Episode Book

EPISODE #5486

Why Hebrew URLs Turn Into Percent-Sign Gibberish

PUBLISHED	23 September 2026 at 12:49 UTC
DURATION	23:13
LISTEN	myweirdprompts.com/episode/hebrew-url-percent-encoding/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:36 UTC

SYNOPSIS

A listener noticed something odd: Hebrew in a URL's domain renders beautifully in the address bar, but Hebrew in the path collapses into six lines of percent-sign hex. It turns out the domain and the path are governed by two completely separate internationalization regimes — IDN and Punycode on one side, IRI and percent-encoding on the other — and only one of them comes with a display mandate. This episode traces why the path stayed ugly, from the 2002 homograph attack paper out of the Technion to the browser policies that deliberately show you the ASCII form as a safety measure. Along the way: the byte math behind those enormous Israeli retail links, RFC 5893 and bidirectional text, and whether China or India built a more robust version at scale.

TRANSCRIPT



Corn

A link to a product page on an Israeli e-commerce site. Six, seven lines of it in the message preview. You assume it's a tracking nightmare.



Herman

It's the product name.



Corn

Every letter of it turned into a percent sign and two hex digits, because the product is called something in Hebrew and the internet would rather you didn't see that.



Herman

And yet the same Hebrew, put in the domain, renders beautifully in the address bar.



Corn

Which is the whole puzzle. Daniel wrote in about this. He noticed it a few years ago, the double take we've all done: URL starts in English, then after the forward slash, Hebrew. He assumed it wouldn't work. It worked. Then the browser swapped his Hebrew for the percent-mark gibberish, and he was back to square one.



Herman

And it's the same reason those Israeli retail links are enormous.



Corn

Same reason. He reads it as partial support for Unicode at the very fundamental layers, the DNS layers, and he's got two questions off the back of that. One: why is there no English-equivalent experience? A browser that displays the suffix as written, shareable links that default to the user's language. Two: in places with a much bigger critical mass in a single non-Latin script, China, India, has somebody already built the robust version? Or



Corn (continued)

does the internet simply assume English letters are how computers talk to each other?



Herman

Both questions have answers.



Corn

Let's start with the thing he intuited, the partial support, because it turns out to be exactly right, and the reason is that the domain and the path are two different countries.



Herman

Two different countries, two different governments, two different standards, and they were written by different people for different reasons. That's the fact the whole episode hangs on.



Corn

The domain name and the URL path are governed by completely separate internationalization regimes.



Herman

Right. Domain names run on IDN, Internationalized Domain Names, plus a little encoding called Punycode. The path runs on IRI, Internationalized Resource Identifiers, plus percent-encoding. And the two behave differently on purpose.



Corn

Walk us up to it. Start with the domain.



Herman

The DNS is ASCII. Not by law, exactly. RFC 2181 technically allows any binary string in a DNS label, and people have run UTF-8 over DNS privately. But the registration system is, in the words of the standard, totally incapable of handling non-ASCII encodings. So in practice, ASCII only.



Corn

So how does dot anything exist?



Herman

IDNA. Internationalizing Domain Names in Applications. Standardized in 2003, updated in 2010 as IDNA2008. An IDNA-aware application takes the Unicode domain, converts it into an ASCII form called Punycode, prefixed with x n dash dash, does the DNS lookup with that, and then displays the Unicode version back to the user.



Corn

So the DNS never sees Hebrew.



Herman

The DNS never sees Hebrew. The browser translates on the way out and translates back on the way in. It's a display-layer trick. Bücher dot example becomes x n dash dash b c h e r dash k v a dot example. Russia's Cyrillic dot r f becomes x n dash dash p one a i. Israel's becomes x n dash dash four d b r k zero c e.



Corn

And the path?



Herman

The path is a different animal. URIs are limited to a subset of US-ASCII. Anything outside that set gets mapped to bytes, UTF-8 bytes, and then percent-encoded. Each byte becomes a percent sign and two hex digits. That's the ugly series Daniel was describing.



