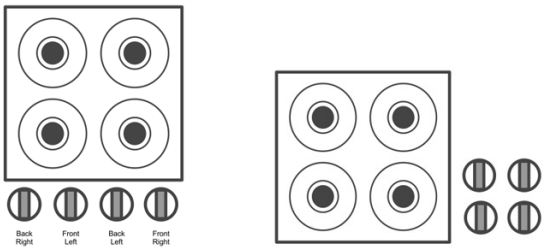
http://www.usabilitypost.com/2010/11/17/the-design-of-everyday-things/

The user of the device often assumes blame by default. They believe that because they were the ones that made the error, or they were the ones who couldn't figure how the thing works, then they're the ones to blame.

Norman argues that in most cases this isn't so. The designer is to blame because they produced something that's not easy to understand or something that lets errors and misuse happen. If we have trouble using something then it's probably because that thing is badly designed, rather than us being stupid.

Interaction Design: Natural Mapping

Controls arrangement:



Completely detached Natural Mapping

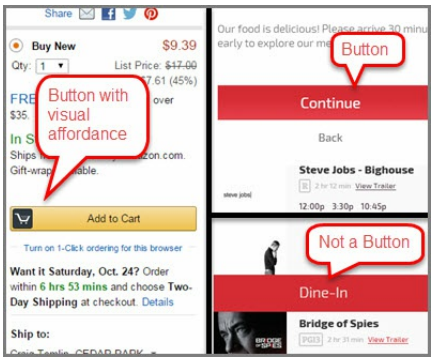
http://www.usabilitypost.com/2010/11/17/the-design-of-everyday-things/

The mappings are not great because the controls don't represent the alignment of the burners, so you always have to refer to the labels when you want to turn them on or off. We can improve this by using a natural mapping, using a spacial analogy to show the relationship between the controls and the burners they operate.

Interaction Design: Perceived Affordances

> It's not you. Bad doors are everywhere.

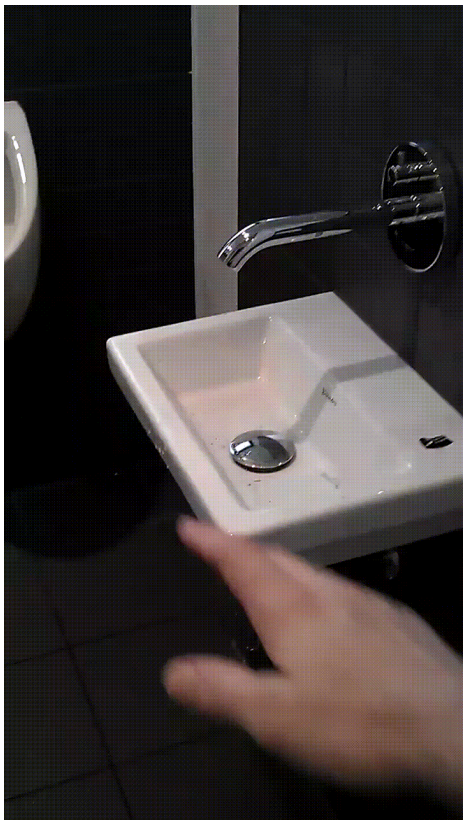
Affordances for Web Design



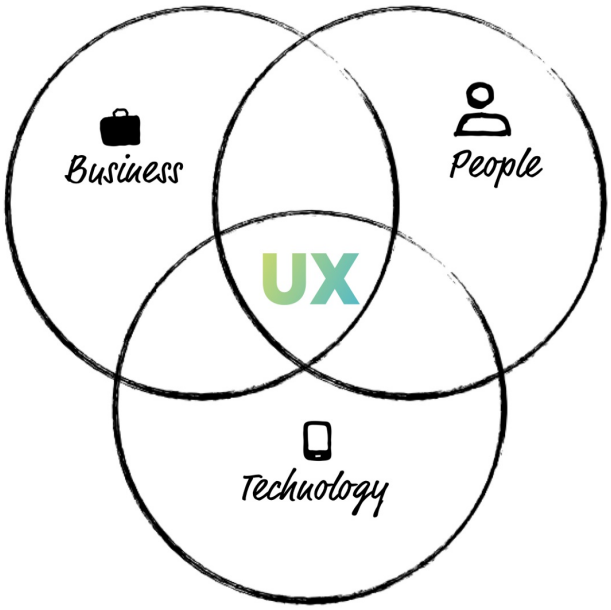
Use Visual Affordance to Ensure Buttons Look Clickable

http://www.usefulusability.com/15-user-experience-details-you-missed/

When Design Fails



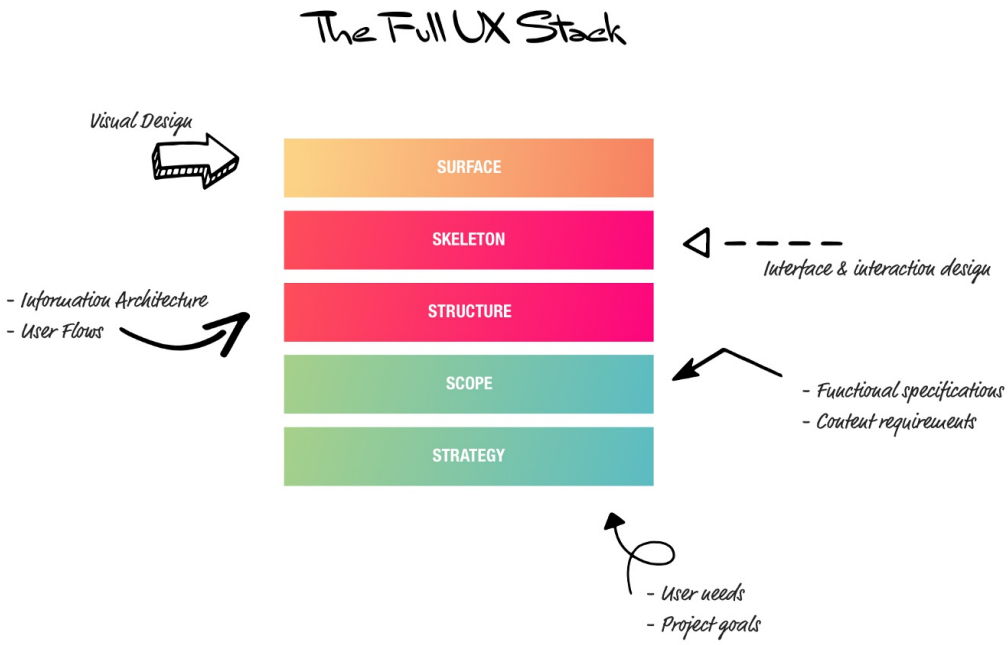
UX: User Experience Design



https://onepotprojects.com/an-introduction-user-experience-design-2a7f8167bf03

While focusing on user needs it is also important for a UX Designer to be aware of balancing business goals with technology constraints (or opportunities). While it is true that a product cannot succeed without a healthy business, a business cannot succeed without a happy customer—and it is the UX Designer's job to be the customer advocate.

UX is not UI



Initial Strategy and Scope Phases

Fundamental business questions (what to build)

- Do users need the product you are making?
- Do they want it enough that they will either pay for it or if it is free, spend time looking for it and learning to use it?
- Are you missing a key feature they will need?
- Are you spending time building features they will never use?

