

```
input
|> hash_input
|> pick_color
end
```

```
def pick_color(image) do
  %Identicon.Image{hex: [r, g, b | _tail]} = image

  [r, g, b]
end
```

```
def hash_input(input) do
  hex = :crypto.hash(:md5, input)
  |> :binary.bin_to_list

  %Identicon.Image{hex: hex}
end
```

```
defstruct hex: nil
```

```
nd
```

```
I
```

```
defstruct hex: nil, color: nil  
nd
```

```
|> hash_input  
|> pick_color  
end
```

```
def pick_color(image) do  
  %Identicon.Image{hex: [r, g, b | _tail]} = image  
  |  
end
```

```
def hash_input(input) do  
  hex = :crypto.hash(:md5, input)  
  |> :binary.bin_to_list  
  
  %Identicon.Image{hex: hex}  
end
```

```
|> hash_input  
|> pick_color  
end
```

```
def pick_color(image) do  
  %Identicon.Image{hex: [r, g, b | _tail]} = image  
  
  %Identicon.Image{image | color: {r, g, b}}  
end
```

```
def hash_input(input) do  
  hex = :crypto.hash(:md5, input)  
  |> :binary.bin_to_list  
  
  %Identicon.Image{hex: hex}  
end
```

```
end
```

ling 2 files (.ex)

```
)> Identicon.main("asdf")
```

```
Identicon.Image{color: {145, 46, 200},
```

```
[145, 46, 200, 3, 178, 206, 73, 228, 165, 65, 6, 141,
```

```
181, 112]}
```

```
)> █
```

```
|> hash_input  
|> pick_color  
end
```

```
def pick_color(image) do  
  %Identicon.Image{hex: [r, g, b | _tail]} = image  
  
  %Identicon.Image{image | color: {r, g, b}}  
end
```

```
def hash_input(input) do  
  hex = :crypto.hash(:md5, input)  
  |> :binary.bin_to_list  
  
  %Identicon.Image{hex: hex}  
end
```



```
input
  |> hash_input
  |> pick_color
end

I

def pick_color(%Identicon.Image{hex: [r, g, b | _tail]} = image) do
  %Identicon.Image{image | color: {r, g, b}}
end

def hash_input(input) do
  hex = :crypto.hash(:md5, input)
  |> :binary.bin_to_list

  %Identicon.Image{hex: hex}
end
end
```



```
)> Identicon.main("asdf")
Identicon.Image{color: {145, 46, 200},
  [145, 46, 200, 3, 178, 206, 73, 228, 165, 65, 6, 141,
  181, 112]}
)> recompile
linking 1 file (.ex)

)> Identicon.main("asdf")
Identicon.Image{color: {145, 46, 200},
  [145, 46, 200, 3, 178, 206, 73, 228, 165, 65, 6, 141,
  181, 112]}
```

```
end
```

```
def pick_color(%Identicon.Image{hex: [r, g, b | _tail]} = image) do  
  %Identicon.Image{image | color: {r, g, b}}  
end
```

```
pick_color: function(image) {  
  image.color = {  
    r: image.hex[0],  
    g: image.hex[1],  
    b: image.hex[2]  
  };  
  
  return image  
}
```

```
end
```

```
def pick_color(%Identicon.Image{hex: [r, g, b | _tail]} = image) do  
  %Identicon.Image{image | color: {r, g, b}}  
end
```

```
I
```

```
def hash_input(input) do  
  hex = :crypto.hash(:md5, input)  
  |> :binary.bin_to_list  
  
  %Identicon.Image{hex: hex}  
end
```

```
end
```

```
identicon.ex  image.ex
1  defmodule Identicon do
2    def main(input) do
3      input
4      |> hash_input
5      |> pick_color
6    end
7
8    def pick_color(%Identicon.Image{hex: [r, g, b | _tail]} = image) do
9      %Identicon.Image{image | color: {r, g, b}}
10   end
11
12   def hash_input(input) do
13     hex = :crypto.hash(:md5, input)
14     |> :binary.bin_to_list
15
16     %Identicon.Image{hex: hex}
17   end
18 end
```



[145, 46, 200,

3, 178, 206,

73, 228, 165,

65, 6, 141,

73, 90, 181,

112]

1 145	2 46	3 200	2 46	1 145
4 3	5 178	6 206	5 178	4 3
7 73	8 228	9 165	8 228	7 73
10 65	11 6	12 141	11 6	10 65
13 73	14 90	15 181	14 90	13 73

```
def main(input) do
```

```
  input
```

```
  |> hash_input
```

```
  |> pick_color
```

```
  |> build_grid
```

```
end
```

```
def build_grid(image) do
```

```
end
```

```
def pick_color(%Identicon.Image{hex: [r, g, b | _tail]} = image) do
```

