XML namespace declaration

xmlns or xmlns:prefix attribute.

```
xmlns(:[a-zA-Z][\w-]*)?="[^"]*"
```

Token-by-token breakdown

Token	Meaning
`xmlns`	literal text “xmlns”
`(`	start of a capturing group
`:`	literal text “:”
`[a-zA-Z]`	any of: lowercase letters, uppercase letters
`[\w-]*`	zero or more: any of: word chars (A–Z, a–z, 0–9, _), “-”
`)?`	end of group, optional (zero or one)
`="`	literal text “=””
`[^"]*`	zero or more: any character except “””
`"`	literal text “””

Quick usage in 7 languages

JavaScript

```
const re = /xmlns(:[a-zA-Z][\w-]*)?="[^"]*"/;
re.test(value);
```

Python

```
import re
re.match(r'xmlns(:[a-zA-Z][\w-]*)?="[^"]*"', value)
```

Java

```
Pattern.compile("xmlns(:[a-zA-Z][\\w-]*)?=\"[^\"]*\"").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"xmlns(:[a-zA-Z][\w-]*)?=""[^""]*""");
```

Go

```
regexp.MustCompile(`xmlns(:[a-zA-Z][\w-]*)?="[^"]*"`).MatchString(value)
```

Ruby

```
/xmlns(:[a-zA-Z][\w-]*)?="[^"]*"/.match?(value)
```

PHP

```
preg_match('~xmlns(:[a-zA-Z][\w-]*)?="[^"]*"~', $value);
```

Common pitfalls

- **Not anchored.** Without ^ and $ this can match a substring anywhere in the input — add anchors if you need the whole value to conform.
- **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 xml namespace declaration against a source of truth (database, API, or checksum) where it matters.