Web Developer

From Matchi Wiki

Contents

Web and Front-end developer Interview Questions and Discussion Points

Interface and User Experience

- The guiding principle of UX design is: *Don't make users think*. Explain.
- Don't display unfriendly errors directly to the user.
- We don't post email addresses in plain text because they will get spammed to death.
- Build well-considered limits into your site (this also enhances security).
- Don't forget to take accessibility into account. It's always a good idea and in certain circumstances it's a legal requirement. WAI-ARIA and WCAG 2 are good resources in this area.
- Why don't we use links that say "click here"? You're wasting an SEO opportunity and it makes things harder for people with screen readers. It is a bad practice from 2 decades ago. If your call-to-action is that unclear that you need to say "click here", then you have a serious UX design issue.
- What are the most popular browsers in use at the moment?
- How do we ensure that our site works reasonably well across all major browsers? At a minimum test against a recent Gecko engine (Firefox), a WebKit engine (Safari and some mobile browsers), Chrome,

your supported IE browsers (take advantage of the Application Compatibility VPC Images), and Opera. Also test how the same browser renders on a different operating systems. Why do we need to test for this? Because different browsers implement standards inconsistently.
- How do we deal with different screen form factors such as smartphones, tablets, etc.? Why, we use Bootstrap CSS of course!

Security

- Essential reading:
  - OWASP development guide: https://www.owasp.org/index.php/OWASP_Secure_Coding_Practices_-_Quick_Reference_Guide
  - The Google Browser Security Handbook
  - The Web Application Hacker's Handbook
- You should never trust user input, nor anything else that comes in the request, which includes cookies and hidden form field values. Why not?
- Hashes
  - What is a hash?
  - Why should one salt a hash? Explain why a *rainbow attack* can be prevented by salting hashes.
  - Why is a slow hashing algorithm such as bcrypt (time tested) or scrypt (even stronger, but newer) preferred for storing passwords compared to and MD5 os SHA hash?
  - How is a hash used for password verification?
- Encryption
  - What is symmetric encryption? Give an everyday example of where this is used.
  - What is asymmetric encryption?
- Why should you not try to come up with your own authentication system?
- What is an SSL certificate?
  - Who issues them?
  - What are the main benefits of using an SSL certificate?
  - What are a downside of using SSL?
- What are following, and how does one prevent them:
  - SQL injection? Add a malicious SQL command to the URL
  - Session hijacking
  - Cross site scripting (XSS)? The user trusts the site, yet the site sends malicious command to the user
  - Cross site request forgeries (CSRF)? The site trusts the user, yet the user sends malicious commands to the site
  - Clickjacking? Malicious UI code on a site intercepts user actions
- Public/Private encryption:
  - How does that this work?
  - Who holds which keys?
  - How does PPK authentication compare to one-way password authentication?
- Are you informed about the latest attack techniques and vulnerabilities affecting PHP, Linux, Joomla, MySQL?
- What is *The principal of least privilege*?
- What is *non-repudiability* in the context of, say, sending a document to someone? How does one establish non-repudiability in a case like this?
- What is *two-factor authentication*? Give an example of everyday-use of this?

Data and Databases

- Understand relational database concepts and third normal form data modelling
- Code data selects, updates and inserts
- Code stored procedures, views and triggers
- Be able to manipulate / munge data en masse using one-liners

MySQL Database Administration

- How does one inspect all the queries
- How does one detect slow qeries?

Graphics

- Be proficient in one of the many graphics packages to produce simple graphics when required
- Have an appreciation of the artistic aspects of website creation.
- Appreciate that the consultant graphic artist - if he/she is any good - has the final say on the look and feel of the website.
- How does one compress the size of graphic files without noticeably reducing the images quality?
- Understand when to use what type of graphic file formats, and why that is.

