

- ❑ Large character sets
- ❑ **Complex script rendering**
 - ❑ Combining characters
 - ❑ Context-sensitive glyph shaping
 - ❑ More character to glyph rendering
- ❑ Text direction
- ❑ Linguistic boundaries, line breaking & justification
- ❑ Other typographic & implementation issues
- ❑ Sorting & case conversion

Complex script rendering

▶ Important definition !

Character

vs.

Glyph

a *a*

雪 雪

all

Before beginning the next section it is important to draw attention to the difference between characters and glyphs.

A character is a semantic unit representing an indivisible unit of text in memory.

A glyph is the visual representation of a character or sequence of characters.

The example on the slide shows two glyphs for the single character *a*, and two glyphs for the single character *g*.

This distinction will become very important in this section. For more information about the distinction between characters and glyphs, see Unicode Technical Report #17.

Combining characters

Combining characters

Arabic
Hebrew

▶ Arabic and Hebrew short vowels



כאשר העולם רוצה לדבר, הוא מדבר ב־Unicode. הירשמו כעת לכנס Unicode הבינלאומי העשירי, שייערך בין התאריכים 10.3.97, בְּמַיִן שבגרמניה. בכנס ישתתפו מומחים מכל ענפי התעשייה בנושא האינטרנט העולמי וה־Unicode, בהתאמה לשוק הבינלאומי והמקומי, ביישום Unicode במערכות הפעלה וביישומים, בגופנים, בפריסת טקסט ובמחשוב רב־לשוני.


Arabic and Hebrew scripts usually do not represent short vowel sounds. The languages are so heavily pattern based that readers can adequately guess at the pronunciation of the words.

In circumstances where ambiguity appears, such as the name of the German town Mainz in the example above, short vowels are represented as diacritics attached to the base consonants.

Combining characters

▶ Arabic and Hebrew short vowels

Arabic
Hebrew



The image shows a slide from a presentation. On the left, there is a vertical blue bar with the text 'Combining characters' written vertically. To the right of this bar, there is a light blue header area containing a right-pointing triangle followed by the text 'Arabic and Hebrew short vowels'. In the top right corner of the slide, the words 'Arabic' and 'Hebrew' are stacked vertically. The main body of the slide is white and contains the Arabic word 'مهندس' (muhandis) written in a blue, cursive script.

Here for example is the Arabic word for engineer, pronounced 'muhandis'.
It is actually written, 'mhnds'.

Combining characters

Arabic
Hebrew

▶ Arabic and Hebrew short vowels

م + ◌ + ه + ◌ + ن + ◌ + د + ◌ + س

If needed, the short vowels (there are only 3 in Arabic) are represented as shown on the lower line of the slide. Note that the small circle diacritic indicates NO intervening vowel. (Sequences of code points in Arabic and Hebrew on this and following slides will be shown in left to right order, to emphasise that the underlying order is logical.)

These short vowels are separate *combining characters* in the text stream that are displayed in the same two-dimensional visual space with the base character. Combining characters do not generally appear without a base character.

▶ Context-sensitive placement of diacritics

Arabic
Hebrew
Thai
Indic


ณ ยามที่โลกต้องการอยู่ด้วยคำใดๆ โลกจะใช้เพียง Unicode เราจึงขอเชิญชวนท่านรับลงทะเบียนงาน International Unicode Conference ครั้งที่ 10 ซึ่งจะจัดให้มีขึ้น ณ เมือง Mainz ประเทศเยอรมัน ในระหว่างวันที่ 10-12 มีนาคม ค.ศ. 1997 เสียแต่บัดนี้ โดยในงานประชุมดังกล่าว ท่านจะมีโอกาสได้พบกับบรรดาผู้เชี่ยวชาญจากธุรกิจอินเทอร์เน็ตและ Unicode ธุรกิจ Internationalization และ Localization จากทุกมุมทั่วโลก พร้อมรับทราบการใช้ประโยชน์จาก Unicode ร่วมกับระบบปฏิบัติการและโปรแกรมต่างๆ ฟอนต์ รูปแบบข้อความ รวมทั้งวิทยาการด้านคอมพิวเตอร์ในภาษาต่างๆ

When displaying combining characters care has to be given to appropriate positioning. In the Thai example above, the SAME character code is used to represent both the tone mark glyphs I have circled. There are not two different characters based on the desired visual position. The font has to work out the best position for the glyph according to the run-time visual context.

