The Jevko Syntax: Standard Grammar Specification

by Darius J Chuck

First Edition: February 2022. Revised: November 2022.

Introduction and scope

Jevko is a versatile minimal syntax for encoding tree-structured information.

It can be used to define simple and portable formats and languages in a variety of domains, such as data interchange, configuration, or text markup.

Jevko is completely programming-language independent and has no inherent semantics. It is a pure syntactic building block which can be joined together with other syntactic or semantic building blocks in a modular way.

These additional building blocks are specified separately. The purpose of this specification is only to describe the basic syntax in terms of Unicode [Unicode] code points, providing a reference for implementations as well as other specifications.

The formal grammar definitions in this document use the ABNF notation [RFC 5234].

Etymology and pronunciation

The name *Jevko* /ˈd͡ʒef.kəʊ/ is derived from Polish *drzewko* /ˈdʐɛf.kɔ/, meaning small tree.

Valid Jevko sequence

A valid Jevko sequence is a sequence of Unicode code points which conforms to the `Jevko` rule defined in this document.

The rules

The syntax of Jevko is extremely minimal.

There are two main grammatical rules: the `Jevko` rule and the `Subjevko` rule.

The `Jevko` rule is the *start symbol* of the grammar.

It refers to the `Subjevko` rule which in turn refers back to `Jevko`: the two rules are mutually-recursive.

This makes Jevko a *recursive grammar*.

This recursion is minimal: if a reference to either `Jevko` or `Subjevko` was removed, the grammar would cease to be recursive.

All remaining rules are non-recursive. Notably:

- The `Character` rule covers all possible code points except three.
- The three remaining code points (`Delimiters`) are the only special characters in the grammar.
- There are no special rules for whitespace. All whitespace characters are captured by the `Character` rule.

All rules are described and formally defined in the following sections. The first section covers rules for delimiters. The remaining sections describe the rest of the rules, top to bottom, starting with `Jevko`.

Delimiters

Jevko is based around three delimiters, named by the Delimiter rule:

```
Delimiter = Opener / Closer / Escaper
```

The delimiters are defined as the following code points:

```
Opener  = %x5b ; left square bracket
Closer  = %x5d ; right square bracket
Escaper = %x60 ; grave accent
```

Equivalently:

```
Opener  = "["
Closer  = "]"
Escaper = "`"
```

Jevko

A `Jevko` is a sequence of zero or more `Subjevko`s followed by a `Suffix`.

```
Jevko = *Subjevko Suffix
```

Subjevko

A `Subjevko` is a `Prefix` followed by an `Opener`, followed by a `Jevko`, followed by a `Closer`.

```
Subjevko = Prefix Opener Jevko Closer
```

Equivalently:

```
Subjevko = Prefix "[" Jevko "]"
```

Prefix

A `Prefix` is an alias for `Text`:

```
Prefix = Text
```

This alias identifies that the `Text` is a part of a `Subjevko`. A `Prefix` is *always* followed by an `Opener`.

Suffix

A `Suffix` is an alias for `Text`:

```
Suffix = Text
```

This alias identifies that the `Text` is a part of `Jevko`. A `Suffix` is *never* followed by an `Opener`.

Text

`Text` is a sequence of zero or more `Symbol`s:

```
Text = *Symbol
```

Symbol

A `Symbol` is either a `Digraph` or a `Character`:

```
Symbol = Digraph / Character
```

Digraph

A `Digraph` is an `Escaper` followed by a `Delimiter`:

```
Digraph = Escaper Delimiter
```

Equivalently:

```
Digraph = "`" ("`" / "[" / "]")
Digraph = "``" / "`[" / "`]"
```

Character

A `Character` is any code point which is not a `Delimiter`:

```
Character = %x0-5a / %x5c / %x5e-5f / %x61-10ffff
```

Normative references

[Unicode] The Unicode Consortium, “The Unicode Standard”, the latest version, http://www.unicode.org/versions/latest/.

[RFC 5234] Crocker, D., Ed. and P. Overell, “Augmented BNF for Syntax Specifications: ABNF”, STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, https://www.rfc-editor.org/info/rfc5234.

See also

- Jevko examples: https://github.com/jevko/examples
- Jevko interactive railroad diagrams: https://jevko.org/diagram.xhtml

Copyright and license

Appendix: The Standard Grammar ABNF in one page

```
; start symbol, main rule #1
Jevko = *Subjevko Suffix
; main rule #2, mutually recursive with #1
Subjevko = Prefix Opener Jevko Closer

; delimiters
Delimiter = Opener / Closer / Escaper

Opener  = %x5b ; left square bracket
Closer  = %x5d ; right square bracket
Escaper = %x60 ; grave accent

; aliases
Suffix = Text
Prefix = Text

; text
Text = *Symbol
Symbol = Digraph / Character
Digraph = Escaper Delimiter
; Character is any code point which is not a Delimiter
Character = %x0-5a / %x5c / %x5e-5f / %x61-10ffff
```