L&R Uddannelse EGMONT

How To Make An Awesome QR Code

By Ryan Bayron, Intern WrightIMC and the folks at SystemID
If you're reading this, it's probably safe to assume you've seen a QR code or two. You may have wondered if it's possible to make your own, or even to take it one step further and make something artistic out of it. Keep reading, my friend.

If you've been to websites like QR Arts, you've seen people do some crazy things with QR codes. Companies like Marc Jacobs and SEATprove that you can completely cover whole sections of the code with other imagery without ruining the code's scan-ability. How does this work? How do they know what they can cover up and still make a working QR code?

First Things First
Before diving in and modifying your QR code with your brand logo, slogan, or a picture of your cat, you need to know what you can't mess with.

In any QR code, a square of either black or white is called a "module." The modules are grouped together into different sections. In this example, the "no-zones" are highlighted in various colors. These are groups of modules that must remain intact if the code is to be scanned.

- The 3 red squares that look like targets define the edges of the code.
- The smaller red square is a position marker. This is a reference point for the scanner, to make sure everything lines up. In larger codes, there are several of these, arranged in a grid pattern.
- The red strips of alternating black and white modules are called timing patterns, and define the horizontal and vertical orientations of the rows and columns of modules.
- The green strips define the format. This tells the scanner whether it's a URL, plain text, Kanji, numbers, or a combination of these.
- The blue segments define the version. The word "version" is simply used to denote the size of the code. The more modules, the higher the version. This only needs to be defined in versions 7 and up. Anything smaller and it automatically detects the version.

The Fun Part
Everything that's not highlighted in the previous example is where the actual message is stored. The modules are sectioned into groups of 8, and each group represents one character in the message. These groups (which I will call "bytes" from now on) are arranged in a systematic, puzzle-like fashion:
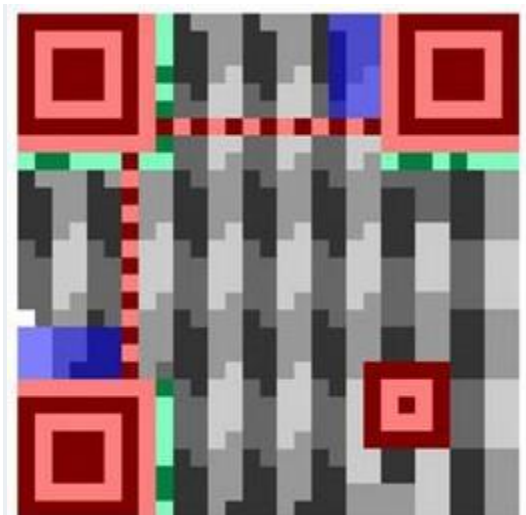
Thanks to the Reed Solomon error correction method, each QR code is designed to include built-in error correction modules. This means that depending on how much free space you have in your code, you can cover up anywhere from 7% to 30% of your code with no detrimental effects on its readability.

Now, what do I mean by "free space?" The example above is a version 3 code, containing 70 bytes, but only 23 alphanumeric characters. That means only 33% of the bytes in the code are actually assigned a value. According to the Version and Maximum Data Capacity Table on Denso-Wave's website, that puts this code at a Level H correction level. Here's what I mean:

- Level L – You can alter up to 7% of the code and still scan it properly. Another way to look at this is out of all the possible bytes allowed by your QR code's version, your message takes up 93% of them.
- Level M – You can alter up to 15% of the code (Your message takes up 85% of the available space in the version).
- Level Q – You can alter up to 25% of the code without damaging it.
- Level H – You can alter as much as 30% of your code.

Now that you know a little bit about the anatomy of a QR code, it's time to dig deeper and look at the behavior of the code.

Generating Your Code
There are several QR code generators out there and here is a list of the best of them - QR Code Generators Online. Most of them are simple: you enter in the text/URL that you want people to see when they scan it, submit your message, and they'll either give you the code immediately or e-mail it to you, depending upon the generator you choose.

One thing to note here is that your code's version will be automatically determined by the length of your message. For example,*http://www.I-love-brand-new-carpet.com/products/shag-carpet/gallery.html* is going to generate a much larger code than*http://www.systemid.com*. The version of your code depends on the sum of A) the number of characters in your message and B) however many extra bytes it needs to have the appropriate level of error correction.

Getting Your Hands Dirty
This is the part where graphic editing software and an experimental mindset come in handy. Once you've got the code, and you've figured out its correction level, it's just a matter of playing around with the bytes. The PDF in the resources list at the bottom is a great source of information on how to tell exactly where each byte lies. In the example below, I isolated a group of bytes in the very center of the code. This was my playground.

First, I rearranged the modules within the red border, and left everything else alone. It still scanned correctly. The next step was to completely eliminate everything within the border (note that the border's shape was determined by selecting a group of whole bytes). After removing all of those modules, it still scanned with

no problems. Last but not least, I replaced the red region with a portrait of one of the most influential figures for geeks everywhere.

It still scanned.

The Original

Scrambled Modules

Completely Erased

Live Long and Prosper

Go Get 'Em

It's just a matter of figuring out what you can and can't do. Once you figure out what you want to do with your QR code, do it, and then scan to make sure it works. If not, go back and figure out why – did you cover up too many bytes? Is your design overlapping one of the "no-zones?" If you add your own design work and it still scans, add more until it doesn't. Before you know it, you'll be turning a giant QR code into an infographic about QR codes. Kind of like this:

Læs videre i følgende artikel fra American Mathematical Sociaty,
af Bill Casselman, University of British Columbia, Vancouver, Canada
How to Read QR Symbols Without Your Mobile Telephone