EatTheBlocks Token Cheatsheet

ERC20

Suitable for	Fungible assets (asset A can be exchanged with asset B)
Use cases	company shares, event ticket, virtual currency, productized service

Main functions

Token transfer

```
function transfer(address _to, uint256 _value) public returns (bool success)
```
=>transfer `_values` tokens to address `_to`

Delegated transfer

```
function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
```
=>transfer `_values` tokens to address `_to` on behalf of `from`.
Token owner must call `approve(sender, _value)` before.

Implementation	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/token/ERC20
Specification	https://eips.ethereum.org/EIPS/eip-20

ERC721

When to use it	For non-fungible assets (asset A cannot be exchanged with asset B)
Use cases	cryptocollectibles, art items, real estate

Main functions

Token transfer (normal & delegated)

```
function transferFrom(address _from, address _to, uint256 _tokenId) external payable
```
=>transfer `_values` tokens identified by `_tokenId` to address `_to`. Sender must be the current owner,
an authorized operator, or the approved address for this (`_tokenId`, `msg.sender`)

Safe transfer (normal & delegated)

```
function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable
```
=>same as before except that if the receiver is a contract it must implement the erc721
receiver interface. This is to avoid to send tokens to contracts that cant handle them.

Implementation	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol
Specification	https://eips.ethereum.org/EIPS/eip-721

ERC223

When to use it	Same as ERC20, except it prevents token being locked in contract that cant handle them. I dont recommend using it. ERC777 is more modern.
Use cases	Same as ERC20

Main functions

Token transfer

```
function transfer(address _to, uint _value, bytes _data) returns (bool)
```
=>transfer `_value` tokens to address `_to`.
If receive is a contract, it must have `tokenFallback(address _from, uint _value, bytes _data)`
`bytes` argument will be forwarded in this case.

Token transfer (compatible with erc20)

```
function transfer(address _to, uint _value) returns (bool)
```
=>same as before except it does not accept a `bytes` argument for compatiblity with erc20 token

Implementation	https://github.com/ethereum/eips/issues/223
Specification	https://github.com/Dexaran/ERC223-token-standard/tree/development/token/ERC223

ERC777

When to use it	Same as ERC20, except it prevents locked coins in contracts + provide function "hooks" in receiving contracts
Use cases	Same as ERC20

Main functions

Token transfer

```
function send(address to, uint256 amount, bytes calldata data) external;
```
=>transfer `amount` tokens to address `to`.
if the recipient is a contract it must implement this function:

```
 function tokensReceived(
        address operator,
        address from,
        address to,
        uint256 amount,
        bytes calldata data,
        bytes calldata operatorData
    ) external
```

Delegated transfer

```
  function operatorSend(
        address from,
        address to,
        uint256 amount,
        bytes calldata data,
        bytes calldata operatorData
    ) external;
```
=>transfer `amount` tokens to address `to` on behalf of `from`.
Token owner must call `authorizeOperator(address operator) external`

Implementation	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/token/ERC777
Specification	https://eips.ethereum.org/EIPS/eip-777

ERC1155

When to use it	For BOTH fungible and non-fungible assets. Good for class of fungible assets.
Use cases	example: event tickets where several tickets are in different categories (premium, economy, etc...)

Main functions

Token transfer (normal & delegated)

```
function safeTransferFrom(address _from, address _to, uint256 _id, uint256 _value, bytes calldata _data) external
```
=>transfer `_value` tokens identified by `_id` to address `_to`, on behalf of `_from`.
Sender must be the current owner, or an authorized operator approved for this (`msg.sender`, `_id`)

if recipient is a contract, it must implement this function:

```
function onERC1155Received(
  address _operator,
  address _from,
  uint256 _id,
  uint256 _value,
  bytes calldata _data) external returns(bytes4);
```

Batch transfer transfer (normal & delegated)

```
function safeBatchTransferFrom(
  address _from,
  address _to,
  uint256[] calldata _ids,
  uint256[] calldata _values,
  bytes calldata _data) external
```
=>same as before except transfer is done in batch

if recipient is a contract, it must implement this function:

```
function onERC1155BatchReceived(
  address _operator,
  address _from,
  uint256[] calldata _ids,
  uint256[] calldata _values,
  bytes calldata _data) external returns(bytes4);
```

Implementation	https://github.com/enjin/erc-1155
Specification	https://eips.ethereum.org/EIPS/eip-1155