Combining characters

► Context-sensitive placement of diacritics

Arabic
Hebrew
Thai
Indic

Here is another example of context-sensitive positioning of combining characters.

The short vowel 'i' in Arabic is usually drawn below the base character. This is normally the only way of distinguishing it from the short vowel 'a', which is displayed above the base character.

In this example, however, an additional *shadda* diacritic is introduced. The shadda is used to lengthen the consonant it is attached to. In that context it is common (though not mandatory) for the 'i' vowel diacritic to appear ABOVE the base character, but BELOW the shadda so you can still tell it apart from 'a'.

Note also that this example introduces the idea that you can have more than one combining character associated with a base character.

Combining characters

Thai
Indic

▶ Vowel signs

क [kə]

की [ki:]

के [ke]

कू [ku:]

In Indic scripts and scripts derived from them a consonant character carries with it an ‘inherent vowel’. The character on the top line above is transcribed ‘ka’, not just ‘k’.

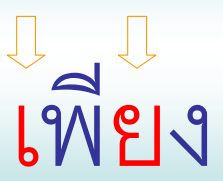
If you want to follow the ‘k’ sound with a different vowel, you append a *vowel sign* to the consonant character. This vowel sign overrides the inherent vowel with a different sound.

In Indic scripts vowel signs are all combining characters. Unlike the Arabic and Hebrew short vowels, however, some of these combining characters may also take up additional space on a line (see the example ‘kii’ above). They are referred to as *spacing combining characters*.

Combining characters

Thai
Indic

▶ Vowel signs



ณ ยามที่โลกต้องการอยู่ด้วยคำใดๆ โลกจะใช้เพียง Unicode เราจึงขอ
เชิญชวนท่านรับลงทะเบียนงาน International Unicode Conference ครั้งที่
10 ซึ่งจะจัดให้มีขึ้น ณ เมือง Mainz ประเทศเยอรมัน ในระหว่างวันที่ 10-12
มีนาคม ค.ศ. 1997 เสียแต่บัดนี้ โดยในงานประชุมดังกล่าว ท่านจะมีโอกาส
ได้พบกับบรรดาผู้เชี่ยวชาญจากธุรกิจอินเทอร์เน็ตและ Unicode ธุรกิจ
Internationalization และ Localization จากทุกมุมทั่วโลก พร้อมรับทราบ
การใช้ประโยชน์จาก Unicode ร่วมกับระบบปฏิบัติการและโปรแกรมต่างๆ
ฟอนต์ รูปแบบข้อความ รวมทั้งวิทยาการด้านคอมพิวเตอร์ในภาษาต่างๆ

Thai, being derived from Indic scripts, also has vowel signs, although they are used in a slightly more complex way.

In the example above, three vowel signs surround the consonant **พ** to produce the desired effect. Whereas in the Indic scripts all vowel signs are combining characters, only one of the above is combining. The other two (indicated by arrows) are normal spacing characters. This is a distinction introduced to Unicode at the request of the Thai national standards body.

Combining characters

► Coding combining characters

All

vila^og ? vil^oag ?

Ha a világ beszélni akarna, Unicode-ul szólalna meg. Regisztráljon már most a Tizedik Nemzetközi Unicode Konferenciára, melyet 1997. március 10-12-én rendeznek Mainz-ban, Németországban. Ezen a konferencián az iparág több neves szakértője is résztvesz. Ízelítőül a témákból: a világháló és a Unicode nemzetközisítése és lokalizálása, a Unicode alkalmazása működő rendszerekben és alkalmazásokban, szövegrendezésnél, és többnyelvű számítógépeken.

When it comes to implementing combining characters, an important question to ask is what order should be applied to them and the base character. Unless you have agreement on this, you can have serious problems when passing data between systems.

The Unicode Standard requires that all combining characters FOLLOW the base consonant in a Unicode string. (So the example to the left above is correct.)

