Regardless of where you stand, there's no denying the simplicity and modularity of OP_CAT. It's a general purpose solution that can be applied in a variety of ways.

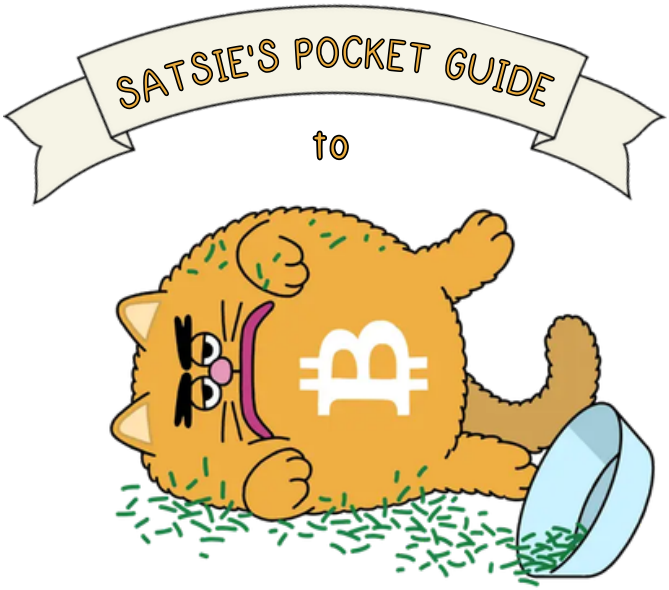
This is arguably good because new ideas can be developed, tested, and presented without committing to more feature specific opcodes. If a feature is expensive to have with OP_CAT, but people are willing to accept the cost, that is a clear indicator of demand. The opposite is also true.

Thanks for reading! That was just a high level overview of Script and OP_CAT. If you enjoyed it, there's much more to dig into. Ready to keep going? Want to download free copies of this and other zines? Visit

https://satsie.dev/zines

This zine is sponsored by the Bitcoin Dev Project. Physical distribution made possible by OpenSats.

BITCOINDEVS.XYZ >_OpenSats



OP_CAT

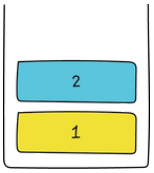
A zine for beginners about bitcoin Script and OP_CAT

@satsie ☆ https://satsie.dev

Script is bitcoin's stack-based programming language. During evaluation, the items in a script are read from left to right and placed in a stack. When a special type of element called an opcode is reached, an action specific to the opcode is performed.

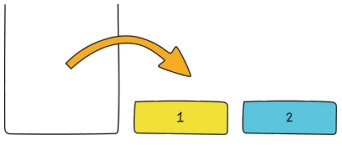
Let's try an example with this script:

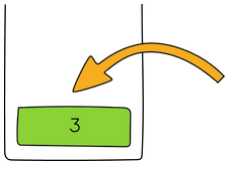
1, 2, OP_ADD ← that's an opcode



First, 1 is added to the stack.
Then 2 is added to the stack.

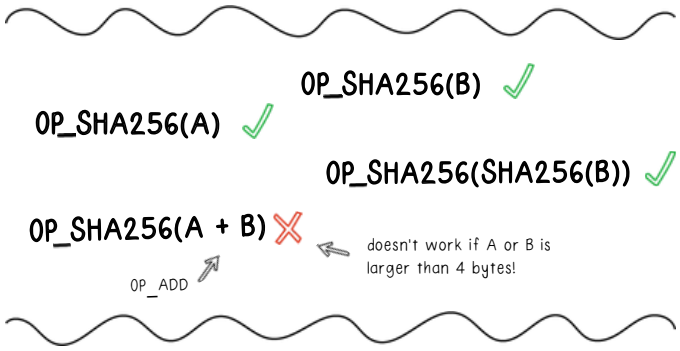
Next, OP_ADD pops two items off the top of the stack.





Finally, OP_ADD adds the two items together and pushes the result, 3 to the top of the stack.

