

Ruby monstas Ruby cheat sheet

Data types and how to use them



Name	Description	Structure	Examples
Integer literal	A whole number		3 -552
Floating point literal	A decimal number		42.23 -0.133742
Addition		$a + b$	5.2 + 6.34 # => 11.54
Subtraction		$a - b$	2.59 - 4.89 # => -2.3
Multiplication		$a * b$	5 * 3.7 # => 18.5
Division	Mind the difference between integer and float divisions!	a / b	6 / 4 # => 1 (integer division) 6 / 4.0 # => 1.5 (float division)
Modulo	Returns the remainder of a division	$a \% b$	13 \% 6 # => 1
String literal	A string of characters, text		“this is a string”
String interpolation	Text with Ruby code embedded in it		“another string with an #{interpolation}”

Arrays

Name	Description	Structure	Examples
Array literal	Creates a new array	<code>[item1, item2, ...]</code>	<code>my_array = [1, 2, 3]</code>
Length	Returns the length of the array (the number of items it contains)	<code>array.length</code>	<code>my_array.length # => 3</code>
Index operator	Lets you access the item at a given position within an array	<code>array[index]</code>	<code>my_array[1] # => 2</code>
<code>delete_at</code>	Deletes the item at a given index and returns it	<code>array.delete_at(index)</code>	<code>my_array.delete_at(1) # => 2</code>
<code>each</code>	Lets you iterate over all elements in an array	<code>array.each do item end</code>	<code>my_array.each do item puts item end</code>
<code>first</code>	Returns the very first item of the array	<code>array.first</code>	<code>my_array.first # => 1</code>
<code>last</code>	Returns the very last item of the array	<code>array.last</code>	<code>my_array.last # => 3</code>
<code>include?</code>	Returns a boolean, whether the array contains a certain element or not	<code>array.include?(item)</code>	<code>my_array.include?(4) # => false</code>
<code>pop</code>	Removes the last item of the array and returns it	<code>array.pop</code>	<code>my_array.pop # => 3</code>
<code>push or <<</code>	Adds an item to the end of the array	<code>array.push(item) array << item</code>	<code>my_array.push(4) my_array << 4</code>
<code>reverse</code>	Returns a copy of the array with the elements in reverse order	<code>array.reverse</code>	<code>my_array.reverse # [3, 2, 1]</code>
<code>sort</code>	Returns a sorted copy of the array	<code>array.sort</code>	<code>[5, 2, 4].sort # [2, 4, 5]</code>
<code>uniq</code>	Returns a copy of the array with duplicates removed	<code>array.uniq</code>	<code>[1, 1, 2, 2].uniq # [1, 2]</code>

Hashes

Name	Description	Structure	Examples
Hash literal	Create a hash	<code>{ "key" => "value" }</code>	<code>hash = []</code> <code>hash = { "key" => "value", "other_key" => 42 }</code>
Hash access	Access a value by its key	<code>hash[key]</code>	<code>hash["key"] # => "value"</code>
Key deletion	Delete a key-value pair by its key	<code>hash.delete(key)</code>	<code>hash.delete("key") # => "value"</code>
Empty hash	Remove all pairs from the hash	<code>hash.clear</code>	<code>hash.clear</code>
Iterate over hash	Iterate over all the pairs in the hash	<code>hash.each do key, value end</code>	<code>hash.each do key, value puts "#{key} has value: #{value}" end</code>
Iterate over pairs	Iterate over all the pairs in the hash	<code>hash.each_pair do key, value end</code>	<code>hash.each_pair do key, value puts "#{key} has value: #{value}" end</code>
Get key	Get a value for a key, with default value if the key does not exist.	<code>hash.fetch(key, default)</code>	<code>hash.fetch("key") # => "value"</code> <code>hash.fetch("xy", "default") => "default"</code>
Key existence	Ask the hash if it has a certain key	<code>hash.has_key?(key)</code>	<code>hash.has_key?("key") # => true</code>
Value existence	Ask the hash if it has a certain value	<code>hash.has_value?(value)</code>	<code>hash.has_value?("xy") # => false</code>
All keys	Get all the keys stored in the hash	<code>hash.keys</code>	<code>hash.keys # => ["key", "other_key"]</code>
All values	Get all the values stored in the hash	<code>hash.values</code>	<code>hash.values # => ["value", 42]</code>
Merge	Merge two hashes	<code>hash.merge(other_hash)</code>	<code>hash.merge({ "a_key" => 23 }) # => { "key" => "value", "a_key" => 23 }</code>