Web Technologies: Report

Jannick Hemelhof, Roberto Ristuccia, Youssef Boudiba

December 19, 2015

Contents

1 Introduction

Intro with some info about the project: basic idea, how we handled being in a group, etc.

2 Design

Design of our web app, maybe show a diagram or two

3 Handling of Requirements

3.1 User can register/log in

3.2 User has a profile

3.3 User can manipulate data

Look at/search/add and edit

3.4 Social aspect is needed

3.5 Usage of AJAX

3.6 Form validity

While thinking about handling this requirement our initial thoughts saw form validation as something that would happen on the server side of our application. Rereading the assignment gave some other insights and discovering the very neat validator.js[1] library showed us that client side validation could look good and was straightforward to implement.

- Server side: We have built in some checks that ensure a consistent status of our database: When registering a username needs to be unique and the required fields need to be filled out. The check for required fields is also present on the client side but since someone can manipulate requests to maybe remove some data after the client side check an insurance policy was needed on the server side. These checks for required fields are present all over the server side where data is being received.
- Client side: We added some basic checks: Required fields need to be filled out and, by using the HTML5 input fields, designated input fields need to have a value that corresponds with the type of field. When there is an error (e.g. a required field wasn't filled in) the field is marked and an error is shown explaining what the problem is. This is possible thank to the validator.js library.

3.7 CSS

3.8 HTML5 features

For this requirement we went ahead and implemented following HTML5 specific features:

- Local Storage: In order to remember the logged in user across different sessions we're using the local storage feature to place a token. That way we can check if there's a user logged in at the moment.
- Geolocation: A user who has just seen a UFO out in the field might be too shocked to remember where he is at the moment. By using the HTML Geolocation API we can just use the location of the device on use that data when submitting the sighting.

[1] Validator, for Bootstrap 3: `http://1000hz.github.io/bootstrap-validator/`

- Notification: Showing the user that an operation was successful would be possible by letting a custom form/modal pop up but we were aiming for a more native solution. By using the Notifications API (supported by Chrome, Firefox, and Safari) we can offer the user a confirmation that his request was handled in a native environment.
- Input fields/Validators: In our input forms we use some new HTML5 input types e.g. email but also the required attribute.

3.9 Web Services

3.10 Provide data

3.11 Responsive design

3.12 Google Maps

4 Conclusion