You can also see in that example that we use utf8_to_uv to get the value of the character; the inverse function uv_to_utf8 is available for putting a UV into UTF-8:

```
    if (!UTF8_IS_INVARIANT(uv))
        /* Must treat this as UTF8 */
        utf8 = uv_to_utf8(utf8, uv);
    else
        /* OK to treat this character as a byte */
        *utf8++ = uv;
```

You **must** convert characters to UVs using the above functions if you're ever in a situation where you have to match UTF-8 and non-UTF-8 characters. You may not skip over UTF-8 characters in this case. If you do this, you'll lose the ability to match hi-bit non-UTF-8 characters; for instance, if your UTF-8 string contains v196.172, and you skip that character, you can never match a chr(200) in a non-UTF-8 string. So don't do that!

70.8.4 How does Perl store UTF-8 strings?

Currently, Perl deals with Unicode strings and non-Unicode strings slightly differently. If a string has been identified as being UTF-8 encoded, Perl will set a flag in the SV, SVf_UTF8. You can check and manipulate this flag with the following macros:

```
    SvUTF8(sv)
    SvUTF8_on(sv)
    SvUTF8_off(sv)
```

This flag has an important effect on Perl's treatment of the string: if Unicode data is not properly distinguished, regular expressions, length, substr and other string handling operations will have undesirable results.

The problem comes when you have, for instance, a string that isn't flagged is UTF-8, and contains a byte sequence that could be UTF-8 - especially when combining non-UTF-8 and UTF-8 strings.

Never forget that the SVf_UTF8 flag is separate to the PV value; you need be sure you don't accidentally knock it off while you're manipulating SVs. More specifically, you cannot expect to do this:

```
    SV *sv;
    SV *nsv;
    STRLEN len;
    char *p;

    p = SvPV(sv, len);
    frobnicate(p);
    nsv = newSVpvn(p, len);
```

The char* string does not tell you the whole story, and you can't copy or reconstruct an SV just by copying the string value. Check if the old SV has the UTF-8 flag set, and act accordingly:

```
    p = SvPV(sv, len);
    frobnicate(p);
    nsv = newSVpvn(p, len);
    if (SvUTF8(sv))
        SvUTF8_on(nsv);
```

In fact, your frobnicate function should be made aware of whether or not it's dealing with UTF-8 data, so that it can handle the string appropriately.

Since just passing an SV to an XS function and copying the data of the SV is not enough to copy the UTF-8 flags, even less right is just passing a char * to an XS function.