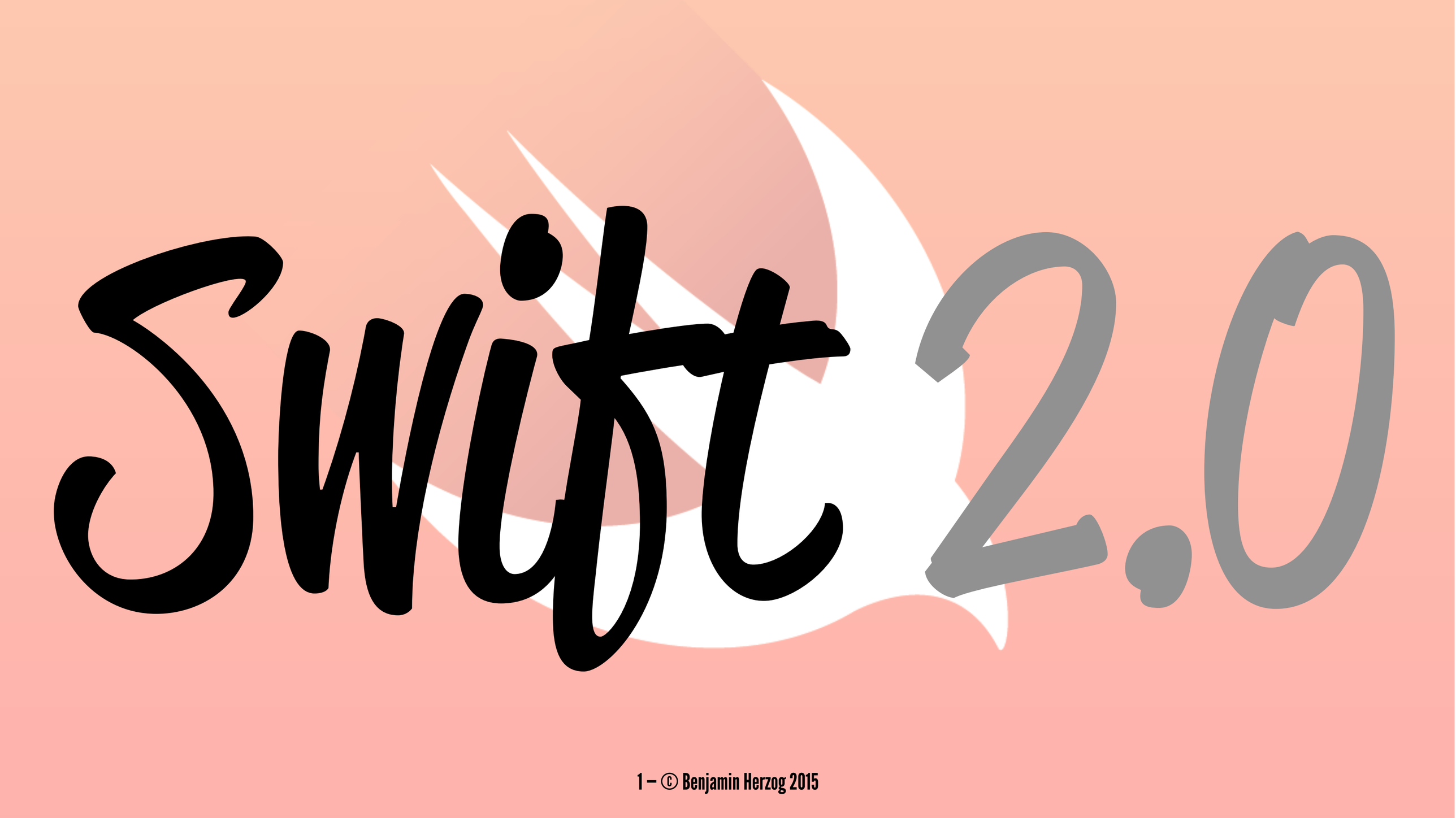

Swift 2.0

- try / catch / throws
- guard
- defer
- protocol extensions
- API availability checks
- minor changes

TRY / CATCH / THROWS

TRY / CATCH / THROWS

```
func downloadFromURL(url: NSURL, inout error: NSError?) -> NSData?
{ ... }

var error: NSError?
downloadFromURL(url, error: &error)

downloadFromURL(url, error: nil)
```

```
func downloadFromURL(url: NSURL) throws -> NSData?
{
    throw MyError.NetworkError
}

do
{
    try downloadFromURL(url)
} catch MyError.NetworkError
{
    print("Network Error")
} catch
{
    print("Some other error")
}
```