Structure and Skeleton Phases

Critical implementation questions (how to build?)

- How should the content be organised so that users can easily find it?
- Will users find your app easy to use? Where do they get confused or lost?
- What content is needed and how should it be written to be most engaging?

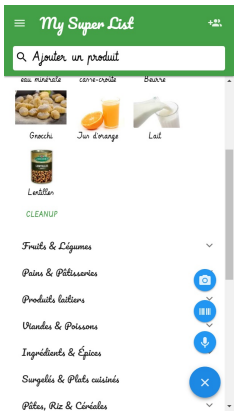
The Surface of the Product

User's first impression is critical

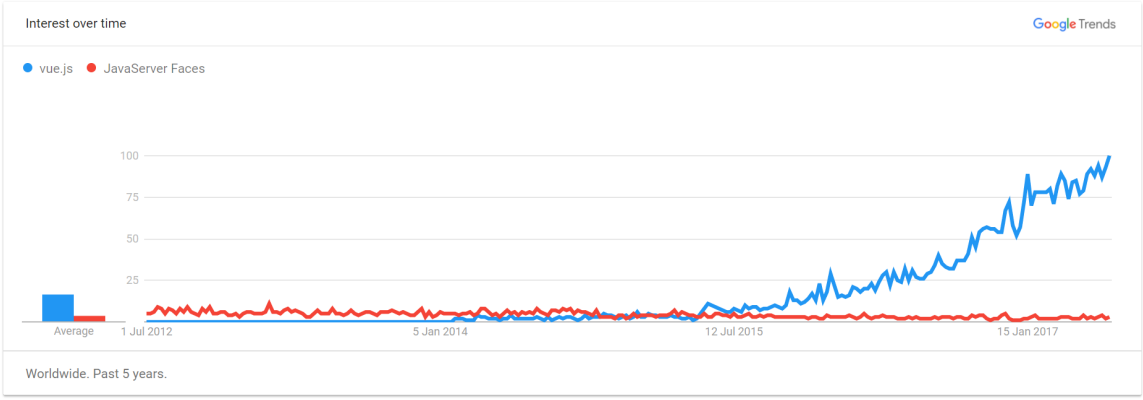
- What should the visual tone of the product be?
- How do users feel when they see your product? Do they trust it?
- It the product visually appealing and does it spark joy?
- Is the visual design usable and accessible?

Objectifs

- Learn SPA with Vue.js
- Learn by Example: Shopping List App
- Only focus on frontent-app
- With Api's and some cloud functions ("serverless")

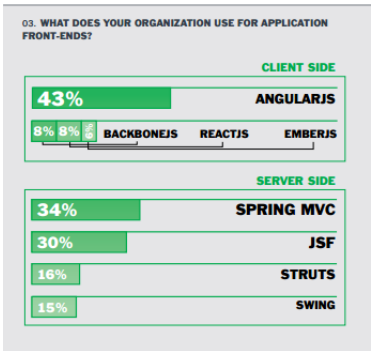


Why Vue.Js?

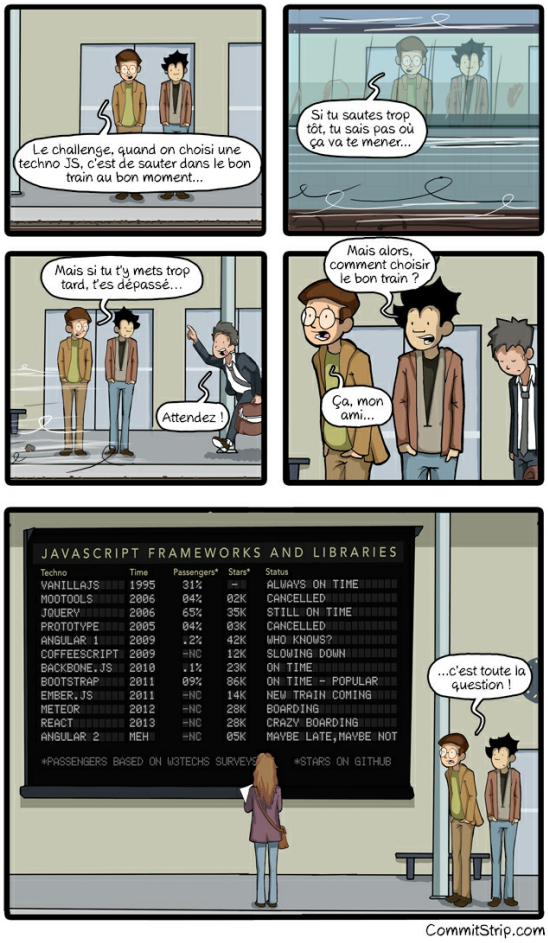


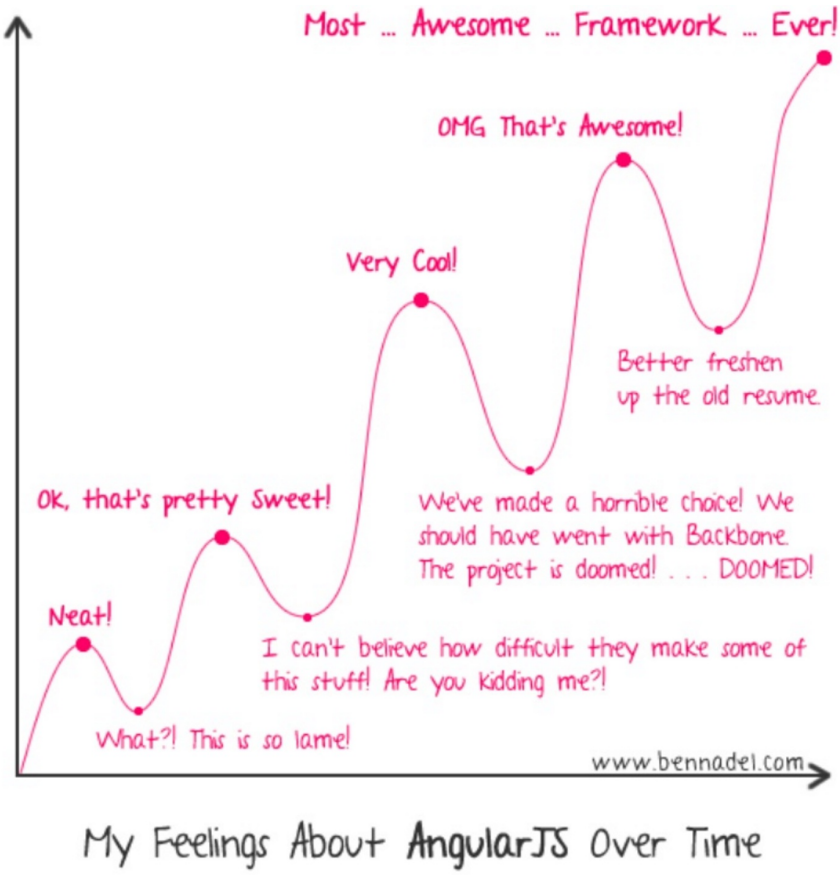
Note:

- address many of the challenges encountered in developing SPA
- large acceptation
- decouple DOM manipulation from application logic
- decouple the client side of an application from the server side
- declarative programming for user interface
- imperative programming for application business logic
- best of anagularjs and react