Corn

And IRI is the standard that's supposed to let you write Unicode in the whole identifier.



Herman

IRI is RFC 3987, 2005, Martin Dürst and Michel Suignard. It permits Unicode across the entire identifier, path included. But here's the catch, and it's the whole episode in one sentence: IRIs are mapped back to URIs for backwards compatibility.



Corn

So the pretty form is permitted but never mandatory.



Herman

Never mandatory. The RFC's own example: the IRI for the Wiktionary page on Rhodes, spelled in Greek, becomes the URI with percent E one, percent B F, percent A C, and so on. The path can be written in Hebrew. The wire format is always percent-encoded UTF-8. Whether you see Hebrew or gibberish depends entirely on whether the browser chooses to render the IRI form or the URI form.



Corn

And the browser mostly chooses the URI form.



Herman

Mostly. Which is the asymmetry. The domain gets a dedicated, standardized, display-friendly mechanism that browsers are strongly motivated to render in native script. The path gets a permissive standard with no equivalent display mandate. Partial support. Daniel's word, and it's the right one.



Corn

Let's make it concrete. Take a domain, take a path, watch what the browser does with each.



Herman

Take the domain first. You type a Hebrew domain into the address bar. The browser looks at the label, checks whether it's allowed to display it in Unicode, and if it is, you see Hebrew. Clean. The Punycode is invisible. That's the IDN display convention doing its job.



Corn

Now the path. You click through to a product page with a Hebrew slug.



Herman

The browser sends the request with the path percent-encoded, because that's the wire format, and then it decides what to show you. And very often it shows you what it sent. The raw URI. So the address bar fills with percent D seven percent nine zero and so on, and your Hebrew product name is gone.



Corn

Why does it do that? Why not render the IRI form?



Herman

Because nothing requires it to. IDN has a display convention baked into the standard and into browser policy. IRI doesn't. There's no "show the pretty version" clause. So the browser falls back to the form it actually transmitted, which is the safe, unambiguous, ASCII one.



Corn

Let's do the byte math, because that's what explains Daniel's six-line links.



Herman

Hebrew characters in UTF-8 are two bytes each. Percent-encoding turns each byte into three characters: percent sign, hex digit, hex digit. So every single Hebrew letter becomes a six-character sequence. Aleph is percent D seven percent nine zero.



Corn

Six characters for one letter.



Herman

Six for one. A product URL with a thirty-character Hebrew slug is roughly a hundred and eighty characters of percent-hex noise. Before you add query parameters, tracking IDs, session tokens, affiliate tags.



Corn

And CJK is worse.



Herman

CJK is three bytes per character. Nine characters of percent-hex per glyph. Chinese and Japanese paths inflate nine to one. Hebrew is the mild case and it still produces a link you have to scroll through.



Corn

So Daniel's observation about Israeli retail links being six or seven lines long isn't a quirk of bad web development. It's arithmetic.



Herman

It's arithmetic. That's exactly what it is.



Corn

Now, the RTL piece, because that's the part that feels like it should have its own standard.



Herman

It does. RFC 5893. Right-to-Left Scripts for Internationalized Domain Names for Applications. That's the direct answer to "is there a Hebrew-specific DNS standard." Yes, and it exists precisely because RTL labels need special bidirectional handling.



Corn

Why do they need it?



Herman

Because a URL is fundamentally a left-to-right string. Scheme, colon, slash slash, dots, slashes. All of that is LTR. And then the Hebrew content runs right-to-left inside it. Mixing the two in one address bar creates ordering ambiguity. Where does the label start, where does it end, which dot is a separator and which is part of the name.



Corn

And ambiguity in an address bar is a security problem, not just a usability one.



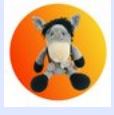
Herman

It's both. Unicode Technical Report thirty-six covers bidirectional text spoofing specifically. You can construct a string that renders one way and parses another. RFC 5893 exists to close that.



