Strings&
Strings&
Strings&
Strings&
Strings&
Strings&
Strings

Swift Strings Seven Ways

a talk presented by Mattt

at try! Swift NYC 2018

Ꭱ Ꭰ Ꮃ Ꮵ Ꮐ Ꮽ Ꮺ Ꮅ Ꮑ Ꮌ Ꭹ Ᏹ Ꮟ Ꮲ

Ꭳ Ꮇ Ꮄ Ꭽ Ꮼ Ꮰ Ꮤ Ᏼ Ꮈ Ꭿ Ꮝ Ꮂ Ꮁ Ꭺ

Ꮷ Ꮍ Ꮞ Ꮦ Ꮯ Ꮸ Ꮿ Ꮜ Ꮖ Ꮓ Ꮔ Ꮳ Ꮢ Ꮒ

Ꮬ Ꮩ Ꭸ Ꮣ Ꭼ Ꮎ Ꭲ Ꮻ Ᏸ Ꮘ Ꮚ Ꭻ Ꮶ Ꮙ

Ꭷ Ꮕ Ꮆ Ꮹ Ꮴ Ꮧ Ꮾ Ꮥ Ꮪ Ᏻ Ꭵ Ꭴ Ꮡ Ꮏ Ꮠ

Ꮱ Ꭾ Ꮀ Ꮋ Ꮫ Ꮉ Ꭶ Ꮭ Ꮮ Ꮨ Ꮗ Ᏺ Ꮊ Ꮛ

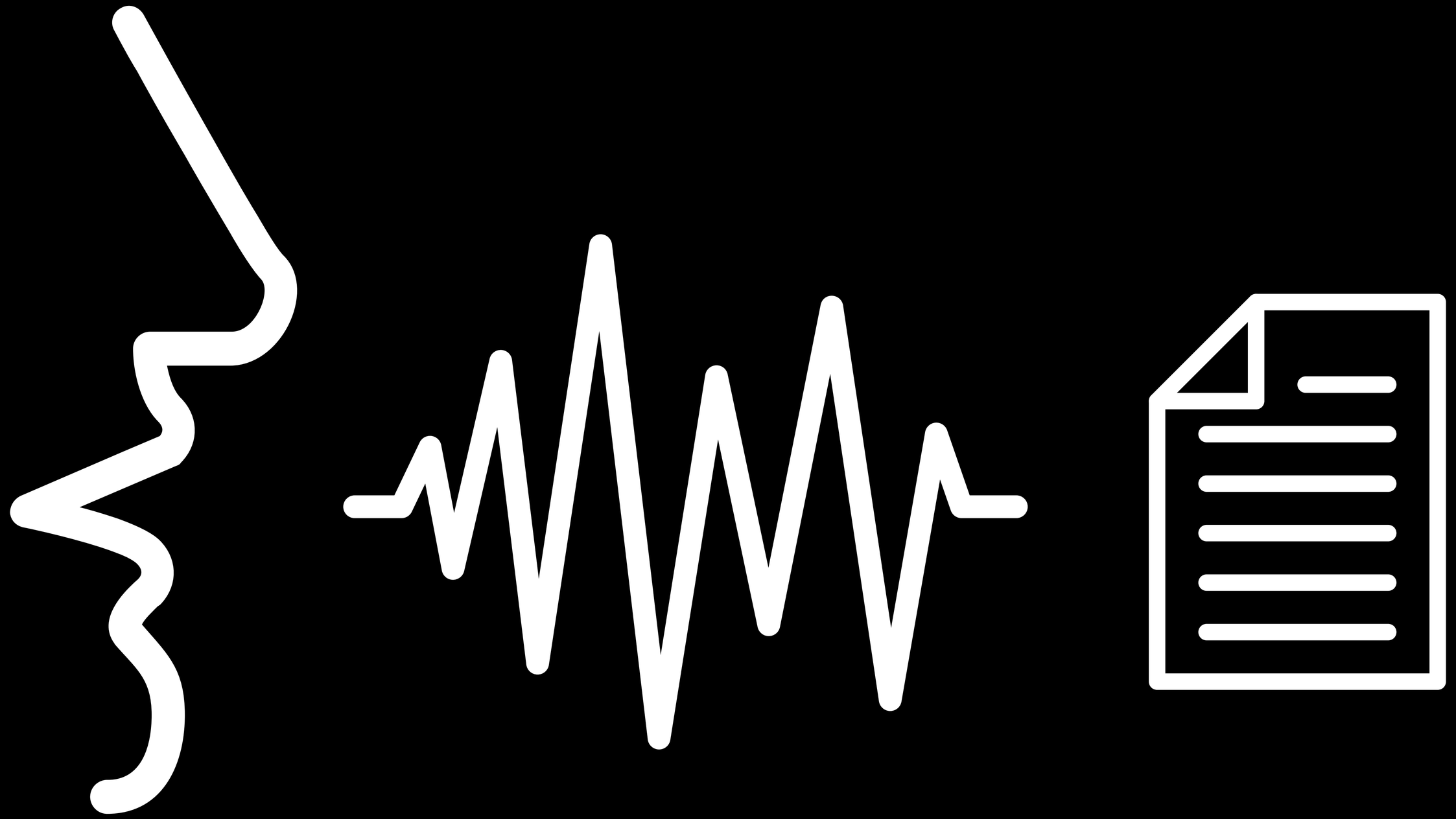
Gwy

/d͡ʒa laˈ gî/

1. Unicode Characters
2. Identifiers
3. Paths
4. Natural Language Text
5. Structured Data
6. Encoded Data
7. Typographic Information

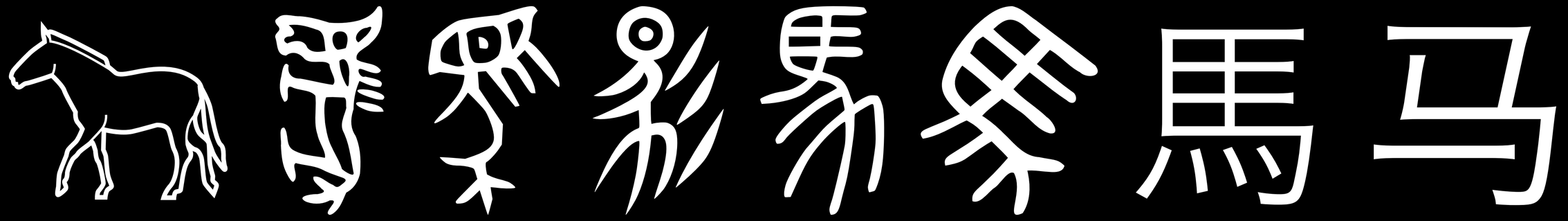
1

Collections of Unicode Characters



Western Writing Systems

Eastern Writing Systems



=

=

Idea

Ideogram

=

Idea	Word
Ideogram	Logogram

=

Idea	Word	Phoneme
Ideogram	Logogram	Alphabet

=

Idea	**Word**	**Phoneme**
Ideogram	Logogram	Alphabet
Consonant + Vowel		
Abugida		

=

Idea	**Word**	**Phoneme**
Ideogram	Logogram	Alphabet
Consonant + Vowel	**Consonant**	
Abugida	Abjad	

=

Idea	**Word**	**Phoneme**
Ideogram	Logogram	Alphabet
Consonant + Vowel	**Consonant**	**Syllable / Mora**
Abugida	Abjad	Syllabary

Alphabets	Latin / Кириллица / Ελληνικό / Հայոց / 한글
Abjads	اُردُو / אָלֶף־בֵּית עִבְרִי / العَرَبِيَّة
Abugidas	देवनागरी / संस्कृतम् / ಕನ್ನಡ / తెలుగు, தமிழ் / ግዕዝ
Syllaberies	ひらがな / カタカナ / 𓀀𓁹𓂋
Logograms	漢字 / 汉字
Ideograms	👋😄 / ✳ / ⚛

Byte

char

ASCII (1960)

American Standard Code for
Information Interchange

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
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	{	|	}	~	DEL

ASCII

“Hello”

H	1000100
e	0101110
l	1100110
l	1100110
o	1111110
NUL	0000000

„Hvåd med mine vøkæler?”

