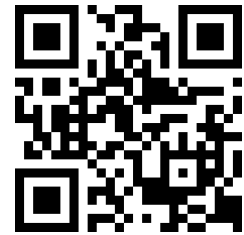




# QR Codes



## Inhalt

1 Einleitung .....	2
2 Aufbau von QR Codes .....	2
3 Codierung des Alphabets .....	5
4 Fehlerkorrektur .....	6
Was ist Fehlerkorrektur? .....	6
Fehlerkorrektur bei QR Codes.....	7
Hamming-Code .....	9
5 Maskierung.....	12
6 Gefahren von QR Codes.....	15
Anhang A: ISO-8859-1.....	16
ISO-8859-1 Tabelle.....	16
Umrechnung Hexadezimalsystem <-> Binärsystem.....	16
Anwendung .....	18
Lösungen .....	19
Quellen.....	23

# 1 Einleitung

Sie haben bestimmt schon viele QR Codes angetroffen. Diese praktischen Bildchen erlauben es, schnell auf Informationen zuzugreifen und werden zum Beispiel auf Werbeplakaten oder Flyern eingesetzt. Doch wie funktioniert das Prinzip? Was steckt dahinter? Das werden Sie in diesem Skript lernen.

QR Code steht für *Quick Response Code*, weil er schnell ausgelesen und verarbeitet werden kann. Solche Codes werden zum Speichern digitaler Daten auf Papier (oder ähnlichen Medien) benutzt. Die Daten werden also in analoger Form gespeichert. Ein Handy (oder ein anderer Barcodeleser) kann dann diese Daten wieder auslesen, digitalisieren und verarbeiten.

QR Codes wurden erstmals 1994 genutzt, erfunden von der Firma Denso, die eine Tochtergesellschaft von Toyota ist. Diese Firma hat die Codes eingesetzt, um die Produktion von Autobauteilen zu automatisieren und zu kontrollieren. Seit 2000 sind QR Codes ein ISO-Standard. Der Standard wurde einige Male erweitert, weshalb es verschiedene Versionen von QR Codes gibt.

## 2 Aufbau von QR Codes

### Aufgabe 1:

Betrachten Sie die beiden QR Codes im Titel des Skriptes und dazu diese fünf:



- a) Welche Gemeinsamkeiten gibt es (rein optisch) zwischen den sieben Codes?
- b) Worin unterscheiden sich die Codes?
- c) Ihnen sind sicher die Quadrate in drei der vier Ecken aufgefallen. Wozu könnten diese dienen? Warum gibt es sie nicht in allen vier Ecken?

(Die Lösungen zu allen Aufgaben finden Sie gegen Ende des Skriptes, ab Seite 19)

Ein QR Code ist immer ein Quadrat, aufgebaut aus sogenannten *Modulen*. Jedes Modul kann entweder weiss oder schwarz gefärbt sein. Ausserdem sind immer die folgenden Bereiche vorhanden:



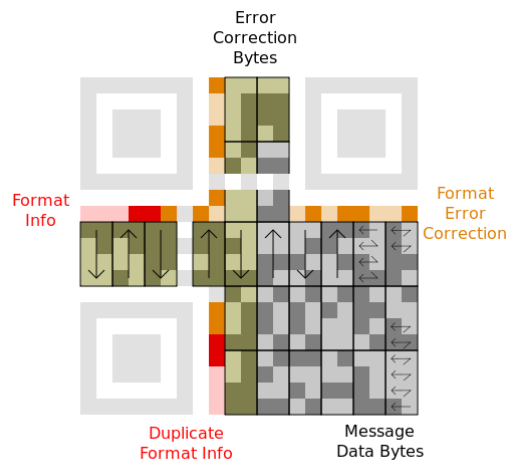
- Die roten Bereiche (*Position detection patterns*) sind dazu da, die Begrenzung und die Orientierung des Codes anzugeben.
- Der Bereich, der direkt an die roten Bereiche angrenzt, wird leer gelassen, damit die roten Bereiche gut sichtbar sind (hier grün markiert).
- Mit zunehmender Grösse (*Version*) des Codes werden weitere Muster (*Alignment patterns*, hier dunkelblau) hinzugefügt, um die Ausrichtung des Codes besser erkennbar zu machen.

Ein QR Code kann verschiedene Grössen haben. Version 1 besteht aus 21x21 Modulen, Version 2 aus 25x25 Modulen, Version 3 aus 29x29 Modulen usw. Die grösste Version, Version 40, besteht aus 177x177 Modulen. Pro Versionsschritt werden immer 4 Module in der Seitenlänge addiert.

- Die *Timing patterns* (hier gelb) helfen dem Decoder, die Lage der einzelnen Module zu bestimmen. Sie geben sozusagen die Zeilen und Spalten vor. Sie bestehen abwechselnd aus schwarzen und weissen Modulen.
- In den pinken Bereichen wird (für grössere QR Codes) die Versionsnummer angegeben, d.h. es wird angegeben, wie gross der Code ist. Die beiden Bereiche enthalten zweimal dieselbe Information – für den Fall dass der eine Bereich nicht gelesen werden kann.
- Im hellblauen Bereich ist Information über das Format des Codes gespeichert. Diese Informationen geben wichtige Details an, z.B. welche Maske verwendet wurde und welche Fehlerkorrektur angewandt wurde. Zu diesen Details folgt später mehr.

Wie die Versions-Information ist auch die Format-Information doppelt vorhanden. Einmal im Gegenuhrzeigersinn um das obere linke *Position detection pattern*, das zweite Mal zuerst links unten aufwärts, dann rechts oben von links nach rechts.

