

INHOUDSOPGAVE

DANKBETUIGING	13
----------------------	-----------

INLEIDING	15
------------------	-----------

Voor wie is dit boek?	16
Wat staat er in dit boek?	16
De bijbehorende website	17
Veel plezier!	17

DEEL 1: XCODE EN SWIFT

1	
HELLO, WORLD!	21

Het installeren van Xcode, de code-editor	22
Je eerste app!	23
Introductie van het storyboard	27
User interface-elementen toevoegen met de Object Library	28
Je werk opslaan	31
De app uitvoeren/tonen op een echt apparaat	32
Wat je geleerd hebt	34

2	
LEREN CODEREN IN EEN PLAYGROUND	35

Constanten en variabelen	37
Wanneer gebruik je constanten of juist variabelen?	39
Constanten en variabelen een naam geven	40
Datatypes	40
Datatypes opgeven	41
Veelgebruikte datatypes	42
Type-inferentie	43
Het transformeren van datatypes door casten	44
Operatoren	45
De bewerkingsvolgorde	48
Bewerkingen forceren met haakjes	49
Operatoren voor samengestelde opdrachten	49
Wat je geleerd hebt	51

3 KEUZES MAKEN **53**

Booleaanse uitdrukkingen	54
Is gelijk aan en is niet gelijk aan	54
Groter dan en kleiner dan	55
Samengestelde Booleaanse uitdrukkingen	56
Voorwaardelijke instructies	58
if instructies	58
switch instructies	62
Wat je geleerd hebt	64

4 CODE IN EEN LOOP SCHRIJVEN **65**

Open de debug area	66
Loops maken via reeksen en verzamelingen met for-in	66
Zeg hallo!	66
Zeg goedemorgen!	68
Het testen van voorwaarden met while loops.	68
Raad mijn getal	69
Krimpen.	70
Welke loop moet je gebruiken?	71
Nesten en bereik	72
Het nesten van blokken code	72
Bereik van constanten en variabelen	74
Wat je geleerd hebt	76

5 JE PROGRAMMA'S VEILIG HOUDEN MET OPTIONALS **77**

Wat is een optional?	78
Optionals maken	78
Optionals uitpakken	79
Een speciaal soort operator: ??	83
Wat je geleerd hebt	84

6 VERZAMELINGEN OPSLAAN IN DICTIONARIES EN ARRAYS **85**

Dingen op volgorde houden met arrays	85
Het gebruik van veranderlijke en onveranderlijke arrays	86
Type-inferentie gebruiken	87
Items in een array opvragen	87
Het bereik in de gaten houden	87
Items toevoegen aan een array	88
Het combineren van arrays	89
Items uit een array verwijderen	89
Items in een array vervangen	90
De eigenschappen van een array gebruiken	91
Een loop toepassen op een array	92

6 INHOUDSOPGAVE

Dictionaries zijn de sleutel!	93
De initialisatie van een dictionary	93
Het benaderen van waarden in een dictionary	94
Items toevoegen aan een dictionary	95
Items uit een dictionary verwijderen	96
Items in een dictionary vervangen	96
De eigenschappen van een dictionary gebruiken	96
Een loop gebruiken in een dictionary	97
Wat je geleerd hebt	98

7 FUNCTIES ZIJN EEN FEESTJE EN JIJ BENT UITGENODIGD 99

Invoer erin, uitvoer eruit	99
Een aangepaste functie schrijven	100
Functies kunnen nog meer met invoerparameters	102
Uitnodigingen voor een feestje maken	102
Al je vrienden tegelijk uitnodigen	104
Je gasten een bericht sturen	105
ArgumentLabels	107
Een aangepast argumentlabel toevoegen	108
Een argumentlabel verwijderen	109
Retourwaarden	110
Welke doos is de grootste?	110
Voorwaardelijk retourneren	111
Wat je geleerd hebt	112

