

UI Blog Programming Test

Prerequisites

To take this test you will need the following:

- A computer with a web browser and Internet access; broadband speed is recommended
- A text editor or programming IDE of your choice, such as Notepad, Eclipse, GEdit, vi, or TextEdit. You are not expected to purchase anything to complete this test. Use the tools you already have, or can use or download freely. If you are unable to meet any of these prerequisites, you should ask your Salesforce.com contact as soon as possible about alternatives.

Test Evaluation Criteria

Your work will be examined and graded based upon these criteria:

1. Feature completeness (see the Test Objectives section below for the list of features)
2. No user apparent bugs (crashes, error dialog, broken UI, corrupted data, bad links, etc.)
3. No other bugs (memory leaks, console errors, faulty logic, invalid HTML or malformed JSON, etc.)
4. Architecture, patterns and original code design
5. Security
6. Efficiency
7. Organization
8. Code documentation
9. Error and exception handling
10. Performance and scalability
11. Maintainability
12. Browser compatibility (IE, Chrome, Firefox, Safari, Opera, etc.)*
13. Device compatibility (Desktop, Tablet and Phone)*
14. Platform compatibility (Windows, OSX, iOS, Linux, Android, etc.)*
15. Internationalization
16. Usability
17. Accessibility
18. Time to complete your work
19. Effective use of third party libraries
20. UI design aesthetics
21. Code formatting

*A Note on Compatibility Writing UI code that is compatible with different browsers, devices and platforms is an important skill at Salesforce.com.

We realize that not everybody has access to multiple test environments for this test; therefore, although you can test and submit your solution for any number of mainstream browsers, we only require you submit a solution for one of these:

Browser OS	Platform Browser	Version
MS Internet Explorer	Windows 7.0 or later	7.0 or later
Google Chrome	Windows, OSX, or Linux	Latest
Mozilla Firefox	Windows, OSX, or Linux	Latest
Apple Safari	OSX Latest	Opera
Windows	OSX or Linux	Latest
Google Chrome for Android	Android	Latest
Apple Mobile Safari	iOS	Latest

Please note your selected primary browser and OS in your comments so we can properly evaluate your solution. You can optionally test and demonstrate your knowledge with other browsers, devices, or platforms using multiple machines, side by side installs, VMs, device emulators, or cloud based free testing services such as these:

- <http://saucelabs.com>
- <http://crossbrowsertesting.com>
- <http://www.webpagetest.com/>

Supplying a solution tested in multiple clients is especially encouraged if you are applying for a senior position. Applicants for mobile focused teams may wish to solve for a mobile browser to demonstrate their domain expertise, but are not required to do so to pass this test. Please note in your comments any additional browsers, devices, and platforms you want us to grade, so we can give you credit for the additional work you put into your solution.

Alternatively, if you know your code would need to change to handle a particular client, but you are unable or unwilling to test it directly, we recommend you include comments in your code explaining the expected differences.

For example:

```
<canvas...><!-- this will not work in IE8 and earlier --></canvas>
```

Create, test and comment your solution

1. Your test is to build the web UI for a Blog. Detailed requirements are listed in the next section.
2. Build your solution on top of the provided structure. Please make sure to use blog.html as your landing page. You can create new files to help you solve the problem.
3. Be sure to include your name at the top of any source code files you create, and add comments throughout your code for clarity.
4. Your work should be contained entirely in static web resource files - HTML, JavaScript, CSS, and images.

Turn in your solution

When ready to turn in your completed test, use these steps:

1. Create a **readme.txt** file in your completed project with:
 - Your full name and email address
 - A list of which browsers you have tested (minimum of one)
 - A list of any libraries you used, and why
 - A list of your source code files, with brief descriptions
 - Any other notes on your solution to augment your code comments
 - Any test objectives you were not able to complete in time, and provide a description of why
2. Zip your test solution's **entire directory** contents into a zip file
3. Upload to HackerRank

Test Objectives

For this test you will be required to build a user interface for an online blog, utilizing an existing server-side API for storing and retrieving blog data, while demonstrating your JavaScript, HTML, and CSS abilities with respect to the Test Evaluation Criteria mentioned above.