Der Rest des QR-Codes besteht aus den Daten, also z.B. dem Text, welche den eigentlichen Inhalt ausmachen. Das "Gerüst" wird wie folgt befüllt:



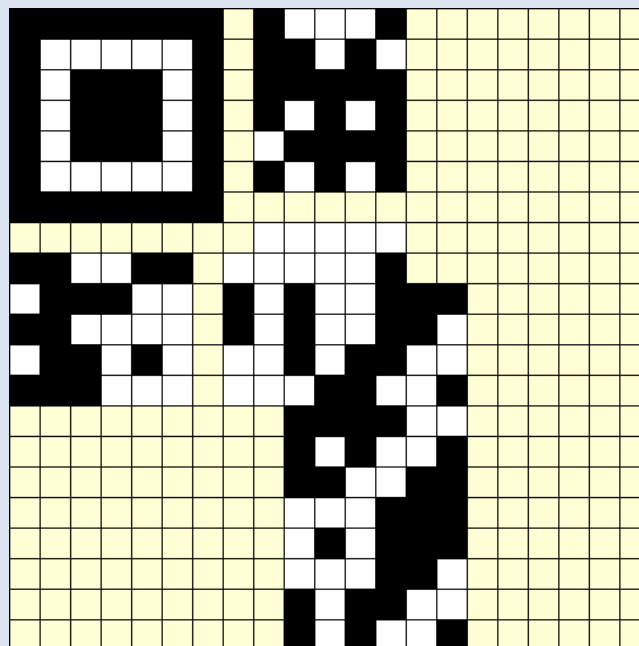
Quelle: [http://upload.wikimedia.org/wikipedia/commons/thumb/1/1e/QR\\_Code\\_Unmasked.svg/528px-QR\\_Code\\_Unmasked.svg.png](http://upload.wikimedia.org/wikipedia/commons/thumb/1/1e/QR_Code_Unmasked.svg/528px-QR_Code_Unmasked.svg.png)

Beginnend bei der unteren rechten Ecke werden die Daten (eine Folge von 0 und 1) eingetragen. Zuerst kommen (in grau) die eigentlichen Daten, danach (in grün) die Bits, welche für die Fehlerkorrektur zuständig sind. Falls am Schluss noch Platz übrig ist, wird dieser einfach leer gelassen.

### Aufgabe 2:

Hier ist ein QR Code, der noch nicht fertig gestellt ist. Füllen Sie die hellgelb markierten Bereiche richtig auf. Die fehlenden Bits für den Anfang in der rechten unteren Ecke lauten:

1000 0011 1000 0110 1000 0101 1100 0001 1010 0001 1101 0110



### 3 Codierung des Alphabets

Am Anfang jedes QR Codes steht eine Information (d.h. ein Text / eine URL / eine Zahl), die dargestellt werden soll. Wie wird diese Information in Nullen und Einsen umgewandelt, so dass man dies im QR Code eintragen kann?

Abhängig davon, welche Zeichen im Text vorkommen, kann man den Text mit verschiedenen Zeichensätzen codieren:

- **Zahlen (0-9)**  
Kommen nur Ziffern vor, kann ein QR Code theoretisch bis 7089 Zeichen speichern
- **Alphanumerisch (0-9A-Z \$%\*+-./:)**  
Kommen nur Grossbuchstaben, Zahlen und 9 vorgegebene Sonderzeichen vor, können bis zu 4296 Zeichen gespeichert werden
- **ISO-8859-1**  
Wenn der Text nur aus Zeichen besteht, die in ISO-8859-1 vorkommen (das sind unter anderem Gross- und Kleinbuchstaben, Ziffern, viele Satzzeichen und kombinierte Buchstaben für den westeuropäischen Sprachraum), können bis zu 2953 Zeichen codiert werden
- **KANJI**  
Wenn der Text nur aus KANJI (japanische Schriftzeichen) besteht, können maximal 1817 Zeichen gespeichert werden.

Nachdem der passende Zeichensatz bestimmt ist, werden die folgenden Informationen in die Bitfolge geschrieben:

1. **Die Kennnummer des Zeichensatzes**  
Jeder der vier Zeichensätze hat eine eindeutige Nummer, sozusagen der Name des Zeichensatzes
2. **Die Anzahl der Zeichen, die der Text hat**  
Damit man weiss, wie viel Text jetzt kommt
3. **Der Text selbst**
4. **Die Ende-Kennung; sie ist immer 0000**  
Wenn man 0000 liest, heisst das, dass der Text jetzt fertig ist

#### ***Beispiel "Märchenbuch":***

1. **Kennnummer:**  
Um den Text "Märchenbuch" zu codieren, wird zuerst der passende Zeichensatz ausgewählt. In diesem Fall ist das ISO-8859-1. Dieser Zeichensatz hat die Kennnummer **0100**.
2. **Anzahl Zeichen:**  
Im zweiten Schritt wird gezählt, wie viele Zeichen der Text enthält. In diesem Fall sind es 11 Zeichen. Bei kleinen QR-Codes wird diese Zahl mit 8 Bit codiert - bei grösseren QR-Codes würde die Zahl mit 16 Bit codiert werden. Die Zahl 11 soll also im Binärsystem geschrieben werden und zwar mit 8 Stellen (8 Bits). Das ergibt **0000 1011**.

### 3. Der Text selbst:

Anschliessend werden die einzelnen Zeichen codiert. Für ISO-8859-1 ist im Anhang A auf Seite 16 angegeben, wie diese Codierung erfolgt. Das "M" wird zu **0100 1101**, das "ä" zu **1110 0100** und so weiter.

### 4. Ende-Kennung:

Den Abschluss bildet die Ende-Kennung. Sie ist immer **0000**, ausser wenn nicht mehr genug Platz ist. Dann wird sie abgekürzt.

Insgesamt wird aus dem Text "Märchenbuch" somit diese Bitfolge:

- 0100
- 0000 1011
- 0100 1101 1110 0100 0111 0010 0110 0011 0110 1000 0110 0101 0110 1110 0110 0010 0111 0101 0110 0011 0110 1000
- 0000

#### **Aufgabe 3:**

Codieren Sie den Text "Kanti Wil" analog zum Beispiel Märchenbuch. Verwenden Sie dazu auch ISO-8859-1. Hinweis: Den Leerschlag finden Sie in der Tabelle unter "NBSP"

## 4 Fehlerkorrektur

Sie wissen bereits, wie aus gegebenen Informationen eine Bitfolge hergestellt wird und wie diese danach in einen QR Code eingetragen wird. Der nächste Schritt ist die Fehlerkorrektur.

### Was ist Fehlerkorrektur?

#### **Aufgabe 4:**

Lassen Sie sich vom Lehrer / der Lehrerin den Zaubertrick mit den Kärtchen demonstrieren.

- a) Wie funktioniert der Trick?
- b) Funktioniert er auch noch, wenn zwei Kärtchen umgedreht werden?

#### **Was steckt hinter dem Trick?**

Die zwei zusätzlichen Zeilen und Spalten dienen der "Fehlerkorrektur". Mit diesen zwei zusätzlichen Reihen wird sichergestellt, dass im ursprünglichen Bild eine gewisse Bedingung erfüllt ist (in jeder Zeile und Spalte eine gerade Anzahl schwarze Kärtchen).