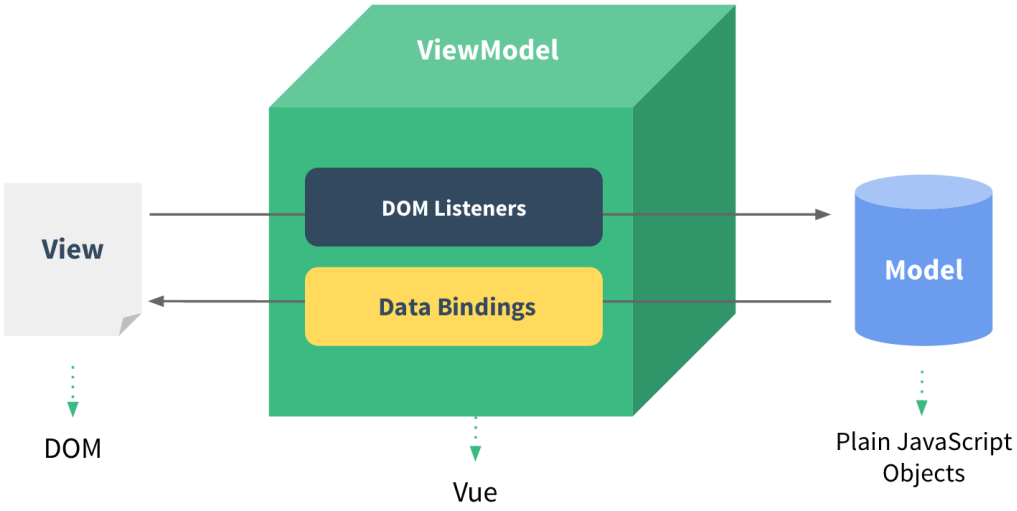


THE DZONE GUIDE TO THE JAVA ECOSYSTEM, 2015





Vue.js a Model View ViewModel (MVVM) framework



Vue constructor and text interpolation

HTML

Javascript

```
<div id="app">
  {{ message }}
</div>


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

```
new Vue({
  el: '#app',
  data() {
    return {
      message: 'Hello Vue.js!'
    };
  }
});
```

Text interpolation

Vue.js expressions are JavaScript-like code snippets that are usually placed in bindings such as `{{ expression }}`.

```
<div>{{ message.toUpperCase() }}</div>       <!-- HELLO VUE.JS! -->
<div>{{ message.slice(0,5) }}</div>          <!-- Hello          -->
<div>You say {{ ok ? 'YES' : 'NO' }}</div>   <!-- You say NO     -->
```

> Template expressions are sandboxed and only have access to a whitelist of globals such as `Math` and `Date`. You should not attempt to access user defined globals in template expressions.

https://vuejs.org/v2/guide/syntax.html#Interpolations

Filters

> used inside mustache interpolations and v-bind expressions primarily designed for text transformation

```
<div>{{ message | reverse }}</div> <!-- !sj.euV olleH -->
```

Filters can be chained

```
{{ message | filterA | filterB }}
```

Filters are JavaScript functions, therefore they can take arguments:

```
{{ message | filterA('arg2', arg3) }}
```

```
<div id="app">
  {{ message | reverse }}
</div>

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

```
new Vue({
  el: '#app',
  data() {
    return {
      message: 'Hello Vue.js!'
    };
  },
  filters: {
    reverse: function(input) {
      return input.split('').reverse().join('');
    }
  }
});
```

Some Directives

Directive	Description
v-model	Create a two-way binding on a form input element or a component.
v-bind	Dynamically bind one or more attributes, or a component prop to an expression.
v-if, v-else-if, v-else	Conditionally render the element based on the truthy-ness of the expression value.

Directive	Description
v-show	Toggle's the element's display CSS property based on the truthy-ness of the expression value.
v-for	Render the element or template block multiple times based on the source data.
v-on:click	Attaches an event listener to the element.

https://vuejs.org/v2/api/#Directives

Directive v-model data-bind input elements

.number - cast input string to numbers

.trim - trim input

```
<div id="app">
  <p><label>Nom: <input v-model.trim="name"></label></p>
  <p>Hello {{name}}</p>
  <pre>{{name}}</pre>
</div>

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

```
new Vue({
  el: '#app',
  data() {
    return {
      name: ''
    };
  },
});
```

Directive show / hide elements

Conditionally render the element based on the truthy-ness of the expression value.

```
<div v-if="type === 'A'"> A </div>
```

Toggle's the element's display CSS property based on the truthy-ness of the expression value.

```
<div v-show="type === 'B'"> B </div>
```

```
<div id="app">
  <div v-if="type === 'A'"> A </div>
  <div v-show="type === 'B'"> B </div>
</div>

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

```
new Vue({
  el: '#app',
  data() {
    return {
      type: 'A'
    };
  },
});
```

Directive conditional

```
<div v-if="type === 'A'">
  A
</div>

<div v-else-if="type === 'B'">
  B
</div>

<div v-else-if="type === 'C'">
  C
</div>

<div v-else>
  Not A/B/C
</div>
```

Directive repeat elements

```
<div v-for="item in items">
  {{ item.text }}
</div>
<div v-for="(val, index) in array"></div>
<div v-for="(val, key) in object"></div>
<div v-for="(val, key, index) in object"></div>
```

`v-bind:key="item.id"` needed on custom elements or for move ordering purpose

Directive dynamique attributes

Dynamically bind one or more attributes, or a component prop to an expression.

```
<div v-bind:attributes=""></div>

<img v-bind:src="'/base/' + n + '.jpg'">
```

A short form exists

```
<div :attributes=""></div>

<img :src="'/base/' + n + '.jpg'">
```

Directive bind multiples attributes

```
<div :attr1="" :attr2=""></div>

<div v-bind="{attr1: '', attr2: ''}"></div>
```

Directives bind dynamic css classes

```
<div v-bind:class="{ active: isActive }"></div>
```

string	:class="'classString'"
variable	:class="classNameVariable"
array	:class="[classVarA, 'classNameB']"
object	:class="{className: someBooleann}"
ternary	:class="bool ? 'active' : 'inactive'"
method	:class="classNameReturningFunction()"

https://speakerdeck.com/bhawkes/introduction-to-vue-js?slide=27

Vue Instance Methods

- Runs whenever an update occurs
- Not cached
- Getter/setter
- Typically invoked from v-on/@, but flexible

```
<div id="app">
  <button v-on:click="addOne">Click Me!</button> {{count}}
</div>

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

```
new Vue({
  el: '#app',
  data() {
    return {
      count: 0
    };
  },
  methods: {
    addOne() {
      this.add(1);
    },
    add(nb) {
      this.count += nb;
    }
  }
});
```

Methods and Events

Methods used to handle events can access event object through $event

For common event manipulation there are helpers

.stop	call event.stopPropagation().
.prevent	call event.preventDefault().
.{keyCode | keyAlias}	only trigger handler on certain keys.
.right	only trigger handler for right button mouse events.
.left	only trigger handler for left button mouse events.
.middle	only trigger handler for middle button mouse events.

