RUBY IF...ELSE, CASE, UNLESS

Ruby offers conditional structures that are pretty common to modern languages. Here, we will explain all the conditional statements and modifiers available in Ruby.

Ruby *if...else* Statement:

Syntax:

```
if conditional [then]
   code...
[elsif conditional [then]
   code...]...
[else
   code...]
end
```

if expressions are used for conditional execution. The values *false* and *nil* are false, and everything else are true. Notice Ruby uses elsif, not else if nor elif.

Executes *code* if the *conditional* is true. If the *conditional* is not true, *code* specified in the else clause is executed.

An if expression's *conditional* is separated from code by the reserved word *then*, a newline, or a semicolon.

Example:

```
#!/usr/bin/ruby

x=1
if x > 2
   puts "x is greater than 2"
elsif x <= 2 and x!=0
   puts "x is 1"
else
   puts "I can't guess the number"
end
```

```
x is 1
```

Ruby *if* modifier:

Syntax:

```
code if condition
```

Executes *code* if the *conditional* is true.

Example:

```
#!/usr/bin/ruby

$debug=1
print "debug\n" if $debug
```

This will produce the following result:

```
debug
```

Ruby *unless* Statement:

Syntax:

```
unless conditional [then]
   code
[else
   code ]
end
```

Executes *code* if *conditional* is false. If the *conditional* is true, code specified in the else clause is executed.

Example:

```
#!/usr/bin/ruby

x=1
unless x>2
   puts "x is less than 2"
 else
  puts "x is greater than 2"
end
```

This will produce the following result:

```
x is less than 2
```

Ruby *unless* modifier:

Syntax:

```
code unless conditional
```

Executes *code* if *conditional* is false.

Example:

```
#!/usr/bin/ruby

$var =  1
print "1 -- Value is set\n" if $var
print "2 -- Value is set\n" unless $var

$var = false
print "3 -- Value is set\n" unless $var
```

This will produce the following result:

```
1 -- Value is set
3 -- Value is set
```

Ruby *case* Statement

Syntax:

```
case expression
[when expression [, expression ...] [then]
   code ]...
[else
   code ]
end
```

Compares the *expression* specified by case and that specified by when using the === operator and executes the *code* of the when clause that matches.

The *expression* specified by the when clause is evaluated as the left operand. If no when clauses match, *case* executes the code of the *else* clause.

A when statement's expression is separated from code by the reserved word then, a newline, or a semicolon.

Thus:

```
case expr0
when expr1, expr2
   stmt1
when expr3, expr4
   stmt2
else
   stmt3
end
```

is basically similar to the following:

```
_tmp = expr0
if expr1 === _tmp || expr2 === _tmp
   stmt1
elsif expr3 === _tmp || expr4 === _tmp
   stmt2
else
   stmt3
end
```

Example:

```
#!/usr/bin/ruby

$age =  5
case $age
when 0 .. 2
    puts "baby"
when 3 .. 6
    puts "little child"
when 7 .. 12
    puts "child"
when 13 .. 18
    puts "youth"
else
    puts "adult"
end
```

This will produce the following result:

```
little child
```