Sobald im geänderten Bild diese Bedingung an einer Stelle nicht mehr erfüllt ist, muss ein "Fehler", d.h. eine Veränderung vorliegen. Weil man sehen kann, in welcher Zeile und in welcher Spalte der Fehler liegt, lässt sich das gedrehte Kärtchen eindeutig bestimmen. Die Fehlerkorrektur sorgt also nicht nur

dafür, dass man feststellen kann, dass irgendetwas falsch ist, sondern sie erlaubt es auch, den Fehler genau zu lokalisieren und damit zu korrigieren.

Wenn Bitfolgen von Nullen und Einsen übertragen werden sollen, dann passieren immer ein paar Übertragungsfehler. Deshalb werden in solchen Fällen zusätzliche Fehlerkorrekturbits angehängt oder eingefügt, um gewisse Bedingungen zu erfüllen, ähnlich wie bei den Kärtchen.

Wird die Bitfolge übertragen, kann der Empfänger jederzeit überprüfen, ob die Bedingungen erfüllt sind. Wenn nicht, ist bei der Übertragung etwas schief gelaufen. Mit Hilfe der Fehlerkorrektur kann der Empfänger sogar genau bestimmen, welches Bit falsch empfangen worden ist und es berichtigen.


## Fehlerkorrektur bei QR Codes




Bei QR Codes sind solche fehlerkorrigierenden Codes sehr wichtig. Da sich die QR Codes meistens auf Papier befinden, können Verschmutzungen vorkommen oder es kann sein, dass das Papier an einer Stelle zerknittert ist. Dann wird der QR Code vom Barcodeleser falsch gelesen und es könnte zu Fehlern kommen.

### Aufgabe 5:

Die folgenden vier QR Codes enthalten alle dieselbe Information. Sie besitzen allerdings verschiedene "Fehlerkorrekturlevels". Das Fehlerkorrekturlevel gibt an, wie viel Fehlerkorrektur zusätzlich zu der eigentlichen Information im QR Code vorhanden ist.

- Falls Sie auf Ihrem Handy noch keine App zum Lesen von QR Codes haben, dann laden Sie sich jetzt eine herunter. Es gibt viele solche Gratis-Apps, z.B. "Barcode Scanner" für Android oder "NeoReader" für iPhone. (Falls Sie mit dem NeoReader arbeiten, schalten Sie in den Einstellungen "Scan bestätigen" ein).
- Untersuchen Sie die unten stehenden Codes. Wie viel Prozent des Codes kann man abdecken / unkenntlich machen / zerstören / ..., so dass die App die im Code enthaltenen Daten noch lesen kann? Schätzen Sie für jeden Code eine Prozentzahl. Sie bekommen die Codes auch auf einem separaten Blatt, etwas grösser, damit Sie gut damit arbeiten können.

QR Code	Fehlerkorrekturlevel	Toleriert ... % Fehler
	M	

QR Code	Fehlerkorrekturlevel	Toleriert ... % Fehler
	H	
	L	
	Q	

Bei Level H kann bis zu 30% des QR Codes verschmutzt oder abgedeckt sein und die Information wird trotzdem noch erkannt. Dank dieser Tatsache können so genannte *Design QR Codes*, die mit Logos oder Bildern versehen sind, immer noch gelesen werden.

Beispiele für Design QR Codes (von <http://www.qrcode-generator.de/design>):





Neben den "verdeckten" Bereichen sind bei Design QR Codes zusätzlich oft die Ecken abgerundet und es werden verschiedene Farben verwendet. Allerdings können gewisse Barcodescanner Probleme mit dem Lesen bekommen, wenn der Code zu fest "verschönert" wird.

Doch wie funktioniert diese Fehlerkorrektur bei QR Codes? Die Korrektur basiert auf dem Solomon-Reed Algorithmus, der leider sehr fortgeschrittene mathematische Kenntnisse erfordert, wenn man ihn verstehen will. Weil dies den Rahmen des Ergänzungsfachs sprengen würde, schauen wir uns ein einfacheres Verfahren zu Fehlerkorrektur an, das zwar nicht bei QR Codes aber sonst in verschiedenen Bereichen der Informatik verwendet wird.

## Hamming-Code

Der Hamming-Code ist ein Verfahren zur Fehlerkorrektur, das von Richard Hamming (1915-1998) entwickelt worden ist. Mit ihm können Bitfolgen so modifiziert werden, dass bei einer Übertragung ein falsches Bit erkannt und korrigiert werden kann.

### *Hamming -Code anhand eines Beispiels:*

#### **Codieren:**

Daten, die codiert werden sollen:

0110100010011

#### **1. Platz für Kontrollbits machen**

An allen Stellen, die Zweierpotenzen sind, wird Platz für ein Kontrollbit gemacht:

      0    110    1000100    11

(Hier sind an den Stellen 1, 2, 4, 8, 16 Kontrollbits)

#### **2. Erstes Kontrollbit**

Für das erste Kontrollbit wird immer ein Bit der Folge angeschaut, dann eines ausgelassen, dann wieder eines angeschaut... Beginn ist beim ersten Kontrollbit (d.h. Bit 1).

      0    110    1000100    11

In den markierten Bits muss eine gerade Anzahl 1 vorkommen. Hier haben wir vier 1 in den markierten Bits, d.h. das Kontrollbit muss eine 0 sein. Somit haben wir:

0    0    110    1000100    11

#### **3. Zweites Kontrollbit**

Für das zweite Kontrollbit werden immer zwei Bits angeschaut, dann zwei ausgelassen usw. Beginn ist beim zweiten Kontrollbit (d.h. Bit 2).

0    0    110    1000100    11

In den markierten Bits muss eine gerade Anzahl 1 vorkommen. Hier haben wir zwei 1 in den markierten Bits, d.h. das Kontrollbit muss eine 0 sein. Somit haben wir:

000    110    1000100    11

#### 4. *Drittes Kontrollbit*

Beginn ist bei Bit 4 (drittes Kontrollbit) und es werden immer 4 Bits angeschaut, dann 4 ausgelassen usw.

000\_110\_1000100\_11

In den markierten Bits muss eine gerade Anzahl 1 vorkommen. Hier haben wir drei 1 in den markierten Bits, d.h. das Kontrollbit muss eine 1 sein. Somit haben wir:

0001110\_1000100\_11

#### 5. *Viertes Kontrollbit*

Beginn ist bei Bit 8 (viertes Kontrollbit) und es werden immer 8 Bits angeschaut, dann 8 ausgelassen usw.