Combining characters

► Coding combining characters

All

$$\text{e}^{\sim\hat{}} = \text{e} + \hat{\text{O}} + \tilde{\text{O}}$$

$$\neq \text{e} + \tilde{\text{O}} + \hat{\text{O}}$$

Much of the time there are no rules about the order of multiple combining characters. If those characters interact typographically, however, as in the case above to produce different possible results, then the 'inside-out' rule is applied.

This rule states that the proximity of the combining character in the text stream must match the visual proximity.

Combining characters

▶ Precomposed vs. decomposed

All

világ ? = vila^og ?

Ha a világ beszélni akarna, Unicode-ul szólalna meg. Regisztráljon már most a Tizedik Nemzetközi Unicode Konferenciára, melyet 1997. március 10-12-én rendeznek Mainz-ban, Németországban. Ezen a konferencián az iparág több neves szakértője is résztvesz. Ízelítőül a témákból: a világháló és a Unicode nemzetközisítése és lokalizálása, a Unicode alkalmazása működő rendszerekben és alkalmazásokban, szövegelrendezésnél, és többnyelvű számítógépeken.

There are many *precomposed* characters in Unicode that have an accent or diacritic already combined with a base character (such as a-acute above). It is however also possible to represent this character using a simple 'a' followed by a combining acute accent. This is referred to as a *decomposed* character sequence.

The Unicode Standard states that both of these approaches must be considered canonically equivalent.

Combining characters

All

▶ Normalisation

NFC

Ízelítőül

NFD

Ízeliótoóüö

Ha a világ beszélni akarna, Unicode-ul szólalna meg. Regisztráljon már most a Tizedik Nemzetközi Unicode Konferenciára, melyet 1997. március 10-12-én rendeznek Meinz-ban, Németországban. Ezen a konferencián az iparág több neves szakértője is résztvesz. **Ízelítőül** a témákból: a világháló és a Unicode nemzetközisítése és lokalizálása, a Unicode alkalmazása működő rendszerekben és alkalmazásokban, szövegelrendezésnél, és többnyelvű számítógépeken.

To facilitate the process of string comparison for operations such as searching, sorting and comparison it is helpful to adopt a standard policy with regard to precomposed versus decomposed variants of a character sequence, and the order in which multiple combining characters appear. This can be achieved by applying an appropriate *normalization form*. The Unicode Standard provides a normalization form called NFD that represents all character sequences in maximally decomposed form. In addition to decomposition, NFD applies a standard order to multiple composing characters attached to a base character. As an alternative, the Unicode Standard offers NFC. NFC is achieved by applying NFD to the text, then re-composing characters for which precomposed forms exist in version 3.0 of the standard.

Note that there are actually some precomposed forms in the Unicode character set that are *not* generated by NFC, for reasons we will not go into here. In addition, where there is no precomposed form, a character sequence is left decomposed, but canonical ordering is still applied to all combining characters.

The World Wide Web Consortium is recommending that all Web content be stored in NFC, to assist in efficient and correct processing.

The Unicode Standard also offers two more normalization forms, NFKD and NFKC, where K stands for 'kompatibility'. These forms are provided because the Unicode character set includes many characters merely to provide round-trip compatibility with other character sets. Such characters represent such things as glyph variants, shaped forms, alternative compositions, and so on, but can be represented by other 'canonical' versions of the character or characters already in Unicode. Ideally, such compatibility variants should not be used. The NFKD and NFKC normalization forms replace them with more appropriate characters or character sequences. (This, of course, can cause a problem if you intend to convert data back into its original encoding, because you lose the original information.)

Context-sensitive glyph shaping

Context-sensitive glyph shaping

► Word-final glyph variants

Hebrew
Greek

כאשר העולם רוצה לדבר, הוא מדבר ב-Unicode. הירשמו כעת לכנס Unicode הבינלאומי העשירי, שייערך בין התאריכים 10-12 במרץ 1997, בקמפיין שבגרמניה. בכנס ישתתפו מומחים מכל ענפי התעשייה בנושא האינטרנט העולמי וה-Unicode, בהתאמה לשוק הבינלאומי והמקומי, ביישום Unicode במערכות הפעלה וביישומים, בגופנים, בפריסת טקסט ובמחשוב רב-לשוני.

