It's an alphanumeric identifier followed by underscore followed by an operator identifier.

They can only be invoked by the primary constructor in the extends statement.

They correspond to the primary constructor, along with all code not in a method or field.

- They are easier to reason about, since they don't change.
- They can be passed around freely without defensive copies.
- They are inherently thread safe.
- They are safe hash table keys.

They are used for a subset of `val`s. Constants would generally not change even between program executions.

Case convention is not caps, just CamelCase.

They are one or more operator characters.

They are internally mangled by Scala as `$char` for each character.

e.g., `+`, `<?>`, `:->`

It starts with a letter or underscore, followed by further letters, digits, or underscores. `$` counts as a letter but should not be used to avoid colliding with the compiler.

By finding the method with the best static type match to arguments.

If there is no unique best-matching version, will return "ambiguous reference" error.

They are literal values with no explanation which, in the worst case, appear multiple times.

`Predef.require(Boolean)`

This throws `IllegalArgumentException`.

It is enclosed by back ticks.

It forces identifier interpretation even over keyword or other restrictions.

- Alphanumeric identifier.
- Operator identifier.
- Mixed identifier.
- Literal identifier.