Performance

- Implement cacheing if necessary, understand multiple levels of cacheing in the technical architecture. Also know about Content Delivery Networks (CDN) such as CloudFlare and how a CDN can help with performance , bandwidth reduction and server loading.
- Optimize images - how does one do this? Who does one do an entire directory of images? When does one use PNGs and when does one use JPEG-encoded images?
- Combine/concatenate multiple stylesheets or multiple script files to reduce number of browser connections and improve gzip ability to compress duplications between files.
- Know and use some of these performance-measuring oper.yahoo.com/performance
  - The YSlow tool
  - Google's Page Speed tool
  - WebpageTest tool at http://webpagetest.org
- Use CSS Image Sprites for small related images like toolbars and icons. Specifically, we use Font Awesome and (at last resort) Bootstrap's Glyph Icons.
- Domain Sharding
  - What is this? Busy web sites should consider splitting components across domains, e.g. static content such as images, CSS, JavaScript
  - What constraint exist for a site sharding solution with respect to cookies? Only content that doesn't need access to cookies should go in a separate domain.
  - What other alternatives to site sharding are there?
    - Content Delivery Network (CDN)
    - Webserver load balancing
- Minimize the total number of HTTP requests required for a browser to render the page.
- JavaScript minification
  - Why do we do this and what does a minified JS file look like?
  - What minification tools are there?
  - What are the alternatives to minification JS files yourself?

SEO (Search Engine Optimization)

- Name 5 very popular search engines
- What is a "search engine friendly" URL?
- What is an XML sitemap? What is its default location? /sitemap.xml. What creates such a file?
- When you have multiple URLs that point to the same content, there is a risk of confusing search engines. How can that be avoided?
  - Use <link rel="canonical" ... /> on the relevant pages
  - Address it in Google Webmaster Tools and Bing Webmaster Tools
- What does the robots.txt file do? Does it control all web spiders?
- What is a 301 Redirect request and when do I use it?
  - A 301 redirect request prevents splitting the search engine ranking between multiple sites.
- There can be badly-behaved spiders. How do I keep them off my site? What is a well-known, badly-behaved web spider?

- .htaccess file
- Apache config
- Webserver config using iptables blacklisting
- Network-edge router blacklisting

Web Technology

- Why do we use different file formats for images?
- HTTP Protocol:
  - WHat is a GET vs. a POST?
  - What are cookies
  - What does it mean when we say the HTTP protocol is *stateless*?
- What is the order in which JavaScript, style sheets, and other resources such as graphics are processed by a typical browser? How do you arrange these resources for optimal perceived performance? Generally, Java Scripts are at the bottom of the page, except for analytics-related scripts.

Issue and Code Management

- Work with an issue management system (we use JIRA): Take instructions from it and update activities by return to it
- Be experienced using a version control system such as Subversion, GIT, MS-Teamview, MS-SourceSafe (VSS), MKS, CVS. Should be able to the following
  - Check code in
  - View file change history
  - Restore code to a given point in time
  - Fork/branch code, merge branches
  - Manage multiple development sandboxes
  - Align coding activities with an issue management system

Coding, Bug fixing and Testing

- You may well spend 20% of your time coding and 80% time maintaining the code, so code accordingly.
- We practise *egoless programming*. Understand that the code belongs to everybody/nobody. Don't attach emotional ownership to code, just enjoy the ride.
- Be confident using an IDE like like Zend, Komodo or Eclipse. It saves on coding time and helps improve code quality and keeps team-coding consistent. It also speeds up debugging, to unit test code, and to efficiently debug errors.
- Document how the application works for future support staff and people performing maintenance.
- Make frequent backups of your own local development environment. Use a DEV branch of the version control system for this if that helps.
- Logging: We need to have sufficient logging in place using frameworks such as log4php, log4perl, log4net etc. so that if something goes wrong on the production site, we have a chance of better understanding the cause. What are *exceptions*?
- Test-driven coding: What is that?
  - Test as you code. Catch and remedy issues early before you code yourself into a corner.
- What is unit testing?
  - Write code so that chunks of it can be easily unit tested. This may mean that one has write separate test harnesses or add commented-out test code in the actual code-base of how to test a chunk or module of code.
  - As a developer you are responsible for doing at least the first round of unit testing of your coded effort before releasing it for integration.
- Coding style:
  - Adhere to common and up-to-date coding standards and paradigms
  - Code defensively: What does this mean?

Deployment and Progressive Enhancement

- Pre-production Staging and testing of fixes: How does one deploy updates in a controlled way without affecting current users or breaking data or architecture.
- Any experience using a continuous integration tool such as Jenson/Jenkins/Hudson, Ant, NAnt, etc.?