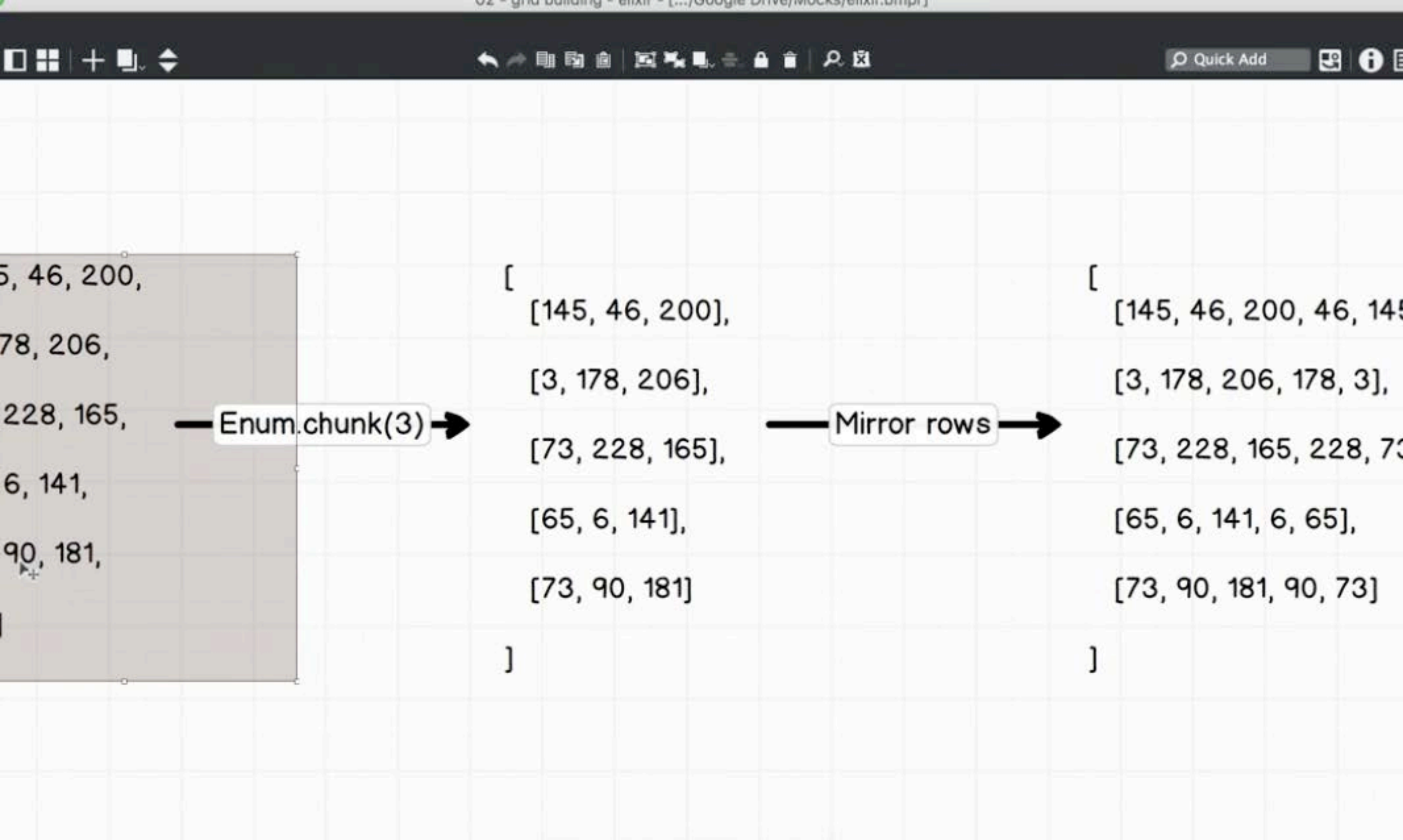
```
  %Identicon.Image{image | color: {r, g, b}}
```

```
end
```

```
def hash_input(input) do
```

```
  hex = :crypto.hash(:md5, input)
```

```
  |> :binary.bin_to_list
```





icon.ex x image.ex x

```
|> hash_input
|> pick_color
|> build_grid
end

def build_grid(%Identicon.Image{hex: hex} = image) do
  hex
  |> Enum.chunk(3)
end

def pick_color(%Identicon.Image{hex: [r, g, b | _tail]} = image) do
  %Identicon.Image{image | color: {r, g, b}}
end

def hash_input(input) do
  hex = :crypto.hash(:md5, input)
  |> :binary.bin_to_list
```

```
x(9)> recompile  
compiling 1 file (.ex)  
warning: variable image is unused  
lib/identicon.ex:9
```