— Dansk

「つまらない質問で恐縮ですが
¥はどこにありますか？」

— 日本語

ISO/IEC 646 (1964)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
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					DEL

ISO/IEC 646 (-INV)

US / IRV	#	$	@	[	\	]	^	`	{	|	}	~
DE	#	$	§	Ä	Ö	Ü	^	`	ä	ö	ü	ß
DK	#	¤	@	Æ	Ø	Å	Ü	`	æ	ø	å	ü
FR	£	$	à	°	ç	§	^	µ	é	ù	è	¨
JP	#	$	@	[	¥	]	^	`	{	|	}	¯
…												

ISO/IEC 646

C:¥Program Files¥

ISO/IEC 8859 (1987)

Part 1	Latin-1 Western European
Part 2	Latin-2 Central European
Part 3	Latin-3 South European
Part 4	Latin-4 North European
Part 5	Latin/Cyrillic
Part 6	Latin/Arabic
Part 7	Latin/Greek
Part 8	Latin/Hebrew
Part 9	Latin-5 Turkish
Part 10	Latin-6 Nordic
Part 11	Latin/Thai
Part 12	Latin/Devanagari
Part 13	Latin-7 Baltic Rim
Part 14	Latin-8 Celtic
Part 15	Latin-9
Part 16	Latin-10 South-Eastern European

$$\text{8-bit} \Rightarrow 2^8 = 256$$

Danish	Ǿ, ǿ
Dutch	IJ, ij, ĳ
Estonian	Š, š, Ž, ž
Finnish	Š, š, Ž, ž
French	Œ, œ, Ÿ
German	ß
Irish	Ḃ, ḃ, Ċ, ċ, Ḋ, ḋ, Ḟ, ḟ, Ġ, ġ, Ṁ, ṁ, Ṡ, ṡ, Ṫ, ṫ
Welsh	Ẁ, ẁ, Ẃ, ẃ, Ŵ, ŵ, Ŷ, ŷ

Languages with Incomplete Coverage for ISO/IEC 8859

Latin 6

يولد جميع الناس أحراراً متساوين في الكرامة والحقوق، وقد وهبوا عقلاً
وضميراً وعليهم أن يعامل بعضهم بعضاً بروح الإخاء.

Latin 5

Bütün insanlar hür, haysiyet ve haklar bakımından eşit doğarlar.
Akıl ve vicdana sahiptirler ve birbirlerine karşı kardeşlik
zihniyeti ile hareket etmelidirler.

106,230

异体字字典

Entries in the Dictionary of Chinese Variant Form

Unicode (1991)

1,114,112

Unicode Code Points

UTF-8

UTF-16

UTF-32

A **String** is a collection of **Characters**

A **String** is a collection of **Characters**

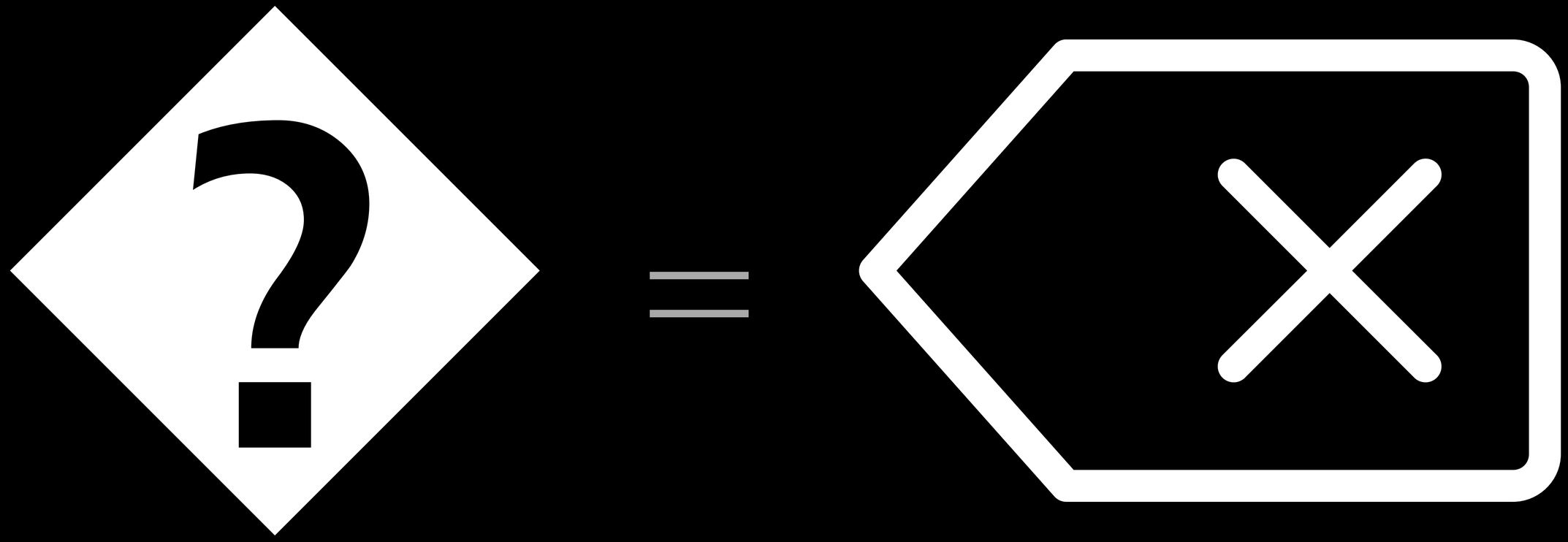
A **Character** consists of one or more **Unicode scalars**