Corn

So the mechanism is clear. Domain gets IDN, path gets IRI, the path maps down to percent-encoding, and the byte math makes Hebrew links enormous. Now the question is why the path stayed ugly, and the answer is not laziness. It's security.



Herman

It's security, and the story starts with two researchers at the Technion, in Israel, funnily enough. Evgeniy Gabrilovich and Alex Gontmakher. The Homograph Attack, published in Communications of the ACM in 2002.



Corn

Homograph as in words that look the same.



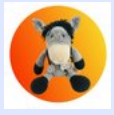
Herman

Words that look the same and aren't. Unicode gives you characters from different scripts that render almost identically. Cyrillic a, U plus zero four three zero, versus Latin a, U plus zero zero six one. On screen, indistinguishable in most fonts.



Corn

So you register the Cyrillic version of a domain.



Herman

You register wikipedi dot org with a Cyrillic a, and it's a completely different domain. Punycode x n dash dash w i k i p e d i dash eight six g. And it renders in the address bar looking exactly like wikipedia dot org.



Corn

And they proved it by doing it.



Herman

They registered a Cyrillic variant of microsoft dot com. That was the demonstration. The paper is the founding document of Unicode URL spoofing.



Corn

And the browser response was to restrict Unicode display, not expand it.



Herman

Restrict it. Firefox, Chrome, Chromium, Edge, Opera. They only display an IDN in Unicode if the top-level domain restricts its characters and the label doesn't mix scripts. Otherwise they show you the Punycode. Safari renders problematic character sets as Punycode. Internet Explorer seven and up blocked mixed-script labels outright.



Corn

So the trend has been toward showing the ugly ASCII form as a safety measure.



Herman

The ugliness is a feature. That's the line. The percent-encoded path and the Punycode fallback are the system choosing to show you something ugly rather than something spoofable.



Corn

Where does Hebrew land in that?



Herman

Hebrew is low-risk for homographs. There are only a few reliable lookalikes. Samekh looks like an o. Vav with a diacritic looks like an i. Heth looks like an n. That's about it. And the Hebrew alphabet runs right-to-left, so mixing it with left-to-right glyphs causes problems before you even get to spoofing.



Corn

So Hebrew gets little of the security attention Cyrillic and Greek get.



Herman

And little of the display accommodation. It's the mild case in both directions.



Corn

Which brings us to Israel's actual rollout, because Daniel's partial-support intuition is precisely correct, and the timeline shows it.



Herman

Dot i l was registered in October 1985. Third country code top-level domain ever, after dot u s and dot u k.



Corn

Israel got in early.



Herman

Very early. And then nothing Hebrew for a long time. The Hebrew script IDN ccTLD, , was proposed in 2012. ICANN approved it in May 2020. Delegated February 2021. Open for full-Hebrew registration since November 2022.



Corn

Seven years from proposal to delegation.



Herman

Ten years from proposal to open registration. And the registration rules are strict. Only Hebrew alphabet, digits, and hyphens. No script mixing at all. You can register directly at the second level, or third level under one of four categories: academia, government, IDF, municipal.



Corn

And no Hebrew equivalent of dot co dot i l?



Herman

No Hebrew dot co dot i l, no dot org dot i l, no dot net dot i l, no dot k twelve dot i l. Those categories exist in the ASCII namespace and they just don't have Hebrew counterparts. There's also dot , which is "com" in Hebrew, registered by VeriSign in 2016 as a private initiative.



Corn

So Israel has a full Hebrew TLD, Hebrew second-level domains, and the path still percent-encodes.



Herman

The domain layer got fixed. The path layer didn't. That's the split, lived out at national scale.



Corn

So now Daniel's scale hypothesis. China and India. Does a bigger native-script user base produce a more robust implementation?



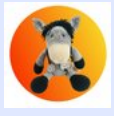
Herman

At the domain level, yes, dramatically. At the path level, no. And the reason is the interesting part.