<

```
x(10)> Identicon.main("asdf")  
[145, 46, 200], [3, 178, 206], [73, 228, 165], [65, 6, 141],  
[73, 90, 181]]  
x(11)> █
```

```
def build_grid(%Identicon.Image{hex: hex} = image) do
  hex
  |> Enum.chunk(3)
end
```

```
def mirror_row(row) do
  # [145, 46, 200]
  [first, second | _tail] = row

  # [145, 46, 200, 46, 145]
  row ++ [second, first]
end
```

I

```
def pick_color(%Identicon.Image{hex: [r, g, b | _tail]} = image) do
  %Identicon.Image{image | color: {r, g, b}}
end
```



```

def build_grid(%Identicon.Image{hex
  hex
  |> Enum.chunk(3)
end

def mirror_row(row) do
  # [145, 46, 200]
  [first, second | _tail] = row

  # [145, 46, 200, 46, 145]
  row ++ [second, first]
end

def pick_color(%Identicon.Image{hex
  %Identicon.Image{image | color:
end

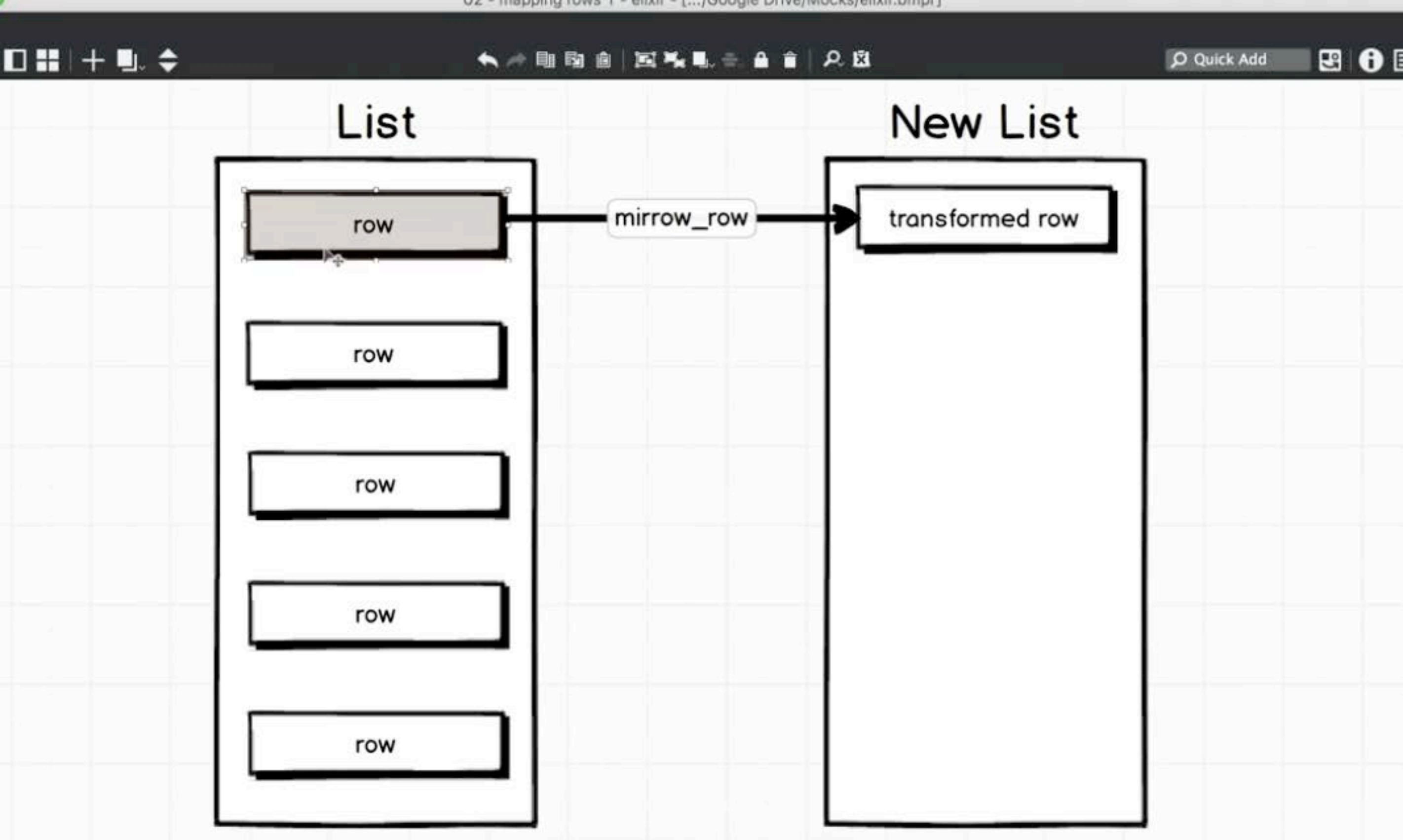
```

```

[[145, 46, 200], [3, 178, 200]
 [73, 228, 165], [65, 6, 141]
 [73, 90, 181]]
iex(12)> recompile
Compiling 1 file (.ex)
warning: variable image is un
ed
lib/identicon.ex:9

:ok
iex(13)> Identicon.mirror_row
145, 46, 200])
[145, 46, 200, 46, 145]
iex(14)>