A **String** is a collection of **Characters**

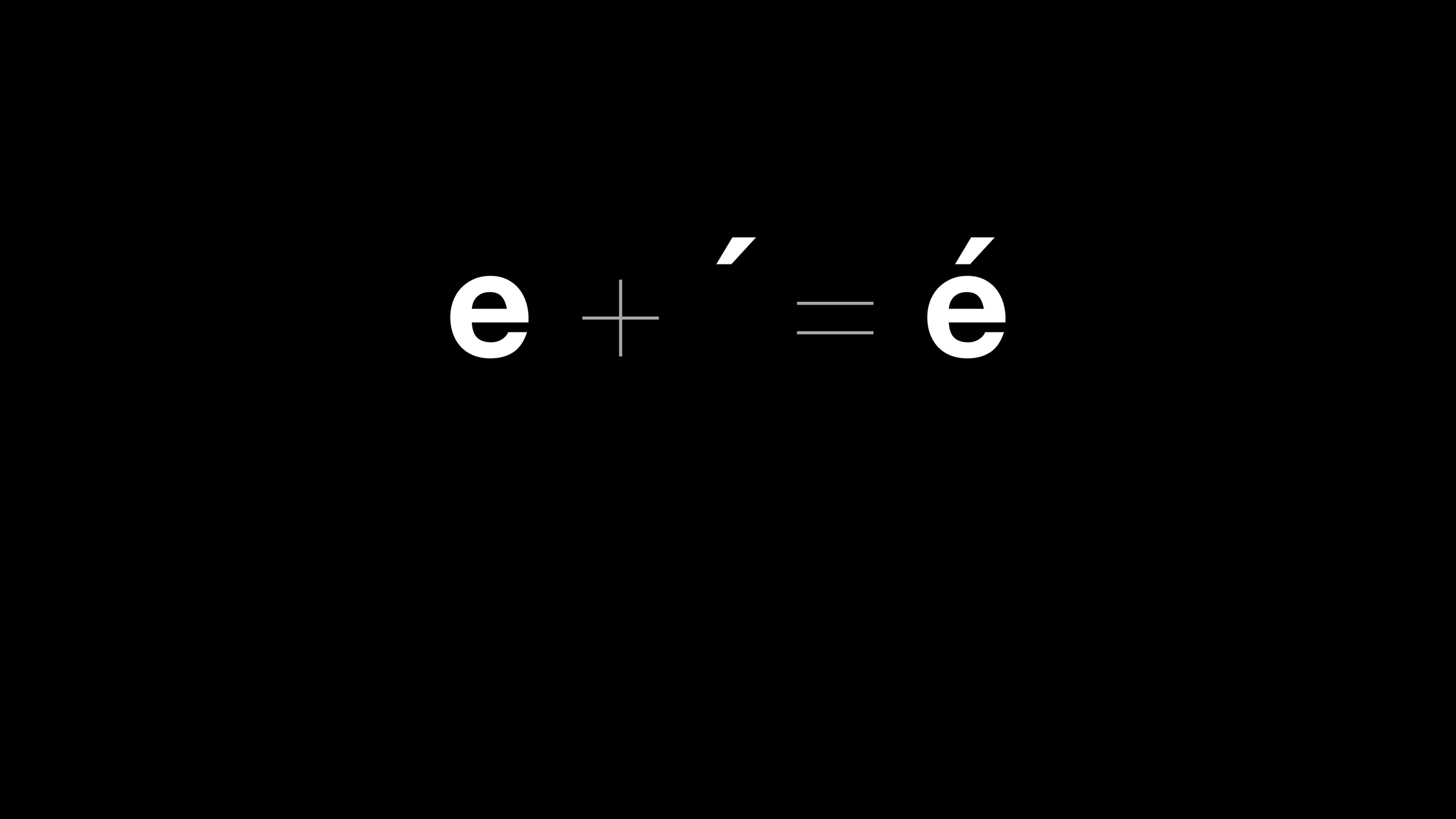
A **Character** consists of one or more **Unicode scalars**

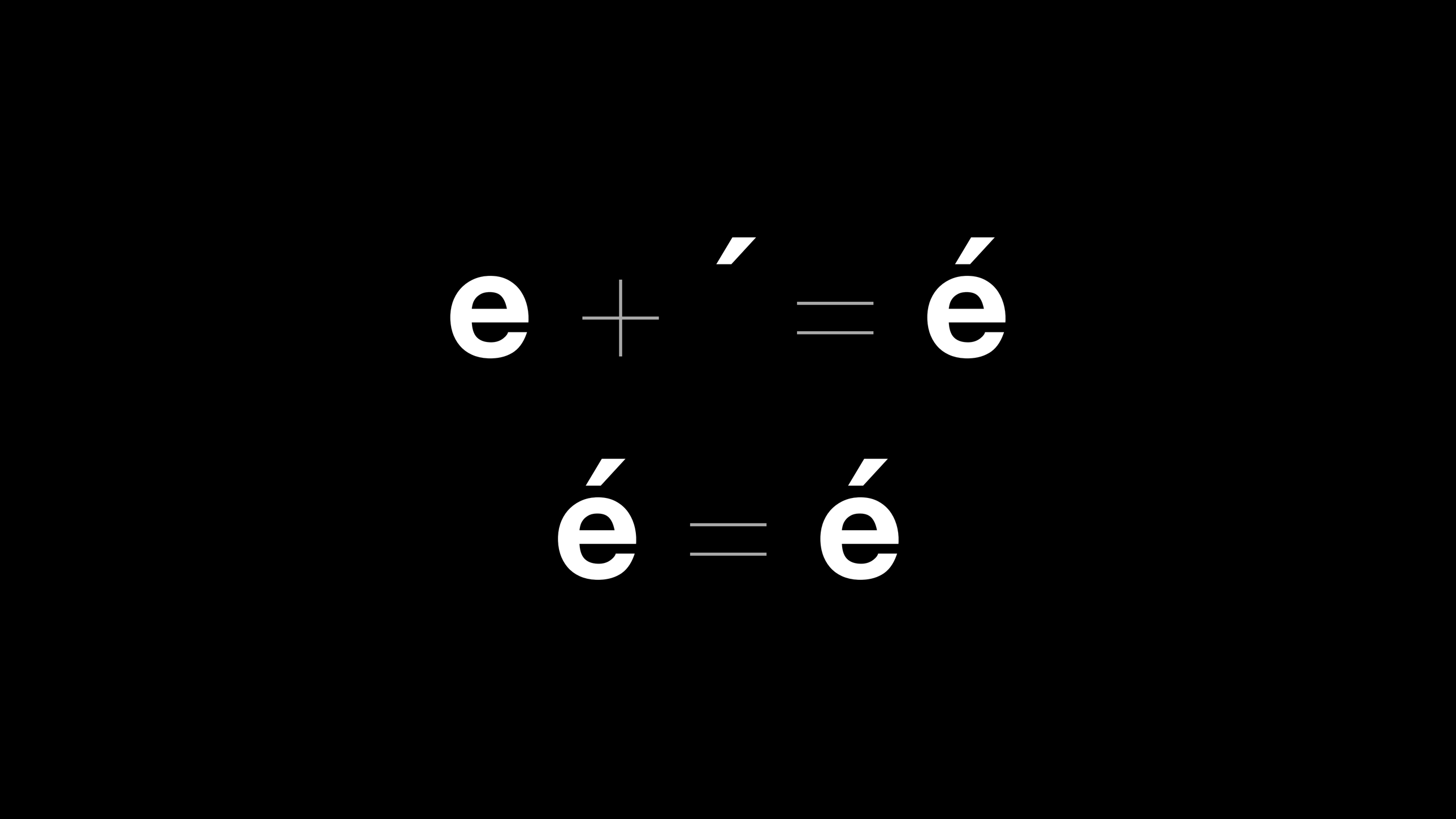
Characters are **encoded** into a **sequence of bytes**

```
let string = "Café 🥖"

for character in string {
    print(character)
}
```



C	a	f	é	SP	


e + ´ = é


e + ´ = é
é = é

```
let character = string[2]
```

```
let character = string[2]
```

```
for character in string {
    print(character)

    for byte in "\(character)".utf8 {
        print(String(byte, radix: 16, uppercase: true))
    }

    for byte in "\(character)".utf16 {
        print(String(byte, radix: 16, uppercase: true))
    }

    for scalar in character.unicodeScalars {
        print(scalar.value)
    }
}
```

