**hyperscript cheatsheet**

**Event listeners**

- `on` event
  - `every` - add event listener
  - `mousemove` - do not queue events
  - `(clientX, clientY)` - event name
  - `[clientX > 100]` - expose the event’s properties
  - `3` - filter events
  - `or 3 to 10` - only respond to 3rd click
  - `or 3 and on` - respond to 3rd, 4th ... 10th click
  - `from #my-form` - respond to all clicks except 1st and 2nd
  - `debounced at 200ms` - filter events
  - `or throttled at 200ms` - every 200ms at most regardless of the number of events
  - `or keyup ...` - specify many events, each with its own from/debounce/...
  - `queue all` - add them to a FIFO queue
  - `or queue none` - discard them
  - `or queue first` - enqueue the first one, discard the rest
  - `or queue last` - enqueue the last one, discard the rest (this is the default)

**Property access**

- `user.data.name` - user’s data’s name
- `user.data.name of user` - name of data of user
- `data.name of user` - name of data of user
- `user's data.name` - user’s data’s name

**CSS literals**

- `#my-form` - Get element by id
- `#{getID()}` - Dynamic ID
- `.active` - Get elements by class
- `.getClass()` - Dynamic class
- `<em, i />` - Query selector all
- `<ul:nth-child($n) />` - Dynamic selector

**Array operations**

- `first in arr = first from arr`
- `= first of arr = first arr`
- also `random arr, last arr`

**Finding elements**

- `closest <section/>` - nearest enclosing section
- `previous <section/>` - from #sec-2 last section that comes before section 2
- `next <input, button, a/>` - from document.activeElement within #form with wrapping element to focus when pressing Tab in a modal dialog

**Variable scopes**

- `foo` - local variable by default
- `:foo` - element scoped variable, persisted
  - can be declared with top-level set
  - behaviors are isolated from one another
- `$foo` - global variable

Honorable mentions:

- `localStorage.foo` - value in local storage
- `@foo` - HTML attribute
**Command index**

required, optional, (?=? default value)

**Ex.** do *argA with argB* and optional *argC*
does stuff with *argA, argB and argC* (?=? default value)

add `.class to *elt`
add `@attribute=value to *elt`
add `{ font-size: ${size}px; }` to *elt*
add classes/attributes/inline styles to *elt* (?=? me)

append *value to *target*
append to strings/arrays/elements, sets *it = target* (?=? it)

async *command | async do *command... end*
run commands in a non-blocking manner

call *expr | get *expr* sets *it = expr*

call *continue* skips to next iteration in a loop

decrement *lvalue* by *amount*
sets *lvalue = lvalue - amount* (?=? 1)

fetch `/url` with *option: value, ...
fetch `'/url/${id}/'` with *option: value, ...
makes an HTTP request, see Fetch API docs for options

**goto** `/url` in new window
**goto** `'//url/${id}/'`
navigate to a URL in the browser

go to top of *elt* -- top/middle/bottom
go to top left of *elt* -- left/center/right
go to left of *elt* smoothly -- /instantly
scroll an element into view

halt the event’s default  prevent default behavior
halt default  same as above, and exits listener
halt the event’s bubbling  stop event bubbling
halt bubbling  same as above, and exits listener
halt the event  stop both default and bubbling
halt all of the above

hide *elt* with *strategy* see show

if *cond* then ... else ... end  if statement

increment  see decrement

**js(var) ... end**  embed JavaScript

log *value* with *func*
logs the value to the console using *func* (?=? console.log)

**make a `<tag#id.class>` called *name***
creates an element with the given *tag, id and classes, sets *name* (?=? id) = the created element

**make a *Class* from *args... called *name***
calls the *Class* constructor with the *args, sets *name* (?=? it) = the created object

**put rvalue into lvalue**  see set

**put content into *elt***
-- into/before/after/at start of/after end of
insert *content* into various parts of the *elt*

remove `.class` from *elt*  see add
remove `@attribute` from *elt*  see add
remove *elt*  removes *elt* (?=? me) from the document

repeat for *name in iterable* index *i ... end*
for *name in iterable* index *i ... end*
loop over an iterable, the loop variable is *name* (?=? it)

repeat until event *e* from *elt* index *i ... end*
Repeat every tick until event *e* is received from *elt* (?=? me)

repeat while *cond | repeat until *cond ... end*
repeat *n* times index *i ... end*
repeat forever ... end

return *value | exit*  return, see also halt

send *evt(args...)* to *elt*
trigger *evt(args...)* on *elt*
dispatch a DOM event on *elt* (?=? me)

set *lvalue to rvalue*

settle  waits for any animations/transitions to end

show *elt* with *strategy* when *cond*
-- strategy: display:~/visibility/opacity/... show *elt* (?=? me) using the *strategy* (?=? display:block) if
*cond* (?=? true), else hide it

take `.class` from *eltA* for *eltB*
remove class from *eltA* (?=? .class) and add it to *eltB* (?=? me)

tell *elt* ... end  set *you = elt*, default to you over *me*

throw *exception*  throws an exception

toggle `.class` on *eltA* for *t* s
**toggle [@attr=value] until *evt from eltB***
toggle between `.class1` and `.class2` on *eltA*
toggle classes and attributes on *eltA* (?=? me)

transition the *elt’s*
**prop-a from value to value ... over *t* s***
Animate style properties

wait *t* s --- or ms  Waits for the given duration

wait for *event* or *event2* or *t* s
waits for one of the events to occur, sets *it* the event