8 AANGEPASTE CLASSES EN STRUCTUREN 113

Een class maken	114
Een definitie van een class schrijven	114
Informatie opslaan in eigenschappen	115
Een exemplaar van een class maken	116
De eigenschappen van een class raadplegen	116
Elke taart aanpassen met initializers	118
Een methode om verjaardagswensen toe te voegen	121
Een helper-methode schrijven	122
Een bijzondere eigenschap genaamd self	124
Class overerving	126
Een superclass maken	126
Een subclass maken	127
Het datatype detecteren door typecasten	128
Het datatype verfijnen door downcasten	131
Waardetypes en referentietypes	132
Structuren gebruiken	135
Wat je geleerd hebt	136

DEEL 2: BIRTHDAY TRACKER

9

KNOPPEN EN SCHERMEN MAKEN OP HET STORYBOARD

139

Een overzicht van je app	140
Een nieuw Xcode-project maken	140
Een app-pictogram toevoegen	143
De verjaardagen van je vrienden weergeven	144
De table view controller toevoegen	145
De navigation controller toevoegen	147
Een knop toevoegen	148
Controle-invoer en labels instellen	151
De namen en verjaardagen van je vrienden toevoegen	151
Zorgen dat je app er perfect uitziet op elk apparaat met auto lay-out	156
Knoppen toevoegen voor opslaan en annuleren	157
Wat je geleerd hebt	158

10

EEN VERJAARDAG CLASS TOEVOEGEN EN GEBRUIKERSINVOER VERWERKEN

159

De Verjaardag class	160
Een nieuw bestand maken	160
De Verjaardag class schrijven	162
Gebruikersinvoer programmeren	162
De Verjaardag toevoegen view controller maken	163
Code aan de input controls koppelen	164
Je code aan het storyboard koppelen	165
Een uiterste geboortedatum instellen	167
Een verjaardag opslaan	168
De knop Opslaan koppelen	168
Tekst in een tekstveld lezen	169
Een datum krijgen van een Date Picker	170
Een verjaardag maken	171
De Annuleren-knop toevoegen	172
Wat je geleerd hebt	172

11

VERJAARDAGEN WEERGEVEN

173

De verjaardagslijst maken	173
De Verjaardagen table view controller maken	174
Cellen toevoegen aan de tabelweergave	176
De Verjaardagen table view controller instellen	179
Verjaardagen weergeven in een tabelweergave	180
Alles komt bij elkaar	184
Delegatie	184
De twee controllers verbinden door het instellen van een gedelegeerde	190
Wat je geleerd hebt	191

8 INHOUDSOPGAVE

12

VERJAARDAGEN OPSLAAN **193**

Verjaardagen opslaan in een database	193
De Verjaardag entiteit	194
De Verjaardag kenmerken	195
De applicatiegedelegeerde	197
Code opschonen	203
Meer functies aan onze app toevoegen	206
Verjaardagen op alfabetische volgorde zetten	206
Verjaardagen verwijderen	207
Wat je geleerd hebt	210

13

MELDINGEN KRIJGEN VAN VERJAARDAGEN **211**

Het raamwerk voor meldingen aan de gebruiker	211
Registreren voor lokale meldingen	212
Een melding plannen	214
Een melding verwijderen	219
De app in het Nederlands uitvoeren	220
Wat je geleerd hebt	220

DEEL 3: SCHOOLHOUSE SKATEBOARDER

14

DE VOORBEREIDING **223**

Waar vind ik de afbeeldingen en geluidseffecten?	224
Spellen maken met de SpriteKit van Xcode	224
Het spelproject maken	225
Afbeeldingen toevoegen	226
Het landschap: Een achtergrondafbeelding weergeven	227
Hoe het wordt afgespeeld: Schermweergave	231
Afbeeldingen aanpassen voor verschillende schermresoluties	233
Wat je geleerd hebt	236

15

VAN SCHOOLHOUSE SKATEBOARDER EEN ECHTE GAME MAKEN **237**

Onze held, de skateboarder	237
Een skater sprite class maken	238
SpriteKit importeren	238
Aangepaste eigenschappen aan de Skater class toevoegen	238
Een exemplaar van de Skater in de scene maken	239
De Skater instellen	240
De Skater op het scherm zien	242
Debugging informatie van SpriteKit begrijpen	243
Bewegende stoeptegels	244
Het maken van stoeptegels	244

