# difference between \w and \b regular expression meta characters

Can anyone explain the difference between \w and \b regular expression meta characters? Both these metacharacters are used for word boundaries. Apart from this, which meta character is efficient for multi lingual content?

regex

edited Aug 8 '12 at 23:05

asked Aug 8 '12 at 22:38
Mahender
2,787 5 30 50

11 `\w` represents a word *character*, while `\b` represents a word *boundary between* a word character and a non-word character. They're not the same thing. – BoltClock ♦ Aug 8 '12 at 22:41

## 5 Answers

The metacharacter `\b` is an anchor like the caret and the dollar sign. It matches at a position that is called a **"word boundary"**. This match is zero-length.

There are three different positions that qualify as word boundaries:

- Before the first character in the string, if the first character is a word character.
- After the last character in the string, if the last character is a word character.
- Between two characters in the string, where one is a word character and the other is not a word character.

Simply put: `\b` allows you to perform a **"whole words only"** search using a regular expression in the form of `\bword\b`. A **"word character"** is a character that can be used to form words. All characters that are not **"word characters"** are **"non-word characters"**.

In all flavors, the characters `[a-zA-Z0-9_]` are word characters. These are also matched by the short-hand character class `\w`. Flavors showing **"ascii"** for word boundaries in the flavor comparison recognize only these as word characters.

`\w` stands for **"word character"**, usually `[A-Za-z0-9_]`. Notice the inclusion of the underscore and digits.

`\B` is the negated version of `\b`. `\B` matches at every position where `\b` does not. Effectively, `\B` matches at any position between two word characters as well as at any position between two non-word characters.

`\W` is short for `[^\w]`, the negated version of `\w`.

edited Feb 4 '15 at 23:58

answered Aug 8 '12 at 23:50
Ωmega
24.6k 21 76 140

`\w` matches a word character. `\b` is a zero-width match that matches a position character that has a word character on one side, and something that's not a word character on the other. (Examples of things that aren't word characters include whitespace, beginning and end of the string, etc.)

`\w` matches `a`, `b`, `c`, `d`, `e`, and `f` in `"abc def"`
`\b` matches the (zero-width) position before `a`, after `c`, before `d`, and after `f` in `"abc def"`

See: http://www.regular-expressions.info/reference.html/

edited Aug 10 '15 at 20:24

answered Aug 8 '12 at 22:42
jwismar
8,198 2 19 35

3 It's more correct to say that it's the boundary between a word character and not a word character because it also matches between a word character and the start or end of a string if that character is at the start/end of the string. – MRAB Aug 8 '12 at 22:47

You're right, that's more correct. I'll edit. – jwismar Aug 8 '12 at 22:49

5 It's still not quite right. `\b` a zero-width assertion; it doesn't match a *character*, it matches a *position*. – Alan Moore Sep 13 '14 at 19:33

@Mahender, you probably meant the difference between `\W` (instead of `\w`) and `\b`. If not, then I would agree with @BoltClock and @jwismar above. Otherwise continue reading.

`\W` would match any non-word character and so its easy to try to use it to match word boundaries. The problem is that it will not match the start or end of a line. `\b` is more suited for matching word boundaries as it will also match the start or end of a line. Roughly speaking (more experienced users can correct me here) `\b` can be thought of as `(\W|^|$)`. [Edit: as @Ωmega mentions below, `\b` is a zero-length match so `(\W|^|$)` is not strictly correct, but hopefully helps explain the diff]

Quick example: For the string `Hello World`, `.+\W` would match `Hello_` (with the space) but will not match `World`. `.+\b` would match both `Hello` and `World`.

edited Aug 9 '12 at 1:33

answered Aug 8 '12 at 23:19

mtariq

**315** 1 9

---

```
\b <= this is a word boundary.
```

Matches at a position that is followed by a word character but not preceded by a word character, or that is preceded by a word character but not followed by a word character.

```
\w <= stands for "word character".
```

It always matches the ASCII characters [A-Za-z0-9_]

Is there anything specific you are trying to match?

Some useful regex websites for beginners or just to wet your appetite.

- http://www.regular-expressions.info
- http://www.javascriptkit.com/javatutors/redev2.shtml
- http://www.virtuosimedia.com/dev/php/37-tested-php-perl-and-javascript-regular-expressions
- http://www.i-programmer.info/programming/javascript/4862-master-javascript-regular-expressions.html

I found this to be a very useful book:

- Mastering Regular Expressions by Jeffrey E.F. Friedl

edited Dec 24 '13 at 12:02

Amal Murali

**56.9k** 12 88 107

answered Dec 19 '13 at 4:22

james emanon

**4,739** 4 22 43

5 This is a good answer, but it is useful to remember that `\w` is not always equivalent to the ASCII characters `[A-Za-z0-9_]` -- it will also match alphanumeric Unicode code points, and may match 8-bit ISO-Latin-1 characters if the locale is set appropriately. – Tim Pierce Dec 19 '13 at 5:14

---

`\w` is *not* a word boundary, it matches any word character, including underscores: `[a-zA-Z0-9_]`. `\b` *is* a word boundary, that is, it matches the position between a word and a non-alphanumeric character: `\W` or `[^\w]`.

These implementations may vary from language to language though.

answered Dec 19 '13 at 4:20

Julián Urbano

**6,971** 22 46

---

**protected** by Ωmega Dec 4 '13 at 13:14

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?