At a minimum, your blog UI should allow users to:

1. Go to a **landing page** at `/blog.html`
2. **View**
 - a. a list of all existing blog posts
 - b. a specific blog post
3. **Create** a new blog post
4. **Edit** a specific blog post
5. **Delete**
 - a. a specific blog post
 - b. all blog posts

Please comment your solution in-line for added context during its evaluation. If you have any project-level comments to add, place them in a `readme.txt` file in your project's root directory.

Most of your blog functionality will be expected to be fully implemented for a successful test. Others that might require server side changes or API changes should be noted as comments, with the best possible client solution given the existing server code implemented.

For example near your [Delete all blog posts] implementation you might include a comment such as:

/ The server should require authentication and limit deletion to just posts by the current user or in the case of an administrator, to posts by a user with a given user ID */*

The exact design of the interface for the blog is up to you, but be aware that we will consider the following aspects of your response, in this order:

- **functionality**: does the blog function according to spec?
- **performance**: do user actions finish in a reasonable amount of time?
- **style**: is the user interface intuitive and appealing?

You are free to download any JavaScript, CSS, or HTML toolkits you would like to use in your project.

If you run into any connectivity issues with the API, please ask your Salesforce.com contact to help you figure out the problem.

API Overview

For the purposes of this test, a blog post contains the following fields:

- **id**: a unique id
- **title**: the post title
- **text**: the post contents
- **timestamp**: the time at which the post was originally created

You will interact with the blog server via a REST API that will be provided to you by your Salesforce.com contact person.

The responses you receive will vary based on the HTTP request type and the parameters in the URL or request body.

All responses return a HTTP 200 (OK) status code is successful, and a HTTP 400 (BAD REQUEST) status code and an exception message if an error occurs.

API Methods

- GET `/api/`
 - Returns all blog posts in JSON format. Posts are ordered from most to least recent.
 - Example response:
 - [{"id":1278002762936,"text":"Thisisthecontentofmyblogpost.", "timestamp":"ThuJul0109:46:02PST2010", "title":"TheTitleofMyBlogPost"}, {"id":1277858967163,"text":"Deardiary,todayImetafishwhocouldtalk.", "timestamp":"TueJun2917:49:27PST2010", "title":"AWeirdThingHappened..."}]
- GET `/api/{id}`
 - Returns a single blog post in JSON format, corresponding to the passed in ID.
 - Example response:
 - {"id":1277858967163,"text":"Deardiary,todayImetafishwhocouldtalk.", "timestamp":"TueJun2917:49:27PST2010", "title":"AWeirdThingHappened..."}
- POST `/api/`