```
<div id="app">
  <form v-on:submit.prevent="onSubmit"></form>

  <!-- chain modifiers -->
  <button @click.stop.prevent="showCoordinates($event)">Where did you click?</button>

  <!-- key modifier using keyAlias -->
  <input @keyup.enter="onEnter">
</div>

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

```
new Vue({
  el: '#app',
  data() {
    return {
      count: 0
    };
  },
  methods: {
    showCoordinates(evt) {
      console.log(evt.clientX, evt.clientY);
    },
    onEnter() {},
    onSubmit() {}
  }
});
```

Computed Properties

- Runs only when a dependency has changed
- Cached
- Should be used as a property, in place of data

```
<div id="app">
  <p>{{date1}} {{date2()}} {{count}}</p>
  <button @click="count+=1">render</button>
</div>

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

```
new Vue({
  el: '#app',
  data() { return { count: 0 };
  },
  computed: {
    date1() { return new Date(); }
  },
  methods: {
    date2() { return new Date(); }
  }
});
```

Computed Properties Setter

Computed properties are by default getter-only, but you can also provide a setter when you need it:

```
// ...
computed: {
  fullName: {
    // getter
    get: function () {
      return this.firstName + ' ' + this.lastName
    },
    // setter
    set: function (newValue) {
      var names = newValue.split(' ')
      this.firstName = names[0]
      this.lastName = names[names.length - 1]
    }
  }
}
// ...
```

GOTCHAS arrays

Arrays mutation are only tracked on the following methods

```
array.push()
array.pop()
array.shift()
array.unshift()
array.splice()
array.sort()
array.reverse()
```

> Setting [index] or .length will not work!

Replace entire array with

```
array.filter()
array.concat()
array.slice()
```

GOTCHAS new attributes

There are also methods on Vue and the vue instance to set new observed attributes onto an array or an object.

```
Vue.set( example1.items, indexOfItem, newValue )

vm.$set( target, key, value )
```

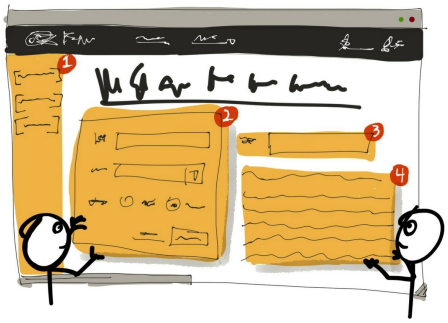
Vue Concepts Summary

Vue.js Concepts	Description
ViewModel	the data shown to the user in the view and with which the user interacts
View	what the user sees (the DOM)
Template	HTML with additional markup
Directives	extend HTML with custom attributes and elements
Components	a special kind of tag used for component-based application structure
Expressions	access variables and functions from the ViewModel
Filter	formats the value of an expression for display to the user
Computed	Computed properties are cached, and only re-computed on reactive dependency changes
Methods	Methods to be mixed into the Vue instance

Component Based Applications

Components provide a way to write small parts with a consistent API that can easily be orchestrated as part of a larger screen, application or system.

An encapsulated set of behaviors or process and logic, with a well-known interface or API to access that component's functionality.

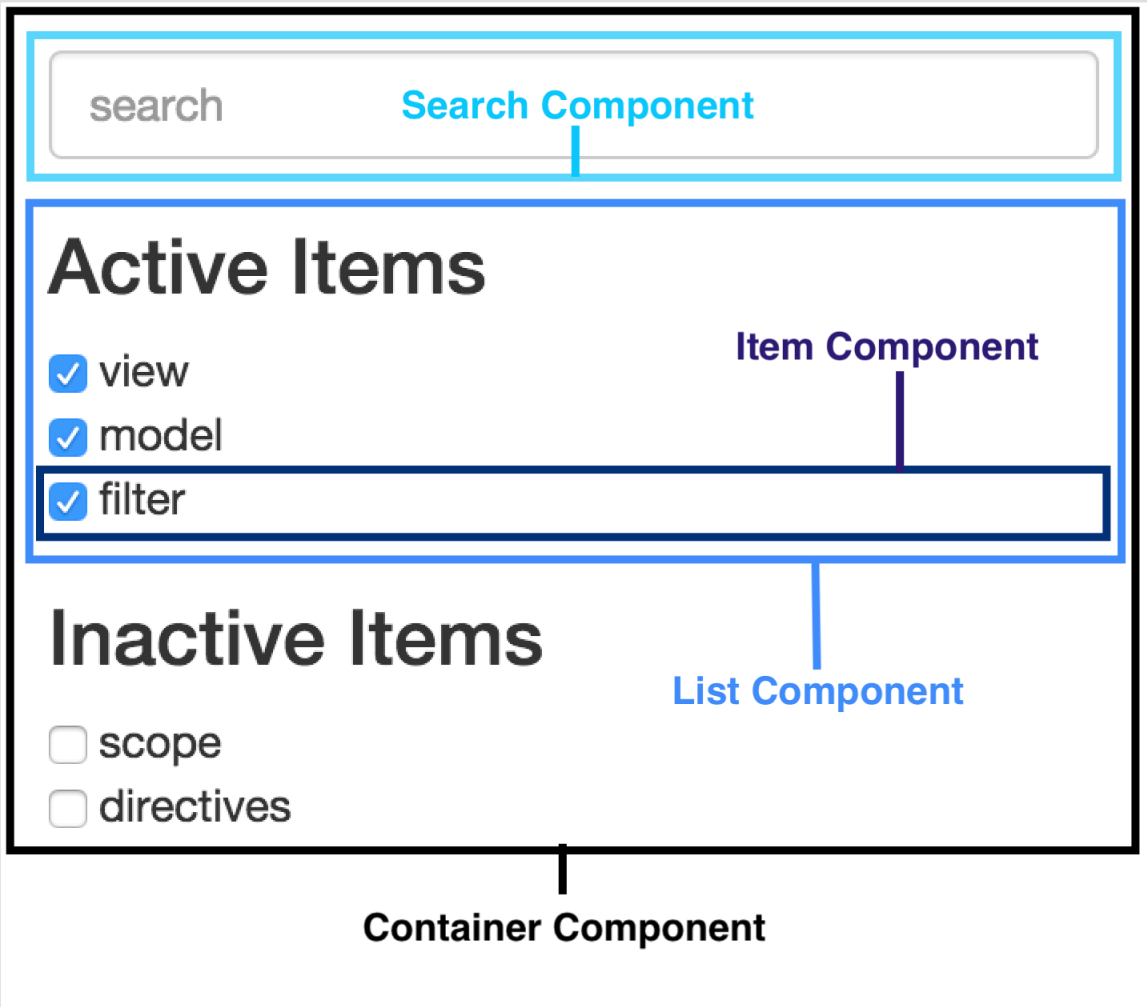


https://derickbailey.com/2015/08/26/building-a-component-based-web-ui-with-modern-javascript-frameworks/

Advantages of Components

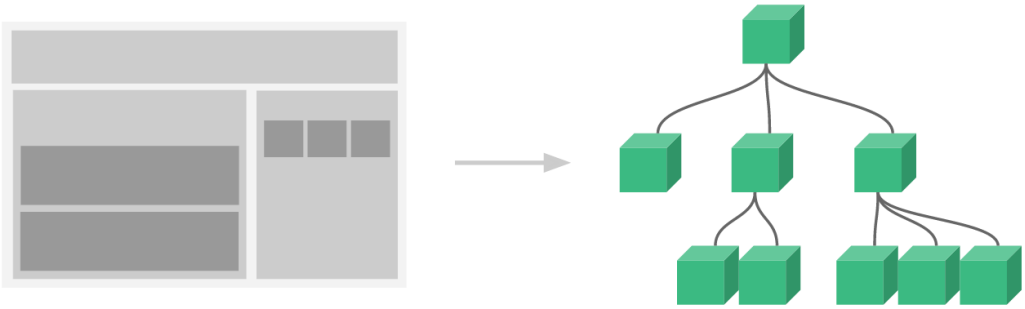
A component is a small, potentially re-usable set of logic, behaviors and interface elements

- Reusable
- Data flow boundaries in/out
- Isolated scope
- Simple or with state easier to predict
- Testable



http://busypeoples.github.io/post/thinking-in-components-angular-js/

Components in Vue.js

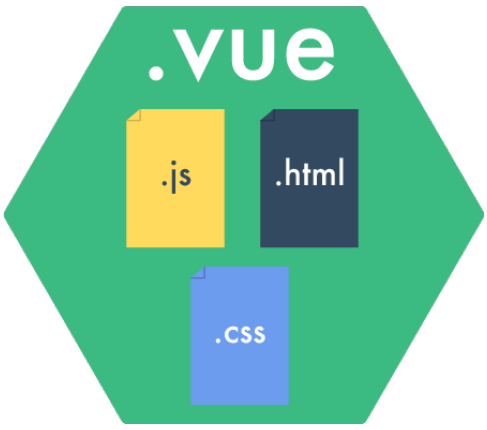