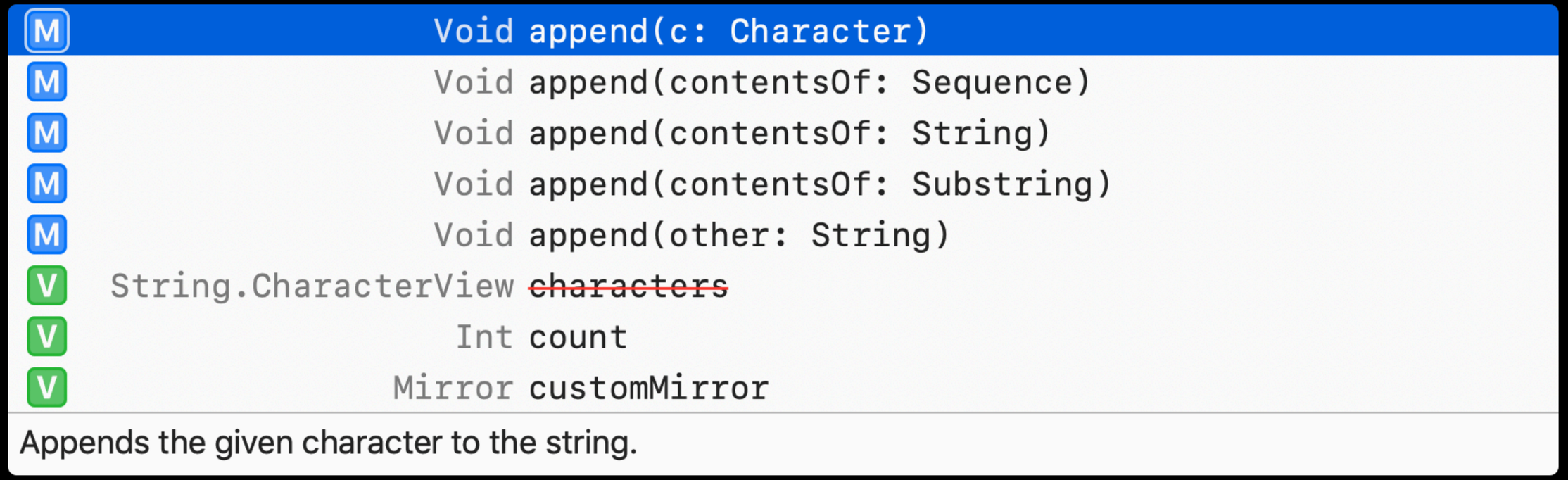
Character	C	a	f	é		SP	🥖			
Codepoint	U+0043	U+0061	U+0066	U+00E9		U+0020	U+1F956			
Scalar Value	67	97	102	233		32	129366			
UTF-8	0x43	0x61	0x66	0xC3	0xA9	0x20	0xF0	0x9F	0xA5	0x96
UTF-16	0x0043	0x0061	0x0066	0x00E9		0x0020	0xD83E		0xDD56	

2 Identifiers

```
struct Product {
    let identifier: String
}
```

```
let coffee = Product(identifier: "☕")
```

```
let coffee = Product(identifier: "☕")

coffee.identifier.
```



```
struct Product {
    struct Identifier: Equatable, Hashable {
        private let value: String
    }

    let identifier: Identifier
}
```

```
let tea = Product(identifier:
            Product.Identifier(value: "🍵")
          )
```

```
let string = "Hello!"
```

```
let id: Product.Identifier = "🍵"
```

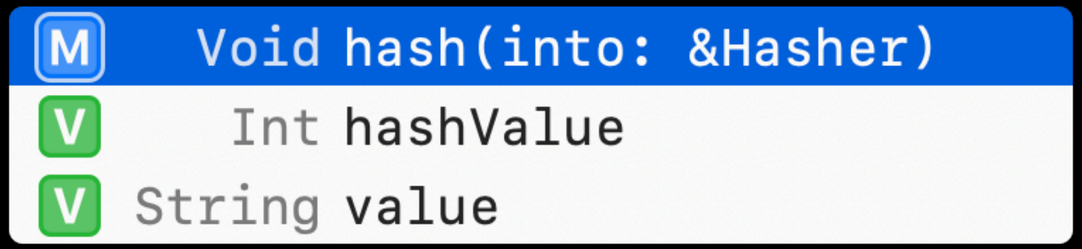
```
extension Product.Identifier:
            ExpressibleByStringLiteral
{
    init(stringLiteral value: String) {
        self.init(value: value)
    }
}
```

```
let id: Product.Identifier = "🍵"
let tea = Product(identifier: id)
```

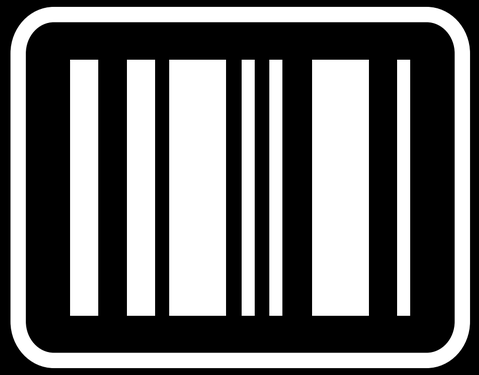
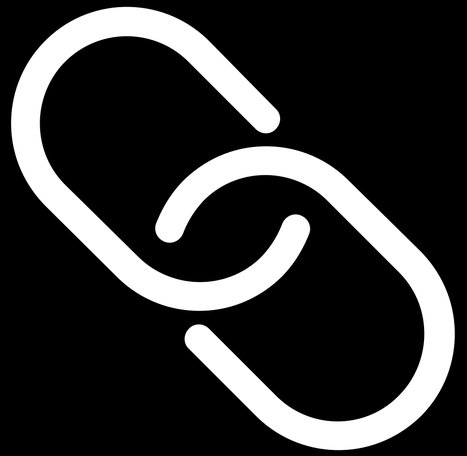
```
let tea = Product(identifier: "🍵")
```

```
let tea = Product(identifier: "🍵")
```

```
let tea = Product(identifier: "🍵")

tea.identifier.
```



```
struct Identifier<Value>: Equatable, Hashable
            where Value : Equatable & Hashable
{
    private let value: Value
}

struct Product {
    let identifier: Identifier<String>
}
```

Validation

```
init(value: String) {
    precondition(!value.isEmpty, "cannot be blank")
    precondition(value.count < 100, "too long")
    for scalar in value.unicodeScalars {
        guard (0x1F344...0x1F37F).contains(scalar.value) else {
            preconditionFailure("must be food emoji")
        }
    }

    self.value = value
}
```

Normalization

```
init(value: String) {
  self.value = value.lowercased()
}
```

3 Paths

scheme://user:password@
hostname:port/path#fragment?query

URL (RFC 3986)

```swift
import Foundation

let url = URL(string: "https://tryswift.co/")!
```

```swift
import Foundation


var components =
  URLComponents(string: "https://tryswift.co/")!

components.path = "/events/2018/nyc/"

components.url
// https://tryswift.co/events/2018/nyc/
```

Percent-Encoding / URI Encoding

```swift
import Foundation

var components =
  URLComponents(string: "https://tryswift.co/")!
components.path = "/events/2018/🗽/"

components.url
// https://tryswift.co/events/2018/%F0%9F%97%BD/
```