0001110\_1000100\_11

In den markierten Bits muss eine gerade Anzahl 1 vorkommen. Hier haben wir zwei 1 in den markierten Bits, d.h. das Kontrollbit muss eine 0 sein. Somit haben wir:

000111001000100\_11

#### 6. *Fünftes Kontrollbit*

Beginn ist bei Bit 16 (fünftes Kontrollbit) und es werden immer 16 Bits angeschaut, dann 16 ausgelassen usw.

000111001000100\_11

In den markierten Bits muss eine gerade Anzahl 1 vorkommen. Hier haben wir zwei 1 in den markierten Datenbits, d.h. das Kontrollbit muss eine 0 sein. Somit haben wir:

000111001000100011

Et voilà: Die Codierung ist fertig!

#### **Merke:**

Beim Kontrollbit an der Stelle n: Beginne bei Bit n und markiere n Bits, lasse n Bits weg usw. Die Anzahl 1 muss immer gerade sein.

### **Überprüfen:**

Zu überprüfende Bitfolge:

010010001000101101

#### 1. *Erstes Kontrollbit überprüfen*

Für das erste Kontrollbit wird immer ein Bit der Daten angeschaut, dann eines ausgelassen, dann wieder eines angeschaut... Beginn ist beim ersten Kontrollbit (d.h. Bit 1).

010010001000101101

Ziel: Eine gerade Anzahl 1. Hier haben wir vier davon, also ist dieses Kontrollbit ok.

#### 2. *Zweites Kontrollbit überprüfen*

Für das zweite Kontrollbit werden immer zwei Bits angeschaut, dann zwei ausgelassen usw. Beginn ist beim zweiten Kontrollbit (d.h. Bit 2).

0**1**001**00**01**00**01**01**10**1**

Ziel: Eine gerade Anzahl 1. Hier haben wir drei davon, also ist dieses Kontrollbit nicht ok.

### 3. *Drittes Kontrollbit überprüfen*

Beginn ist bei Bit 4 (drittes Kontrollbit) und es werden immer 4 Bits angeschaut, dann 4 ausgelassen usw.

010**0100**0100**0101**101

Ziel: Eine gerade Anzahl 1. Hier haben wir drei davon, also ist dieses Kontrollbit nicht ok.

### 4. *Viertes Kontrollbit überprüfen*

Beginn ist bei Bit 8 (viertes Kontrollbit) und es werden immer 8 Bits angeschaut, dann 8 ausgelassen usw.

0100100**01000101**101

Ziel: Eine gerade Anzahl 1. Hier haben wir drei davon, also ist dieses Kontrollbit nicht ok.

### 5. *Fünftes Kontrollbit überprüfen*

Beginn ist bei Bit 16 (fünftes Kontrollbit) und es werden immer 16 Bits angeschaut, dann 16 ausgelassen usw.

010010001000101**101**

Ziel: Eine gerade Anzahl 1. Hier haben wir zwei davon, also ist dieses Kontrollbit ok.

### 6. *Falsche Kontrollbits addieren*

Die falschen Kontrollbits sind die Bits 2, 4 und 8. Die Summe davon ist  $2 + 4 + 8 = 14$ , d.h. bei der vorliegenden Zeichenfolge ist das 14. Bit falsch. Mathe-Magie! :-)

0100100010001**0**1101 falsches Bit

0100100010001**1**1101 so wäre es richtig

#### **Merke:**

Beim Kontrollbit an der Stelle n: Beginne bei Bit n und markiere n Bits, lasse n Bits weg usw. Die Anzahl 1 muss gerade sein, sonst ist das Kontrollbit falsch.

Addiere alle falschen Kontrollbits um das gekippte Bit zu erhalten.

#### **Aufgabe 6:**

- Codieren Sie die Bitfolge 1001011010111001 mit dem Hamming-Code.
- Sie empfangen die Folge 11100001110010101010. Wurde sie richtig übertragen? Wenn nein: Welches Bit ist falsch?

#### **Wie funktioniert der Hamming-Code?**

Das erste Kontrollbit schaut auf die Stellen 1, 3, 5, 7, 9, ... Schreibt man diese Zahlen um als Summen von Zweierpotenzen, so sind es die Stellen 1,  $2+1$ ,  $4+1$ ,  $4+2+1$ ,  $8+1$ , ... also alle Stellen, die in dieser Summendarstellung "+1" enthalten.

**Aufgabe 7:**

Füllen Sie die Tabelle aus. Geben Sie jeweils an, welche Stellen das Kontrollbit überprüft und wie die Stellennummer als Summe von Zweierpotenzen geschrieben wird.

1. Kontrollbit		2. Kontrollbit		3. Kontrollbit		4. Kontrollbit	
Stelle	Summe	Stelle	Summe	Stelle	Summe	Stelle	Summe
1	1						
3	2+1						
5	4+1						
7	4+2+1						
9	8+1						
Alle Stellen, die folgendes enthalten: +1							

Wenn das erste Kontrollbit (Stelle 1) also falsch ist, muss der Fehler an einer Stelle sein, deren Summe "+1" enthält. Wenn das zweite Kontrollbit (Stelle 2) falsch ist, muss der Fehler an einer Stelle sein, die "+2" enthält, und so weiter.

Erhält man als falsche Kontrollbits die Stellen 1, 2 und 8, so muss die fehlerhafte Stellennummer "+1", "+2" und "+8" enthalten – und sonst nichts, denn sonst wäre ein weiteres Kontrollbit falsch! Die falsche Stelle ist also  $1 + 2 + 8 = 11$ .

## 5 Maskierung

Nun wissen Sie, wie man den gesamten QR Code mit den schwarzen und weissen Modulen ausfüllt. Den Anfang machen die Datenbits, danach folgen Fehlerkorrekturbits. Was jetzt noch passieren kann, ist dass es sehr grosse zusammenhängende schwarze oder weisse Flächen gibt.

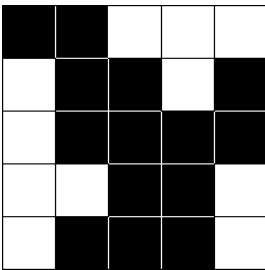
Wenn Sie sich fertige QR Codes anschauen, dann stellen Sie fest, dass es selten grosse einfarbige Flächen gibt – meist sind die Codes sehr "unruhig". Beim letzten Schritt der QR Code-Erstellung, der sogenannten Maskierung, wird sichergestellt, dass genau dies der Fall ist.