De bakstenen voor de stoep bijwerken	246
Het scherm met bakstenen vullen	249
Ruimtes laten om te springen	250
De game-loop	251
Het bijhouden van de update-tijd	252
De verstreken tijd voor elke update berekenen	252
De scrollsnelheid aanpassen op basis van de verstreken tijd	253
Het bijwerken van de bakstenen	254
Omhoog, omhoog en weg: De skater laten springen	255
Een tap gesture recognizer gebruiken	255
Op een eenvoudige manier zwaartekracht simuleren	257
De landing controleren	258
Wat je geleerd hebt	260

16 DE SPRITEKIT PHYSICS-ENGINE GEBRUIKEN 261

De fysieke wereld instellen	262
Fysieke lichamen	263
Vorm geven aan de fysieke lichaam	263
Eigenschappen instellen voor fysieke lichamen	265
De skater sprite een fysiek lichaam geven	266
Fysieke lichamen aan bakstenen toevoegen	268
Contacten en botsingen	268
Contacten en botsingen afhandelen	269
Reageren op contacten	272
Krachten toepassen op fysieke lichamen	273
Het spel starten en beëindigen	274
Het spel starten	274
Het spel eindigen	277
Wat je geleerd hebt	278

17 MOEILIKHEIDSGRAAD INSTELLEN, EDELSTENEN VERZAMELEN EN SCORE BIJHOUDEN 279

De boel versnellen	279
Multilevel-platforms toevoegen	280
Meerdere niveaus van bakstenen definiëren	281
Wijzigen hoe bakstenen voortgebracht worden	283
Edelstenen toevoegen om te verzamelen	284
Edelstenen voortbrengen en bijhouden	285
Besluiten wanneer een edelsteen wordt voortgebracht	286
Edelstenen verwijderen	287
Edelstenen bijwerken	288
Edelstenen verzamelen	290
Scores en labels toevoegen	291
Labels maken	291
Het bijhouden van de score	295
Labels bijwerken	295
De score van de speler bijwerken	297
De edelstenen waardevol maken	298

10 INHOUDSOPGAVE

Het bijhouden van de topscore	299
De gameplay afstellen	299
Wat je geleerd hebt	301

18 STATUS VAN HET SPEL, MENU'S, GELUIDEN EN SPECIALE EFFECTEN 303

De status van het spel bijhouden	303
Een menusysteem toevoegen	306
De MenuLayer class aanmaken	306
De menulagen weergeven als dat nodig is	310
De menulaag verwijderen	312
Geluiden maken	313
De geluidsbestanden toevoegen	314
De geluiden op het juiste moment afspelen	314
Vonken schieten	315
Wat je geleerd hebt	322

HULPMIDDELEN 323

Fouten oplossen	324
Apple documentatie	324
Xcode sneltoetsen	325
IOS-Simulator sneltoetsen	326
Xcode versies	327

INDEX 329

3

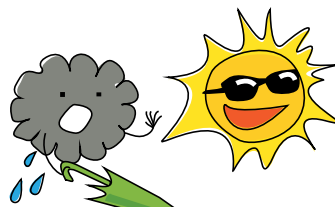
KEUZES MAKEN



Nu we hebben besproken hoe je constanten en variabelen maakt, ben je klaar om te leren hoe je jouw computer vertelt keuzes te maken. Dit hoofdstuk gaat over de flow van een computerprogramma: hoe vertel je de computer welke route hij door het programma moet nemen. Als we het over *flow hebben*, bedoelen we de volgorde waarin de instructies van het programma worden uitgevoerd.

Tot nu toe werden de uitdrukkingen alleen uitgevoerd in de volgorde waarin je ze getypt had. Je hebt daar al gave dingen mee gedaan. Maar door de computer te vertellen hoe hij keuzes kan maken over de volgorde waarin hij de instructies uitvoert, kun je nog meer doen. Om de computer een keuze te laten maken, gebruiken we *voorwaardelijke instructies* die de computer vertellen een code uit te voeren, gebaseerd op de waarde van een voorwaarde.

