

IIT Madras BSc Degree

Copyright and terms of use

IIT Madras is the sole owner of the content available in this portal - onlinedegree.iitm.ac.in and the content is copyrighted to IIT Madras.

- Learners may download copyrighted material for their use for the purpose of the online program only.
- Except as otherwise expressly permitted under copyright law, no use other than for the purpose of the online program is permitted.
- No copying, redistribution, retransmission, publication or exploitation, commercial or otherwise of material will be permitted without the express permission of IIT Madras.
- Learner acknowledges that he/she does not acquire any ownership rights by downloading copyrighted material.
- Learners may not modify, publish, transmit, participate in the transfer or sale, create derivative works, or in any way exploit, any of the content, in whole or in part.

Assorted Vues

Persistent Storage

What?

- Vue data lost (or set to initial state) whenever page reloaded
- ***Persistent*** state without need for server

Why?

- True persistence possible at server
- Offline:
 - Work without requiring server connection
- Simple apps:
 - no need to go to server for simple requests
- Configuration
 - Local configurations possible - not needed by server

How?

- Cookies
 - JS `setCookie()` can be used for simple data
 - Limited storage - usually session temporary - removed on browser restart
- `localStorage`:
 - API to save simple key -> value entries
 - Complex objects should be stringified - use JSON
- `IndexedDB`:
 - Transactional database system
 - Object-oriented JS-based DB
 - store and retrieve objects with a key

localStorage

- WebStorage API
 - https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API
- sessionStorage
 - storage ends with session (browser restart)
 - more storage than cookies (~ 5MB)
- localStorage
 - persists across browser restarts
 - browsers may implement limits to avoid overload

Example: <https://vuejs.org/v2/cookbook/client-side-storage.html>

Form Validation

Validation

- Check whether data entered in form meet certain criteria
- Simple checks in browser
 - Text field contains number, email address etc
 - Select field has at least one entry
 - Empty fields
- Server side checks
 - Essential in many cases for security
 - More costly, increase server load

Vue and Validation

- Data binding and reactivity
 - Easy updates of parts of DOM
 - Selectively display error messages: `v-if`, `v-show`
- `v-model` connects fields with JS variables for easy processing
- `preventDefault()` - stop normal processing of form unless check successful
 - connect as `submit` event handler

Custom Validation

- Example: custom email check
 - Specific domain, specific number of characters etc.
- Example: check for certain overall condition
 - All numbers add up to given value
- Need to prevent regular form validation
 - `novalidate=true` in form definition

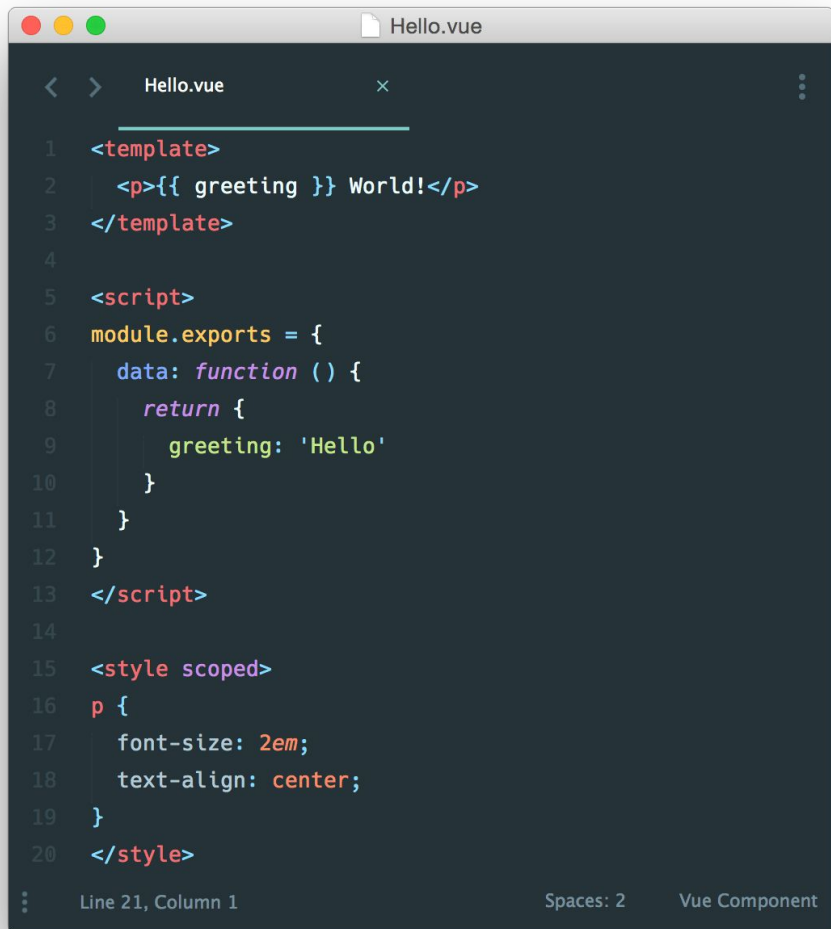
<https://vuejs.org/v2/cookbook/form-validation.html>

Managing Components

Need for Single File Components

- Global namespace
 - Unique names needed for each component
 - (probably good anyway, but difficult when importing from other sources)
- String templates
 - Harder to edit and manage with regular editors
- CSS!
 - No block scoping - only global CSS
 - Not modular unlike HTML (templates) and JS (components)
- No build step
 - Backwards compatibility not easy, cannot use tools like Babel

SFC structure



```
1 <template>
2   <p>{{ greeting }} World!</p>
3 </template>
4
5 <script>
6 module.exports = {
7   data: function () {
8     return {
9       greeting: 'Hello'
10    }
11  }
12 }
13 </script>
14
15 <style scoped>
16 p {
17   font-size: 2em;
18   text-align: center;
19 }
20 </style>
```

Line 21, Column 1 Spaces: 2 Vue Component

Src: <https://vuejs.org/v2/guide/single-file-components.html>

Separation of Concerns

- Would normally like separation
 - Semantic content - HTML
 - Presentation - CSS
 - Logic - JS
- This does not require separate files!

Extra “Tooling”

- JS cannot directly read .vue files
- Compilation step needed
 - Convert .vue to .js + .css + .html
- Webpack, ESBuild, Vite etc. - package files
- npm: Node Package Manager
 - Systematically managed JS modules
 - Import new modules as needed

Mostly managed from Command Line Interface

Testing

Testing Vue applications

- Unit testing
 - As with any other software process
 - Components are good units to test
 - Can “mount” into a testing DOM
- E2E (End-to-End) testing
 - Full application including backend
- Cross-browser testing
 - Compatibility with older browsers
 - Diminishing returns - is it worth supporting all older browsers?

Test mechanisms

- Set up fixtures:
 - Prepared data
- Test Suite
 - Collect several tests together
 - Test one component
- Tools
 - mocha, chai, jest: supporting functions to test for presence/absence of elements

<https://vuejs.org/v2/cookbook/unit-testing-vue-components.html>