```
<div id="app">
  <my-tag></my-tag>
</div>

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

```
Vue.component('my-tag', {
  template: `<div>{{ text }}<div>`,
  data() {
    return {
      text: 'Hello From My Tag!'
    }
  }
});

new Vue({
  el: '#app'
});
```

> Note that Vue does not enforce the W3C rules for custom tag names (all-lowercase, must contain a hyphen) though following this convention is considered good practice.

Components using .vue



https://speakerdeck.com/bhawkes/introduction-to-vue-js

Global vs Local Registrations

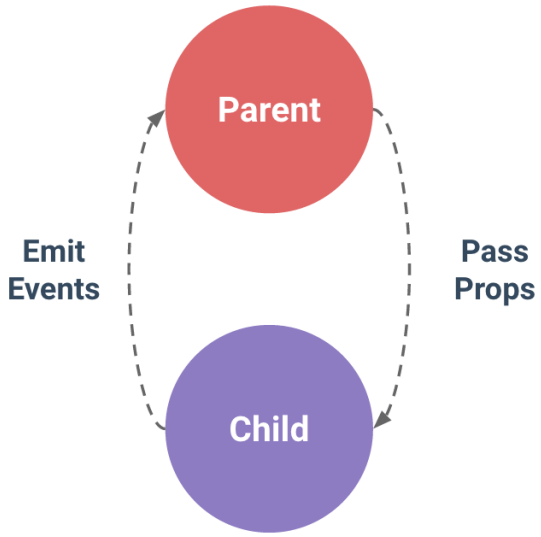
Global

```
Vue.component('my-component', {
  // options
})
```

Local

```
import MyTag from './components/MyTag';
const Child = {
  template: '<div>A custom component!</div>'
}
new Vue({
  // ...
  components: {
    // <my-component> will only be available in parent's template
    'my-component': Child,
    MyTag
  }
})
```

Components Communication in Vue.js



Component receive data through attributes binding by exposing properties.

Component send changes up to the parent by emiting events, to avoid mutations!

Components Properties in Vue.js

In javascript use **camelCasing**, which will be converted to **kebab-case** in HTML.

In compoonent expose properties

```
{
  //...
  props: ['myprop', 'myProp2'],
  // ...
  mounted(){
    this.myProp2;
  }
}
```

```
<my-comp v-bind:myprop="" :my-prop2></my-comp>
```

Warning: changing an attribute of a bound object mutates it's state outside the scope of the component.

Components Properties Validation in Vue.js

Define

```
Vue.component('myCheckbox', {
  props: {
    isOk: {
      type: Boolean,
      required: true,
      default: false
    },
    name: {
      type: String,
      required: true,
      default: 'Bob'
    }
  },
  template: `<label>{{name}} <span v-if="isOk">✓</span></label>`
});
```

Use

```
<my-checkbox :is-ok="booleanValue"></my-checkbox>
```

Components Properties Validation GOTCHA

Objects and arrays need their defaults to be returned from a function:

```
text: {
  type: Object,
  default: function () {
    return { message: 'hello mr. magoo' }
  }
}
```

Example (https://codepen.io/sdras/pen/63d98696878200f6c0e987cd58341c39)

To listen for a native event on the root element of component, .native has to be added:

<my-checkbox @click.native="doSomething"></my-checkbox>

By default, v-model on a component uses **value** as the prop and **input** as the event

Components Emitting Events

Define

```
Vue.component('myCheckbox', {
  methodes: {
    change() {
      this.$emit('hello', 'world');
    }
  },
  template: `<label>{{name}} <span @click="change">✓</span></label>`
});
```

Use

```
<my-checkbox @hello="doSomethingWithWorld"></my-checkbox>
```

Custom Events

Events can also be used to communicated between components. Vue.js events do not *bubble up* or *trickle down* the tree.

We attach and emit on a common object for example the **$root**.

```
// in component A
const clickHandler = clickCount => {
  console.log(`Oh, that's nice. It's gotten ${clickCount} clicks! :)`)
}
this.$root.$on('i-got-clicked', clickHandler);

// in component B
this.$root.$emit('i-got-clicked', this.clickCount);

// in component A to stop listening
// Stop listening.
this.$root.$off('i-got-clicked', clickHandler);
```

Content Distribution Inside a Component with Slots

```
<!-- app-layout component -->
<div class="container">
  <header>
    <slot name="header"></slot>
  </header>
  <main>
    <slot></slot>
  </main>
  <footer>
    <slot name="footer"></slot>
  </footer>
</div>
```

```
<app-layout>
  <h1 slot="header">Here might be a page title</h1>
  <p>A paragraph for the main content.</p>
  <p>And another one.</p>
  <p slot="footer">Here's some contact info</p>
</app-layout>
```

Multiples Views and Router

A SPA has to support multiple virtual views to simulate pages.

