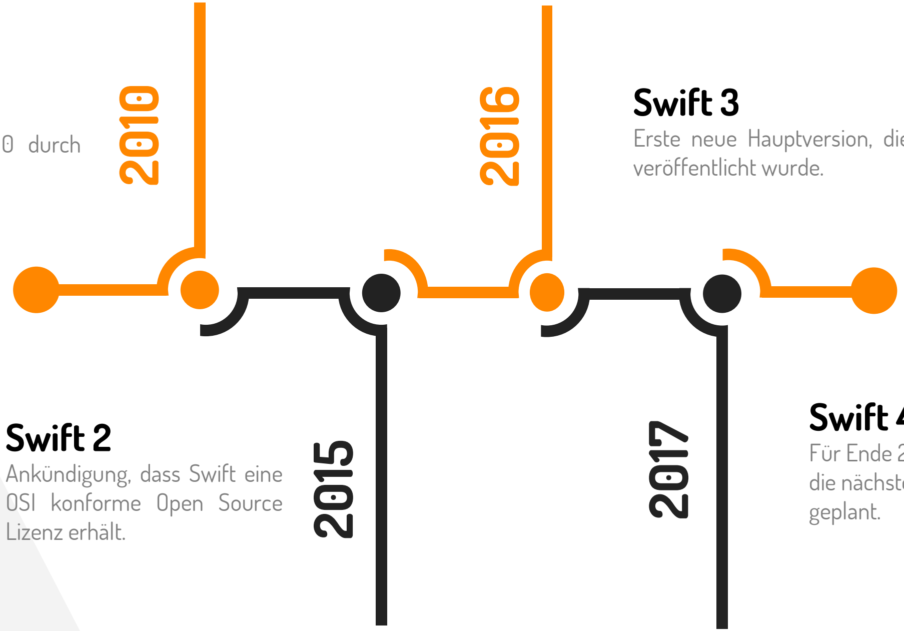
E-PORTFOLIO
SWIFT

Die Geschichte von Swift



2010

Entwicklung

Begann im Juli 2010 durch Chris Lattner.

2015

Swift 2

Ankündigung, dass Swift eine OSI konforme Open Source Lizenz erhält.

2016

Swift 3

Erste neue Hauptversion, die veröffentlicht wurde.

2017

Swift 4

Für Ende 2017 ist mit Swift 4 die nächste Hauptversion geplant.

Modern

Generische und Funktionale Programmierung

Safe

Statischer und starke Typisierung, kein Null-Pointer by Default, Nicht-Veränderbarkeit von Daten

Fast and powerful

Virtual Functions statt Message Passing, Structs, hinreichend intelligenter Compiler

Modern: Funktionale Programmierung

Objective C

```
NSArray *numbers = @[@1, @2, @3, @4, @5, @6];
NSMutableArray *squaredNumbers = [NSMutableArray array];
for (NSNumber *number in numbers)
{
    NSNumber *squaredResult = [NSNumber numberWithInt:[number intValue] * [number intValue]];
    [squaredNumbers addObject:squaredResult];
}
NSLog(@"squared numbers: %@", squaredNumbers);
```

Swift

```
let numbers = [1, 2, 3, 4, 5, 6]
let squaredNumbers = numbers.map({x in x * x})
println("square numbers: \(squaredNumbers)")
```

```
let sum = numbers.reduce(0, +)
```