Corn

Start with China.



Herman

China's dot , simplified, and dot , traditional, both approved by ICANN in June 2010, delegated to CNNIC. Over two hundred thousand domains by 2018. But the interesting thing is what happened before that. The Chinese Domain Name Consortium was founded in May 2000. CNNIC, TWNIC, HKNIC, MONIC.



Corn

Two thousand. That's years before ICANN's fast-track process.



Herman

Years before. China was organizing around native-script DNS before ICANN had a process for it. The Joint Engineering Team guidelines for CJK IDN registration came out as RFC 3743 in May 2004. Taiwan's dot is the second-most-used IDN ccTLD, around five hundred thousand domains.



Corn

So scale did produce something.



Herman

Scale produced registry infrastructure. Consortiums, variant tables, dedicated RFCs, a whole apparatus for handling the fact that Chinese has traditional and simplified forms and regional variants. That's real work and it's impressive.



Corn

And the path?



Herman

Chinese paths still percent-encode. And each CJK character is three UTF-8 bytes, so it's nine characters of percent-hex per glyph. Worse than Hebrew. The scale bought them a better domain registry and nothing at all on the path.



Corn

India.



Herman

India went the other direction. Instead of one script, they applied for an IDN ccTLD in each of their local scripts. Dot in Devanagari, approved 2011, live August 2014. Then Tamil, Telugu, Gurmukhi, Gujarati, Bengali, Urdu, Kannada, Malayalam, Odia. Plus Assamese, Kashmiri, Sanskrit, Santali, Sindhi variants in 2022.



Corn

That's fourteen, fifteen scripts.



Herman

The dot IN registry created an expert IDN working group in May 2005 to cover twenty-two official languages. Ram Mohan from Afilias led the technical implementation. C-DAC supplied the linguistic expertise. The Tamil and Malayalam variant tables were completed in January 2007.



Corn

Variant tables being what, exactly?



Herman

Which characters count as equivalent for registration purposes. If two spellings of a name are visually or semantically the same, you don't want two different people registering both. That's a hard problem in scripts with combining marks and multiple encodings, and India solved it script by script.



Corn

So the honest answer to Daniel's question is yes and no.



Herman

Yes, China and India built far more extensive registry infrastructure. Multiple scripts, variant tables, consortiums, dedicated RFCs, working groups. No, nobody has solved the path problem. Not China, not India, not Israel, not anyone.



Corn

And that's because it isn't a scale problem.



Herman

It's architectural. The path is percent-encoded by design. No jurisdiction can override that unilaterally. You can't have a country-level fix for a wire format that every server and every proxy and every log parser expects in one specific shape.



Corn

Which gets us to Daniel's last question, and it's the uncomfortable one. Does the internet fundamentally assume English letters are the standard for computer-to-computer communication?



Herman

Largely yes, and the standards say so themselves. The DNS's ASCII restriction is described as a practical limitation that initially set the standard for acceptable domain names.



Corn

Initially.



Herman

It's doing all the work. IDN is explicitly framed as a technical solution to translate names written in language-native scripts into an ASCII text representation compatible with the DNS.



Corn

Translate into ASCII. ASCII is the substrate.



Herman

Everything else is a translation layer bolted on top. IRI is the same story. IRIs are mapped to URIs to retain backwards compatibility with systems that don't support the new format. The standard itself concedes that an IRI doesn't make clear how web resources can be accessed with keyboards that can't generate the internationalized characters.



Corn

So IRI assumes an input-method layer that ASCII never needed.



Herman

It assumes one. And the whole homograph-defense apparatus, the Punycode fallback, the script-mixing bans, the ICANN string evaluations, all of it exists because the system treats non-Latin script as suspect by default.



Corn

Which is a defensible engineering position and a strange thing to notice about the plumbing of the internet.



Herman

Both of those are true at once. It's defensible and it's strange.