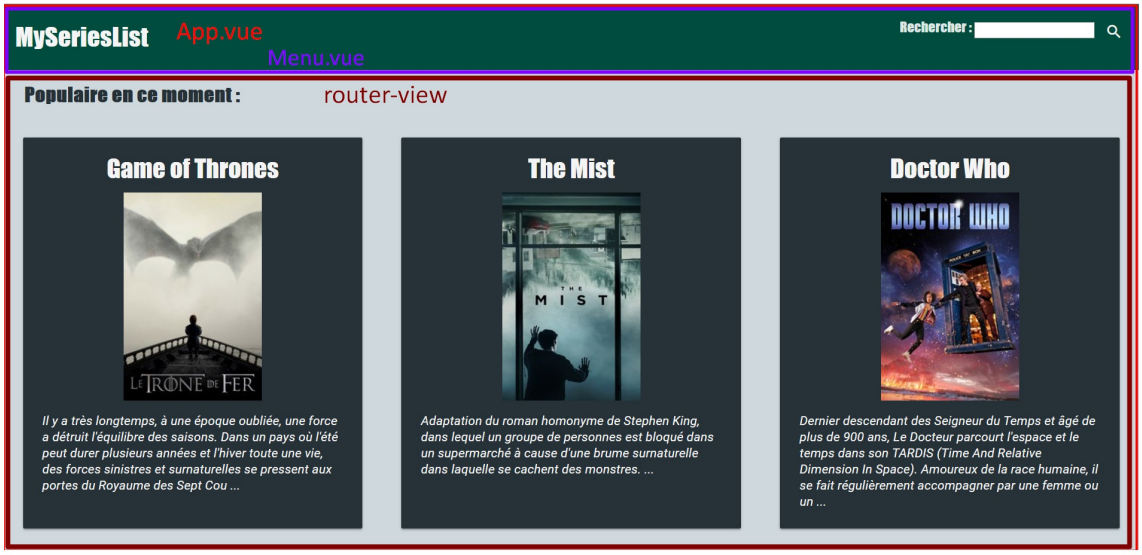
This can be achieved with a router, routes and components.

```
const router = new VueRouter({
  routes: [
    // dynamic segments start with a colon
    { path: '/user/:id', name: 'user', component: User }
  ]
})
```

```
 <!-- will host the component corresponding to the route -->
<router-view></router-view>

<!-- Vue instance has a special property with route params -->
<div>{{ $route.params.id }}</div>

<!-- create links by lookup of the route -->
<router-link :to="{ name: 'user', params: { userId: 123 }}">User</router-link>
```

```
// changing route in code.
this.$router.push({ name: 'user', params: { userId: 123 }})
```

Example Flow of Components and Router



Vue.js Advanced Stuff

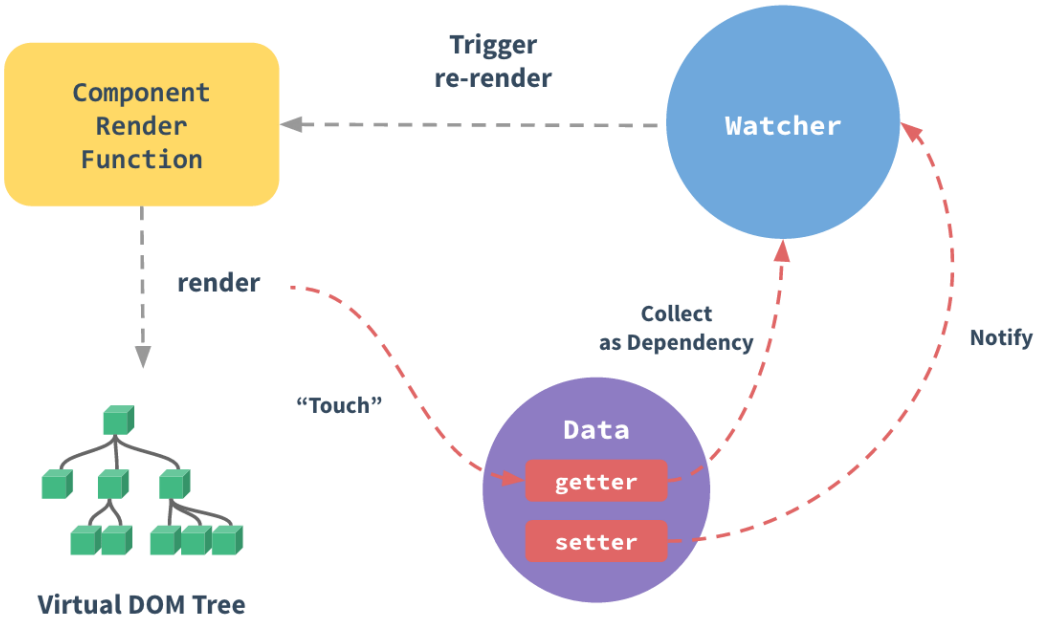
Access DOM through Refs

ref is used to register a reference to an element or a child component. The reference will be registered under the parent component's $refs object.

```
<div id="app">
  <div ref="someID"></div>
</div>

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

```
new Vue({
  el: '#app',
  mounted() {
    this.$refs.someID.innerText = 'DOM Direct Manipulation is BAD!';
  }
});
```

Watches



Lifecycle Hooks

https://vuejs.org/v2/guide/instance.html

created, mounted, ...

router also has some hooks

```
beforeRouteUpdate (to, from, next) {
  // react to route changes...
  // don't forget to call next()
}
```

Transitions

```
<transition name="fade">
  <p v-if="show">hello</p>
</transition>

<transition-group name="flip-list" tag="ul">
  <!-- multiple elements / move animations -->
</transition-group>
```

```
.fade-enter-active, .fade-leave-active {
  transition: opacity .5s
}
.fade-enter, .fade-leave-to /* .fade-leave-active in <2.1.8 */ {
  opacity: 0
}

.flip-list-move {
  transition: transform 1s;
}
```

Vue.js more

- Transitioning State
- Mixins
- Custom Directives
- Vue.nextTick
- Plugins
- Server-Side Rendering
- Webpack Code Splitting
- Route lazy loading
- State Management: VueX

Resources

- https://vuejs.org/v2/guide/
- https://vuejs.org/v2/api/
- https://www.grafikart.fr/formations/vuejs
- https://speakerdeck.com/bhawkes/introduction-to-vue-js

Setup environement:

- init a new vue webpack project
- install dependencies
- Adapt ESLint to our coding style
- Inside `build/webpack.dev.conf` Fix debugging by changing

```
devtool: 'source-map'
```

- install vue chrome devtool extension (https://chrome.google.com/webstore/detail/vuejs-devtools/nhdogjmejiglipccpnnnanhbledajbpd)
- install vscode **vetur** extension

- change vscode settings eslint.validate

```
"eslint.validate": [
        "javascript",
        "javascriptreact",
        "vue"
    ]
```

JSON

JavaScript Object Notation is a lightweight data-interchange format. It is easy for *humans* to **read and write**. It is easy for *machines* to **parse and generate**. It is based on a subset of the JavaScript Programming Language

```
{
  "key_string": "hello",
  "key_number": 3,
  "key_array": ["some text", 34]
  "key_object": {
    "other_key": "value"
    "key_boolean": true,
    "null possible": null
  }

}
```

http://json.org/

JSON API in JavaScript

JSON.stringify(*object*)	create a JSON_string
JSON.parse(*JSON_string*)	create an object from a string
angular.toJson(*object*)	create a JSON_string (removing angular's internal variables)
angular.fromJson(*JSON_string*)	create an object from a string

```
//optional formatter and indentation spacing for pretty-print
JSON.stringify( {hello: {text: 'world'}}, null, 2 )
//results in the following string
'{
  "hello": {
    "text": "world"
  }
}'
```

localStorage

Interface of the Web Storage API provides access to storage for a particular domain.

localStorage.length	Returns an integer representing the number of data items stored in the Storage object.
localStorage.key(number)	will return the name of the nth key in the storage.
localStorage.getItem(key)	will return that key's value.
localStorage.setItem(key, value)	will add that key to the storage, or update that key's value if it already exists.
localStorage.removeItem(key)	will remove that key from the storage.
localStorage.clear()	will empty all keys out of the storage.

localStorage content can be viewed in chrome developer tools resource tab

https://developer.mozilla.org/en-US/docs/Web/API/Storage

Exercice: localStorage and routes

Asynchronous programming techniques

We have already seen that JavaScript relies on asynchronous programming:

- The JS engine is single-threaded. For this reason, IO operations have to be non-blocking.
- An event loop is used both in the browser and on the server (node.js):
  - As the program executes, events are added to a queue. Every event has an associate callback function.
  - A dispatcher takes the next event in the queue and invokes the callback function (on the single thread).
  - When the callback function returns, the dispatcher takes the next event in the queue, and continues forever (it's an event loop).

Callback

```
setTimeout( function() {
    console.log("the callback has been invoked");
}, 2000);
```

An event will be added to the queue in 2000 ms. In other words, the function passed as the first argument will be invoked in 2 seconds or more (the thread might be busy when the event is posted...).

