

Perl 6 / Rakudo

An Introduction

Install

Install

- `wget http://github.com/downloads/rakudo/rakudo/rakudo-2010.06.tar.gz`

Install

- `wget http://github.com/downloads/rakudo/rakudo/rakudo-2010.06.tar.gz`
- `tar xzf rakudo-2010.06.tar.gz`

Install

- `wget http://github.com/downloads/rakudo/rakudo/rakudo-2010.06.tar.gz`
- `tar xzf rakudo-2010.06.tar.gz`
- `perl Configure.pl --gen-parrot`

Install

- `wget http://github.com/downloads/rakudo/rakudo/rakudo-2010.06.tar.gz`
- `tar xzf rakudo-2010.06.tar.gz`
- `perl Configure.pl --gen-parrot`
- `make`

Install

- `wget http://github.com/downloads/rakudo/rakudo/rakudo-2010.06.tar.gz`
- `tar xzf rakudo-2010.06.tar.gz`
- `perl Configure.pl --gen-parrot`
- `make`
- `make install`

Basics

Basics

```
./perl6 -e 'say "Hello World"'
```

Basics

```
./perl6 -e 'say "Hello World"'
```

```
#Hello World
```

Basics

```
./perl6 -e 'say "Hello World"'
```

```
#Hello World
```

```
./perl6 -e 'my $x = 5; say $x;'
```

Basics

```
./perl6 -e 'say "Hello World"'
```

```
#Hello World
```

```
./perl6 -e 'my $x = 5; say $x;'
```

```
#5
```

Basics

```
./perl6 -e 'say "Hello World"'
```

```
#Hello World
```

```
./perl6 -e 'my $x = 5; say $x;'
```

```
#5
```

Looks just like Perl to me!

Chained Comparisons

Chained Comparisons

```
my $weight = 150;
```

Chained Comparisons

```
my $weight = 150;
```

```
if ( 140 <= $weight <= 147) {
```

Chained Comparisons

```
my $weight = 150;
```

```
if ( 140 <= $weight <= 147) {  
    say "is a welterweight";  
}
```

Chained Comparisons

```
my $weight = 150;
```

```
if ( 140 <= $weight <= 147) {  
    say "is a welterweight";  
}
```

Chained Comparisons

```
my $weight = 150;
```

```
if ( 140 <= $weight <= 147) {  
    say "is a welterweight";  
}  
else {
```

Chained Comparisons

```
my $weight = 150;
```

```
if ( 140 <= $weight <= 147) {
```

```
    say "is a welterweight";
```

```
}
```

```
else {
```

```
    say "out of the weight class";
```

Chained Comparisons

```
my $weight = 150;
```

```
if ( 140 <= $weight <= 147) {  
    say "is a welterweight";  
}  
else {  
    say "out of the weight class";  
}
```

Switch Case

Switch Case

```
my $x = 'blue';
```

Switch Case

my \$x = 'blue';

given \$x {

Switch Case

```
my $x = 'blue';
```

```
given $x {  
  when 'green' { say "envious" }  
}
```

Switch Case

```
my $x = 'blue';
```

```
given $x {  
  when 'green' { say "envious" }  
  when 'blue' { say "sad" }  
}
```

Switch Case

```
my $x = 'blue';
```

```
given $x {  
  when 'green' { say "envious" }  
  when 'blue' { say "sad" }  
  when 'red' { say "angry" }  
}
```

Switch Case

```
my $x = 'blue';
```

```
given $x {  
  when 'green' { say "envious" }  
  when 'blue' { say "sad" }  
  when 'red' { say "angry" }  
  default { say "happy" }
```

Switch Case

```
my $x = 'blue';
```

```
given $x {  
  when 'green' { say "envious" }  
  when 'blue' { say "sad" }  
  when 'red' { say "angry" }  
  default { say "happy" }  
}
```

Switch Case

```
my $x = 'blue';
```

```
given $x {  
  when 'green' { say "envious" }  
  when 'blue' { say "sad" }  
  when 'red' { say "angry" }  
  default { say "happy" }  
}
```

Now Available in Perl 5.12

Reading Files

Reading Files

```
my $filename = 'test.txt';
```

Reading Files

```
my $filename = 'test.txt';
```

```
if (my $handle = open $filename, :r) {
```

Reading Files

```
my $filename = 'test.txt';
```

```
if (my $handle = open $filename, :r) {  
    for $handle.readlines -> $line {
```

Reading Files

```
my $filename = 'test.txt';
```

```
if (my $handle = open $filename, :r) {  
  for $handle.readlines -> $line {  
    say $line;  
  }  
}
```

Reading Files

```
my $filename = 'test.txt';
```

```
if (my $handle = open $filename, :r) {  
    for $handle.readlines -> $line {  
        say $line;  
    }  
}
```

Reading Files

```
my $filename = 'test.txt';
```

```
if (my $handle = open $filename, :r) {  
    for $handle.readlines -> $line {  
        say $line;  
    }  
    $handle.close;
```

Reading Files

```
my $filename = 'test.txt';
```

```
if (my $handle = open $filename, :r) {  
    for $handle.readlines -> $line {  
        say $line;  
    }  
    $handle.close;  
}
```

Reading Files

```
my $filename = 'test.txt';
```

```
if (my $handle = open $filename, :r) {  
    for $handle.readlines -> $line {  
        say $line;  
    }  
    $handle.close;  
}  
else {
```

Reading Files

```
my $filename = 'test.txt';
```

```
if (my $handle = open $filename, :r) {  
    for $handle.readlines -> $line {  
        say $line;  
    }  
    $handle.close;  
}  
else {  
    die "Couldn't open $filename"}
```

