Vim：文本对象操作

回顾

上节介绍了操作符+移动的操作逻辑：`[count]{operator}{motion}`

用当前光标与 `motion` 之间的区域作为 `operator` 操作的对象

本节介绍 `{operator}{textobjects}` 这一操作逻辑

文本对象操作

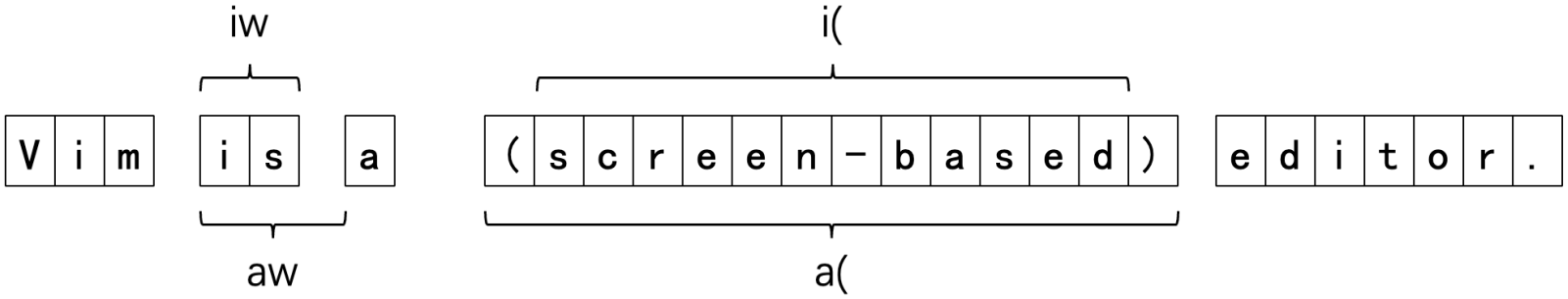
`textobjects`：语义化的文本片段

格式：`i` / `a` + 对象

常见的对象：

- `w` / `W`，`s`，`p`：单词、句子、段落
- `(` / `)`，`[` / `]`，`{` / `}`，`<` / `>`，`'` / `"`：配对符定义的对象

`i` 代表“inner”，内部；`a` 代表 “a”，额外包括周围的空格或配对符



文本对象操作：例子

文本对象提供了为文本赋予了**结构化**的含义，允许我们**以一个语义对象作为操作单元**

`[count]{operator}{textobjects}`

- `diw`：删除一个单词
- `ci(`：修改小括号内部
- `yi{`：复制大括号内部

通过组合 `operator` 与 `textobjects`，可以对不同的语义对象实施不同的操作，不仅十分灵活，而且语义明确，容易记忆

配合 `.` 命令或 `[count]` 可以简单地完成多次对特定语义对象的操作

textobjects VS motion

`{operator}{motion}` 与 `{operator}{textobjects}`

解耦了操作与操作的对象，大大提升了操作效率

- `motion` 是能够移动光标的命令，可以独立使用（如 `wbe`）
- 文本对象只能跟在 `operator` 后面，不能独立使用（如 `iw`）
- `motion` 通过光标的移动确定 `operator` 的作用范围，范围更加灵活但不够明确
- `textobjects` 则是显式地指定操作的对象，范围明确

扩展

`{operator}{motion}` 与 `{operator}{textobjects}`

可以通过自定义 `operator`、`motion`、`textobjects` 进行扩展，实现更强大的操作

- vim-easymotion：扩展 `motion`，更强大的跳转功能
- vim-surround：定义了添加配对符（括号、引号等）的 `operator`
- vim-commentary：定义了添加注释的 `operator`
- targets.vim：扩展 `textobjects`，定义了新的文本对象，如函数参数等

操作符与命令补充

- `gu` / `gU` / `g~` ： 操作符，转小写/转大写/翻转大小写
- `J` ： join，连接两行
- `<Ctrl-a>` / `<Ctrl-x>` ： 增加数字/减少数字
- `g<Ctrl-A>` ： 创建递增序列
- `<` / `>` ： 左/右缩进

建议：让你的命令更模块化

尽量使你的命令更模块化，具有清晰的含义与作用范围，以便于与 `.` 等命令协同

例如： `daw` 比 `dw` 具有更清晰的语义，也更模块化

练习技巧：用 `v` 先可视化修改的范围

一开始不熟练时，可能不确定 `motion` 或 `textobjects` 具体覆盖了哪部分文本

可以先用 `v{motion}` / `v{textobjects}` 将范围选中，再执行 `c` / `d` / `y` 等操作

但是，当熟练掌握后，尽可能直接使用完整的命令进行操作，因为相比于 `viwd` ， `diw` 更加模块化，也更容易进行重复操作

小结

- 了解常用的文本对象
- （**重要**）掌握 `{operator}{textobjects}` 的操作哲学
- 尽量使你的命令模块化，语义清晰，易于重复