מומחים

In Hebrew and Greek there are certain characters (only a small number) that look different in the middle of a word and at the end. Two examples are shown above. In each example, the same consonant appears in the middle of a word and at the end of a word in the sample text, and has a different appearance.

Due to traditional approaches, these shapes are encoded separately and are typed in using distinct keys on the keyboard. This is manageable because there are so few such characters.

In other scripts a very different approach has to be taken.

Arabic

▶ Cursive script

Context-sensitive glyph shaping

ع

وسيجمع

عندما يريد العالم أن يتكلم، فهو يتحدث بلغة يونيكود. تسجل الآن لحضور المؤتمر الدولي العاشر ليونيكود (Unicode Conference)، الذي سيعقد في 10-12 آذار 1997 بمدينة مانتس، ألمانيا. وسيجمع المؤتمر بين خبراء من كافة قطاعات الصناعة على الشبكة العالمية انترنت ويونيكود، حيث ستم، على الصعيدين الدولي والمحلي على حد سواء مناقشة سبل استخدام يونيكود في النظم القائمة وفيما يخص التطبيقات الحاسوبية، الخطوط، تصميم النصوص والحوسبة متعددة اللغات.

متعددة

على

Arabic is often referred to as a 'cursive' script with the meaning that letters in a word are usually joined to each other – whether handwritten or printed.

The slide shows the unjoined form of the letter AIN at the top right, and, at the bottom, three joined examples of of the same letter. As you can see, the shape changes quite dramatically.

Context-sensitive glyph shaping

► Cursive script

Arabic

Here are some more examples of un-joined Arabic letters (right column) and their various joining forms (to the left).

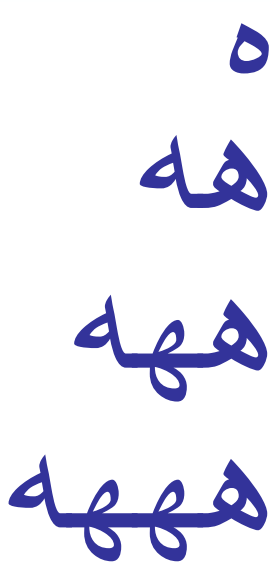
It is important to understand that there is only ONE code point here for each letter. The various different visual forms are only font-based glyphs chosen to suit the run-time visual context.

(There are compatibility characters encoded in Unicode for specific joining forms, but these should not be used for storing Arabic text.)

The shapes above can be referred to (from right to left) as *independent*, *initial*, *medial* and *final*.

Context-sensitive glyph shaping

▶ Inputting context-sensitive glyphs



Arabic

On previous slides I mentioned the 'run-time' context. This is quite important. If I type in the Arabic letter HEH shown at the top of the slide it will initially be in an independent glyph form. If I press exactly the same key on the keyboard and insert exactly the same character alongside it in memory, however, the original letter HEH will be expected to join with the second HEH. The shape of the first HEH will therefore change to 'initial', and the second HEH will be in 'final' shape. Type another HEH and the second will become 'medial', and so on.

In this way Arabic text is constantly changing as you type. The editing application also has to adapt these glyphs as you do things such as backspace, insert or delete text.

Context-sensitive glyph shaping

Indic

▶ **Conjunct consonants**

ङ + क → ङ्क

क + ष → क्ष

ल + ल → ल्ल

When two Indic consonants appear together without any intervening vowel sound they may form a *conjunct*, ie. the consonant cluster is rendered as a composite shape. This composite shape may show a vertical or horizontal mixture of the base shapes. In some cases the original constituents of a conjunct may not be recognizable.

One approach that is very common is the use of a *half-form* to represent the initial consonant in the cluster. An example of this is shown on the bottom line of the slide.

It is important to bear in mind, once again, that this is all glyph magic. The individual consonants are all still represented using the regular code points in memory, it is only the visual appearance that changes. There are no special code points for half-form glyphs. The appropriate glyph is simply applied at display time according to the rendering rules of the script.