**Aufgabe 8:**

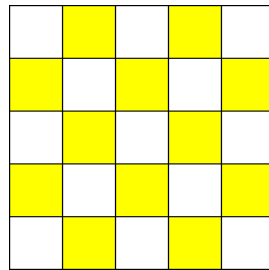
Warum macht man diesen letzten Schritt? Was könnte theoretisch passieren, wenn ein QR Code einfach mit den erhaltenen Daten und Fehlerkorrekturbits gefüllt wird?

Bei der Maskierung wird eine Art "Folie" über den QR Code gelegt. Auf der Folie sind Bereiche markiert, in welchen die Module des QR Codes umgedreht werden sollen, d.h. aus schwarz wird weiss und umgekehrt.

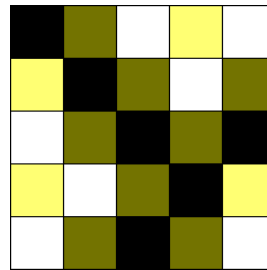
**Beispiel:**



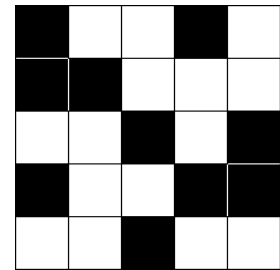
In diesem QR Codes gibt es grosse einfarbige Flächen



Diese Maske wird über den Code gelegt. Ein weisses Feld bedeutet, dass der Code so belassen wird, gelb heisst, die Farbe wird an dieser Stelle geändert



Der ursprüngliche Code plus die darübergelegte Maske

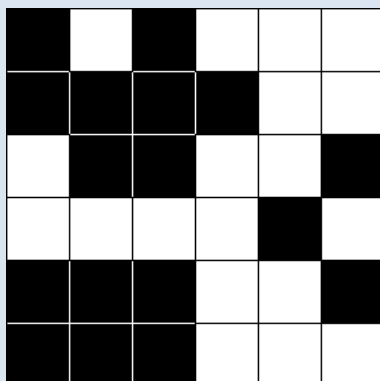


Im Endresultat gibt es viel weniger einfarbige Flächen

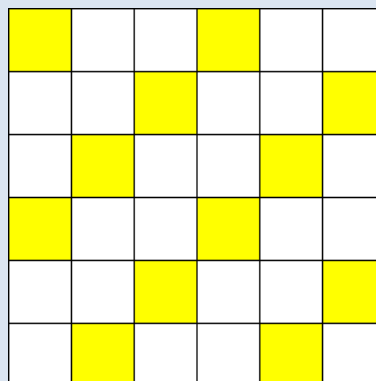
**Aufgabe 9:**

Maskieren Sie den gegebenen QR Code mit der Maske:

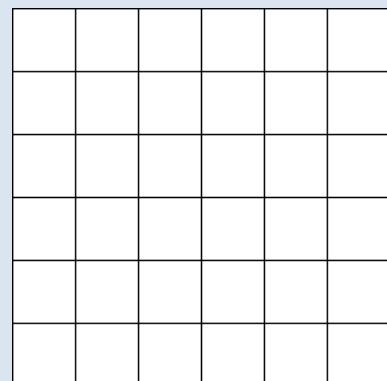
**Code**



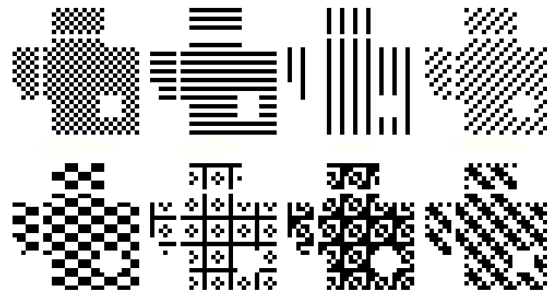
**Maske**



**Lösung**



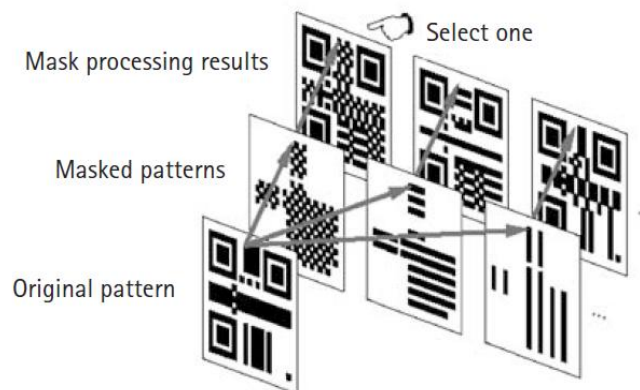
Für QR Codes gibt es acht verschiedene Masken:



Quelle: <http://research.swtch.com/qart>

Die Masken bestehen jeweils aus einem Muster, das immer wiederholt wird. Man sieht bei den obigen Masken, dass wichtige Bereiche wie die *Position Detection Patterns* in den Ecken, die *Alignment Patterns* und natürlich auch Versions- und Formatinformationen nicht maskiert werden. Denn bei diesen Informationen ist es zwingend, dass sie unverändert und sofort lesbar sind.

Wenn ein QR Code fertig ausgefüllt ist, werden alle dieser acht Masken darauf angewendet. Dies ergibt acht verschiedene fertige QR Codes. Von diesen acht wird derjenige ausgewählt, der am wenigsten "störende Eigenschaften" aufweist. Diese "störenden Eigenschaften" sind im Standard für QR Codes festgelegt. Für die acht möglichen Codes wird anhand von vorgegebenen Kriterien ausgerechnet, wie "schlecht" der Code ist. Derjenige QR Code mit den wenigsten "Strafpunkten" wird als Endresultat gewählt.



[http://qrbcn.com/imatgesbloc/Three\\_QR\\_Code.pdf](http://qrbcn.com/imatgesbloc/Three_QR_Code.pdf)

Damit ist der QR Code endgültig fertig und kann von Anwendern auf der ganzen Welt gescannt werden.

## 6 Gefahren von QR Codes

QR Codes sind praktisch. Man kann damit schnell auf eine Webseite verlinken, Nachrichten austauschen, Kontaktdaten verschicken etc. Doch wie so vieles heutzutage bieten sie auch Angriffspunkte für Hacker.

### **Aufgabe 10:**

- a) Recherchieren Sie im Internet. Welche Gefahren können von QR Codes ausgehen?
- b) Was kann ein Hacker im schlimmsten Fall mit den Daten auf Ihrem Handy anfangen? Was wäre für Sie das Schlimmste?
- c) Für wie gefährlich halten Sie QR Codes? Wie realistisch ist ein Angriff?
- d) Wie kann man sich gegen die Gefahren schützen?

