DYNAMIC IMAGE DATA THEME

ADAM LLOYD – R00117318 – ADVANCED WEB PUBLISHING APPS

https://github.com/Radicis/node-gallery-module

MOTIVATION

The motivation behind this project was to create a reusable theme to display a data set consisting of images and meta data in a manner that would be appealing to end users.

POTENTIAL USES

This theme would be well suited for any user that needs a fun and dynamic way to manage a large set of images on their web sites such as schools, social groups, or events.

It also serves as a solid standalone gallery theme for those wanting to focus the design of their site on the image content completely or to display their content in a mobile web app.

STYLE CHOICES

The simplistic layout lends its self to integration with pre-existing websites if it were imported as a gallery module as it does not have any dominant features which would look out of place on varying web site designs. This simplicity also lends itself well to being a standalone gallery web or mobile app.

The choice to remove any conventional “GO” or “Search” buttons from the search field and have the script monitor for key presses instead further solidify the simplistic and clean design.

COLLAGE STYLE DISPLAY

The dynamic “collage” style display provides the ability to open the full-size image using the standard lightbox library, view optional meta data and provide the ability to cycle through the loaded images.

The theme is fully responsive because of multiple media queries dictating the CSS column-count property for each notable screen width. The column-count property allows for the images to fit into the collage somewhat seamlessly regardless of height, without the whitespace that would normally be present.

This concept was improved upon by allowing the user to set a column width in the configuration which switches the gallery from responsive CSS mode to adaptive JavaScript mode. This mode calculates the max width of the collage container and the width of the contained images using the column count variable and the width of the smallest images to ensure that there are always X columns on the screen. It also ensures that the container does not expand past the point where the images would be stretched or would have space between columns.

Sorting buttons are featured which integrate seamlessly with the search bar and call the MixItUp library to provide a stylish animation and add extra flair to the theme.

The site identity, search and sorting controls react to the window scroll event and pop into a fixed floating menu when a threshold is reached.

DATA SET AGNOSTIC

The theme was created to be data set agnostic by only modelling the object schema using the values stored within to conform the data set to one compatible with the theme.

This was achieved by mapping the required fields, such as title and thumbnail, to corresponding properties of the object. For example, an object must have a thumbnail image URL as one of its properties to be compatible, however it is unlikely that all data sets would have a property called “thumbnail”. To handle this, the

“thumbnail” property is mapped to the corresponding property of the object, such as “thumbnailUrl” by storing it is the database. The theme looks up that property whenever the “thumbnail” property is requested and renders the correct value.

The configuration for this is accessed via the /config route which displays a form allowing the user to map the properties of any data set within a Mongo DB database to corresponding ones within the theme.

STYLE CONFIGURATION

Within the configuration form, the user can also configure the look and feel of the theme by setting colours, item limits and toggling interface elements.

To add an extra layer configuration, it is possible enter and compatible CSS values for the background and font colour which are reflected as the user types to provide a preview of the changes before saving them.

FUTURE WORK

If I were to develop this further I would certainly remove the dependency on Mongo DB and create a database adapter class for each of the popular databases or, alternatively, allow the user to fetch the data and simply manage a JSON collection. This approach would lend itself well to consuming data sets originating from restful APIs.

It would also be possible to link it to a social media account to add more dynamism with minimal work.

I would also remove the dependency on express and handlebars. Moving all the functionality into a JavaScript module which would enable it to be packaged more efficiently.