Click or press 's' or Ctrl+P to

Variable sized arrays / Vectors

let mut vec: Vec<T> =
Vec::new();
= Vec::with_capacity();
= vec![];
=
Vec::from(slice|str|VecDeque|CStri
= othervec.clone(); ▶ Details

Accessing

vec[3]; ▶ Details
vec.len();
.is_empty();
.first_mut(); .last_mut();
▶ Details
.get_mut(index); ▶ Details
.contains(needle); ▶ Details
.iter().find(|&T| -> bool);
▶ Details
.binary_search(x:&T); ▶ Details

Adding

.push(3); ▶ Details
.insert(index, element);
.extend(iterable);
.extend_from_slice(&[T]);
.append(other : Vec); ▶ Details

Removing

.pop(); ▶ Details
.remove(index); ▶ Details
.swap_remove(index); ▶ Details
.drain(range); ▶ Details
.clear();
.retain(|i| -> bool); ▶ Details

Manipulating

.sort(); ▶ Details
.sort_by(|&T|->Ordering);
▶ Details
.sort_by_key(|&T|->Key);
▶ Details
.reverse(); ▶ Details
.swap(index1, index2);

Transforming (Iter, as_, to_)

.iter_mut(); ▶ Details
.into_iter(); ▶ Details
.chunks_mut(cnk_sz); ▶ Details
.windows(wnd_sz); ▶ Details
.as_ref(); ▶ Details
.as_mut_slice(); ▶ Details

Memory

.reserve(100); ▶ Details
.reserve_exact(100); ▶ Details

Split

.split_at_mut(mid); ▶ Details
.split_mut(|&T| -> bool);

Hashmaps / Dicts

use std::collections::HashMap;
▶ Details
let mut foo: HashMap<K, V> =
HashMap::new();
= HashMap::with_capacity();
▶ Details
= other.clone(); ▶ Details

Access

foo[key];
foo.len();
.iter_mut(); ▶ Details
.into_iter(); ▶ Details
.keys(); ▶ Details
.values_mut(); ▶ Details
.is_empty(); ▶ Details
.contains_key(k:Q); ▶ Details

Manipulate

.get_mut(k:&Q); ▶ Details
.entry(key); ▶ Details
.drain(); ▶ Details
.clear();
.extend(iter : <Item=(&K,&V)>);
.insert(k,v); ▶ Details
.remove(k:&Q); ▶ Details
.from_iter(iter : <Item=(K,V)>);
▶ Details

Manage

.capacity();
.reserve(additional);
.shrink_to_fit();
.clone_from(source); ▶ Details

Comparision

.eq() .ne(); ▶ Details

Special Hasher

let hm =
HashMap::with_hasher(b);
=
HashMap::with_capacity_and_hasl
hm.hasher(b); ▶ Details

Traits

Clone clone() clone_from() |
PartialEq eq() ne() | Eq | Debug
fmt() | Default default() | Index
index() | IntoIterator into_iter() |
FromIterator from_iter() |
Extend extend() |

Option

let foo : Option = Some(T::new());
= None;

If

.is_some();
.is_none(); ▶ Details

&

.as_ref(); ▶ Details
.as_mut(); ▶ Details
.cloned(); ▶ Details
.iter_mut(); ▶ Details

Retrieve T

.unwrap(); ▶ Details
.expect(msg); ▶ Details
.unwrap_or(default:T); ▶ Details
.unwrap_or_default(); ▶ Details
.unwrap_or_else(|| -> T);
▶ Details
mutableopt.take(); ▶ Details

Manipulate (map)

.map(|t| -> U); ▶ Details
.map_or(default:U, |t| -> U);
▶ Details
.map_or_else(|| default -> U, |t| ->
U); ▶ Details

to Result<>

.ok_or(err:E); ▶ Details
.ok_or_else(|| err -> E); ▶ Details

Boolean Combinations

a.and(b : Option<U>); ▶ Details
a.and_then(|| b -> Option<U>);
▶ Details
a.or(b : Option<T>); ▶ Details
a.or_else(|| b -> Option<T>);
▶ Details

Traits

Hash hash() | Debug fmt() | Ord
cmp() | Eq | PartialOrd
partial_cmp() lt() le() gt() ge() |
PartialEq eq() ne() | Copy |
Clone clone() clone_from() |
Default default() | IntoIterator
into_iter() | FromIterator
from_iter() |

Result

let foo : Result = Ok(T::new());
= Err(E::new());

If

.is_ok();
.is_err(); ▶ Details

&

.as_ref(); ▶ Details
.as_mut(); ▶ Details
.iter_mut(); ▶ Details

Retrieve T

.unwrap(); ▶ Details
.expect(msg); ▶ Details
.unwrap_or(default:T); ▶ Details
.unwrap_or_else(|err| default ->
T); ▶ Details

Retrieve E

.unwrap_err(); ▶ Details

Manipulate (map)

.map(|t| -> U); ▶ Details
.map_err(|e| -> F); ▶ Details

to Option<>

.ok(); ▶ Details
.err(); ▶ Details

Boolean Combinations

a.and(b : Result<U,E>); ▶ Details
a.and_then(|| b -> Result<U,E>);
▶ Details
a.or(b : Result<T,E>); ▶ Details
a.or_else(|| b -> Result<T,E>);
▶ Details

Traits

Hash hash() | Debug fmt() | Ord
cmp() | Eq | PartialOrd
partial_cmp() lt() le() gt() ge() |
PartialEq eq() ne() | Copy |
Clone clone() clone_from() |
IntoIterator into_iter() |
FromIterator from_iter() |

Rust Cheatsheet

Contribute at
github.com/phaiax/rust-
cheatsheet

▶ Details

.splitn_mut(n, |&T| -> bool);

.rsplitn_mut(_); ▶ Details

.split_off(mid); ▶ Details

Comparision

Traits

From<BinaryHeap> from() | BorrowMut borrow /_mut() | Clone clone/_from() | Hash hash/_slice() | IndexMut index/_mut() DerefMut deref/_mut() | FromIterator from_iter() | IntoIterator into_iter() | Extend extend() | PartialEq eq() ne() | PartialOrd partial_cmp() lt() le() gt() ge() | Eq | Ord cmp() | Drop drop() | Default default() | Debug (if T:Debug) fmt() | AsRef AsMut as_ref() as_mut() | From from() | Write write() write_all() flush() by_ref() .. |