Je gebruikt elke dag al voorwaardelijke instructies om keuzes te maken! Voordat je 's ochtends van huis vertrekt, bekijk je bijvoorbeeld het weerbericht. Als de zon schijnt, zet je misschien een zonnebril op. Als het regent, pak je je paraplu. In beide gevallen controleer je een voorwaarde. Als de voorwaarde “het regent” waar is, neem je je paraplu mee als je naar buiten gaat. Wanneer de voorwaarde waar of onwaar kan zijn, heet dat een *Booleaanse uitdrukking*. Het `Bool` datatype waarover je geleerd hebt in Hoofdstuk 2 wordt gebruikt om de waarde waar (`true`) of onwaar (`false`) weer te geven.



BOOLEAANSE UITDRUKKINGEN

Een veelvoorkomende Booleaanse uitdrukking is er een die twee waarden vergelijkt met behulp van een *vergelijkende operator*. Er zijn zes vergelijkende operatoren. Laten we beginnen met twee eenvoudige: *is gelijk aan* en *is niet gelijk aan*.

IS GELIJK AAN EN IS NIET GELIJK AAN

Je gaat de *is gelijk aan* en *is niet gelijk aan* vergelijkende operatoren veel gebruiken. *Is gelijk aan* wordt geschreven als twee is-gelijktokens naast elkaar, zoals deze: `==`. *Is niet gelijk aan* wordt geschreven als een uitroepteken en één is-gelijk teken, zoals deze: `!=`.

Laten we ze allebei uitproberen in de playground!

❶ <code>3 + 2 == 5</code>	<code>true</code>
<code>4 + 5 == 8</code>	<code>false</code>
❷ <code>3 != 5</code>	<code>true</code>
<code>4 + 5 != 8</code>	<code>true</code>
// Dit is verkeerd en levert een foutmelding op	
❸ <code>3 + 5 = 8</code>	<code>error</code>

In gewoon Nederlands, zegt de regel bij ❶ “drie plus twee is vijf” en dit is een ware bewering. De uitvoer in het deelvenster rechts zal dit bevestigen zodra je klaar bent met typen. Bij ❷ zegt de regel, “drie is niet gelijk aan vijf” en dat is ook een ware bewering. Merk op dat ❸ een fout is. Weet je waarom? Hoewel `=` en `==` veel op elkaar lijken, kent een enkel is-gelijk teken (`=`) waarden toe. Die instructie luidt: “Plaats de waarde van 8 in iets dat `3 + 5` heet” en dat werkt niet.

In Swift werkt de `==` operator ook met andere datatypes, en niet alleen met getallen. Laten we proberen enkele andere vergelijkingen te maken.

<pre>// Strings vergelijken let mijnNaam = "Anne" mijnNaam == "Melissa" mijnNaam == "Anne" ❶ mijnNaam == "anne" var mijnLengte = 171 mijnLengte == 171 // Dit is verkeerd en levert een foutmelding op ❷ mijnLengte == mijnNaam</pre>	<pre>"Anne" false true false 67.5 true error</pre>
---	--

De regel bij ❶ is een lastige. Had je verwacht dat die waar zou zijn? Deze twee strings lijken veel op elkaar, maar ze zijn niet precies hetzelfde. Een *is gelijk* vergelijking is alleen waar als de twee waarden precies gelijk zijn. De constante `mijnNaam` heeft de waarde "Anne" met een hoofdletter A en dat is niet hetzelfde als "anne" met een kleine letter a.

Herinner je je nog uit Hoofdstuk 2 dat we zeiden dat je geen rekenkundige operatoren zoals + en * kunt gebruiken voor dingen die niet hetzelfde datatype zijn? Hetzelfde geldt voor vergelijkingen. Je kunt geen dingen vergelijken die van verschillende types zijn. De regel bij ❷ zal een foutmelding geven, omdat de ene een string is en de andere een Double.