While the max size of a script element is 520 bytes, elements over 4 bytes cannot influence one another. For example, using OP_ADD to compute A + B only works if elements A and B are each less than 4 bytes.



OP_CAT is the simplest way to solve this problem where elements larger than 4 bytes are siloed from one another. Why does this matter? Signatures, public keys, and hashes are all examples of data larger than 4 bytes. If Script is able to work with larger stack elements like these, it can do new things like build Merkle trees and hash multiple elements.

Did you know bitcoin lets you attach spending conditions directly on individual coins? With the bitcoin Script programming language, you can define the exact rules for spending a coin

Suppose you are building your savings and don't want to spend your bitcoin until a certain point in the future. You can attach a rule for this directly onto one or more of your coins. If you try to prematurely spend a coin that has the rule attached, the network will reject the transaction.

Another popular example of a spending condition is multisig where multiple signatures are needed to move coins. This is often seen in treasury management.

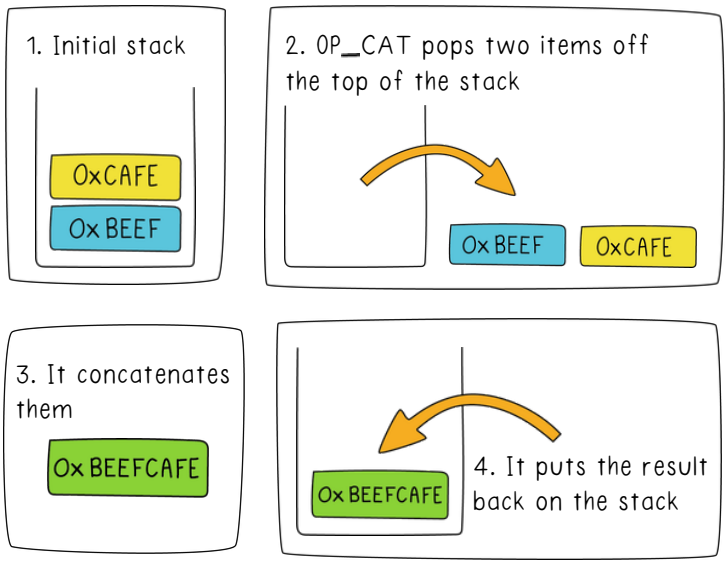
Opcode History

Bitcoin has many opcodes. When the code base was released in 2009, Satoshi included over eighty! Here are a few examples:

★Opcode★	★Function★
OP_SUB	Subtract one element from another
OP_SHA256	Hash one element with SHA-256
OP_CHECKMULTISIG	Check if a certain number of signatures correspond to the public keys listed in the script. Used in multisig transactions.

In 2010, Satoshi disabled 15 opcodes as part of a commit labeled "misc changes". We don't know why this was done but one theory is it came from a need to remove OP_LSHIFT which had a severe bug. The other 14 opcodes may have been untested and removed as a precaution.

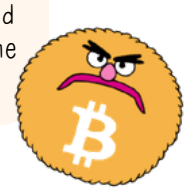
One of the opcodes removed was OP_CAT. "CAT" is short for concatenate. OP_CAT pops the top two elements off the stack, concatenates (joins) them, and pushes the result back on the stack.



In 2023, after years of discussion, a formal proposal (BIP-347) was made by **Ethan Heilman** and **Armin Sabouri** to bring OP_CAT back.

This means OP_CAT can support some pretty cool features like covenants, zero-knowledge proofs, and more powerful layer 2 protocols.

OP_CAT has tradeoffs. The things people want it for are likely to require complex, inefficient, and expensive scripts. But it gets the job done.



Pushback

While OP_CAT has the potential to bring a lot of new things to bitcoin, it is not supported by everyone. Some just don't care for the feature set that it would enable.

Others may want the new functionality, but consider it inefficient or too general of a solution. Certain engineers prefer a more tightly scoped opcode that only enables a specific feature.