%F0%9F%97%BD

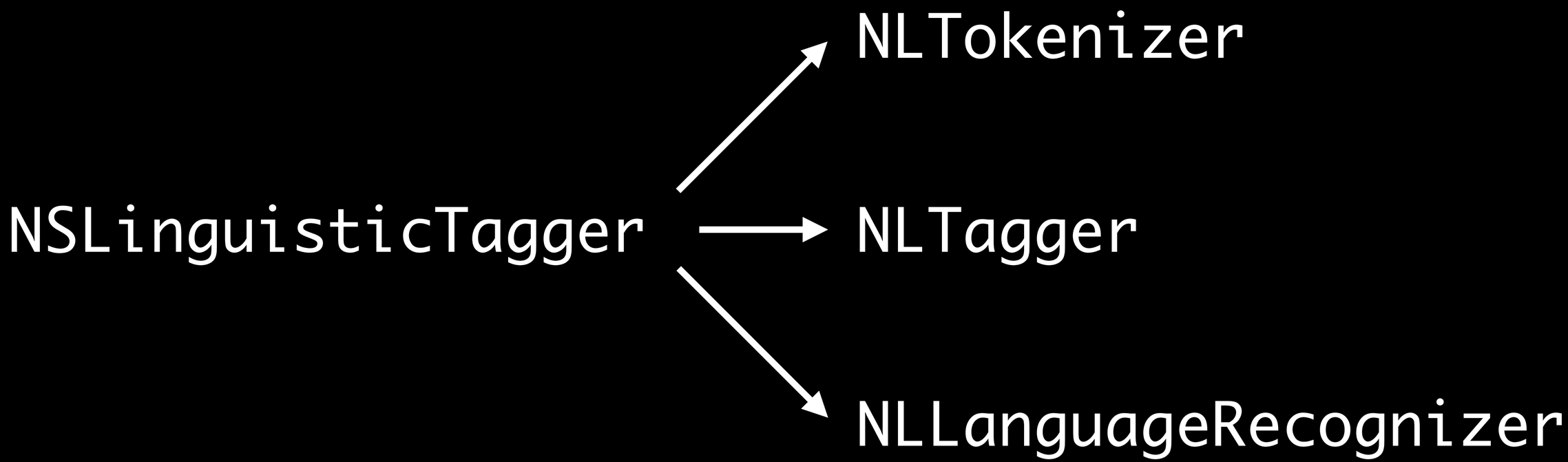
STATUE OF LIBERTY
(U+1F5FD)

UTF-8	F0	9F	97	BD

4

Natural Language Text

Natural Language Framework


NSLinguisticTagger
NLTokenizer
NLTagger
NLLanguageRecognizer

Tokenization

All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.

Tokenization

All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.

Tokenization

All human beings are born free and equal in dignity and rights.
They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.

Tokenization

All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.

Tokenization

All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.

Tokenization

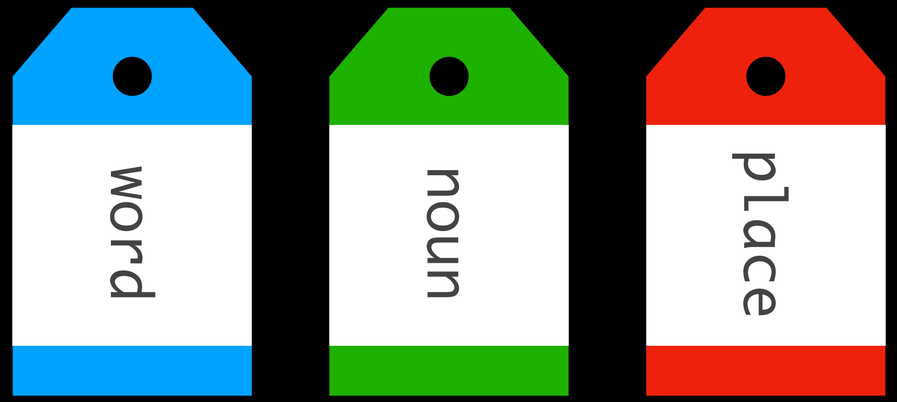
All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.

```
let string = """
 All human beings are born free and equal in dignity and rights.
"""

let paragraphs = string.split(separator: "\n")
let sentences = paragraphs.flatMap{ $0.split(separator: ".") }
let words = sentences.flatMap{ $0.split(separator: " ") }
```

```
let string = """
    人人生而自由,在尊严和权利上一律平等。
"""
```

```
let tokenizer = NLTokenizer(unit: .word)
tokenizer.string = string

let stringRange = string.startIndex..<string.endIndex
tokenizer.enumerateTokens(in: stringRange) {
    (tokenRange, attributes) in

    let token = string[tokenRange]
    // …

    return true // continue processing
}
```

Tagging



Part of Speech Tagging

```
let string = "The quick brown fox jumps over the lazy dog."

let tagger = NLTagger(tagSchemes: [.lexicalClass])
tagger.string = string

let options: NLTagger.Options = [.omitWhitespace, .omitPunctuation]
tagger.enumerateTags(in: string.startIndex..<string.endIndex,
                     unit: .word,
                     scheme: .lexicalClass,
                     options: options) { (tag, range) in
    if let tag = tag, tags.contains(tag) {
        print("\(string[tokenRange]): \(tag.rawValue)")
    }

    return true // continue processing
}
```

Part of Speech Tagging

The	quick	brown	fox	jumped	over	the	lazy	dog
Determiner	Adjective	Adjective	Noun	Verb	Preposition	Determiner	Adjective	Noun

Named Entity Recognition

```
let string = "Tim Cook is the CEO of Apple Inc, based in Cupertino, California."

let tagger = NLTagger(tagSchemes: [.nameType])
tagger.string = string

let options: NLTagger.Options = [.omitPunctuation, .omitWhitespace, .joinNames]
let tags: Set<NLTag> = [.personalName, .placeName, .organizationName]

// ...
```

Named Entity Recognition

Tim Cook	`PersonalName`
Apple Inc	`OrganizationName`
Cupertino	`PlaceName`
California	`PlaceName`

Custom Tagging Model

```
import CreateML

let trainingData: [(tokens: [MLWordTagger.Token], labels: [String])] = [
    (
        ["Check", "out", "my", "new", "iPhone!"],
        ["",      "",    "",   "",    "“"]
    ),
    (
        ["How", "do", "you", "like", "your", “iPad?"],
        ["",    "",   "",    "",     "",     "“"]
    )
]

let wordTagger = try MLWordTagger(trainingData: trainingData)
```

Custom Tagging Model

```
import NaturalLanguage

let customModel = try NLModel(mlModel: wordTagger.model)

let appleProductScheme = NLTagScheme(rawValue:  "appleProduct")


let string = "Try out the app on your iPhone or iPad.”

let tagger = NLTagger(tagSchemes: [appleProductScheme])
tagger.setModels([customModel], forTagScheme: appleProductScheme)
tagger.string = string
```

iPhone	
iPad	

Trånslîtérätiôn

Transliteration

```
import Foundation

let string = "Tråñslîtérätiôn"

string.applyingTransform(.stripDiacritics,
                         reverse: false)
// Transliteration
```

