|  |  |
| --- | --- |
| TEA  PRIMITIVE | SEMANTICS |
| D: | |  |  | | --- | --- | | NAME | Delete | | PURPOSE | Delete something from the AI | | SYNTAX  & SEMANTICS | |  | | --- | | d: | | INERT | | d:REGEX  d:RX1:RX2:…:RXN | | Delete from AI all sections matching the single regular expression REGEX or any of the given regular expressions RX1, RX2,…, RXN | | d.:REGEX | | The Exceptional Delete Instruction, that deletes from AI all sections matching the single regular expression REGEX. It doesn’t accept multiple parameters, and so, everything after the first “:” is treated as part of the pattern. It is one of few TEA commands using the “.” Qualifier. | | d!: | | Delete all white-space from the AI (same as g:) | | d!:REGEX  d!:RX1:RX2:…:RXN | | The Inverse of d:, for which only sections not matching the given patterns are deleted from AI. | | d\*:vREGEX  d\*:vRX1:vRX2:…:vRXN | | Like the d:REGEX and d:RX1:…:RXN except, referencing the patterns stored in the named vaults. | | d\*!:vREGEX  d\*!:vRX1:vRX2:…:vRXN | | Like the d!:REGEX and d!:RX1:…:RXN except, referencing the patterns stored in the named vaults. | |  | | NOTES | One can appreciate **D:** by the following illustrative examples:  i!: “bC CB BA aB” | #(=“bC CB BA aB”)  d:[aA] |#(=“bC CB B B”)  But  i!: {bC CB BA aB} |#(=“bC CB BA aB”)  d:aA |#(=“bC CB BA aB”) because no pattern “aA”  While  i!: {bC CB BA aB} | #(=“bC CB BA aB”)  d!: |#(=“bCCBBAaB”)  And  i!: {bC CB BA aB} |#(=“bC CB BA aB”)  d:[aA] |#(=“bC CB B B”)  d:.B |#(=“bC”)  Or rather  i!: {bC CB BA aB} | d:[aA]:.B | #(=“bC”)  Note that parameterized inverse of the Delete command makes implementing powerful complex filters easier. For example, we eliminate anything in AI that isn’t a punctuation mark using the following terse program:  d!:[.,?;:] |  | |