## Anhang A: ISO-8859-1

### ISO-8859-1 Tabelle

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	<i>nicht belegt</i>															
1...																
2...	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{		}	~	
8...	<i>nicht belegt</i>															
9...																
A...	NBSP	ı	ø	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
B...	°	±	²	³	´	µ	¶	·	,	¹	º	»	¼	½	¾	¿
C...	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D...	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E...	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F...	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Quelle: [http://de.wikipedia.org/wiki/ISO\\_8859-1](http://de.wikipedia.org/wiki/ISO_8859-1)

### Umrechnung Hexadezimalsystem <-> Binärsystem

Dec	Hex	Oct	Binaire	Dec	Hex	Oct	Binaire	Dec	Hex	Oct	Binaire	Dec	Hex	Oct	Binaire
0	0	000	00000000	16	10	020	00010000	32	20	040	00100000	48	30	060	00110000
1	1	001	00000001	17	11	021	00010001	33	21	041	00100001	49	31	061	00110001
2	2	002	00000010	18	12	022	00010010	34	22	042	00100010	50	32	062	00110010
3	3	003	00000011	19	13	023	00010011	35	23	043	00100011	51	33	063	00110011
4	4	004	00000100	20	14	024	00010100	36	24	044	00100100	52	34	064	00110100
5	5	005	00000101	21	15	025	00010101	37	25	045	00100101	53	35	065	00110101
6	6	006	00000110	22	16	026	00010110	38	26	046	00100110	54	36	066	00110110
7	7	007	00000111	23	17	027	00010111	39	27	047	00100111	55	37	067	00110111
8	8	010	00001000	24	18	030	00011000	40	28	050	00101000	56	38	070	00111000
9	9	011	00001001	25	19	031	00011001	41	29	051	00101001	57	39	071	00111001
10	A	012	00001010	26	1A	032	00011010	42	2A	052	00101010	58	3A	072	00111010
11	B	013	00001011	27	1B	033	00011011	43	2B	053	00101011	59	3B	073	00111011
12	C	014	00001100	28	1C	034	00011100	44	2C	054	00101100	60	3C	074	00111100
13	D	015	00001101	29	1D	035	00011101	45	2D	055	00101101	61	3D	075	00111101



14	E	016	00001110	30	1E	036	00011110	46	2E	056	00101110	62	3E	076	00111110
15	F	017	00001111	31	1F	037	00011111	47	2F	057	00101111	63	3F	077	00111111
<b>Dec</b>	<b>Hex</b>	<b>Oct</b>	<b>Binaire</b>	<b>Dec</b>	<b>Hex</b>	<b>Oct</b>	<b>Binaire</b>	<b>Dec</b>	<b>Hex</b>	<b>Oct</b>	<b>Binaire</b>	<b>Dec</b>	<b>Hex</b>	<b>Oct</b>	<b>Binaire</b>
64	40	100	01000000	80	50	120	01010000	96	60	140	01100000	112	70	160	01110000
65	41	101	01000001	81	51	121	01010001	97	61	141	01100001	113	71	161	01110001
66	42	102	01000010	82	52	122	01010010	98	62	142	01100010	114	72	162	01110010
67	43	103	01000011	83	53	123	01010011	99	63	143	01100011	115	73	163	01110011
68	44	104	01000100	84	54	124	01010100	100	64	144	01100100	116	74	164	01110100
69	45	105	01000101	85	55	125	01010101	101	65	145	01100101	117	75	165	01110101
70	46	106	01000110	86	56	126	01010110	102	66	146	01100110	118	76	166	01110110
71	47	107	01000111	87	57	127	01010111	103	67	147	01100111	119	77	167	01110111
72	48	110	01001000	88	58	130	01011000	104	68	150	01101000	120	78	170	01111000
73	49	111	01001001	89	59	131	01011001	105	69	151	01101001	121	79	171	01111001
74	4A	112	01001010	90	5A	132	01011010	106	6A	152	01101010	122	7A	172	01111010
75	4B	113	01001011	91	5B	133	01011011	107	6B	153	01101011	123	7B	173	01111011
76	4C	114	01001100	92	5C	134	01011100	108	6C	154	01101100	124	7C	174	01111100
77	4D	115	01001101	93	5D	135	01011101	109	6D	155	01101101	125	7D	175	01111101
78	4E	116	01001110	94	5E	136	01011110	110	6E	156	01101110	126	7E	176	01111110
79	4F	117	01001111	95	5F	137	01011111	111	6F	157	01101111	127	7F	177	01111111
<b>Dec</b>	<b>Hex</b>	<b>Oct</b>	<b>Binaire</b>	<b>Dec</b>	<b>Hex</b>	<b>Oct</b>	<b>Binaire</b>	<b>Dec</b>	<b>Hex</b>	<b>Oct</b>	<b>Binaire</b>	<b>Dec</b>	<b>Hex</b>	<b>Oct</b>	<b>Binaire</b>
128	80	200	10000000	144	90	220	10010000	160	A0	240	10100000	176	B0	260	10110000
129	81	201	10000001	145	91	221	10010001	161	A1	241	10100001	177	B1	261	10110001
130	82	202	10000010	146	92	222	10010010	162	A2	242	10100010	178	B2	262	10110010
131	83	203	10000011	147	93	223	10010011	163	A3	243	10100011	179	B3	263	10110011
132	84	204	10000100	148	94	224	10010100	164	A4	244	10100100	180	B4	264	10110100
133	85	205	10000101	149	95	225	10010101	165	A5	245	10100101	181	B5	265	10110101
134	86	206	10000110	150	96	226	10010110	166	A6	246	10100110	182	B6	266	10110110
135	87	207	10000111	151	97	227	10010111	167	A7	247	10100111	183	B7	267	10110111
136	88	210	10001000	152	98	230	10011000	168	A8	250	10101000	184	B8	270	10111000
137	89	211	10001001	153	99	231	10011001	169	A9	251	10101001	185	B9	271	10111001
138	8A	212	10001010	154	9A	232	10011010	170	AA	252	10101010	186	BA	272	10111010
139	8B	213	10001011	155	9B	233	10011011	171	AB	253	10101011	187	BB	273	10111011
140	8C	214	10001100	156	9C	234	10011100	172	AC	254	10101100	188	BC	274	10111100
141	8D	215	10001101	157	9D	235	10011101	173	AD	255	10101101	189	BD	275	10111101
142	8E	216	10001110	158	9E	236	10011110	174	AE	256	10101110	190	BE	276	10111110
143	8F	217	10001111	159	9F	237	10011111	175	AF	257	10101111	191	BF	277	10111111
<b>Dec</b>	<b>Hex</b>	<b>Oct</b>	<b>Binaire</b>	<b>Dec</b>	<b>Hex</b>	<b>Oct</b>	<b>Binaire</b>	<b>Dec</b>	<b>Hex</b>	<b>Oct</b>	<b>Binaire</b>	<b>Dec</b>	<b>Hex</b>	<b>Oct</b>	<b>Binaire</b>
192	C0	300	11000000	208	D0	320	11010000	224	E0	340	11100000	240	F0	360	11110000
193	C1	301	11000001	209	D1	321	11010001	225	E1	341	11100001	241	F1	361	11110001
194	C2	302	11000010	210	D2	322	11010010	226	E2	342	11100010	242	F2	362	11110010
195	C3	303	11000011	211	D3	323	11010011	227	E3	343	11100011	243	F3	363	11110011
196	C4	304	11000100	212	D4	324	11010100	228	E4	344	11100100	244	F4	364	11110100
197	C5	305	11000101	213	D5	325	11010101	229	E5	345	11100101	245	F5	365	11110101
198	C6	306	11000110	214	D6	326	11010110	230	E6	346	11100110	246	F6	366	11110110
199	C7	307	11000111	215	D7	327	11010111	231	E7	347	11100111	247	F7	367	11110111
200	C8	310	11001000	216	D8	330	11011000	232	E8	350	11101000	248	F8	370	11111000
201	C9	311	11001001	217	D9	331	11011001	233	E9	351	11101001	249	F9	371	11111001
202	CA	312	11001010	218	DA	332	11011010	234	EA	352	11101010	250	FA	372	11111010
203	CB	313	11001011	219	DB	333	11011011	235	EB	353	11101011	251	FB	373	11111011
204	CC	314	11001100	220	DC	334	11011100	236	EC	354	11101100	252	FC	374	11111100
205	CD	315	11001101	221	DD	335	11011101	237	ED	355	11101101	253	FD	375	11111101
206	CE	316	11001110	222	DE	336	11011110	238	EE	356	11101110	254	FE	376	11111110
207	CF	317	11001111	223	DF	337	11011111	239	EF	357	11101111	255	FF	377	11111111