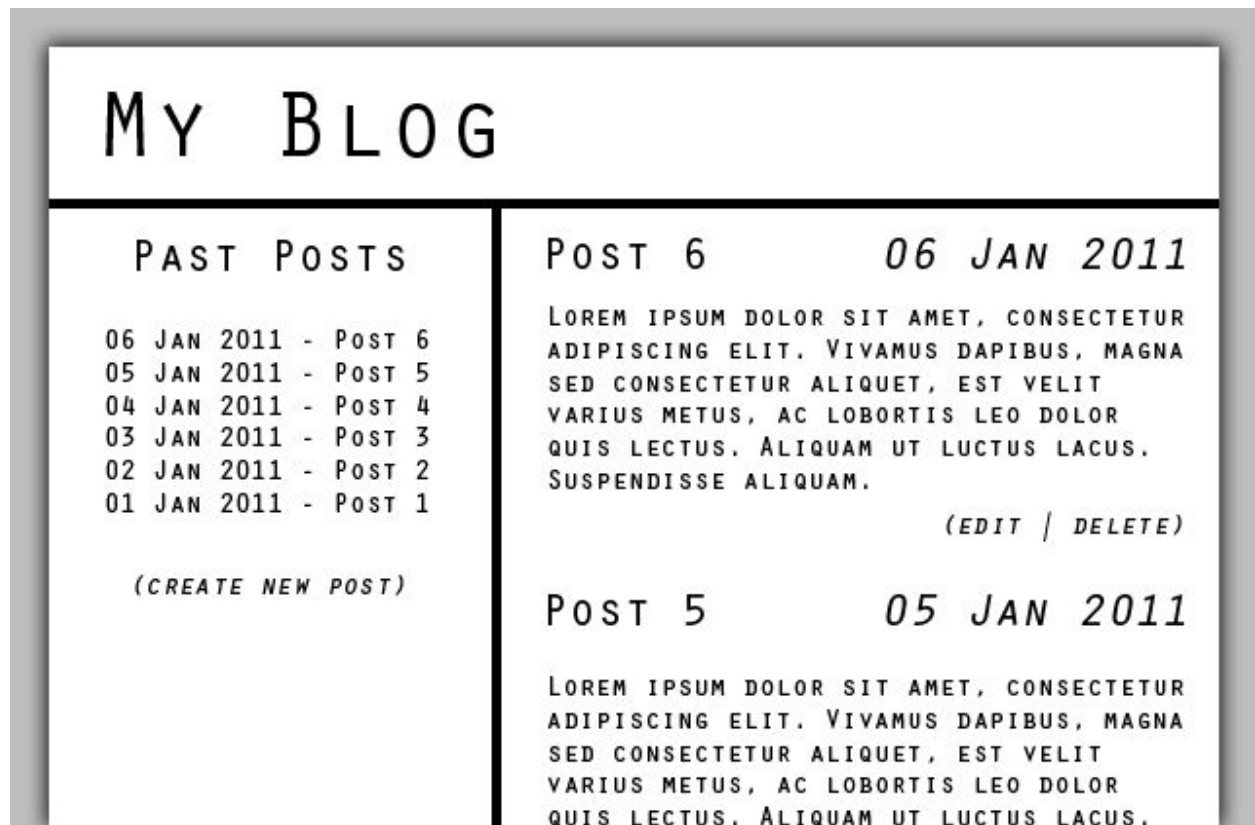
- Creates a new blog post
 - Must include "text" and "title" parameters in the payload of the request
 - Returns the post that was created
- POST */ {name} /api / {id}*
 - Updates an existing blog post, corresponding to the passed in ID.
 - Must include "text" and "title" parameters in the post data of the request.
 - Updating a post does not change its chronological ordering.
 - Returns the updated post
- DELETE */ {name} /api /*
 - Deletes all the posts in the blog
- DELETE */ {name} /api / {id}*
 - Delete the blog post that matches the provided ID
- GET */ {name} /api /generateSampleData*
 - Loads sample data into the blog so you can display and delete as you see fit without having to continually recreate your own content

