Web technologies

Code Challenge

GDC TT&L Solutions

Version	1.0.0
Last revised	02.11.15
Status	Done
Author	dschwank

Protection category: Internal

1 Introduction

This code challenge contains several web technology related tasks which should be solved in a specific amount of time. The focused web technologies / frameworks are

- HTML 5,
- CSS3,
- JavaScript and
- AngularJS.

The challenge is not only based on development / implementation tasks – it also contains several tasks concerning design, test and documentation. Especially the design task is used to identify that you as developer have even a mind for UI / UX design (appearance, animation, usability, ...). Furthermore it should show that you test your code in a sufficient way and document your code as much as possible, thus other developers would be able to work with your code too. Therefore the challenge is separated in the following four topics:

- Design
- Development
- Test
- Documentation

Furthermore the challenge consists of two groups of requirements – core and additional requirements. The core requirements are those which should be focused on. The additional requirements are nice to have, but they are not required to pass this challenge.

Please try to use as many standard tools / frameworks / modules as possible, to make it as easy, general and good as possible (e.g. bower, gulp, karma, ...).

The time limit for this challenge is *3 days*.

Have fun and enjoy the challenge!

2 Task

Goal of this code challenge is to write a notification service. The notification service should display the user a notification message (see 3.1), if something happened on the backend side and should give the user the ability to react on a message. Furthermore the notification service should be bundled into one module, thus it would be possible to include the service in any application you like.

2.1 Requirements

The requirements are separated into core and additional requirements. The core requirements are those requirements which you should focus on to pass this challenge. During the implementation of the requirements focus on “finish tasks as good as possible” and not on “start as many tasks as possible”. It is more important to make one requirement perfect, instead of starting all without finishing any of them.

The additional requirements are nice to have – if you still have a lot of time left, have a look at those requirements!

2.1.1 Core requirements

1. The notification service should display the notifications as overlay in front of the page.
2. The notifications shouldn’t interrupt the user – they should be displayed in an area where the user notices the messages, but does not lose the focus on his work.
3. The notifications should have a title and a body.
4. The notifications should have a different appearance based on the categories.
5. The notifications should be closable.
6. The notifications should be closed automatically after 90 seconds.
7. The notification service should display max 5 notifications at the same time.
8. If the max amount of notifications is reached, the notification service should combine the oldest notifications into one group, thus the max amount is satisfied again.

2.1.2 Additional requirements

1. The notification service should be able to read / receive data from the backend side according to 3.2.
2. The notification service should be able to display notifications with confirmations.
3. The notification service should be able to send the confirmation result to the backend according to 3.3.

2.2 Topics

The realization for this challenge is separated into four small topics which you should work on. You don’t have to care about the order, but you should be aware to work on each different topic.

- Design
- Development
- Test
- Documentation

2.2.1 Design

Create a draft / mockup with one of your favorite tools. The following questions should be answered within the mockup:

- Placement of the notifications?
- How should the notifications look like?
- How is the behavior of the notifications? What happens if a new notification is added, removed, closed, …
- How should the user interact with the notifications?

2.2.2 Development

Create a prototype based on *AngularJS, HTML5 and CSS3*. Write a simple SPA where the notification service is included and can be tested in. Therefore the SPA should have the following three input elements (or something similar), thus a new notification can be simulated:

- Input fields for notification header and body
- Selection for the category
- Button to simulate the notification

2.2.3 Test

Write unit tests to check the functionality of the notification service. It is not necessary to check the SPA itself – reduce the test cases to the notification service. Ensure that the *test coverage* is about *80%*.

2.2.4 Documentation

Document the resulting module / code with *ngDoc* as good as possible, thus other developers might be able to work on your code too.

Add the finally generated documentation to the result or give an introduction how to generate the documentation.

3 Backend / API simulation

3.1 Notification message

Each notification message consists of the following properties:

- **id**: The id of the notification message.
- **from**: Identifier of the emitter.
- **category**: Category of the notification message. Possible categories are “info”, “warning” and “error”.
- **type**: Type of the notification – Possible types are “note”, “ok_confirm” and “ok_cancel_confirm”.
- **header**: Header of the notification.
- **content**: Content of the notification.

Example:

```
{
    id: 1337,
    from: 'userManagement',
    category: 'info',
    header: 'Password expiration',
    content: 'Your password expires in the next 2 days, please
             change it using the user management interface.',
    type: 'note'
}
```

3.2 Notification request

The notifications can be requested using the following backend interface:

```
/api/notification/list
```

The backend responses with an array containing the notification objects (see 3.1).

Example:

```
[
    {id: …},
    …
]
```

3.3 Notification response

If the backend requested a confirmation / response, use the following backend interface to send the response:

```
/api/notification/confirm
```

The response should contain the following information:

- **id**: The id of the notification message.
- **from**: Identifier of the emitter
- **result**: Result of the confirmation. The result is a number value – either “1” for true / “ok” or “0” for false / cancel.

Example:

```
{
    id: 1337,
    from: 'userManagement',
    result: 1
}
```

4 Finalize the challenge

The result of the challenge should be a packed archive containing the wireframe, source code, test and generated documentation. It should be possible to open the SPA example directly opening the *index.html* file or using a simple *http-server*.

Furthermore add a description how to use your example and how to generate the documentation.

Change History / Release Notes

Version	Last revised	Author/Editor	Reason	Changes/Comments
1.0.0	02.11.2015	dschwank		Create “web” code challenge version 1.0.0

Table 1.1: Change history