GROTER DAN EN KLEINER DAN

Laten we nu eens kijken naar vier andere vergelijkende operatoren. We beginnen met *groter dan* (geschreven als >) en *kleiner dan* (geschreven als <). Je hebt waarschijnlijk al een goed idee van hoe deze werken. Een Booleaanse uitdrukking als $9 > 7$, wat wil zeggen "9 is groter dan 7" is waar. Vaak wil je weten of iets groter dan of gelijk is aan iets of kleiner dan of gelijk aan iets. Hiervoor bestaan er nog twee operatoren: *groter dan of gelijk aan* (dat eruit ziet als >=) en *minder dan of gelijk aan* (dat eruit ziet als <=). Laten we deze eens uitproberen met wat meer voorbeelden:

<pre>// Groter dan 9 > 7 // Kleiner dan 9 < 11 // Groter dan of gelijk aan ❶ 3 + 4 >= 7 ❷ 3 + 4 > 7 // Kleiner dan of gelijk aan ❸ 5 + 6 <= 11 ❹ 5 + 6 < 11</pre>	<pre>true true true false true false</pre>
---	--

Let op het verschil tussen *groter dan of gelijk aan* bij ❶ en *groter dan* bij ❷. De som van $3 + 4$ is niet groter dan 7, maar het is groter dan of gelijk aan 7. Op dezelfde manier is $5 + 6$ minder dan of gelijk aan 11 ❸, maar niet kleiner dan 11 ❹.

Tabel 3-1 geeft een overzicht van de zes vergelijkende operatoren.

Tabel 3-1: Vergelijkende operatoren.

Symbool	Definitie
==	Is gelijk aan
!=	Is niet gelijk aan
>	Is groter dan
<	Is kleiner dan
>=	Is groter dan of gelijk aan
<=	Is kleiner dan of gelijk aan

Je zult merken dat je deze operatoren vaak gebruikt wanneer je voorwaardelijke instructies schrijft.

SAMENGESTELDE BOOLEAANSE UITDRUKKINGEN

Samengestelde Booleaanse uitdrukkingen zijn eenvoudige Booleaanse uitdrukkingen die samengevoegd zijn. Het lijkt wel wat op het samenvoegen van zinnen in het Nederlands met de woorden *en* en *of*. Bij programmeren is er nog een derde mogelijkheid: *niet*. In Swift noemen we deze woorden *logische operatoren*. We gebruiken de Engelse woorden “and”, “or” en “not”. Een logische operator combineert een Booleaanse uitdrukking met een andere of ontkracht die. De drie logische operatoren in Swift kun je vinden in tabel 3-2.

Tabel 3-2: Logische operatoren.

Symbool	Definitie
&&	Logisch AND
	Logisch OR
!	Logisch NOT

Met logische operatoren kun je uitdrukkingen schrijven die testen of een waarde binnen een bepaald bereik valt, zoals: “Ligt de leeftijd van deze persoon tussen de 10 en 15?” Dit kun je doen door te testen of de leeftijd tegelijkertijd groter is dan 10 *en* kleiner dan 15, zoals hier:

<pre>var leeftijd = 12 leeftijd > 10 && leeftijd < 15</pre>	<pre>12 true</pre>
---	--------------------

De uitdrukking `leeftijd > 10 && leeftijd < 15` is waar, omdat beide voorwaarden waar zijn: leeftijd is groter dan 10 en kleiner dan 15. Een “AND”-uitdrukking is alleen waar als beide voorwaarden waar zijn.

Verander de waarde van leeftijd maar eens in 18 en kijk wat er gebeurt:

<pre>var leeftijd = 18 leeftijd > 10 && leeftijd < 15</pre>	18 false
---	-------------

Omdat we leeftijd nu veranderd hebben in 18, is maar één kant van de uitdrukking waar. De variabele leeftijd is nog steeds groter dan 10, maar niet meer kleiner dan 15, dus onze uitdrukking wordt onwaar.

Probeer nu “OR” uit door de volgende code in je playground in te voeren:

<pre>let naam = "Jacqueline" naam == "Jack" ❶ naam == "Jack" naam == "Jacqueline"</pre>	"Jacqueline" false true
--	-------------------------------

Eerst verzinnen we een persoon met de naam Jacqueline door de constante naam in te stellen op "Jacqueline". Vervolgens testen we enkele voorwaarden om te zien of ze waar of onwaar zijn. Omdat naam == "Jacqueline" waar is, is de “OR”-uitdrukking bij ❶ waar, hoewel naam == "Jack" onwaar is. In het Nederlands zegt deze uitdrukking: “De naam van deze persoon is Jack *of* de naam van deze persoon is Jacqueline”. In een “OR”-uitdrukking hoeft slechts één van de voorwaarden waar te zijn om de hele uitdrukking waar te laten zijn.