Solution Designing and Critical thinking

Simple Web Application Walk-through

First Round Design

Design a high-level web application showing block diagrams of data tables and workflows processes and any known framework, that does the following:

- Serves the entire humanity of users
- A user can self-register on the web application, and choose what type of member he/she wants to be
- A user can log in after having completed the registration process
- The user can view certain content, depending on what type of member the user chose to at registration time

Questions:

- How do we ensure that real people register and prevent spambots from registering?
- What authentication mechanism do we use? What other options are available?
- How are authentication credentials securely communicated between user and web application?

Next design iteration

The user needs to be able to upload images. The user can mark images as publically visible, in which case Search Engines have access to them. Other images can be marked a private, in which case we need to prevent search engines from accessing those imaged. How do we do this? You also need to consider:

- UX issues
- Storage issues and Scalability
- Performance issues

Simple Application Integration Walk-through

A process needs to scale uploaded images into thumbnails. Given the immense high volume of images that are constantly begin uploaded,

Possible approaches:

- File inotify process
- Message-queued architecture
- Image resizing service on a separate server

Design evaluation

The design's quality is measured in terms of:

- Elegance and simplicity
- Extensibility

- Scalability
- Performance
- Implementation feasibility

Fault Finding

- A web application that I find is still running but users are getting a "connection refused" error when they try to log in. How do I diagnose the problem?
  - Check that the application is actually running using `top/ps/etc...`
  - And more!

Back-end scripting

Simple one-liner

This is the content of /etc/hosts:

```
$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4 mapp1 mapp01
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6 mapp1 mapp01
```

- What does this line of code do?

```
nice_hostname=$(grep "127.0.0.1" /etc/hosts | cut -f2- -d" " | sed -e 's/\s/\n/g' | grep -v "^\s*$" | awk '{print length($
```

Code snippet

This is what the file /var/log/matchi looks like:

```
[2016/04/14 00:00:04][FATAL][notification_dispatch_daemon.sh][MERGE] [234] SQL Error on mdb01.merge_20160405: select event
[2016/04/14 00:00:08][FATAL][notification_dispatch_daemon.sh][TEST] [234] SQL Error on mdb01.test_20160229: select event_i
[2016/04/14 00:00:10][FATAL][notification_dispatch_daemon.sh][PROD] [234] SQL Error on mdb01.build_20151105: select event_
```

- What does this (long) line of code do?

```
SLACKBOT_CHANNEL=system_log
SLACKBOT_TOKEN=xxxxxxxxxxxxxxxxxxxxxx
SLACKBOT_TEAM=matchi-innovation

tail --follow=name /var/log/matchi -s 16 | \
  sed -e "s/^/Server ${nice_hostname}:/" -u | \
  grep --line-buffered -E '\[(ERROR|FATAL|SECURITY)\]' | \
  xargs -I '{}' curl -d '{}' $"https://${SLACKBOT_TEAM}.slack.com/services/hooks/slackbot?token=${SLACKBOT_TOKEN}&channel=
  1>/dev/null 2>&1  &
```

- What is the *1>/dev/null* for?
- What is the *2>&1* for?
- What is the trailing *&* for?
- What benefit do I get if I put a *nohup* in front of this line?

System Administration

Log file management

You see the following log files:

```
root@s16972619 mdb01 /var/log/mysql # ls -al
total 141292
drwxr-xr-x  2 mysql mysql      4096 Jun 17  2015 .
drwxr-xr-x 10 root  root      12288 Apr 17 04:03 ..
-rw-rw----  1 mysql mysql 142118921 Apr 18 00:00 general.log
-rw-r-----  1 mysql mysql   2161666 Apr 18 04:04 mysqld.log
-rw-rw----  1 mysql mysql    225299 Jun 17  2015 slow.log
```

What is wrong here?

- Massive log file that starts from the Pre-Cambrian era and holds redundant data
- There is no limitation on how large the log files can get, nor housekeeping to truncate old data

How do we fix this elegantly?

- Set up a log-rotate configuration for each of these files

Database Skills

TODO

Infrastructure Architecture

TODO