Universal Typewriter

When creating paperclips, all materials should be dedicated to increasing production. So, why should you still get access to a GPU?

Beat *[Universal Paperclips](https://www.decisionproblem.com/paperclips/)* without the UI.

Gameplay

Beat the game only using text commands. By opening Chrome's developer console, you gain access to the ability to call internal functions and navigate the UI without the mouse. Use these to beat the game without any visuals

__Prented to be the mouse:__
Navigate the different UI elements by searching through the defined variables and calling the elements' `click()` function. Keep clicking the buttons just like you would with the normal UI and mouse.

__Hack the game:__
Call internal methods and navigate internal structures that aren't normally available. Use these to gain an advantage and quickly cheat some paperclips into existance.

Do whatever is needed to get to `30,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000,000` paperclips. Just don't use your eyes.

Setup

Open the [game](https://www.decisionproblem.com/paperclips/) in *Chrome* and click on the *Universal Paperclips* image.

On your keyboard, press the `F12` key to open the *DevTools* window. Click on the __Console__ tab at the top of the *DevTools* window.

Click on the three dots (`⋮`) in the top right coner of the *DevTools* to see advanced options. Click on the first icon (`Undock into seperate window`) after `Dock Side`. This should open the *DevTools* into a seperate window. Minimize the window with the game running. Expand the *DevTools* window.

You should now only be seeing the *DevTools* console. The UI for the game should not be visible.

Rules

- You cannot click on the UI; the *DevTools* window must take up the full screen for the duration of play (the UI of the game should not be visible)[^1][^2]
- You cannot view any source files[^3]
- You cannot view the __Elements__ tab within the *DevTools* window
- You cannot use assignment operators (`=`, `++`, `+=`, `--`, `-=`, etc...)[^4][^5]
- You cannot do mutliple function calls on a single line[^6]
  - Once you type `()`, you must hit `enter`
- Functions cannot be passed arguments
  - All functions must be called using the format `functionName()` and __not__ `functionName(a,b,c)`
- Any function which uses a dot operator (`.`) or an index operator (`[]`) to be called (i.e. any function contained in a namespace, an object, or a list/array)

cannot be called
  - A function must called using the format `singleName()` and not `name1.name2()` nor `name1[i]()` nor any combination or the two
  - This rule can be ignored if the function is `click` method[^7][^8]. In this case, the click method can be called with any number of proceeding `.` or `[]`. e.g. the following formats of calling functions is allowed and encouraged:

    ```js
    variableNameElement.click();
    variableName.element.click();
    namespace1.namespace2.variableElement.click();
    list[index].click();
    namespace.list[index1][index2].variable.element.click();
    ```

    Of note, the following declarations or similar declarations are not allowed:

    ```js
    variableName.click().element.click();
    variableName.element.click().click();
    variableName.element.click().element.click();
    ```

    Note in almost all cases `click()` will return `undefined` and thus this will throw an error, nevertheless these types of calls are still prohibitted

- Functions cannot be stored as global variables[^9]
- jQuery calls are not permited. (i.e. any call using the variable `$`)
- Loops are not permited. (i.e. you cannot use neither `for` nor `while` loops in the console)
- The game is over once `clips == 3e+55` evaluates to `true`.

[^1]: This rule can be ignored on start-up wherein the user can click the paperclip box to start the game
[^2]: Clicking, both right and left clicks, are permitted within the *DevTools* window as long as it does not overrule any other rules (i.e. you cannot click on the __Sources__ nor the __Elements__ tab)
[^3]: This includes all files with the extension `.css`, `.js`, and `.html`
[^4]: The full list of assignment operators includes `=`,`+=`,`-=`,`*=`,`/=`,`%=`,`**==`,`<<=`,`>>=`,`>>>=`,`&=`,`^=`,`|=`,`&&=`,`||=`,`??=`,`++`,`--`
[^5]: This includes defining a function using `function foo(){...}`
[^6]: Although the console can handle new line characters and hence multiple lines per console entry, this is not allowed. Thus, one function call per console entry.
[^7]: This only refers to the rule which prohibits the usage of `.` and `[]` within function calls. All previous and proceeding rules still apply in this case.
[^8]: This can only be used with a `click` function that is attached to an HTML button. Using `variable.click()` is prohibitted when the code

    ```js
    variable instanceof HTMLElement && variable.nodeName === 'BUTTON'
    ```

    evaluates to `false`
[^9]: Any variable that is not a function is not subject to this rule. Thus, you could save the result of a function call as a global variable to have a shorter name for future reference