Workshop: Intro to Vue Cheatsheet

Installation

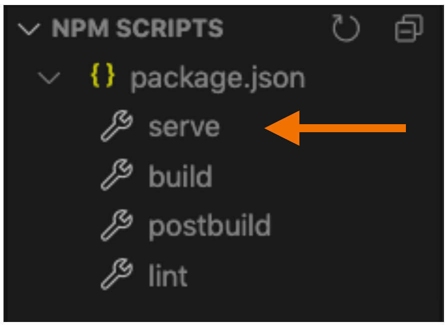
The steps below assume that you have VSCode installed (see here to download https://code.visualstudio.com/).

If you are on Windows, please use the command prompt to type in the commands. If you are using a Mac, please use terminal to type in the commands.

1. Check if you have node installed

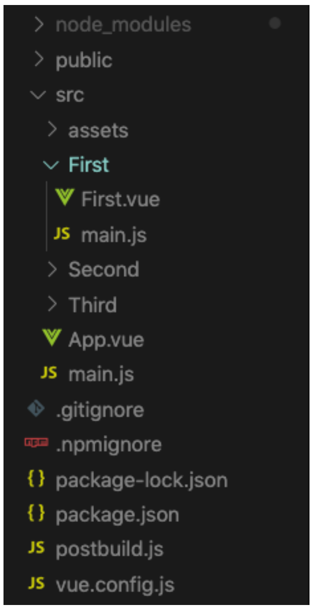
   A quick way is to run "`node -v && npm -v`" and see if it outputs an error.
2. If not, install from https://nodejs.org/en/download/
3. Run "`npm -g install @impvis/cli`" to install the ImpVis command-line
4. After completion, change directory to the one where you want your project to be located in using the "`cd ~~directory path~~`" command
5. Run "`impvis create ~~project name~~`" to create your project (it is recommended to select the current template.

Now to actually run your code, using VSCode, open the directory which the command-line just created. After everything finishes loading, click the "NPM Scripts" panel in the bottom left-hand side, hover over the serve item, and click the play triangle that appears.



A terminal should pop out. Once it is done loading, please open the link that appears with your desired browser application.

Basics



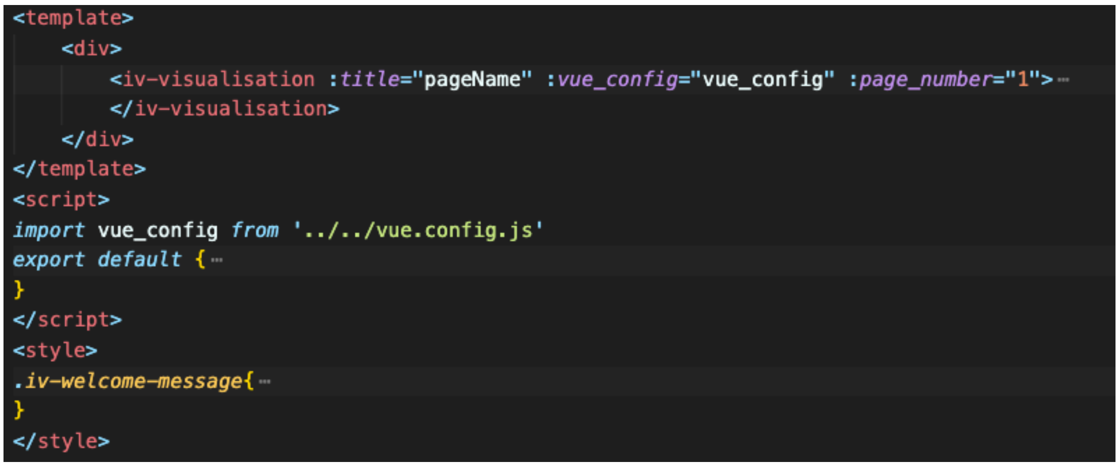
```
> node_modules
> public
v src
  > assets
  v First
     First.vue
     main.js
  > Second
  > Third
   App.vue
   main.js
 .gitignore
 .npmignore
 package-lock.json
 package.json
 postbuild.js
 vue.config.js
```

Key files

(* is a wildcard, denoting any string of letters)

