JWT token

JSON Web Token — three base64url segments.

```
^[A-Za-z0-9_-]+\.[A-Za-z0-9_-]+\.[A-Za-z0-9_-]*$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`[A-Za-z0-9_-]+`	one or more: any of: uppercase letters, lowercase letters, digits, “_”, “-”
`\.`	literal “.”
`[A-Za-z0-9_-]+`	one or more: any of: uppercase letters, lowercase letters, digits, “_”, “-”
`\.`	literal “.”
`[A-Za-z0-9_-]*`	zero or more: any of: uppercase letters, lowercase letters, digits, “_”, “-”
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^[A-Za-z0-9_-]+\.[A-Za-z0-9_-]+\.[A-Za-z0-9_-]*$/;
re.test(value);
```

Python

```
import re
re.match(r"^[A-Za-z0-9_-]+\.[A-Za-z0-9_-]+\.[A-Za-z0-9_-]*$", value)
```

Java

```
Pattern.compile("^[A-Za-z0-9_-]+\\.[A-Za-z0-9_-]+\\.[A-Za-z0-9_-]*$").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^[A-Za-z0-9_-]+\.[A-Za-z0-9_-]+\.[A-Za-z0-9_-]*$");
```

Go

```
regexp.MustCompile(`^[A-Za-z0-9_-]+\.[A-Za-z0-9_-]+\.[A-Za-z0-9_-]*$`).MatchString(value)
```

Ruby

```
/^[A-Za-z0-9_-]+\.[A-Za-z0-9_-]+\.[A-Za-z0-9_-]*$/.match?(value)
```

PHP

```
preg_match('~^[A-Za-z0-9_-]+\.[A-Za-z0-9_-]+\.[A-Za-z0-9_-]*$~', $value);
```

Common pitfalls

- **Anchored to the whole string.** This pattern uses ^ and $, so it requires the entire input to match. To find it inside a longer text, drop the anchors and use the global (g) flag.
- **ASCII letters only.** [a-zA-Z] excludes accented and non-Latin letters (é, ü, ß, ñ, and non-Latin scripts). For international input use Unicode properties like \p{L} with the u flag.
- **Escape it correctly per language.** In Java and JavaScript strings each backslash must be doubled (\\d); in Python, Go, and C# use raw/verbatim strings so the backslashes survive.
- **Validate beyond format.** Matching the format doesn't guarantee the value is real. Confirm the jwt token against a source of truth (database, API, or checksum) where it matters.