Layout Examples

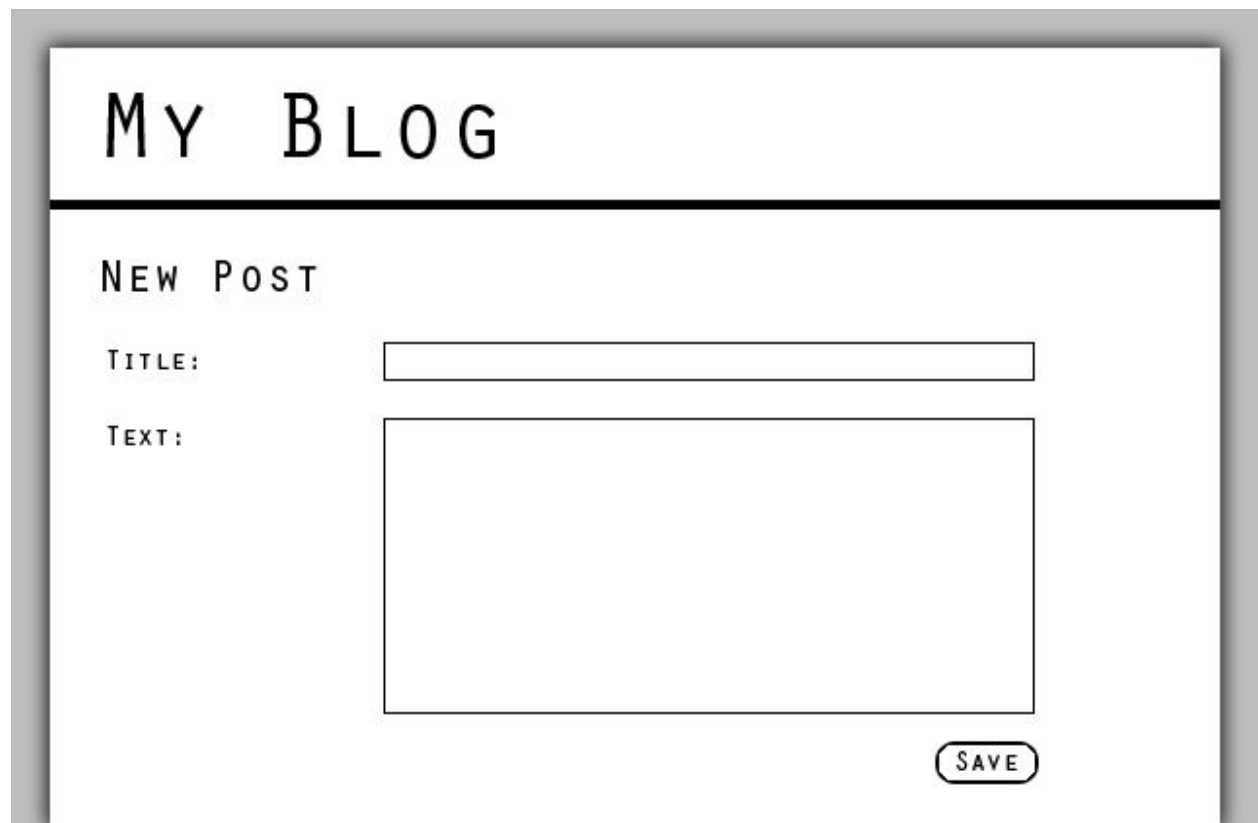
The following three images are screenshots of a very simple blog layout. Feel free either to work toward these designs, seek inspiration from public Blog sites, or to invent your own designs. Your blog can be either a multiple page based application, or a single page Ajax based application, or a mix of both.

You should code with readability, usability, accessibility, web standards, and best practices in mind, and be prepared to talk about your design decisions in the debrief following the test.

Sample Blog Landing Page Screen



Sample Blog Post Create Screen



A wireframe of a blog post creation screen. The screen has a header area with the text "MY BLOG". Below the header is a horizontal line. Underneath the line is the section title "NEW POST". To the left of the form fields are the labels "TITLE:" and "TEXT:". To the right of "TITLE:" is a single-line text input field. To the right of "TEXT:" is a larger, multi-line text area. At the bottom right of the form is a rounded rectangular button labeled "SAVE". The entire form is enclosed in a light gray border.

MY BLOG

NEW POST

TITLE:

TEXT:

SAVE

Sample Blog Post Edit Screen

MY BLOG

EDITING POST

TITLE:

TEXT:

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR
ADIPISCING ELIT. VIVAMUS DAPIBUS, MAGNA
SED CONSECTETUR ALIQUET, EST VELIT
VARIUS METUS, AC LOBORTIS LEO DOLOR
QUIS LECTUS. ALIQUAM UT LUCTUS LACUS.
SUSPENDISSE ALIQUAM, TORTOR IN ACCUMSAN
CONVALLIS, QUAM NUNC EUISMOD RISUS, SED
IMPERDIET TELLUS LEO MALESUADA ANTE. UT
SEM TURPIS, LAOREET ID RUTRUM EU.