Tags
Each tag is an array of one or more strings, with some conventions around them. Take a look at the example below:

```
{
  "tags": [
    ["e", "5c83da77af1dec6d7289834998ad7aafbd9e2191396d75ec3cc27f5a77226f36", "wss://nostr.example.com"],
    ["p", "f7234bd4c1394dda46d09f35bd384dd30cc552ad5541990f98844fb06676e9ca"],
    ["a", "30023:f7234bd4c1394dda46d09f35bd384dd30cc552ad5541990f98844fb06676e9ca:abcd", "wss://nostr.example.com"],
    ["alt", "reply"],
    // ...
  ],
  // ...
}
```

The first element of the tag array is referred to as the tag *name* or *key* and the second as the tag value. So we can safely say that the event above has an e tag set to "`5c83da77af1dec6d7289834998ad7aafbd9e2191396d75ec3cc27f5a77226f36`", an `alt` tag set to "`reply`" and so on. All elements after the second do not have a conventional name.

This NIP defines 3 standard tags that can be used across all event kinds with the same meaning. They are as follows:

- The `e` tag, used to refer to an event: `["e", <32-bytes lowercase hex of the id of another event>, <recommended relay URL, optional>]`
- The `p` tag, used to refer to another user: `["p", <32-bytes lowercase hex of a pubkey>, <recommended relay URL, optional>]`
- The `a` tag, used to refer to an addressable or replaceable event:
  - for an addressable event: `["a", <kind integer>:<32-bytes lowercase hex of a pubkey>:<d tag value>, <recommended relay URL, optional>]`
  - for a normal replaceable event: `["a", <kind integer>:<32-bytes lowercase hex of a pubkey>:, <recommended relay URL, optional>]`

As a convention, all single-letter (only english alphabet letters: a-z, A-Z) key tags are expected to be indexed by relays, such that it is possible, for example, to query or subscribe to events that reference the event "`5c83da77af1dec6d7289834998ad7aafbd9e2191396d75ec3cc27f5a77226f36`" by using the {"`#e`": ["`5c83da77af1dec6d7289834998ad7aafbd9e2191396d75ec3cc27f5a77226f36`"]} filter. Only the first value in any given tag is indexed.

Kinds
Kinds specify how clients should interpret the meaning of each event and the other fields of each event (e.g. an "`r`" tag may have a meaning in an event of `kind 1` and an entirely different meaning in an event of kind `10002`). Each NIP may define the meaning of a set of kinds that weren't defined elsewhere. This NIP defines two basic kinds:

- `0`: **user metadata**: the content is set to a stringified JSON object `{name: <username>, about: <string>, picture: <url, string>}` describing the user who created the event. Extra metadata fields may be set. A relay may delete older events once it gets a new one for the same pubkey.
- `1`: **text note**: the content is set to the **plaintext** content of a note (anything the user wants to say). Content that must be parsed, such as Markdown and HTML, should not be used. Clients should also not parse content as those.

And also a convention for kind ranges that allow for easier experimentation and flexibility of relay implementation:

- for kind n such that `1000 <= n < 10000 || 4 <= n < 45 || n == 1 || n == 2`, events are **regular**, which means they're all expected to be stored by relays
- for kind n such that `10000 <= n < 20000 || n == 0 || n == 3`, events are **replaceable**, which means that, for each combination of pubkey and `kind`, only the latest event MUST be stored by relays, older versions MAY be discarded
- for kind n such that `20000 <= n < 30000`, events are **ephemeral**, which means they are not expected to be stored by relays
- for kind n such that `30000 <= n < 40000`, events are **addressable** by their kind, pubkey and `d` tag value -- which means that, for each combination of kind, pubkey and the d tag value, only the latest event MUST be stored by relays, older versions MAY be discarded.

In case of replaceable events with the same timestamp, the event with the lowest id (first in lexical order) should be retained, and the other discarded.

When answering to `REQ` messages for replaceable events such as `{"kinds":[0],"authors":[<hex-key>]}`, even if the relay has more than one version stored, it SHOULD return just the latest one.

These are just conventions and relay implementations may differ.