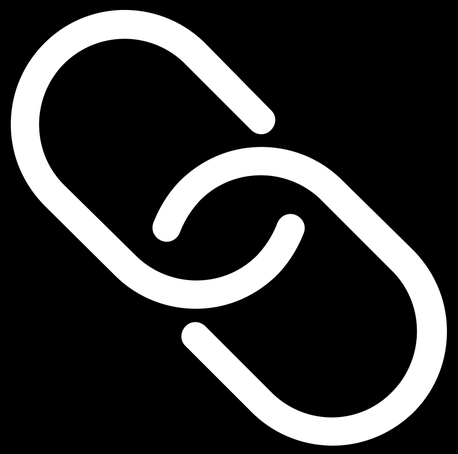
```swift
import Foundation

let string = "모든 인간은 태어날 때부터 자유로우며 그 존엄과 권리에 있어 동등하다."

string.applyingTransform(.toLatin, reverse: false)
/* "modeun ingan-eun taeeonal ttaebuteo jayuloumyeo
    geu jon-eomgwa gwonlie iss-eo dongdeunghada."   */
```


Search

5

Structured Data




```
import Foundation

let string = """
    try! Swift NYC starts at 9 AM on September 5th.
    The venue is at 340 W 50th St.
    For more information, see https://tryswift.co/
"""

let detector =
    try NSDataDetector(types: NSTextCheckingAllTypes)
```

```
detector.enumerateMatches(in: string,
                          options: [],
                          range: range) { (match, flags, _) in
    guard let match = match else {
        return
    }

    switch match.resultType {
    case .date:
        let date = match.date
        print(date)
    case .address:
        let street = match.components?[.street]
        print(street)
    case .link:
        let url = match.url
        print(url)
    default:
        return
    }
}
```

2018-09-05 13:00:00 +0000

340 W 50th St.

https://tryswift.co/

6

Encoded Data

	0	1	2	3	4	5	6	7
0	A	B	C	D	E	F	G	H
1	I	J	K	L	M	N	O	P
2	Q	R	S	T	U	V	W	X
3	Y	Z	a	b	c	d	e	f
4	g	h	i	j	k	l	m	n
5	o	p	q	r	s	t	u	v
6	w	x	y	z	0	1	2	3
7	4	5	6	7	8	9	+	/

Base64

All human beings are born free and equal in dignity and rights.


```
QWxsIGh1bWFuIGJlaW5ncy
BhcmUgYm9ybiBmcmVlIGFu
ZCBlcXVhbCBpbiBkaWduaX
R5IGFuZCByaWdodHMu
```

```swift
import Foundation

let string = "Hello, world!"

let utf8Data = string.data(using: .utf8)!
let base64EncodedString = utf8Data.base64EncodedString()
// "ICAgIEhlbGxvLCB3b3JsZCE="

let base64DecodedData =
    Data(base64Encoded: base64EncodedString)!
let decodedString = String(data: base64DecodedData,
                           encoding: .utf8)
// "Hello, world!
```


```
To: to@swift.org
From: from@swift.org
Subject: MIME Email Example
MIME-Version: 1.0
Content-Type: multipart/alternative;boundary = "+++"
Message-Id: <20180901000000.123456789@iMac.local>
Date: Wed, 5 Sep 2018 16:10:00 -0500 (EST)

--+++
Content-Type: text/txt; charset=utf-8
Content-Transfer-Encoding: base64

QWxsIGh1bWFuIGJlaW5ncy+
BhcmUgYm9ybiBmcmVlIGFu+
ZCBlcXVhbCBpbiBkaWduaX+
R5IGFuZCByaWdodHMu
```

```
@font-face {
    font-family: 'CustomTypeface';
    src: url(data:font/woff2;
             charset=utf-8;
             base64,…==)
         format('woff2'),
    font-weight: normal;
    font-style: normal;
}
```


```
import Foundation
import UIKit

let base64EncodedPNG = """
iVBORw0KGgoAAAANSUhEUgAAAAEAAAAB
CAAAAAA6fptVAAAACklEQVR4nGP6DwAB
BQECz6AuzQAAAABJRU5ErkJggg==
"""

let data = Data(base64Encoded: base64EncodedPNG)!
let image = UIImage(data: data)
```

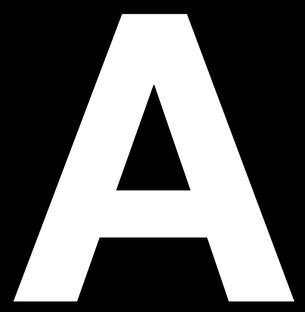
7

Typographic Information

Characters

≠

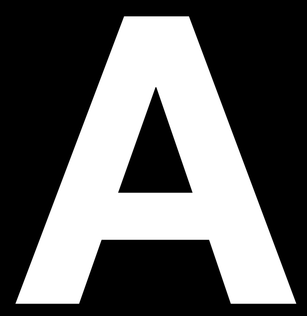
Glyphs



LATIN CAPITAL LETTER A

(U+0041)


A
LATIN CAPITAL LETTER A
(U+0041)

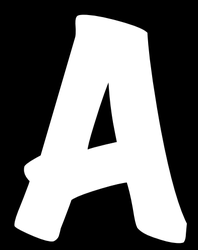


LATIN CAPITAL LETTER A

(U+0041)

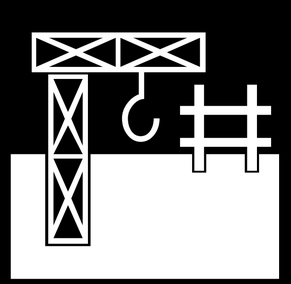
A LATIN CAPITAL LETTER A
(U+0041)

