Major Web Project

Part 2 - Static Design, Responsive Design and Wordpress

Login Information

My Wordpress site is available at http://janv.frwaw.itu.dk. You can use the following administrator user to play around with the custom post types when verifying the dynamic behaviour of the site:

- Username: “examiner”
- Password: “grade12”
- Backend URL: http://janv.frwaw.itu.dk/wp-admin

The code for Part A, B and C is also made available on GitHub:

- https://github.com/jvmk/ITU-FRWAW-MWP

Differences Between Part A and Part C

I did not manage to implement all functionality in Wordpress as a result of being quite ambitious when designing the look, layout and functionality of the site in the first assignment. The missing features are primarily features that require more sophisticated database accesses than those taught as part of the course. For example, I was planning to include frontend user posting on the Blogs page. This functionality should also include a rating system, i.e. a way for users to rate the movie blog postings of other users. Obviously this requires some sort of database table that maps user ratings to movie blog posts. I also did not manage to implement a recommendation system for the Movies page.

I consider the above limitations to the functionality warranted, given that we have only been taught how to use Wordpress backend posting and how to query these backend posts using The Loop.

To compensate for the lack of functionality on the Blogs and Movies pages, I have developed the About page (which was not part of my three visual mockups in assignment 1). The About page has dynamic content in the form of the partners section on the right hand side. Partners can be added/removed using the Wordpress backend.

Another thing missing is the frontend login functionality. Again, I do not recall that we have been taught how to implement this. I could have investigated how to do this on my own, but I did not have the time to do so.

Dynamic Content

All content on the Wordpress site is dynamic in the sense that you can add new posts using the backend. On the sidebar on the left hand side of the backend user interface, you will find the different post types:

- Screening: add/remove upcoming screening events (for the Movies page).

- News Post: add/remove general news items (news posts on the front page).
- Partner: add/remove sponsors and affiliates (partner information on the About page).
- Movie Blog Post: add/remove a blog post about a movie or actor (blog posts on the Blogs page)

The only exception is the Organization section of the About page which is purely static (hardcoded in the php file).

Code Structure

The different pages of my site uses the same overall structure for the main content: the primary content is in a (or possibly multiple) box(es) on the left hand side (LHS), while secondary content resides in one or more boxes on the right hand side (RHS).
In order to avoid code duplication, I make heavy use of template files. These template files are located in the `partials` folder of the theme. What follows is an example of how these template files are used.

By assigning an array of arrays to a specific variable (`$rhs_content_boxes`) and including the rhs-content.php file, one can easily create multiple RHS content boxes for a new page. For example:

```
$rhs_content_boxes = array(
   array(
       'heading' => 'Upcoming Screenings',
       'items' => $upcoming_screenings
   ),
   array(
       'heading' => 'Recent Blogs',
       'items' => $recent_blogs
   )
);
include(locate_template('partials/rhs-content.php'));
```

This will construct one box with a box heading of “Upcoming Screenings” followed by a box with a box heading of “Recent Blogs”. The `items` key in an inner array is used to denote the array of posts to display within that box. `partials/rhs-content.php` inspects the type of posts in the aforementioned array and looks for a corresponding `entry-`*`[post_type]`*`.php` file (located in `partials/rhs-entries`) that contains the code with the markup for that specific post type.
Using this approach, adding content to the RHS of a new page is simply a matter of writing the database query for the posts to be displayed and creating a `entry-`*`[post_type]`*`.php` that defines how to structure the content of each post within the sidebar box (and optionally adding a CSS file for custom styling).

The LHS content is produced in a similar fashion.