```



```
|> build_grid  
end
```

```
def build_grid(%Identicon.Image{hex: hex} = image) do  
  hex  
  |> Enum.chunk(3)  
  |> Enum.map(&mirror_row/1)  
end
```

```
def mirror_row(row) do  
  [first, second | _tail] = row  
  
  row ++ [second, first]  
end
```

```
def pick_color(%Identicon.Image{hex: [r, g, b | _tail]} = image) do  
  %Identicon.Image{image | color: {r, g, b}}
```



```
.compile/5
```

```
(mix) lib/mix/task.ex:296: Mix.Task.run_task/3
```

```
(elixir) lib/enum.ex:1184: Enum."-map/2-lists^map/1-0-"/2
```

```
(elixir) lib/enum.ex:1184: Enum."-map/2-lists^map/1-0-"/2
```

```
x(14)> recompile
```

```
compiling 1 file (.ex)
```

```
warning: variable image is unused
```

```
lib/identicon.ex:9
```

```
<
```

```
x(15)> Identicon.main("asdf")
```

```
[145, 46, 200, 46, 145], [3, 178, 206, 178, 3],
```

```
[73, 228, 165, 228, 73], [65, 16, 141, 6, 65],
```

```
[73, 90, 181, 90, 73]]
```

```
x(16)>
```



```
identicon.ex — /Users/stephengrider/w
identicon.ex  x  image.ex  x
3      input
4      |> hash_input
5      |> pick_color
6      |> build_grid
7  end
8
9  def build_grid(%Identicon.Image{he
10     hex
11     |> Enum.chunk(3)
12     |> Enum.map(&mirror_row/1)
13 end
14
15 def mirror_row(row) do
16     [first, second | _tail] = row
17
18     row ++ [second, first]
19 end
20
```

```
1. lex -S mix (beam.smp)
x  lex (beam.smp)  1  x  ..rod/identicon (2...  2
iex(14)> recompile
Compiling 1 file (.ex)
warning: variable image is unus
ed
lib/identicon.ex:9

:ok
iex(15)> Identicon.main("asdf")

[[145, 46, 200, 46, 145], [3, 1
78, 206, 178, 3],
 [73, 228, 165, 228, 73], [65,
6, 141, 6, 65],
 [73, 90, 181, 90, 73]]
iex(16)>
```

```
identicon.ex — /Users/stephengrider/w
identicon.ex x image.ex x
4 |> hash_input
5 |> pick_color
6 |> build_grid
7 end
8
9 def build_grid(%Identicon.Image{hex
10 hex
11 |> Enum.chunk(3)
12 |> Enum.map(&mirror_row/1)
13 |> List.flatten
14 end
15
16 def mirror_row(row) do
17   [first, second | _tail] = row
18
19   row ++ [second, first]
