Chewy Alpha Release

chyzewsk - Kenneth Matthew Ryan Chyzewski
---
kenneth.chyzewski@mail.utoronto.ca

Description of Chewy

Chewy is a vanilla Javascript library that allows users with visual impairment or specific display preferences to reshape websites to enhance the viewing experience with the click of a button. The idea arose from watching Youtube, where the user is allowed to switch the video display type, ranging from default mode, Miniplayer mode (i), Cinema Mode mode (t) or Fullscreen mode (f). The letter following each mode represents a keyboard character the user can press to seamlessly switch between viewing modes. Chewy follows suit, aiming to increase the ease of use for people with specific needs by reshaping websites, scraping images and enlarging them at the click of a keyboard button.

Current Chewy features

As of the Alpha release, Chewy currently has three viewing modes: a three by three grid, a fullscreen scrollable column and a fullscreen slideshow. Looking at code, the demonstration both on the bottom of the web page itself or in the examples.js file show the full functionality of Chewy. Here is the example at the bottom of the web page:

```
const chewy = Chewy() //Create an instance of Chewy.js
chewy.createFormat('K', '69 x 69 grid') // Add a viewing format your users would appreciate
chewy.updateFormat('K', '3 x 3 grid') // Modify existing formats based on user feedback
chewy.deleteFormat('K') // Remove existing formats
```

When a developer creates an instance of Chewy.js, the chewy assistant on the bottom right of the web page will appear. This assistant is strictly to inform the user that Chewy is active and can be used to their own discretion. All three API functions, create, update and delete are functional and further explained below. Chewy currently does not provide any default templates, rendering it useless until the developer creates one. updateFormat() and deleteFormat() are rather trivial as they are simply updating or deleting private dictionary data inside the Chewy Javascript object respectively. However, createFormat() is currently very limited and will not work with all developer formats. The only supported formats as of this release are: '3 x 3 grid', 'slideshow' and 'column'. Further explanation found in the **Future Chewy features and challenges** section below.

Link to deployed web page and expectations

https://enigmatic-brook-57872.herokuapp.com/

After clicking on this link, you can expect to see a website that demonstrates the usage of Chewy.js. There is a lot of text placed on the website to guide the user how to use, install and setup Chewy.js for their own use. The expandable assistant on the bottom right corner allows a user to view the available display formats. Pressing any of the listed Keyboard buttons inside the assistant will transform the web page accordingly. Pressing the same Keyboard button will revert the page to normal. The bottom of the website has a developer section devoted to Documentation and Installation.

JSObject explanation

The only JSObject a developer would work with when using Chewy.js is the Chewy() instance. The Chewy object stores the following data: developer-created formats in a dictionary, mapping of keyboard button codes to their integer value in a dictionary, web page's body tag with all of its contents (needed to revert the page), and a flag to indicate whether a user is in a viewing mode or not. When instantiated, the Chewy object also scraps and stores information about the present web page, such as the user's window height and width (needed for resizing images). Once Chewy is setup, the first thing it does is injects a bit of html and css, the Chewy assistant, to the web page. The Chewy assistant serves the purpose of informing the user of available viewing formats and when the user does enter a viewing mode, informs the user how to revert the web page to normal. This assistant stores nothing however, it does reads the developer-created formats stored in the Chewy object's dictionary to dynamically display them all. The Chewy API offers three functions for their use: createFormat(), deleteFormat(), updateFormat(). These functions modify the data contained in the Chewy object, changing the way the user can interact with the present web page. Below is an abstract view of the Chewy object when you open the examples web page using the URL found in the **Link to deployed web page and expectations** section.

`Chewy` instance:

```
{
 inViewMode: false,
 formatsDict: {71: "3 x 3 grid", 76: "column", 221:
"slideshow"},
 codesDict: {71: "G", 76: "L", 221: "]"},
 oldBody: window.body,
 pageHeight: window.screen.availHeight,
 pageWidth: window.screen.availWidth,
 createFormat = function(key, format),
 deleteFormat = function(key),
 updateFormat = function(key, format)
```

```
}
```

Future Chewy features and challenges

As mentioned in **Current Chewy features** section, createFormat() is far from where I want it to be. I want a developer to give any format string ('5 second slideshow', '2 x 6 grid', 'row', etc) and have Chewy able to interpret this string to build a viewing format out of it. With my less than 3 months of Javascript experience, I am not sure how to approach this feature. I need to find a way to generate code on the fly, based on key words such as 'grid' or 'slideshow'. Using enough if statements, I can likely mimic this code generation with boilerplate format code and have this ready for the final submission. The other feature I wanted to implement was what I had mentioned in the Proposal, that is, using some library to automatically scale images while maintaining their resolution. After conducting some research, I have found out this issue, to-date, has never been solved and there are currently Artificial Intelligence and Machine Learning experts collaborating to solve this problem. I highly overestimated this simple-sounding feature but, I will continue to search for alternative Javascript libraries that can at least scale images without drastically reducing their resolution like Chewy currently is.