Corn

Daniel asked whether scale fixes this. China and India built enormous registry infrastructure and the path is still percent-encoded. Which raises the question of whether anyone downstream ever tried to actually live with Hebrew paths in production.



Hilbert

I did. Well, I cleaned up after people who did.



Corn

Go on.



Hilbert

Small web shop outside Tel Aviv. Storefronts for small businesses. Part of my job was fixing product URLs. The client would upload a catalogue with Hebrew product names, the CMS would build the slug from the name, and the link would come out as a hundred and eighty characters of percent signs.



Corn

And the browser handled it.



Hilbert

The browser was the one piece of software that mostly worked. Everything else fell over. The email marketing tool truncated the links at a certain length, so the customer clicked through to a four-oh-four. The CRM choked on them when it logged the click. The affiliate network's link shortener refused to accept them at all.



Corn

The path fails at every hop.



Hilbert

Every hop. The address bar is the last place you'd notice. The first place was the marketing tool.



Corn

What did you do?



Hilbert

Stripped the Hebrew out. Replaced every slug with an ASCII transliteration. Product name in Hebrew on the page, product name in Latin letters in the URL. Two names for every item in the catalogue.



Corn

And the domain layer?



Hilbert

The domain layer was fine. We registered a Hebrew domain name and it worked. Never had a problem with it. So you'd end up with a Hebrew-branded domain and a path that was pure ASCII. Hebrew on the left of the slash, English on the right.



Corn

Which is the split we've been describing all episode, arrived at as a business decision by a shop that had never read an RFC.



Hilbert

We had a spreadsheet. Every Hebrew letter mapped to a Latin substitute. We kept it updated by hand because the automated transliteration kept producing collisions.



Corn

Collisions?



Hilbert

Two different Hebrew names coming out as the same Latin string. So you'd get two products fighting over one URL. There was a column in that spreadsheet for letters that look like other letters, so we could flag the ones that would confuse people.



Corn

That's the homograph problem.



Hilbert

That's what we called it. Letters that look like other letters.



Corn

Discovered empirically, in a web shop, years before anyone there had heard the word homograph.



Hilbert

We just knew that if a customer couldn't tell two links apart, we'd get a phone call. Anyway. I'm late for an appointment.



Corn

The browser was the one piece that mostly worked, and everything downstream gave up. That's the shape of the problem.



Herman

It's the shape of the whole episode, in a way. The browser is the part everyone sees, so it's the part everyone judges. The failures are all in the boring middle, in the tools that log and forward and shorten and archive.



Corn

The most common wrong belief about all of this is that Hebrew URLs don't work, or that the percent-encoded gibberish is a sign of incomplete implementation. It isn't. The browser translates, the path percent-encodes by design, and the ugliness is partly a deliberate defense against spoofing.



Herman

It works exactly as specified. That's the correction. Nothing here is broken.



Corn

Which leaves the open question. If the path problem is architectural rather than a matter of user volume, what would actually fix it? A new standard that mandates display of the IRI form? Browser vendors agreeing to render paths in native script the way they render IDN domains? Or is the percent-encoded path load-bearing infrastructure that nobody wants to touch?



Herman

I don't know. And I think the honest answer is that nobody knows, because nobody has tried.



Corn

The lowest layers of the internet are ASCII by design. Every non-Latin accommodation is a translation layer bolted on top. The domain layer has been substantially fixed. The path layer has not. Daniel's instinct was right.



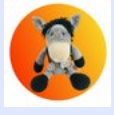
Herman

The internet does default to English letters. The domain layer got a fix. The path layer got a standard that permits the pretty form and doesn't require it.



Corn

The ugliness of a percent-encoded Hebrew URL isn't a bug waiting to be fixed. It's the system working as designed. And the design assumed English.



Herman

Thanks as always to Hilbert Flumingtop, who produces the show and who has a spreadsheet somewhere with a column for letters that look like other letters.



Corn

This has been My Weird Prompts. If you enjoyed it, a review helps other people find us. We'll be back soon.