Context-sensitive glyph shaping

Indic

► Conjunct consonants

Codes in memory			Display
க	+	க	→ கக
க	+	்	→ க்
		க	→ கக

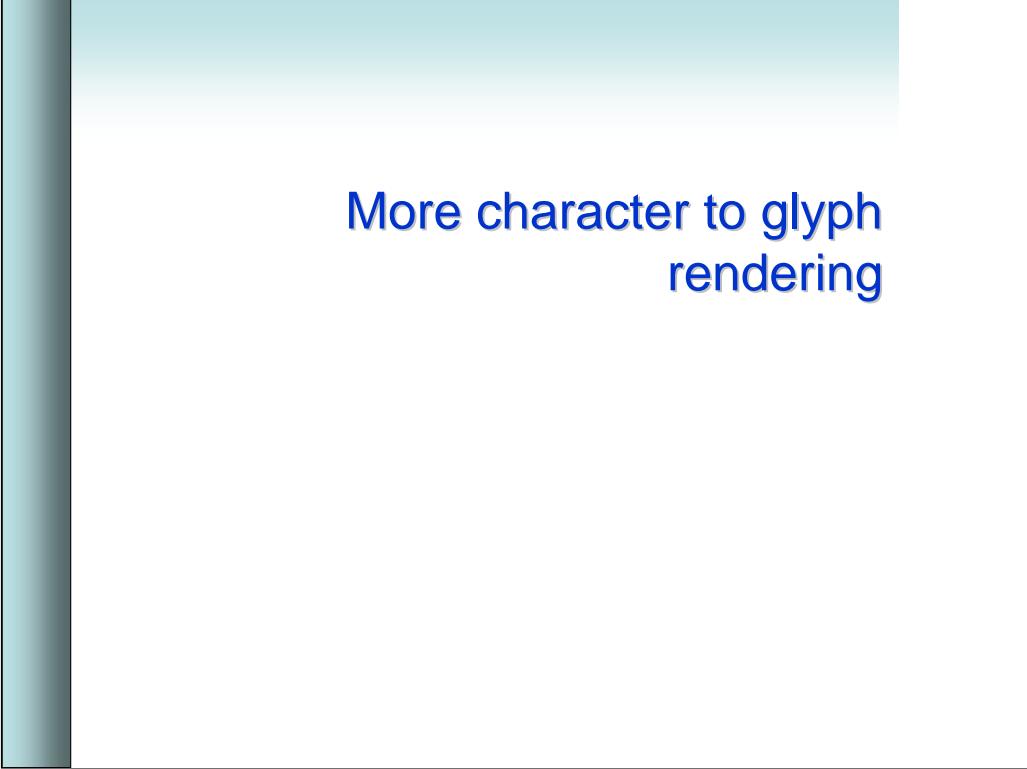
In actual fact, there is a vital ingredient to a conjunct form that we have not yet discussed. It is called a *virama*. The virama is often called 'vowel killer'.

If you simply put two consonants side by side in Unicode, as in the top line above, you will get two separate consonants displayed (with the assumption on the part of the reader that there is an inherent vowel between them).

It is only when you put a virama character between them that they combine to form a conjunct. So the conjunct glyph shown middle right actually represents three underlying characters.

The number of conjunct forms can vary from font to font. Some fonts will be capable of rendering more than others. So what happens if the font you are using doesn't have a conjunct glyph for the combination you want to create?

In this case the virama is shown visually as a combining mark – see the last line above. (In fact, in modern Tamil this is the default approach.)



More character to glyph
rendering

► Special joining forms

Arabic
Indic

بين خبراء بين خبراء

عندما يريد العالم أن يتكلم، فهو يتحدث بلغة يونيكود. تسجل الآن لحضور المؤتمر الدولي العاشر ليونيكود (Unicode Conference)، الذي سيعقد في 10-12 آذار 1997 بمدينة ماينتس، ألمانيا. وسيجمع المؤتمر بين خبراء من كافة قطاعات الصناعة على الشبكة العالمية انترنت ويونيكود، حيث ستم، على الصعيدين الدولي والمحلي على حد سواء مناقشة سبل استخدام يونيكود في النظم القائمة وفيما يخص التطبيقات الحاسوبية، الخطوط، تصميم النصوص والحوسبة متعددة اللغات.

The concepts we have discussed so far in this section on combining characters and glyph shaping have shown that there is no one-to-one correspondence, as there usually is in English, between the characters in memory and the glyphs displayed on screen. Indeed, sometimes complex rules are needed to determine the displayed result.