20 end
21
```

```
1. lex -S mix (beam.smp)
x lex (beam.smp) 261 x ..rod/identicon (z... 262
iex(14)> recompile
Compiling 1 file (.ex)
warning: variable image is unus
ed
lib/identicon.ex:9

:ok
iex(15)> Identicon.main("asdf")

[[145, 46, 200, 46, 145], [3, 1
78, 206, 178, 3],
 [73, 228, 165, 228, 73], [65,
6, 141, 6, 65],
 [73, 90, 181, 90, 73]]
iex(16)>
```



```
identicon.ex — /Users/stephengrider/w
identicon.ex  x  image.ex  x
4      |> hash_input
5      |> pick_color
6      |> build_grid
7  end
8
9  def build_grid(%Identicon.Image{hex
10    hex
11    |> Enum.chunk(3)
12    |> Enum.map(&mirror_row/1)
13    |> List.flatten
14  end
15
16  def mirror_row(row) do
17    [first, second | _tail] = row
18
19    row ++ [second, first]
20  end
21
```

```
1. lex -S mix (beam.smp)
lex (beam.smp)  1  x  ..rod/identicon (z...  2
[73, 90, 181, 90, 73]]
iex(16)> recompile
Compiling 1 file (.ex)
warning: variable image is unused
lib/identicon.ex:9

:ok
iex(17)> Identicon.main("asdf")

[145, 46, 200, 46, 145, 3, 178,
206, 178, 3, 73, 228, 165,
228, 73, 65, 6, 141, 6, 65,
73, 90, 181, 90, 73]
iex(18)>
```



```
identicon.ex — /Users/stephengrider/w
identicon.ex  x image.ex  x
5      |> pick_color
6      |> build_grid
7  end
8
9  def build_grid(%Identicon.Image{hex
10    hex
11    |> Enum.chunk(3)
12    |> Enum.map(&mirror_row/1)
13    |> List.flatten
14    |> Enum.with_index
15  end
16
17  def mirror_row(row) do
18    [first, second | _tail] = row
19
20    row ++ [second, first]
21  end
22
```

```
1. lex -S mix (beam.smp)
lex (beam.smp)  1  x ..rod/identicon (z...  2
[73, 90, 181, 90, 73]]
iex(16)> recompile
Compiling 1 file (.ex)
warning: variable image is unused
lib/identicon.ex:9

:ok
iex(17)> Identicon.main("asdf")

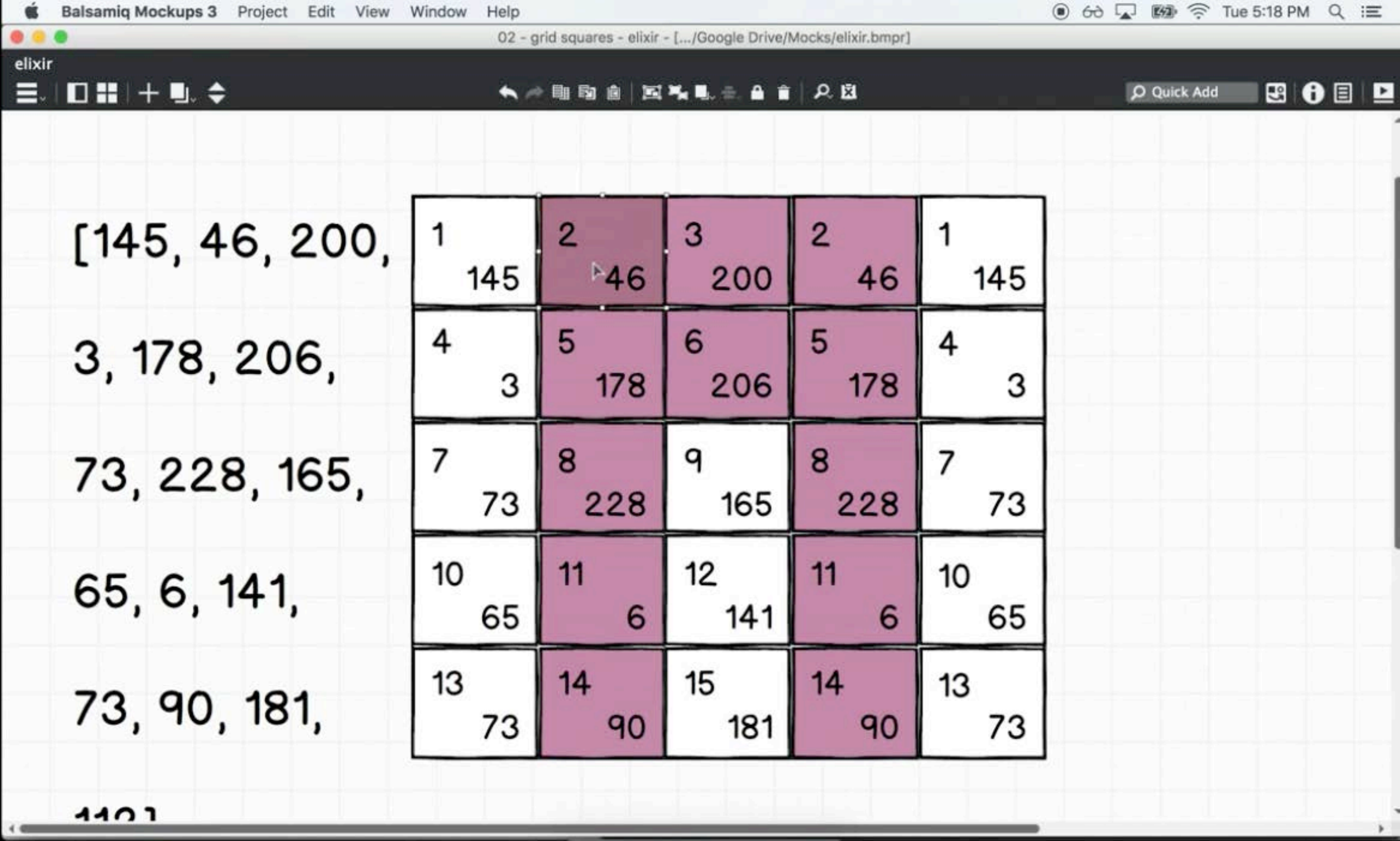
[145, 46, 200, 46, 145, 3, 178,
 206, 178, 3, 73, 228, 165,
 228, 73, 65, 6, 141, 6, 65,
 73, 90, 181, 90, 73]
iex(18)>
```



```
identicon.ex — /Users/
identicon.ex  x  image.ex  x
5      |> pick_color
6      |> build_grid
7  end
8