NUR ÜBER EIGENEN ERROR-TYP

```
enum MyError: ErrorType {
    case NetworkError
    case UserError
}
```

NUR ÜBER EIGENEN ERROR-TYP

```
enum MyError: ErrorType {
    case NetworkError(httpStatusCode: Int)
    case UserError(error: String)
}
```

```swift
enum MyError: ErrorType {
    case NetworkError(httpStatusCode: Int)
    case UserError
}

func downloadFromURL(url: NSURL) throws -> NSData?
{
    throw MyError.NetworkError(httpStatusCode: 500)
}

do{
    try downloadFromURL(url)
} catch MyError.NetworkError(let status) {
    print("Network Error (statusCode \(status))")
} catch {
    print("Some other error")
}
```

```
enum MyError: ErrorType {
    case Fatal
}

func machEtwasGefährliches() throws {
    throw MyError.Fatal
}

func machEtwas() throws {
    try machEtwasGefährliches()
}
```

GUARD

```
let json = getJsonFromUrl(url) // json ist ein [String: AnyObject]

if let name = json["name"] {
    if let alter = json["alter"] {
        if let adresse = json["adresse"] {
            if let score = json["score"] {
                liste.append(User(name: name, alter: alter,
                                  adresse: adresse, score: score))
            }
        }
    }
}
```

```
let json = getJsonFromUrl(url) // json ist ein [String: AnyObject]

if let name = json["name"] {
    if let alter = json["alter"] {
        if let adresse = json["adresse"] {
            if let score = json["score"] {
                liste.append(User(name: name, alter: alter,
                                  adresse: adresse, score: score))
            }
            else { println("kein score") }
        }
        else { println("keine adresse") }
    }
    else { println("kein alter") }
}
else { println("kein name") }
```

SWIFT 1.2

```
let json = getJsonFromUrl(url) // json ist ein [String: AnyObject]

if let name = json["name"], alter = json["alter"],
    adresse = json["adresse"], score = json["score"] {
                liste.append(User(name: name, alter: alter,
                                  adresse: adresse, score: score))
}
else {
    println("User konnte nicht erstellt werden.")
}
```

SWIFT 2.0

SWIFT 2.0

```
guard let name = json["name"] as? String else { throw Error.NoName; return }
guard let alter = json["alter"] as? String else { throw Error.NoAge; return }
guard let adresse = json["adresse"] as? String else { throw Error.NoAdress; return }
guard let score = json["score"] as? String else { throw Error.NoScore; return }

liste.append(User(name: name, alter: alter,
                            adresse: adresse,
                              score: score)))
```

DEFER

DEFER

```
func readFromFile()
{
    let file = openFile()

    for line in file {
        read()
    }
    file.close()
}
```

DEFER

```
func readFromFile()
{
    let file = openFile()

    for line in file {
        if !read() {
            file.close()
            return
        }
    }
    file.close()
}
```

DEFER

```
func readFromFile()
{
    let file = openFile()
    defer {
        file.close()
    }

    for line in file {
        if !read() { return }
    }
}
```

PROTOCOL EXTENSIONS

PROTOCOL EXTENSIONS

```
protocol Fahrzeug {
    var anzahlRäder: Int { get }
}
```

PROTOCOL EXTENSIONS

```
protocol Fahrzeug {
    var anzahlRäder: Int { get }
}

extension Fahrzeug {
    func fahr()
    {
        print("Ich fahre mit \(anzahlRäder) Rädern!")
    }
}
```

PROTOCOL EXTENSIONS

```
struct Auto: Fahrzeug {
    var anzahlRäder: Int {
        get{
            return 4
        }
    }
}

var audi = Auto()

audi.fahr()
```

PROTOCOL EXTENSIONS

```
protocol TransportMittel {
    var volumen: Int { get }
}

extension TransportMittel {
    func transportiere() {
        print("Ich transportiere \(volumen)cm^3")
    }
}
```

PROTOCOL EXTENSIONS

```
struct Auto: Fahrzeug, TransportMittel {
    var anzahlRäder: Int {
        get{ return 4 }
    }
    var volumen: Int {
        get { return 1500 }
    }
}

var audi = Auto()

audi.fahr()
audi.transportiere()
```

PROTOCOL EXTENSIONS

https://developer.apple.com/videos/wwdc/2015/?id=408

API AVAILABILITY CHECKS

API AVAILABILITY CHECKS

```
if (NSClassFromString("UIAlertController") != nil)
{
    // we are at iOS 8 \o/
}
```