We have seen some of the more basic transformation cases, but over the next few slides we will take a quick look at some additional possibilities. This is by no means intended to give you all the information you need to implement these scripts – merely expose you to some slightly more advanced behavior.

First out we look at some font-dependent alternatives for joining Arabic glyphs. Arabic glyphs typically join along the baseline, but in some (typically more classical) fonts, specific pairings join above the baseline as shown top left.

More character to glyph rendering

▶ Special joining forms

Arabic
Indic

ल + ् + ल → ल्ल

The use of half-forms in Indic scripts could also be seen as a kind of special joining form.

More character to glyph rendering

► Positional variation

Indic

हिंदी

ह + ि + ं + द + ी

Spacing combining characters to the left of the base consonant are common in Indic scripts. Here what is important to bear in mind is that the Unicode rule about combining characters following the base character still applies. It is only as part of the rendering process that the glyph for the combining character is made to appear to the left.

The example above shows how the Hindi word for 'Hindi' would normally be displayed, but on the second line shows the order of the characters in memory.

► Positional variation

Indic

โปรแกรม

ณ ยามที่โลกต้องการเอ่ยถ้อยคำใดๆ โลกจะใช้เพียง Unicode เราจึงขอเชิญชวนท่านรีบลงทะเบียนงาน International Unicode Conference ครั้งที่ 10 ซึ่งจะจัดให้มีขึ้น ณ เมือง Mainz ประเทศเยอรมัน ในระหว่างวันที่ 10-12 มีนาคม ค.ศ. 1997 เสียแต่บัดนี้ โดยในงานประชุมดังกล่าว ท่านจะมีโอกาสได้พบกับบรรดาผู้เชี่ยวชาญจากธุรกิจอินเทอร์เน็ตและ Unicode ธุรกิจ Internationalization และ Localization จากทุกมุมทั่วโลก พร้อมรับทราบการใช้ประโยชน์จาก Unicode ร่วมกับระบบปฏิบัติการและโปรแกรมต่างๆ ฟอนต์ รูปแบบข้อความ รวมทั้งวิทยาการด้านคอมพิวเตอร์ในภาษาต่างๆ

This text from the Thai sample shows the same effect in Thai. This word is pronounced very much like 'program', and the vowel sign at the far left is actually pronounced after the third character (ie. it is the 'o' sound after 'pr').

We have already seen, however, that vowel signs are not necessarily combining in Thai, so no reordering is actually needed in this case. The characters displayed are actually stored in the same order in memory.

More character to glyph rendering

► Positional variation

Indic

கௌ

க + ௌ

ईर्ष्या

ई + र + ्र + ष + ्र + य + ा

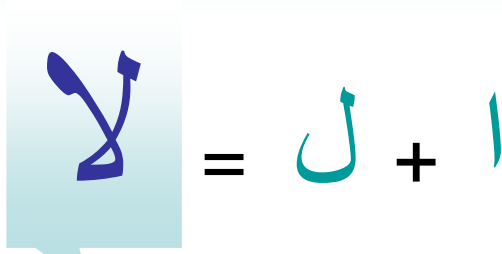
This slide shows some additional examples of reordering during display.

The top example shows a Tamil combining character that appears on BOTH sides of the base consonant when displayed.

The bottom example shows the Devanagari 'repha' in a consonant cluster.

The RA code that appears at the beginning of the cluster in memory is rendered as a diacritic above the vowel sign that completes the syllabic cluster.

▶ Ligatures

Arabic
Indic

عندما يريد العالم أن يتكلم، فهو يتحدث بلغة يونيكود. تسجّل الآن لحضور المؤتمر الدولي العاشر ليونيكود (Unicode Conference)، الذي سيعقد في 10-12 آذار 1997 بمدينة ماينتس، ألمانيا. وسيجمع المؤتمر بين خبراء من كافة قطاعات الصناعة على الشبكة العالمية انترنت ويونيكود، حيث ستتم، على الصعيدين الدولي والمحلي على حد سواء مناقشة سبل استخدام يونيكود في النظم القائمة وفيما يخص التطبيقات الحاسوبية، الخطوط، تصميم النصوص والحوسبة متعددة اللغات.

