

# Typesets

## Table of Contents

1. Abstract .....	1
2. Available typesets in Red .....	1
2.1. all-word! .....	1
2.2. any-block! .....	2
2.3. any-function! .....	2
2.4. any-list! .....	2
2.5. any-object! .....	2
2.6. any-path! .....	2
2.7. any-string! .....	2
2.8. any-type! .....	2
2.9. any-word! .....	2
2.10. default! .....	2
2.11. external! .....	3
2.12. immediate! .....	3
2.13. internal! .....	3
2.14. number! .....	3
2.15. scalar! .....	3
2.16. series! .....	3

## 1. Abstract

Typesets represents sets of datatype values stored in a compact array of bits (up to 96 bits), allowing for high performance runtime type-checking support.

Datatypes in a typeset may share common traits or behaviors, but that is not a requirement. A typeset can be created based on any criteria that suits the users needs.

See: [typeset!](#)

## 2. Available typesets in Red

### 2.1. all-word!

- make typeset! [[word!](#) [set-word!](#) [lit-word!](#) [get-word!](#) [refinement!](#) [issue!](#)]

## 2.2. any-block!

- make typeset! [block! paren! path! lit-path! set-path! get-path! hash!]

## 2.3. any-function!

- make typeset! [native! action! op! function! routine!]

## 2.4. any-list!

- make typeset! [block! paren! hash!]

## 2.5. any-object!

- make typeset! [object! error!]

## 2.6. any-path!

- make typeset! [path! lit-path! set-path! get-path!]

## 2.7. any-string!

- make typeset! [string! file! url! tag! email! ref!]

## 2.8. any-type!

- make typeset! [datatype! unset! none! logic! block! paren! string! file! url! char! integer! float! word! set-word! lit-word! get-word! refinement! issue! native! action! op! function! path! lit-path! set-path! get-path! routine! bitset! object! typeset! error! vector! hash! pair! percent! tuple! map! binary! time! tag! email! handle! date! image! event!]

## 2.9. any-word!

- make typeset! [word! set-word! lit-word! get-word!]

## 2.10. default!

- make typeset! [datatype! none! logic! block! paren! string! file! url! char! integer! float! word! set-word! lit-word! get-word! refinement! issue! native! action! op! function! path! lit-path! set-path! get-path! routine! bitset! object! typeset! error! vector! hash! pair! percent! tuple! map! binary! time! tag! email! handle! date! image! event!]

## 2.11. external!

- make typeset! [event!]

## 2.12. immediate!

- make typeset! [datatype! none! logic! char! integer! float! word! set-word! lit-word! get-word! refinement! issue! typeset! pair! percent! tuple! time! handle! date!]

## 2.13. internal!

- make typeset! [unset! float! percent!]

## 2.14. number!

- make typeset! [integer! float! percent!]

## 2.15. scalar!

- make typeset! [char! integer! float! pair! percent! tuple! time! date! money!]

## 2.16. series!

- make typeset! [block! paren! string! file! url! path! lit-path! set-path! get-path! vector! hash! binary! tag! email! image!]

A series in Red is defined as a sequence of elements, and a starting position which can be moved along the sequence of elements from the first position (**head**), to the last position (**tail**). The starting position of an empty series is at the last position (**tail**).

Several references can be set to the same series with different starting positions:

```
>> a: "hello"
== "hello"

>> b: next a
== "ello"

>> index? a
== 1

>> index? b
== 2

>> same? a b
== false

>> same? a head b
== true

>> append a " world"
== "hello world"

>> b
== "ello world"
```

The type of the elements in a series is dependant on the [datatype!](#) of the series. For example, a [block!](#) series can contain values of any type. A [string!](#) series can only contain [char!](#) values, etc.

[Series!](#) provides an index variable that can be leveraged by all series [action!](#) values.