```
$(document).mousemove( function(event) {
    $("span").text(event.pageX + ", " + event.pageY);
});
```

An event will be added to the queue whenever the mouse moves. In each case, the callback function has access to the event attributes (coordinates, key states, etc.).

```
$.get( "ajax/test.html", function( data ) {
    $( ".result" ).html( data );
    alert( "Load was performed." );
});
```

An event will be added when the AJAX request has been processed, i.e. when a response has been received. The callback function has access to the payload.

Beyond simple callbacks...

- The principle of passing a callback function when invoking an asynchronous operation is pretty straightforward.
- Things get more tricky as soon as you want to coordinate multiple tasks. Consider this simple example...

Do this first...

... when done, do this.

A first attempt...

```
var milkAvailable = false;

function milkCow() {
  console.log("Starting to milk cow...");
  setTimeout(function() {
    console.log("Milk is available.");
    milkAvailable = true;
  },  2000);
}

milkCow();
console.log("Can I drink my milk? (" + milkAvailable + ")");
```

FAIL

Fixing the issue with a callback...

```
var milkAvailable = false;

function milkCow(done) {
  console.log("Starting to milk cow...");
  setTimeout(function() {
    console.log("Milk is available.");
    milkAvailable = true;
    done();
  }, 2000);
}

milkCow( function() {
  console.log("Can I drink my milk? (" + milkAvailable + ")");
});
```

SUCCESS

Beyond simple callbacks...

- Ok... but what happens when I have more than 2 tasks that I want to execute in sequence?
- Let's say we want to have the sequence B, C, D, X, Y, Z, E, F, where X, Y and Z are asynchronous tasks.

```
function f() {
  syncB();
  syncC();
  syncD();
  asyncX();
  asyncY();
  asyncZ();
  syncE();
  syncF();
}
```

```
B  result  available
C  result  available
D  result  available
E  result  available
Z  result  available
Y  result  available
F  result  available
X  result  available
```

Sequence with callbacks

```
function f() {
  syncB();
  syncC();
  syncD();
  asyncX(function() {
    asyncY(function() {
      asyncZ(function() {
        syncE();
        syncF();
      });
    });
  });
}
```

```
B  result  available
C  result  available
D  result  available
X  result  available
Y  result  available
Z  result  available
E  result  available
F  result  available
```

But welcome to the **"callback hell"** aka **"callback pyramid"**

Callback parallel tasks

- Now, let's imagine that we have 3 asynchronous tasks. We want to invoke them in parallel and wait until all of them complete.
- Typical use case: you want to send several AJAX requests (to get different data models) and update your DOM once you have received all responses.

```
function f( done ) {
 async1( function( r1 ) {
    reportResult( r1 );
  });
  async2( function( r2 ) {
    reportResult(r2);
  });
  async3( function( r3 ) {
    reportResult( r3 );
  })
  done();
}
```

Double fail: not only is done() invoked to early, but also there is no result to send back...

Callback parallel tasks with counter

```
function f( done ) {
  <span class="fragment highlight-current-red" data-fragment-index="1">var numberOfPendingTasks = 3;</span>
  var results = [];
  <span class="fragment highlight-current-red" data-fragment-index="2">
  function reportResult( result ) {
    result.push( result );
    numberOfPendingTasks -= 1;
    if ( numberOfPendingTasks === 0 ) {
      done( null, results );
    }
  }
  </span><span class="fragment highlight-current-red" data-fragment-index="3">
  async1( function( r1 ) {
    reportResult( r1 );
  });
  async2( function( r2 ) {
    reportResult( r2 );
  });
  async3( function( r3 ) {
    reportResult( r3 );
  });</span>
}
```

When this reaches 0, I know that all the tasks have completed. I can invoke the "done" callback function that I received from the client. I can pass the array of results to the function.

When a task completes, it invokes this function and passes its result. The result is added to the array and the number of pending tasks is decremented.

The three tasks are asynchronous, so they pass their own callback functions and receive a result when the operation completes.

Async libs to the rescue: Promise

A **promise** must be in **one of three states**: *pending*, *fulfilled*, or *rejected*.

When *pending*, a promise:

- may transition to either the *fulfilled* or *rejected* state.

When *fulfilled*, a promise:

- **must not transition** to any other state.
- must have a **value**, which must not change.

When *rejected*, a promise:

- **must not transition** to any other state.
- must have a **reason**, which must not change.

https://github.com/promises-aplus/promises-spec

A promise must provide a then method to access its current or eventual value or reason. A promise's `then` method accepts two arguments:

- `promise.then( onFullfilled, onRejected )`
- If `onFulfilled` is a function:
  - it must be called after promise is *fulfilled*, with promise's value as its first argument.
  - it must not be called before promise is *fulfilled*.
  - it must not be called more than once.
- If `onRejected` is a function,
  - it must be called after promise is *rejected*, with promise's reason as its first argument.
  - it must not be called before promise is *rejected*.
  - it must not be called more than once

https://github.com/promises-aplus/promises-spec

then must return a promise.

```
promise2 = promise1.then(onFulfilled, onRejected);
```

- If either onFulfilled or onRejected returns a value x, run the Promise Resolution Procedure Resolve(promise2, x).
- If either onFulfilled or onRejected throws an exception e, promise2 must be rejected with e as the reason.
- If onFulfilled is not a function and promise1 is fulfilled, promise2 must be fulfilled with the same value as promise1.
- If onRejected is not a function and promise1 is rejected, promise2 must be rejected with the same reason as promise1.

https://github.com/promises-aplus/promises-spec

Deferred objects in JQuery

"a **promise** represents a value that is not yet known, a **deferred** represents work that is not yet finished"

http://blog.mediumequalsmessage.com/promise-deferred-objects-in-javascript-pt1-theory-and-semantics

```
var d1 = new $.Deferred();
var d2 = new $.Deferred();
$.when( d1, d2 ).done(function ( v1, v2 ) {
  console.log( v1 ); // "Fish"
  console.log( v2 ); // "Pizza"
});
d1.resolve( "Fish" );
d2.resolve( "Pizza" );
```

Promise in ECMAScript 2015

```
const promise = new Promise(function(resolve, reject) {
  // do a thing, possibly async, then…

  if (/* everything turned out fine */) {
    resolve("Stuff worked!");
  }
  else {
    reject(Error("It broke"));
  }
});

promise.then(function(result) {
  console.log(result); // "Stuff worked!"
}, function(err) {
  console.log(err); // Error: "It broke"
});
```

https://developers.google.com/web/fundamentals/getting-started/primers/promises

Chaining Transforming values

```
const promise = new Promise(function(resolve, reject) {
  resolve(1);
});

promise.then(function(val) {
  console.log(val); // 1
  return val + 2;
}).then(function(val) {
  console.log(val); // 3
})
```

Wait for all

