Строгий не null режим(Rust mode)

Если вы компилируете ваш код с ключом --enable-experimental-non-null, то компилятор Vala будет строго проверять каждый тип на не null, если только явно не объявлено обратное с помощью вопросительного знака:

```
Object o1 = new Object(); // не может быть null
Object? o2 = new Object(); // может быть null
```

Компилятор будет следить за тем, чтобы ссылки, которые могут содержать null не были присвоены ссылкам, которые не могут быть null, т.е. такого рода присвоения будут невозможны:

```
o1 = o2;
```

o2 может быть null, а o1 объявлен не null, поэтому такое присвоение запрещено. Тем не менее, вы можете переопределить такое поведение с помощью приведения ссылки к не null, если есть уверенность, что o2 не null:

```
o1 = (!) o2;
```

Строгая проверка в не null режиме помогает избежать использования нежелательного использования null ссылок. Эта возможность покажет свой потенциал полностью, если все типы в биндингах будут отмечены на предмет содержания null ссылок правильным образом, что пока еще встречается редко.

Литералы регулярных выражений(regexp)

Регулярные выражения - это мощное средство поиска по шаблону в строках. Vala имеет экспериментальную поддержку литералов для регулярных выражений. Например:

```
string email = "tux@kernel.org";
if (/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$/i.match(email)) {
   stdout.printf("Valid email address\n");
}
```

i в конце делает выражение не чувствительным к регистру. Вы можете хранить регулярное выражение в переменной типа Regex:

Regex regex = /foo/;

A example of regular expression replacement:

```
var r = /(foo|bar|cow)/;
var o = r.replace ("this foo is great", -1, 0, "thing");
print ("%s\n", o);
```

The following trailing characters can be used:
• i, letters in the pattern match both upper- and lowercase letters
• m, the "start of line" and "end of line" constructs match immediately following or immediately before any newline in the string, respectively, as well as at the very start and end.
• s, a dot metacharater . in the pattern matches all characters, including newlines. Without it, newlines are excluded.
• x, whitespace data characters in the pattern are totally ignored except when escaped or inside a character class.

Цепочки связанных выражений

Эта возможность позволяет писать сложные связанные выражения типа

```
if (1 < a && a < 5) {}
if (0 < a && a < b && b < c && c < d && d < 255) {
   // do something
}
```

в более естественном виде

```
if (1 < a < 5) {}
if (0 < a < b < c < d < 255) {
   // что-то делаем
}
```