Georgia

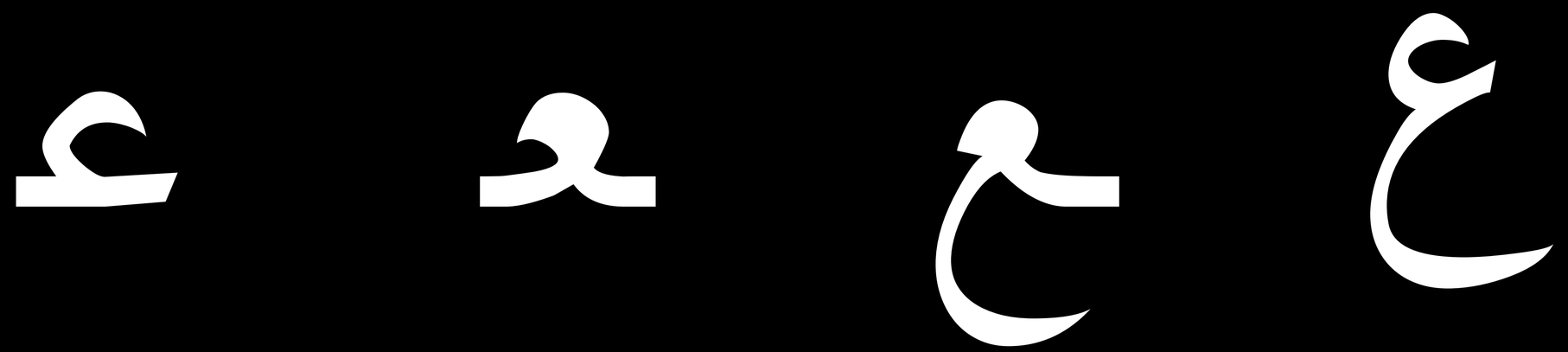


Marker Felt

Zapfino



Webdings

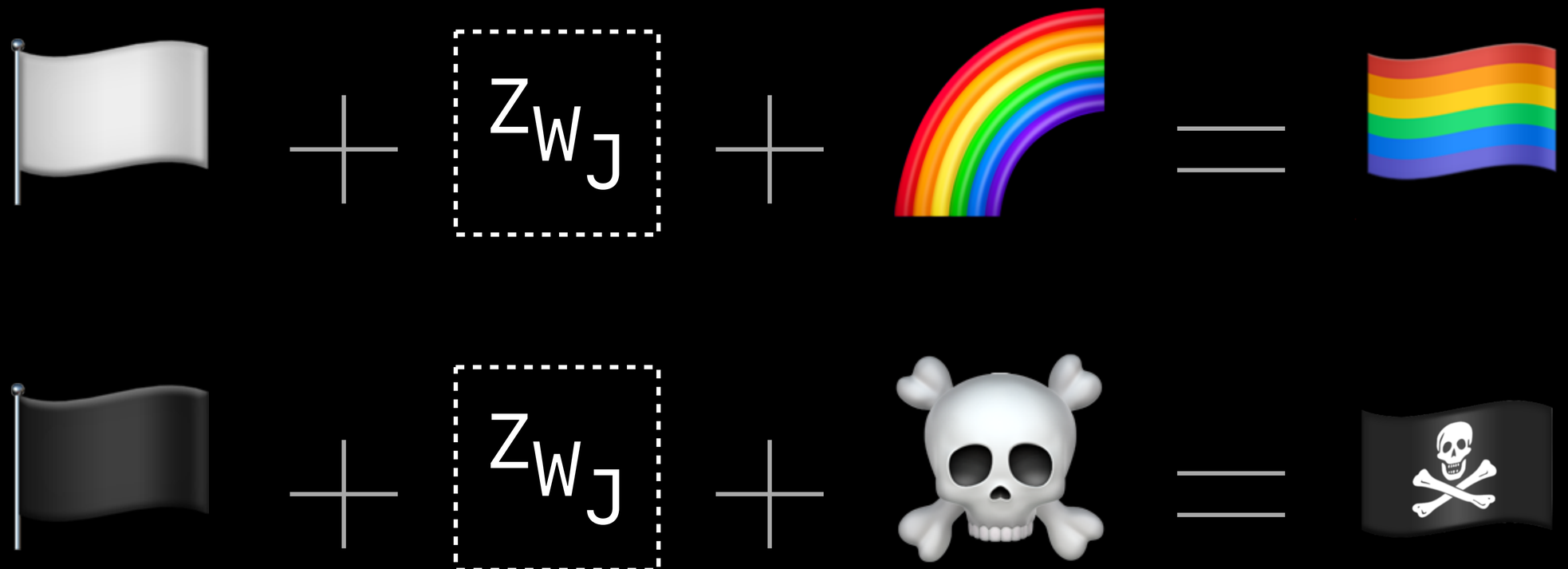


Initial Medial Final Isolated

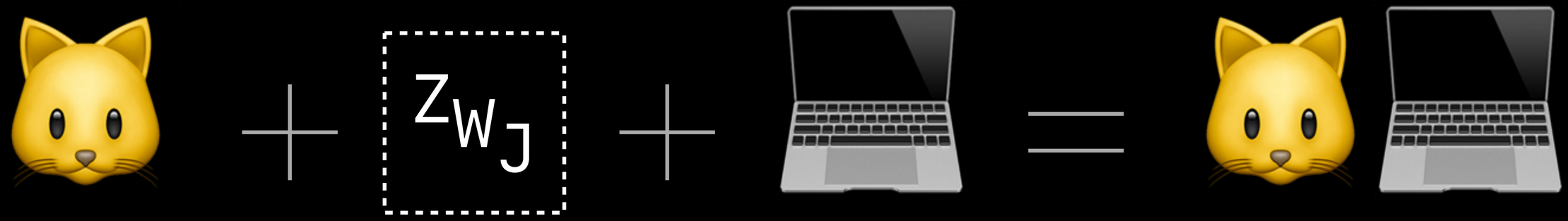
ffi

ffi

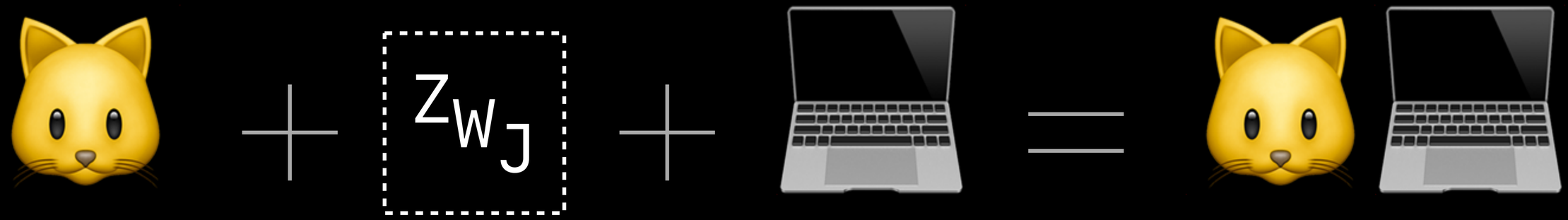
U + S = 


ZWJ
ZWJ