- `*.vue` - these are where all the Vue code goes
- `main.js` - this tells the browser how to load and display the Vue file
- `.gitignore` - tells git which files not to save to repository
- `package-lock.json` - a snapshot of all the node packages you have installed, never to be touched by the user
- `package.json` - contains the basic information for your package, most importantly: scripts what you are running when you click serve and dependencies/devdependencies are where you tell node what packages you want
- `vue.config.js` - used by ImpVis to define what pages are in your visualisation

Overview of a basic vue file



```
<template>
    <div>
        <iv-visualisation :title="pageName" :vue_config="vue_config" :page_number="1">…
        </iv-visualisation>
    </div>
</template>
<script>
import vue_config from '../../vue.config.js'
export default {…
}
</script>
<style>
.iv-welcome-message{…
}
</style>
```

There are three parts to any Vue file: template, script, and style.

TEMPLATE: The space inside your template tag is like your html file. However, you are able to use more powerful Vue constructs inside, which can sometimes take the place of Javascript code. In addition, you can tell Vue to embed these small components inside others.

In the template, you can use `"iv-*"` tags to refer to components that ImpVis has developed.

SCRIPT: This is where you write your Javascript code to create interaction on the page. Particular to Vue, you first important all the components that you need (in the picture, we import an object `vue_config`), then we use the construct `"export default { … }"` to give Vue the information it needs about the different properties of the component. This is in the form of an object.

Vue has a list of special key names that it will look at to do specific things on the page. For example, you might see a `"name"` key in your object, which specifies the name of the page (and later component, but how this works is for another day).

Below, I have picked a few of the most important key names that you might come across. Please look at the Vue documentation for a complete explanation and other key names that exist.

- `name` - the name of the page or component as a string

- `components` - once you have imported other vue files in the script section, this is where you list those other components for Vue to make available for you to use in the template

- `data` - a dictionary of variable names and values that you want to be available to use in your template that can change dynamically. Often, you will see this written as a function because then, each new instance of the Vue file will have a different data object.

- `methods` - a collection of methods/functions that you can call in code or in your template using events to trigger certain behaviour. For example, if you want to update a data value whenever a slider is changed, you would add a prop to the slider that tells it to call a method that you define with the event.

- `mounted` - a set of code that the page runs on first start. This is where you would put your code if you are moving from a HTML+JS project.

- `props` - like how a function accepts a list of parameters, each Vue file can also accept what are called props. Here, you define the names of the props that this file accepts and their expected type.

STYLE: You can write CSS code within these tags to use in your template.

Key ImpVis Components

General Structure of Components

The following shows how you might nest the different components to create a sidebar, explanations, and hotspots.

```
<iv-visualisation>
      <template #hotspots>
            <iv-toggle-hotspot> or <iv-fixed-hotspot>
                  <iv-slider>

      <iv-pane>
            <iv-sidebar-content>
                  <iv-sidebar-section>
                        <p>
                        <iv-equation-box>

      <div> *embed your main visualisation within this div
```

Explanation of Individual Components

I have listed some of the most important attributes for each tag. For example, if I want to set the title to "`Hello!`" and theme to "`orange`", I would write:

```
      <iv-sidebar-section title="Hello!" theme="Orange">.
```

`iv-visualisation`: sets up header and enables adding of hotspots

- *`title`* - *string*; title of page
- *`hotspotColumnWidth`* - *string*; space to put around the hotspots (e.g. "100px")
- *`hotspotColumnHeight`* - *string*; same as above, but for height

`iv-toggle-hotspot`: a hotspot to put additional information or interaction elements in
`iv-fixed-hotspot`: same as toggle, except is fixed (cannot be hidden)

- *`title`* - *string*; title of hotspot
- *`position`* - *string*; where to put the hotspot, available positions are:
  ["left","right","top","bottom","topleft","topright","bottomright","bottomleft"]

`iv-slider`: a slider

- *`:init_val`* - *float*; initial value of slider (write it in code as a string though)
- *`:min`* - *float*; min value of slider
- *`:max`* - *float*; max value of slider
- *`:step`* - *float*; step size of slider
- *`@sliderChanged`* - *string*; name of function in methods to be called with new value
- *`theme`* - *string*; see themes below (essentially colour)
- *`ref`* - *string*; name of slider to reference using "`this.$ref`"

`iv-pane`: adds a pane to the page

`position` - *string*; position to put plane (normally “left”)
`format` - *string*; either “push”, “full”, or “overlay"

