# ð Fhenix Differences For Developers

You might be familiar with fhevm, which is a fork of the Ethereum Virtual Machine that supports homomorphic encryption by Zama.

While Fhenix uses a similar FHE cryptography, it does not use fhevm. However, in order to make the FHE ecosystem as accessible as possible, we have chosen to maintain compatability in most interfaces, except when we felt that the developer experience was significantly improved by making changes.

In this page, we try and document the differences that developers should be aware of.

## Differencesâ

- fhenix.js is the recommended Javascript library for interacting with Fhenix smart contracts.
- FHE library is available at the npm repository@fhenixprotocol/contracts
- .
- cmux
- is namedselect
- .
- reencrypt
- is namedsealoutput
- orseal
- .
- Operations can be called directly as properties of encrypted types (e.g.euint32.add(euint32)
- instead ofFHE.add(euint32, euint32)
- ).
- Operations between encrypted types expect the types to match (e.g.euint32 + euint32
- instead ofeuint32 + euint64)
- ).
- In Fhenix, we recommend using theinEuintXX
- input types instead of raw bytes when receiving encrypted data.
- Conversion to other encrypted types can be done using the.toUxx
- functions. E.g.euint32 b = a.toU32();
- Division by zero will return aMAX_UINT
- value instead of throwing an error (e.g.euint8(1) / euint8(0)
- will returneuint8(255)
- instead of throwing an error).
- Permits
- andPermissioned
- contracts are the recommended way to handle access to sensitive data in Fhenix. To read more about permits and access control, seeAccess Control
- .
- Sealing and Decryption can be accessed using.seal
- and.decrypt
- respectively.
- Large bit sizes are supported (includingeaddress
- ), with a limited instruction setEdit this page

Previous ð Limitations Next Changelog