Nu proberen we enkele “NOT”-uitdrukkingen. Voer de volgende code in je playground in:

<pre>let isEenMeisje = true ❶ !isEenMeisje && naam == "Jack" isEenMeisje && naam == "Jacqueline" ❷ (!isEenMeisje && naam == "Jack") (isEenMeisje && naam == "Jacqueline")</pre>	true false true true
--	-------------------------------

De ! operator wordt gebruikt in de samengestelde Booleaanse uitdrukking ❶, die je kunt lezen als “Onze persoon is *niet* een meisje *en* onze persoon heet Jack”. Deze uitdrukking heeft twee logische operatoren, namelijk ! en &&. Je kunt bij het schrijven van samengestelde Booleaanse uitdrukkingen zoveel logische operatoren combineren als je wilt.

Soms is het een goed idee om haakjes te gebruiken om de computer te laten weten wat er eerst beoordeeld moet worden. Haakjes maken de code ook beter leesbaar. Dit is vergelijkbaar met de manier waarop je haakjes gebruikt wanneer je verschillende rekenkundige bewerkingen in één vergelijking gebruikt, zoals je in “Bewerkingen forceren met haakjes” op pagina 49 hebt kunnen lezen. Bij ❷ gebruiken we haakjes om de computer te vertellen dat hij eerst moet controleren op !isEenMeisje && naam == "Jack" en vervolgens op isEenMeisje && naam == "Jacqueline". Na evaluatie van de twee delen kan de computer het “OF”-gedeelte beoordelen voor de gehele verklaring (die waar zal zijn, omdat het tweede deel waar is). Nogmaals, in een “OR”-verklaring is de hele uitdrukking waar als één van de voorwaarden waar is.

In tabel 3-3 zie je de drie logische operatoren en de samengestelde uitdrukkingen die je ermee kunt maken, met de bijbehorende Booleaanse waarden.

Tabel 3-3: Samengestelde Booleaanse uitdrukkingen met logische operatoren.

Logische operator	Samengestelde uitdrukking	Waarde
NOT (!)	!true	false
NOT (!)	!false	true
AND (&&)	true && true	true
AND (&&)	true && false	false
AND (&&)	false && true	false
AND (&&)	false && false	false
OR ()	true true	true
OR ()	true false	true
OR ()	false true	true
OR ()	false false	false

Het eerste item in de tabel laat zien dat iets dat NIET waar is, onwaar is. En iets dat NIET onwaar is, is dus waar.

Met de “AND”-operator is alleen iets dat waar && waar is waar. Dit betekent dat de uitdrukkingen aan beide kanten van de && operator waar moeten zijn voordat de && uitdrukking waar is. Een samengestelde uitdrukking die waar && onwaar is, geeft onwaar. En een samengestelde && uitdrukking waarin beide voorwaarden onwaar zijn, zal ook onwaar opleveren.

Als het gaat om de “OR”-operator hoeft maar één van de uitdrukkingen aan beide kanten van de || operator waar te zijn om de || uitdrukking waar te laten zijn. Dus een waar || waar is waar, en een waar || onwaar is ook waar. Alleen een samengestelde “OR”-uitdrukking waarin beide zijden onwaar zijn, wordt uiteindelijk onwaar. Je ziet dat in de tabellen en de voorbeelden de Engelse woorden “true” (waar) en “false” (onwaar) gebruikt worden, omdat de Swift-programmeertaal in het Engels is.

VOORWAARDELIJKE INSTRUCTIES

Er bestaan twee soorten voorwaardelijke instructies: if instructies en switch instructies. Deze instructies geven de computer een voorwaarde op basis waarvan de computer een keuze maakt.

IF INSTRUCTIES

Een if instructie begint met het trefwoord if, gevolgd door een *voorwaarde* die altijd een Booleaanse uitdrukking is. De computer onderzoekt de voorwaarde en voert de code in de if instructie uit als de conditie waar is, en