`iv-sidebar-content`: encloses sidebar contents

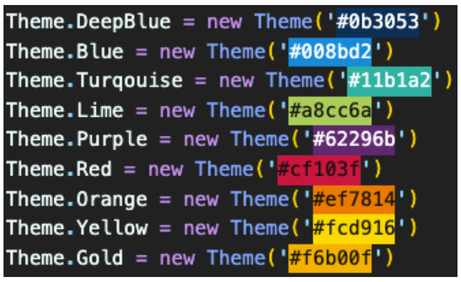
`iv-sidebar-section`: starts a new section in your sidebar

`title` - *string*; title of the section
`theme` - *string*; essentially colour of the section (themes available are listed below)
`icon` - *string*; name of icon from the `font-awesome` icon library to display

`iv-equation-box`: displays a typeset, LaTeX equation

`:stylise` - *true or false*; whether to enclose the equation in a box
`equation` - *string*; LaTeX string representing the equation you want to display

Themes



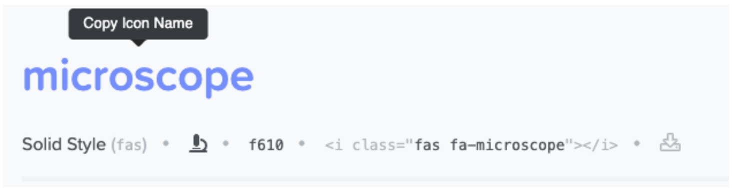
```
Theme.DeepBlue = new Theme('#0b3053')
Theme.Blue = new Theme('#008bd2')
Theme.Turqouise = new Theme('#11b1a2')
Theme.Lime = new Theme('#a8cc6a')
Theme.Purple = new Theme('#62296b')
Theme.Red = new Theme('#cf103f')
Theme.Orange = new Theme('#ef7814')
Theme.Yellow = new Theme('#fcd916')
Theme.Gold = new Theme('#f6b00f')
```

The available theme names are listed to the right in the form `Theme.[theme_name]`. Use `theme_name` when setting a theme in `iv-sidebar-section`.

Icons

https://fontawesome.com/icons?d=gallery

Using the link above, search and select a non-pro icon. Then use the icon name in your code to use a custom icon.



Useful Tips

Git - when you initialise a git repository with your project, you might find that it is asking to commit 1k+ files at once. If this is the case, you need to create a `".gitignore"` file and add the line `"node_modules/"` to it. This tells git to ignore that folder, which contains files for all the modules you installed.

Working on Someone Else's Project - when you download a project from Git, it probably has only the code, not the node_modules folder. To install the modules, change directory to the project folder and then run `"npm ci"` (if this doesn't work, run `"npm install"`).

Semicolon Attributes - when you see an attribute that starts with a semicolon, e.g. `":title=…"`, Vue interprets its value as Javascript code. In other words, `":title='pageName'"` actually sets the title to the *variable* `"pageName"` instead of the raw string.

Getting Data in Code - a reference to the component that the code lives in is available in `"this"`. Keep a reference to this to read or write to the value in code. See the example project for specifics of how to do this.

Slider Updates - For now, to get continuous updates from the slider, the best way is to use a reference to the slider (a bit of a hack, as this requires an update to the ImpVis component library). An example of this is included in the example project.

Jargon

Object - in JS, an object can be thought of as a dictionary. A dictionary contains a list of keys that each have a value attached. For example, I could have a list of keys that are names, each with the height of that person as the value. I would define the object as:

```
{
    Tom: 4,
    Cathy: 10,
    Jamie: 12
}
```

Objects can be nested and also contain functions as their value.

Tag - things that look like <…> </…>, the logical unit of the html language. The first word inside the tag is the name of the tag.

Additional Resources

Official Vue Guide - https://vuejs.org/v2/guide/

ImpVis Tutorial - https://impvis.co.uk/launch/impvis-tutorial-unpublished/index.html

ImpVis Vue Components Code - https://github.com/Imperial-visualizations/Vue-Components

Exercises

1. Add another section to the sidebar
    - Add another equation to that section
    - Change the colour and icon of the section
2. Add another hotspot with another slider
    - Link the slider to some other variable (e.g. change in gravity)
3. Add a style to the style section and link it to the main visualisation