```
Promise.all(arrayOfPromises).then(function(arrayOfResults) {
  //...
})
```

https://developers.google.com/web/fundamentals/getting-started/primers/promises

API and Remote Data

POSTMAN (https://chrome.google.com/webstore/detail/postman-rest-client-packa/fhbjgbiflinjbdggehcddcbncdddomop) a tool to test apis

https://www.getpostman.com/docs/introduction

Google Cusom Search API

https://developers.google.com/custom-search/json-api/v1/reference/cse/list

```
`https://www.googleapis.com/customsearch/v1?
cx=011288001747608865807:a7rxzv4srri&q=${item.name}&searchType=image&safe=high&key=AIzaSyBlh2KvC84vD0cebFOlMSnLe0-Dx1mc-2A`
```

Getting JSON content with Fetch

```
fetch('./api/some.json')
  .then( response => {
      return response.json();
    }
  )
  .then(data => {
      console.log(data);
  })
  .catch(function(err) {
    console.log('Fetch Error :-S', err);
  });
```

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Getting JSON content with axios

```
npm install axios --save
```

```
import axios from 'axios';

axios.get('/user?ID=12345')
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

```
axios.get(`https://www.googleapis.com/customsearch/v1?
cx=011288001747608865807:a7rxzv4srri&q=${this.$route.params.name}&searchType=image&safe=high&key=AIzaSyBlh2KvC84vD0cebFOlMSnLe0-
Dx1mc-2A`)
  .then((response) => {
      console.log(response);
  })
  .catch(error => {
      console.log(error);
  });
```

Load Data in Vue.js

```
new Vue({
  el: '#app',
  data: function () {
    return {
      dataLoaded: false,
      apiReply: {}
    };
  },
  methods: {
    loadData: function() {
      axios.get('/api').then(response => {
        this.apiReply = response.data;
        this.dataLoaded = true;
      });
    }
  },
  created() { // or mounted
    this.loadData();
  }
});
```

Async / Await

```
async function myFirstAsyncFunction() {
  try {
    const fulfilledValue = await promise;
  }
  catch (rejectedValue) {
    // …
  }
}
```

```
function logFetch(url) {
  return fetch(url)
    .then(response => response.text())
    .then(text => {
      console.log(text);
    }).catch(err => {
      console.error('fetch failed', err);
    });
}

async function logFetch(url) {
  try {
    const response = await fetch(url);
    console.log(await response.text());
  }
  catch (err) {
    console.log('fetch failed', err);
  }
}
```

```
// map some URLs to json-promises
const jsonPromises = urls.map(async url => {
  const response = await fetch(url);
  return response.json();
});
```

https://developers.google.com/web/fundamentals/getting-started/primers/async-functions

Exercice real data

For now we leave the first page as is and extend the app to display movies from an external web service

Add the following to the moviedb service or even better make them constants.

```
var apiUrl = 'http://api.themoviedb.org/3/'; //proxy
var baseUrl = 'http://image.tmdb.org/t/p/';
```

- `getMovie()` returns a promise http of a movie (http://apiUrlEndpoint/movie/id?api_key=apiKey&append_to_response=similar,releases,credits&language=fr)
- `getMovieResults()` retrieve movie results array
- `searchMovies()` http://apiUrlEndpoint/search/movie?api_key=apiKey&language=fr&query=movie_name updates internal movieResults array
- `upcomingMovies()` http://apiUrlEndpoint/movie/upcoming?api_key=apiKey&language=fr updates internal movieResults array

Using Material instead of bootstrap

https://vuetifyjs.com/

```
$ npm install vuetify --save
```

```
<link rel="stylesheet" href="//fonts.googleapis.com/icon?family=Material+Icons">
```

```
import Vue from 'vue';
import Vuetify from 'vuetify';
import('./node_modules/vuetify/dist/vuetify.min.css');

Vue.use(Vuetify)
```

Read the docs, copy examples, ...

Material Design Ressources

- https://www.materialpalette.com/green/blue
- https://materialdesignicons.com/

Web App Manifest

Web App Manifests are one of the key pieces to making your web app look and feel like a native app

https://tomitm.github.io/appmanifest/

- Add to Homescreen
- Fullscreen
- Notifications

- Meta viewport
- Colors
- Zoom, touch interactions

Synchronised persistent datastorage

Firebase Database

Install Firebase

```
$ npm install -g firebase-tools
```

Available commands: login, init, serve, deploy

Integration with Vue.js

```
$ npm install firebase vuefire --save
```

https://github.com/vuejs/vuefire

To use login UI for firebase authentification

```
$ npm install firebaseui --save
```

https://github.com/firebase/FirebaseUI-Web

Documentation

https://firebase.google.com/

https://firebase.google.com/docs/auth/web/github-auth

```
// firebase.js
import Vue from 'vue';
import firebase from 'firebase';
import VueFire from 'vuefire';
Vue.use(VueFire);

// Initialize Firebase
// Copy from google firebase console (Authentication>Web Setup)
const config = {
    apiKey: 'AIzaSyAauSkomxxlcgMqlYABWkzuEvHkxaT0xHc',
    authDomain: 'ptw.firebaseapp.com',
    databaseURL: 'https://ptw.firebaseio.com',
    projectId: 'firebase-ptw',
    storageBucket: 'firebase-ptw.appspot.com',
    messagingSenderId: '281865054216'
};
export default firebase.initializeApp(config);
```

Lab: Firebase Messages

```
import firebase from '../firebase';
export default {
    data() {
        return {
            newMsg: '',
        };
    },
    firebase() {
        return {
            messages: {
                source: firebase.database().ref('/demo/messages')
            }
        };
    },
    methods: {
        send() {
            this.$firebaseRefs.messages.push({txt: this.newMsg});
            this.newMsg = '';
        },
        remove(p) {
            this.$firebaseRefs.messages.child(p['.key']).remove();
        }
    }
};
```

Lab: Firebase Game

https://gist.github.com/bfritscher/f15258ad2161eda24a32159632738bcc

Setup Firebase User Security

```
{
    "rules": {
       "users": {
      "$uid": {
        // grants write access to the owner of this user account whose uid must exactly match the key ($uid)
        ".write": "auth !== null && auth.uid === $uid",
        ".read": "auth !== null && auth.uid === $uid"
        }
      }
    }
}
```

https://firebase.google.com/docs/database/security/quickstart

Serverless: Firebase Cloud Functions

https://github.com/firebase/functions-samples/tree/master/exif-images

https://firebase.google.com/docs/reference/functions/functions.storage.ObjectMetadata

https://cloud.google.com/vision/docs/reference/libraries#client-libraries-install-nodejs

https://firebase.google.com/docs/functions/config-env

Firebase Functions: Keeping Secrets

```
$ firebase functions:config:set service.name="value"
firebase functions:config:get
```

```
functions.config().someservice.id

exports.groupA = {
  function1: functions.https.onRequest(...);
  function2: functions.database.ref('\path').onWrite(...);
}
exports.groupB = require('./groupB');
```

Deploy to Firebase

```
$ firebase deploy --only functions
$ firebase deploy --only hosting
$ firebase deploy --only functions:function1,function2
```

Possible next steps after deploy

- Analytics & SPA
  - Virutal page views
  - Events
- A/B testing your site!