Reading Files

```
my $filename = 'test.txt';
```

```
if (my $handle = open $filename, :r) {  
    for $handle.readlines -> $line {  
        say $line;  
    }  
    $handle.close;  
}  
else {  
    die "Couldn't open $filename"  
}
```

Arrays

Arrays

```
my @colors = ("red", "green", "blue");
```

Arrays

```
my @colors = ("red", "green", "blue");
```

```
say "{@colors}";
```

Arrays

```
my @colors = ("red", "green", "blue");
```

```
say "{@colors}";
```

```
# red green blue
```

Arrays

```
my @colors = ("red", "green", "blue");
```

```
say "{@colors}";
```

```
# red green blue
```

```
for @colors -> $color {
```

Arrays

```
my @colors = ("red", "green", "blue");
```

```
say "{@colors}";
```

```
# red green blue
```

```
for @colors -> $color {  
    say $color  
}
```

Arrays

```
my @colors = ("red", "green", "blue");
```

```
say "{@colors}";
```

```
# red green blue
```

```
for @colors -> $color {  
    say $color  
}
```

Arrays

```
my @colors = ("red", "green", "blue");
```

```
say "{@colors}";
```

```
# red green blue
```

```
for @colors -> $color {  
    say $color  
}
```

```
# one per line
```

Hashes

Hashes

```
my %car = (color => 'black', year => '2001', make =>
'Honda');
```

Hashes

```
my %car = (color => 'black', year => '2001', make =>
'Honda');
```

```
if (%car.exists('color')) {}
```

Hashes

```
my %car = (color => 'black', year => '2001', make =>
'Honda');
```

```
if (%car.exists('color')) {}
```

```
for %hash.keys.sort -> $key { } # sorted by keys
```

Hashes

```
my %car = (color => 'black', year => '2001', make =>
'Honda');
```

```
if (%car.exists('color')) {
```

```
for %hash.keys.sort -> $key { } # sorted by keys
```

```
for %hash.kv -> $key, $value {
```

Hashes

```
my %car = (color => 'black', year => '2001', make =>
'Honda');
```

```
if (%car.exists('color')) {
```

```
for %hash.keys.sort -> $key { } # sorted by keys
```

```
for %hash.kv -> $key, $value {
    say "The $key is $value.";
```

Hashes

```
my %car = (color => 'black', year => '2001', make =>
'Honda');
```

```
if (%car.exists('color')) {}
```

```
for %hash.keys.sort -> $key { } # sorted by keys
```

```
for %hash.kv -> $key, $value {
  say "The $key is $value.";
}
```

Subroutines

Subroutines

normal

Subroutines

normal

```
sub add ($a, $b) { }
```

Subroutines

```
# normal  
sub add ($a, $b) { }  
add(3,4);
```

Subroutines

normal

```
sub add ($a, $b) { }
```

```
add(3,4);
```

optional

Subroutines

normal

```
sub add ($a, $b) { }
```

```
add(3,4);
```

optional

```
sub format_name ($first, $last?) { }
```

Subroutines

normal

```
sub add ($a, $b) { }
```

```
add(3,4);
```

optional

```
sub format_name ($first, $last?) { }
```

```
format_name('JT');
```

Subroutines

normal

```
sub add ($a, $b) { }  
add(3,4);
```

optional

```
sub format_name ($first, $last?) { }  
format_name('JT');
```

named and optional

Subroutines

normal

```
sub add ($a, $b) { }  
add(3,4);
```

optional

```
sub format_name ($first, $last?) { }  
format_name('JT');
```

named and optional

```
sub add_user ( $name, :$twitter?, :$email?) { }
```

Subroutines

normal

```
sub add ($a, $b) { }  
add(3,4);
```

optional

```
sub format_name ($first, $last?) { }  
format_name('JT');
```

named and optional

```
sub add_user ( $name, :$twitter?, :$email?) { }  
add_user('JT', email => 'jt@madmongers.org');
```

Objects

Objects

```
class Point {
```

Objects

```
class Point {  
  has $.x;
```

Objects

```
class Point {  
  has $.x;  
  has $.y;
```

Objects

```
class Point {  
  has $.x;  
  has $.y;  
  method reset { $.x = 0, $.y = 0 };
```

Objects

```
class Point {  
  has $.x;  
  has $.y;  
  method reset { $.x = 0, $.y = 0 };  
};
```

Objects

```
class Point {  
  has $.x;  
  has $.y;  
  method reset { $.x = 0, $.y = 0 };  
};  
  
my $point = Point.new(x => 23, y => 42);
```

Objects

```
class Point {  
  has $.x;  
  has $.y;  
  method reset { $.x = 0, $.y = 0 };  
};  
  
my $point = Point.new(x => 23, y => 42);  
say $point.x; # 23
```

Objects

```
class Point {  
  has $.x;  
  has $.y;  
  method reset { $.x = 0, $.y = 0 };  
};  
  
my $point = Point.new(x => 23, y => 42);  
say $point.x; # 23  
$point.x = 10; # assignment to 10
```

Objects

```
class Point {  
  has $.x;  
  has $.y;  
  method reset { $.x = 0, $.y = 0 };  
};  
  
my $point = Point.new(x => 23, y => 42);  
say $point.x; # 23  
$point.x = 10; # assignment to 10  
say $point.WHAT; # Point
```

Objects

```
class Point {  
  has $.x;  
  has $.y;  
  method reset { $.x = 0, $.y = 0 };  
};
```

```
my $point = Point.new(x => 23, y => 42);  
say $point.x; # 23  
$point.x = 10; # assignment to 10  
say $point.WHAT; # Point  
$a.reset;
```