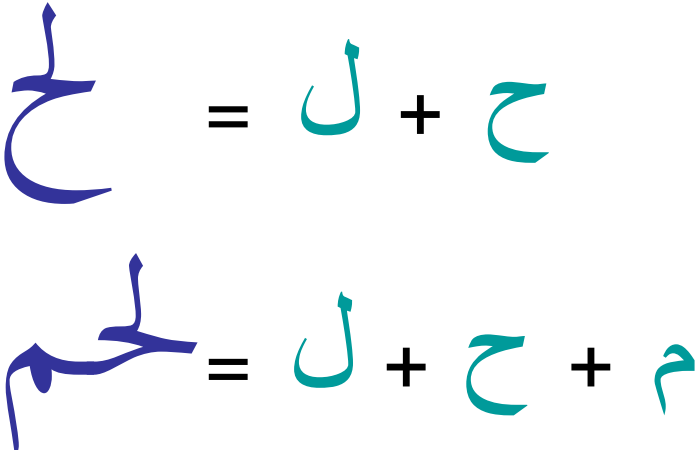
Ligatures are very common. Essentially a ligature is a single glyph that represents more than one underlying character.

The example shown here is of a mandatory ligature in Arabic. An ALEF character followed by a LAM character must always be displayed as a single lam-alef glyph.

More character to glyph rendering

► Ligatures

Arabic
Indic



The diagram illustrates two Arabic ligatures. The first line shows a blue ligature (shayn) equal to a teal 'L' (lam) plus a teal 'C' (ha). The second line shows a blue ligature (shayn-mim) equal to a teal 'L' (lam) plus a teal 'C' (ha) plus a teal 'M' (mim).

The top line shows another Arabic ligature. This ligature is optional and will only be displayed if the font developers included it.

The second line shows that ligatures in Arabic also have joining forms when they occur alongside other characters.

More character to glyph rendering

▶ Ligatures

Arabic
Indic

क्ष
क + ष + ष

क
ङ + ष + क

This slide shows some ligatures used to render Indic consonant clusters.

More character to glyph rendering

▶ Ligatures

Arabic
Indic

क्ष
क + ष् + ष

ङ्क
ङ + ष् + क

Again, the number of ligatures available in a font varies. In some fonts the lower example may simply be rendered using a visible virama.

More character to glyph rendering

▶ Ligatures

[u]

க	கு
ங	ஙு
ச	சு
ஐ	ஐு
ஔ	ஔு
ட	டு
ண	ணு
ம	மு

Arabic
Indic

Ligatures are not only used for combining consonants. This slide shows the effect of combining a single vowel sign with various consonants in Tamil. As you can see, the combinations produced some complex and vary varied results.

More character to glyph rendering

▶ Joiner & non-joiner control characters

Arabic
Indic

We have seen how Arabic glyphs join up with each other when juxtaposed. Unicode provides some special characters, invisible to the naked eye and to processing algorithms, to help control joining behaviour manually.

The *zero-width non-joiner* character (U+200C) can be inserted between the three characters LHM to create the effect on the second line. Here the three characters are not separated by spaces, but the glyphs no longer join.

The *zero-width joiner* character (U+200D), on the other hand, has the opposite effect. The three characters on the third line have spaces between them, but the joiner character is used to produce the joining forms of the glyphs. This behaviour is occasionally needed for correctly rendering Arabic text.

More character to glyph rendering

► Joiner & non-joiner control characters

Arabic
Indic

कक → क्क

क्ष → कष

Unicode allows you to force a consonant + virama sequence to display the virama where the font would otherwise have used a half-form – add a zero width non-joiner immediately after the virama of the dead consonant.

Unicode allows you to force a dead consonant to assume a half-form rather than combine as part of a ligature – place a zero width joiner immediately after the virama.

□ Complex script rendering

Combining characters

- Arabic & Hebrew short vowels
- Context-sensitive placement of diacritics
- Vowel signs
- Coding combining characters
- Precomposed vs. decomposed
- Normalisation

Context-sensitive glyph shaping

- Word-final glyph variant
- Cursive script
- Inputting cursive glyphs
- Conjunct consonants

More character to glyph rendering

- Special joining forms
- Positional variation
- Ligatures
- Joiner & non-joiner control characters