## Anwendung

Gesucht: Binäre Bitfolge, die das Zeichen "B" in ISO 8859-1 codiert.

Vorgehen:

- Das Zeichen in der ISO-Tabelle suchen:  
Der Code dafür ist 42 (hexadezimal)
- Den hexadezimalen Code mit der Umrechnungstabelle in Binärcode umwandeln:  
42 Hex entspricht 01000010 Binär

Die Codierung von B ist also 01000010.



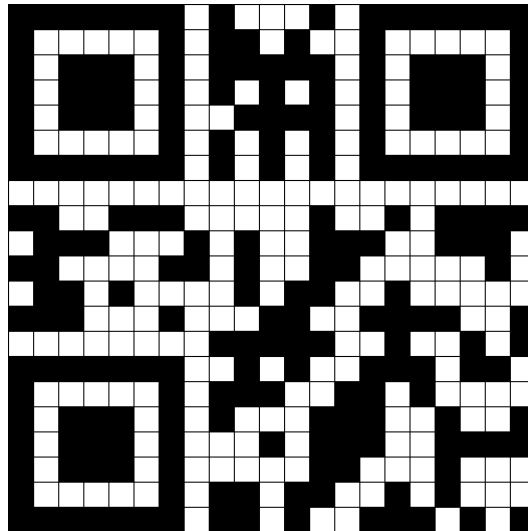
# Lösungen



## Aufgabe 1:

- Gleiches Muster in drei Ecken (oben links und rechts, unten links), bei grösseren Codes eine kleine Version dieses Quadrates in regelmässigen Abständen.
- In der Grösse (Anzahl "Module"), bei grösseren Codes gibt es mehr von den kleinen Quadraten
- Die so genannten "Position Detection Patterns" sind dazu da, die Begrenzung des QR Codes zu erfassen. Es gibt sie nur in drei Ecken, damit die Orientierung des Codes immer klar ist, auch wenn man ihn auf dem Kopf oder gedreht einscannt.

## Aufgabe 2:



## Aufgabe 3:

- 0100
- 0000 1000
- 0100 1011 0110 0001 0110 1110 0111 0100 0110 1001 1010 0000 0101 0111 0110 1001 0110 1100
- 0000

**Aufgabe 4:**

- a) Der Lehrer hat sichergestellt, dass in jeder Zeile und in jeder Spalte eine gerade Anzahl  $x$  vorkommt. Wird ein Kärtchen vertauscht, dann sorgt das dafür, dass in der betreffenden Zeile und der betreffenden Spalte die Anzahl  $x$  ungerade wird. Damit kann nach der Vertauschung schnell festgestellt werden, welches das umgedrehte Kärtchen ist.
- b) Ja, wenn zwei Kärtchen aus verschiedenen Zeilen und Spalten umgedreht werden. Wenn es zwei Kärtchen aus der gleichen Zeile sind, dann kann man im Nachhinein nur feststellen, dass es a) zwei Kärtchen aus einer Zeile waren und b) in welchen Spalten sie liegen. Die betreffende Zeile lässt sich aber nicht angeben.

**Aufgabe 5:**

Fehlerkorrekturlevel	Toleriert ... % Fehler
M	15%
H	30%
L	7%
Q	25%

**Aufgabe 6:**

- a) 101000100110101111001
- b) Die Kontrollbits 1, 2 und 8 sind falsch, d.h. das 11. Bit müsste eine 1 sein statt einer 0.

### Aufgabe 7:

1. Kontrollbit		2. Kontrollbit		3. Kontrollbit		4. Kontrollbit	
Stelle	Summe	Stelle	Summe	Stelle	Summe	Stelle	Summe
1	1	2	2	4	4	8	8
3	2+1	3	2+1	5	4+1	9	8+1
5	4+1	6	4+2	6	4+2	10	8+2
7	4+2+1	7	4+2+1	7	4+2+1	11	8+2+1
9	8+1	10	8+2	12	8+4	12	8+4
Alle Stellen, die folgendes enthalten: +1		Alle Stellen, die folgendes enthalten: +2		Alle Stellen, die folgendes enthalten: +4		Alle Stellen, die folgendes enthalten: +8	

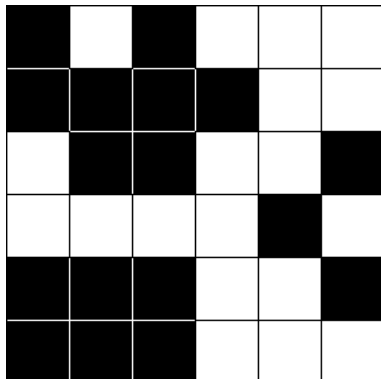
### Aufgabe 8:

Die Maskierung sorgt dafür, dass bestimmte Muster, wie das Position Detection Pattern, in Datenmodulen unterdrückt wird. Dadurch wird eine schnelle Erkennung des Symbols garantiert.

