ACT

Cheat Sheet

main

Render text

```
main('Hello')
```

Render tag

```
main(['h1', 'Hello'])
```

Render function

```
const view = () => ['h1', 'Hello']

main(view)
```

Render function with model

```
const view = (val) => ['em', val]
const model = 10

main(view, { model })
```

With reducer

```
const reducer = (state = 0, {type, payload}) => {
  switch(type) {
    case 'em'
      return state + 1
    default:
      return state
  }
}

main(view, { reducer })
```

With subscriptions

```
const subscriptions = {
  scroll
}

main(view, { subscriptions }
```

Render on a specific DOM node

```
const node = document.getElementById('app')

main(view, { node })
```

With different history

```
main(view, { historyClass: TraversableHistory })
```

views

Simple tag

```
['h1']
```

Tag with id

```
['h1#title']
['h1', {id: 'title'}]
```

Tag with class

```
['div.main']
['div', {class: 'main'}]
```

Tag with attributes

```
['input', {type: 'checkbox', checked: true}]
```

Tag with styles

```
['i', {style: 'color: red; margin: 5px'}]
['i', {style: {color: 'red', margin: 5}}]
```

Tag with css modules (constant)

```
['i', {style: [styles, 'main']}]
```

Tag with css modules (conditional)

```
['i', {style: [styles, [open, 'is_open']]}]
['i', {
  style: [styles, [open, 'is_open', 'is_closed']]
}]
```

Tag with text child

```
['h1', 'Hello']
['em', 12]
```

Tag with child

```
['main', ['h1', 'Hello']]
```

Tag with children

```
['main', [
  'Hello',
  ['br'],
  'World'
]]
```

actions

Emit action with type and no payload

```
['button', {click: 'add'}]
```

Emit action with type and constant payload

```
['button', {click: {add: 1}}]
```

Emit action with type and signal value payload

```
['input', {keyup: {update: valueOnEnter}}]
```

Emit multiple actions with type and signal value payloads

```
['input', {keyup: {
  update: valueOnEnter,
  clear: valueOnEsc,
}}]
```

Execute side effect with original value

```
['input', {change: sideEffect}]
```

Execute side effect with signal value

```
['input', {change: [sideEffect, value]}]
```

Execute multiple side effects with signal value

```
['input', {change: [
  [sideEffect1, value],
  [sideEffect2, onKeyCode(33)]
]}]
```

Mixing all of the above

```
['input', {change: [
  ['event', identity],
  ['add', always(1)],
  ['update', valueOnEnter],
  [sideEffect, onKeyCode(66)]
]}]
```

Side effect function

```
const sideEffect = (history, payload) {
  const result = doSomeSideEffect(payload)
  history.push({
    type: 'side_effect',
    payload: result
  })
}
```

subscriptions

Subscriptions emiting action from signal

```
const subscriptions = {
  actionType: signal
}
const subscriptions = {
  scroll
}
```

Subscription calling side effect from signal

```
const subscriptions = [
  [sideEffect, signal]
]
```