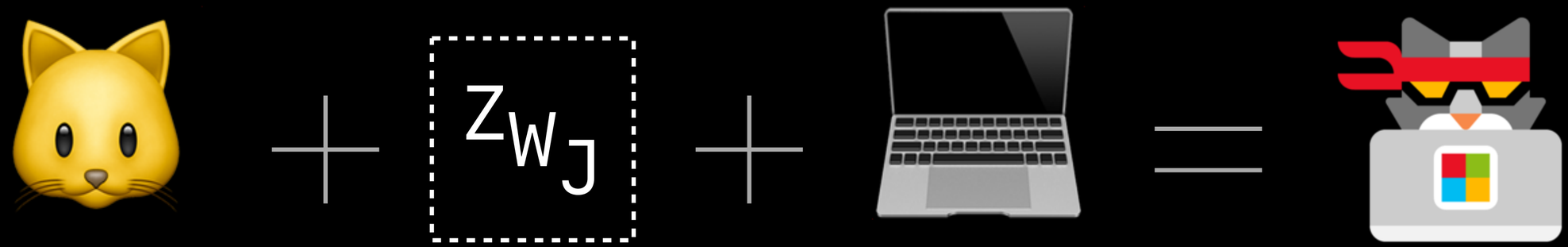
Apple Platforms

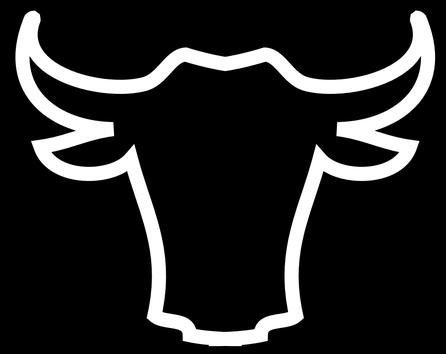


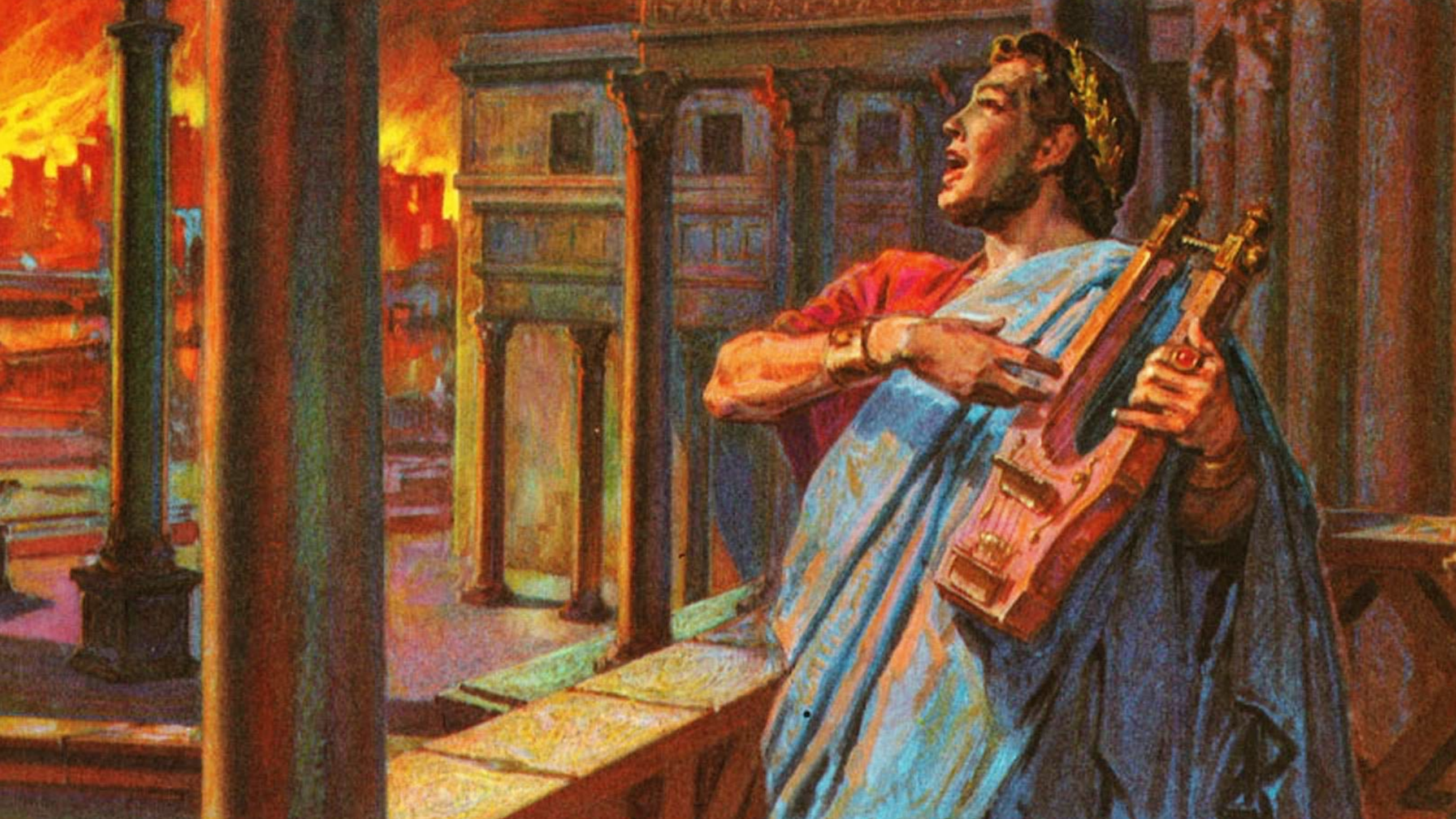
Apple Platforms

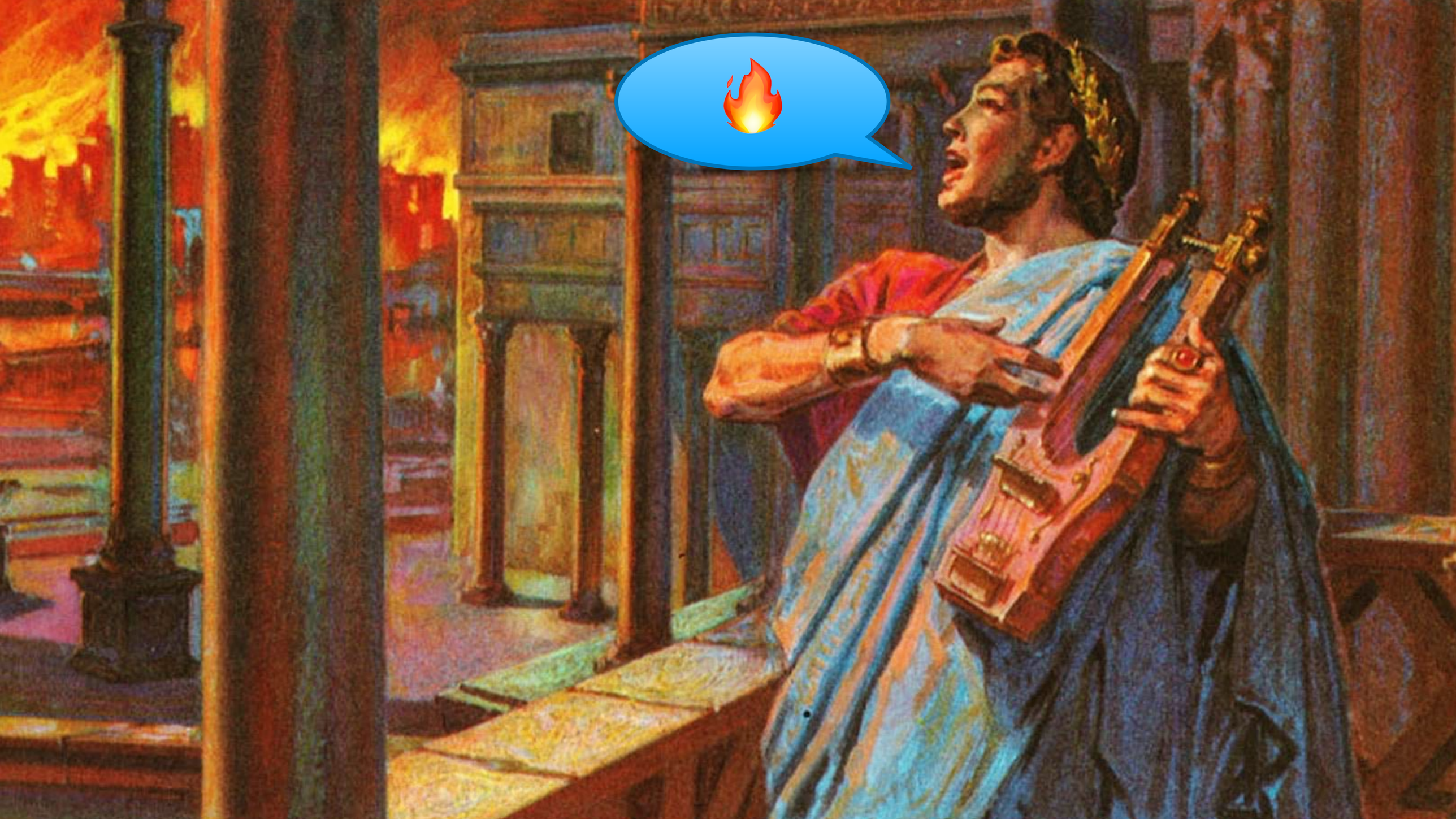


Microsoft Platforms



 ≠ A A 





Thanks&
Thanks&
Thanks&
Thanks&
Thanks&
Thanks&
Thanks