Modern: Generische Programmierung

C

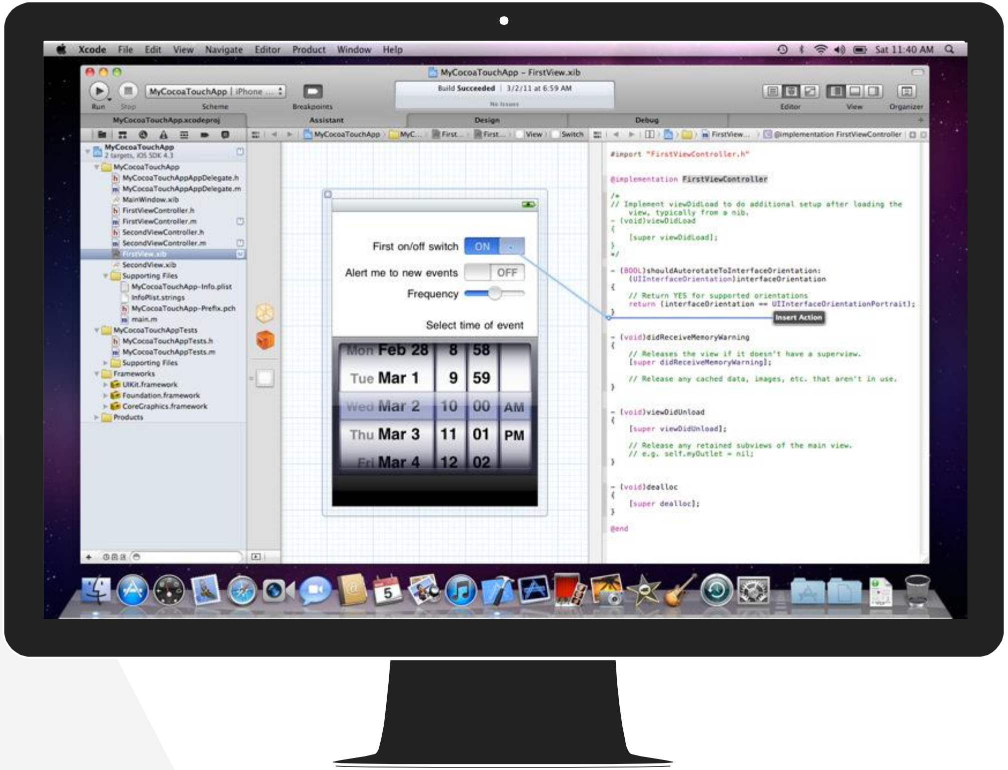
```
int maxInt(int a, int b) {
    if (a > b) {
        return a;
    }
    return b;
}

float maxFloat(float a, float b) {
    if (a > b) {
        return a;
    }
    return b;
}
```

Swift

```
id<Comparable> max(id<Comparable> a, id<Comparable> b) {
    if ([a compare:b] == NSOrderedDescending) {
        return a;
    }
    return b;
}
```

Apple exclusive

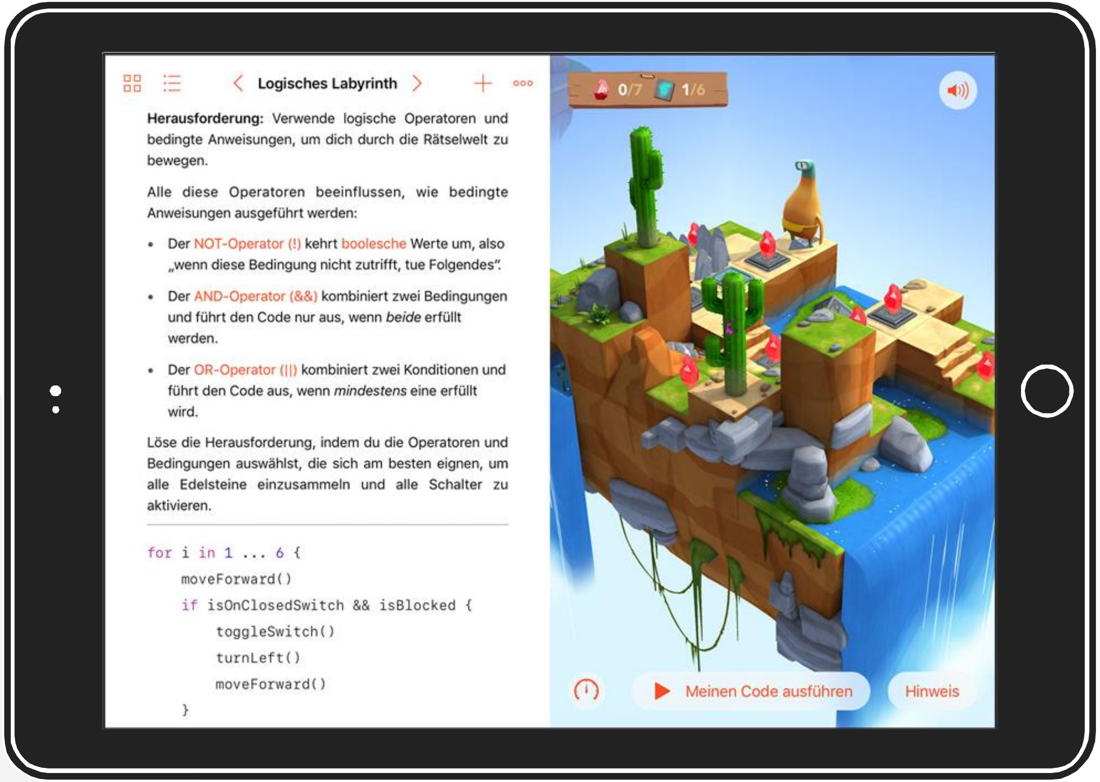
Xcode als Entwicklungsumgebung



```
#import "FirstViewController.h"


@implementation FirstViewController

/*
// Implement viewDidLoad to do additional setup after loading the
    view, typically from a nib.
- (void)viewDidLoad
{
    [super viewDidLoad];
}
*/

- (BOOL)shouldAutorotateToInterfaceOrientation:
    (UIInterfaceOrientation)interfaceOrientation
{
    // Return YES for supported orientations
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}


- (void)didReceiveMemoryWarning
{
    // Releases the view if it doesn't have a superview.
    [super didReceiveMemoryWarning];

    // Release any cached data, images, etc. that aren't in use.
}


- (void)viewDidUnload
{
    [super viewDidUnload];

    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}


- (void)dealloc
{
    [super dealloc];
}

@end
```