API AVAILABILITY CHECKS

```
func doSomethingFancy() {
    guard #available(iOS 9, *) else {
        doSomethingNotSoFancy()
    }
    // we are at iOS 9 \o/
}
```

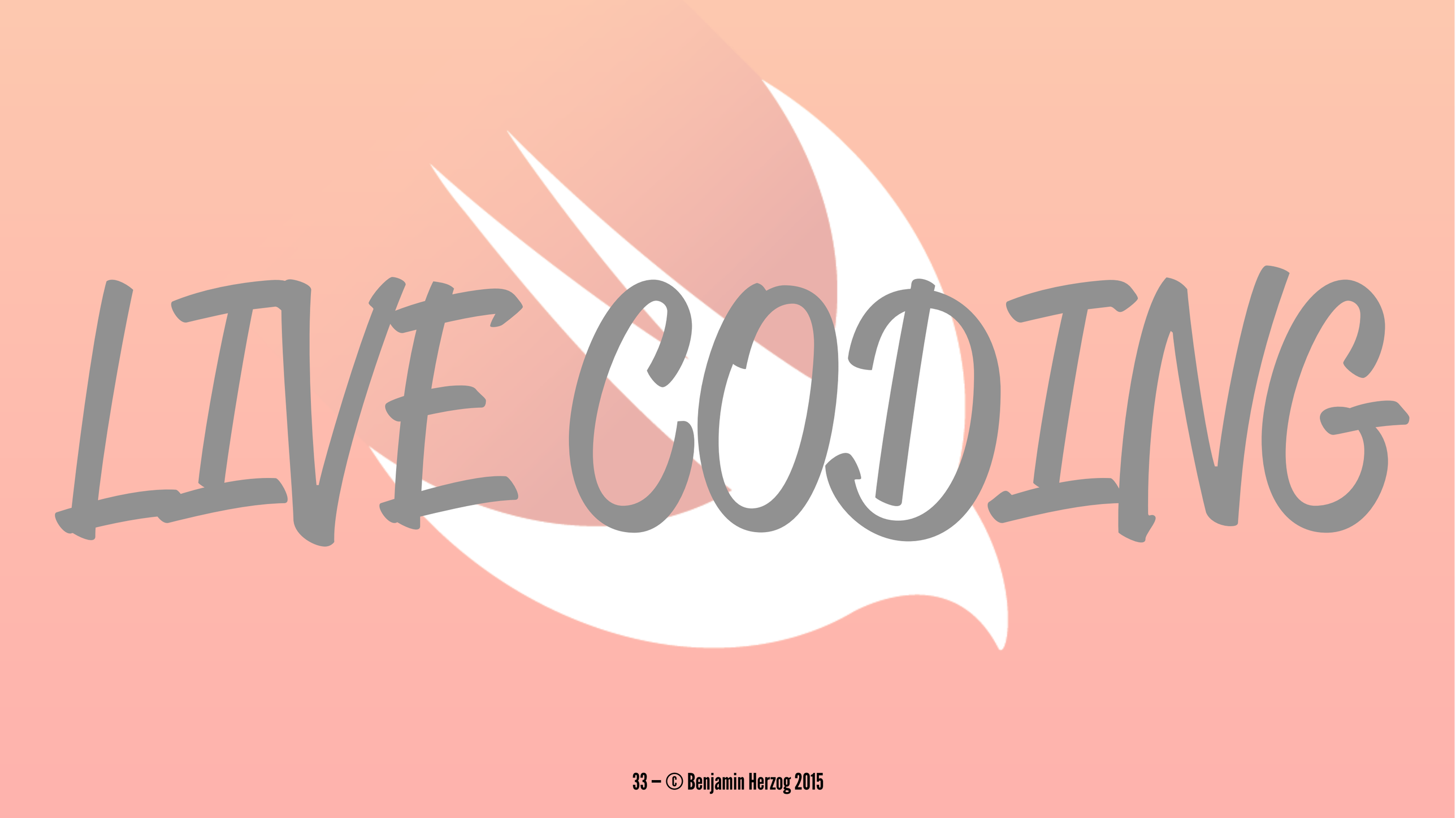
API AVAILABILITY CHECKS

```
if #available(iOS 9, *) {
    // iOS 9
}
else {
    // ¯\(ツ)/¯
    let stackView = UIStackView ()
}
```

MINOR CHANGES

MINOR CHANGES

- print statt println
- mutability warnings
- variable not used warnings
- String lenght > string.characters.count
- generated interfaces


LIVE CODING

DANKE!

Twitter: @BenchR
Youtube: SwiftDeTut
Facebook: benschr