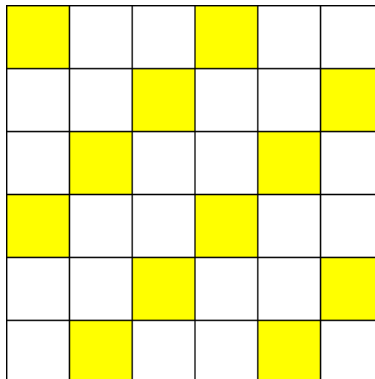
Es könnte sonst passieren, dass die Daten- oder Fehlerkorrekturbits im Code per Zufall genau so angeordnet sind, dass fälschlicherweise ein Muster entsteht, das dem Alignment Pattern oder dem Position Detection Pattern entspricht. Dann hätte ein Decoder Mühe, den Code richtig zu lesen.

### Aufgabe 9:

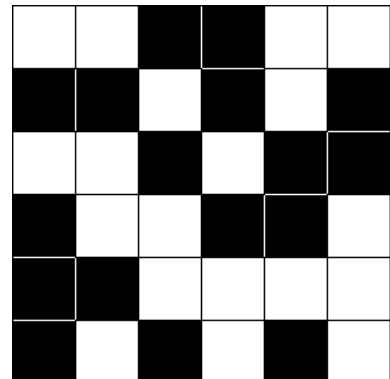
Code



Maske



Lösung



### **Aufgabe 10:**

- a) Wenn man mit seinem Handy einen QR Code scannt, kann dieser einen Link auf eine Webseite enthalten. Je nachdem, wie die App eingestellt ist, führt sie einen eventuell direkt auf die Webseite – ohne Nachfrage. Theoretisch kann von dieser Website aus Schadsoftware auf das Handy geladen werden, ohne dass der Benutzer es merkt. Diese läuft dann in Zukunft im Hintergrund mit und kann Daten aufzeichnen etc.
- b) Mögliche Szenarien sind
- Auslesen von Passwörtern
  - Daten werden hoch- und runtergeladen, was zu einer höheren Handyrechnung führen kann
  - Teure Telefonnummern können angerufen werden
  - Die Kontakte können ausgelesen werden. Einige Leute sind ziemlich naiv und speichern ihre Passwörter dort...
  - "Der Feind" kann GPS-Daten auslesen, d.h. er weiss, wo man sich befindet
  - Die Fotos auf dem Handy können ausgelesen werden
  - Der gescannte Link kann auf eine Phishing-Seite führen. Die sieht vielleicht aus wie eine ganz normale Seite (paypal, ebay,...), ist aber ein Fake und nur dazu da, Passwörter auszulesen.
  - Wenn es ein Firmenhandy ist, dann freut sich der Hacker natürlich umso mehr, wenn er Zugangsdaten und Passwörter bekommt...
  - ...
- c) Wenn ein russischer Hacker mich damit angreifen will, muss er zuerst in die Schweiz kommen und dort einen QR Code an irgendeine Bahnhofswand oder so kleben (oder einfacher: übers Internet). Diese muss ich dann scannen und ihr blind vertrauen. Dann muss ich auch noch genau das richtige Betriebssystem auf dem Handy haben, damit der Angriff erfolgreich ist. Es gibt Angriffe in dieser Form, sie sind auch schon erfolgreich gewesen, aber es gibt deutlich gefährlichere Dinge, die man im Internet anstellen kann.
- d)
- Man sollte seine Handy-App so einstellen, dass sie nicht direkt auf eine Website führt
  - Sehr grosse QR Codes sollten einem suspekt vorkommen.
  - Bei Android-Smartphones: Besser keine Apps via QR Code installieren! Das könnte irgendeine suspekte App sein!
  - Auch auf Werbeplakaten von bekannten Firmen finden sich gefährliche QR Codes. Je nach Plakat ist es nämlich sehr einfach, den vorhandenen Code mit einem anderen zu überkleben.

## Quellen

- <http://www.qrcode.com>, Stand 20.12.12
- Einführung in QR Codes von Martin Stoev  
<http://martinstoev.de/public/articles/qrcode/ausarbeitung/index.html>, Stand 20.12.12
- ISO/IEC18004:2000
- Wikipedia-Seite zu QR Codes  
<http://de.wikipedia.org/wiki/QR-Code>, Stand 22.12.12
- Aufbau QR Codes und BCH Codes:  
[http://en.wikiversity.org/wiki/Reed%E2%80%93Solomon\\_codes\\_for\\_coders](http://en.wikiversity.org/wiki/Reed%E2%80%93Solomon_codes_for_coders), Stand 1.1.13
- Understanding QR Codes  
<http://marksprague.wordpress.com/qr-codes-technology/understanding-qr-codes/>, Stand 3.1.13
- Abenteuer Informatik, IT zum Anfassen – von Routenplaner bis Onlinebanking, Jens Gallenbacher, 3. Auflage, 2012
- CS Unplugged  
<http://www.csunplugged.org/error-detection>, Stand 3.1.13
- QR Code Generator  
<http://goqr.me/de/>, Stand 1.1.13
- Bilder Design QR Codes  
<http://www.qrcode-generator.de/design>, Stand 9.1.13
- ISO-8559-1:  
[http://de.wikipedia.org/wiki/ISO\\_8559-1](http://de.wikipedia.org/wiki/ISO_8559-1), Stand 10.1.13
- Umrechnung Hex-Bin:  
<http://www.table-ascii.com/>, Stand 10.1.13
- QR Code Tutorial:  
<http://www.thonky.com/qr-code-tutorial/introduction/>, Stand 10.1.13
- QR Art Codes:  
<http://research.swtch.com/qart>, Stand 10.3.13
- [http://qrbcn.com/imatgesbloc/Three\\_QR\\_Code.pdf](http://qrbcn.com/imatgesbloc/Three_QR_Code.pdf), Stand 10.3.13