I hope that by making programming more approachable and fun, we'll appeal to the next generation of programmers and to help redefine how Computer Science is taught.

Swift Playgrounds



Herausforderung: Verwende logische Operatoren und bedingte Anweisungen, um dich durch die Rätselwelt zu bewegen.

Alle diese Operatoren beeinflussen, wie bedingte Anweisungen ausgeführt werden:

- Der NOT-Operator (!) kehrt boolesche Werte um, also „wenn diese Bedingung nicht zutrifft, tue Folgendes".
- Der AND-Operator (&&) kombiniert zwei Bedingungen und führt den Code nur aus, wenn *beide* erfüllt werden.
- Der OR-Operator (||) kombiniert zwei Konditionen und führt den Code aus, wenn *mindestens* eine erfüllt wird.

Löse die Herausforderung, indem du die Operatoren und Bedingungen auswählst, die sich am besten eignen, um alle Edelsteine einzusammeln und alle Schalter zu aktivieren.

```
for i in 1 ... 6 {
    moveForward()
    if isOnClosedSwitch && isBlocked {
        toggleSwitch()
        turnLeft()
        moveForward()
    }
```

Demo

https://swift.sandbox.bluemix.net/#/repl

Variablen & Konstanten

```
//Variablen
var str = "Hallo"
str = "Tschüss"

//Konstanten
let name = "Katharina"
~~name = "Andre"~~
```

Type Inferrence & Type Annotation

```
//Type Inferrence
var age = 19  //Int
var pi = 3.14  //Double
var name = "Katharina"  //String
~~age = true~~

//Type Annotation
~~var heightInCm~~
var heightInCm: Int
heightInCm = 172

let surname: String
surname = "Heer"
~~surname = "Harbrecht"~~
```

Datentypen

```
//String
var name = "Katharina"

//Int
var age = 19

//Double
var pi = 3.14

//Bool
let isfemale = true
var xChromosomen: Int

if isfemale {
xChromosomen = 2
}
else {
xChromosomen = 1
}
```

Typumwandlung

```
var pi = 3 + 0.141  //pi = 3.141

var aInt = 3
var aDouble = 0.141
~~var againpi = aInt + aDouble~~
var againpi = Double(aInt) + aDouble  // againpi = 3.141
var withlost = aInt + Int(aDouble)  // withlost = 3

var x = 3
var y = 10
var z = y / x  // z = 3
var w = Double(x) + Double(y)  // z = 3.33…

var n = 10/3  // n = 3
```

Optionals

```
var errorcode: Int?
errorcode = 404
print(errorcode)  // Optional(404)

//forced unwrapping
if errorcode != nil {
print(errorcode!)  // 404
}

//implicit unwrapping
var errormessage: String!
errormessage = nil
if errormessage != nil {
print(errormessage)
}
```

```
// guard
func addiere (x:Int?, y:Int?)->Int?
{
guard let x = x else { return nil }
guard let y = y else { return nil }
return x+y
}

//if let
var age: Int?
age = 19
if let myage = age {
print(myage)  //19
}

```

Funktionen

```
func addiere(x:Int,y:Int)->Int {
return x+y
}

var summe = addiere(x:1,y:5)  //summe = 6

func fibonacci(x:Int)->Int {
if x==1 { return 1 }
else if x==0 { return 0 }
else {
            return (fibonacci(x:x-1)+fibonacci(x:x-2))
     }
}

var fibo22 = fibonacci(22)  //fibo22 = 17711
```