9  def build_grid(%Identicon.Image) do
10    hex
11    |> Enum.chunk(3)
12    |> Enum.map(&mirror_row/1)
13    |> List.flatten
14    |> Enum.with_index
15  end
16
17  def mirror_row(row) do
18    [first, second | _tail] = Enum.take(row, 2)
19
20    row ++ [second, first]
21  end
22
```

```
1. lex -S mix (beam.smp)
lex (beam.smp)  %1  ..rod/identicon (z...  %2
warning: variable image is unused
lib/identicon.ex:9

:ok
iex(19)> Identicon.main("asdf")
[{145, 0}, {46, 1}, {200, 2},
 {46, 3}, {145, 4}, {3, 5},
 {178, 6}, {206, 7}, {178, 8},
 {3, 9}, {73, 10}, {228, 11},
 {165, 12}, {228, 13},
 {73, 14}, {65, 15}, {6, 16},
 {141, 17}, {6, 18}, {65, 19},
 {73, 20}, {90, 21}, {181, 22},
 {90, 23}, {73, 24}]
iex(20)>
```





```
1 defmodule Identicon.Image do
2   defstruct hex: nil, color: nil, grid: nil
3 end
4
```



```
identicon.ex  x  image.ex  x
6      |> build_grid
7  end
8
9  def build_grid(%Identicon.Image{hex: hex} = image) do
10    grid =
11      hex
12      |> Enum.chunk(3)
13      |> Enum.map(&mirror_row/1)
14      |> List.flatten
15      |> Enum.with_index
16
17    %Identicon.Image{image | grid: grid}
18  end
19
20  def mirror_row(row) do
21    [first, second | _tail] = row
22
23    row ++ [second, first]
```

```
identicon.ex — /Users/
identicon.ex x image.ex x
6     |> build_grid
7 end
8
9 def build_grid(%Identicon.Image)
10     grid =
11         hex
12         |> Enum.chunk(3)
13         |> Enum.map(&mirror_row)
14         |> List.flatten
15         |> Enum.with_index
16
17     %Identicon.Image{image |> Enum.map(&mirror_row)}
18 end
19
20 def mirror_row(row) do
21     [first, second | _tail] = Enum.chunk(row, 2)
22
23     row ++ [second, first]
```

```
1. iex -S mix (beam.smp)
lex (beam.smp) 1
...rod/identicon [2... 2
:ok
iex(19)> Identicon.main("asdf")
[{145, 0}, {46, 1}, {200, 2},
 {46, 3}, {145, 4}, {3, 5},
 {178, 6}, {206, 7}, {178, 8},
 {3, 9}, {73, 10}, {228, 11},
 {165, 12}, {228, 13},
 {73, 14}, {65, 15}, {6, 16},
 {141, 17}, {6, 18}, {65, 19},
 {73, 20}, {90, 21}, {181, 22},
 {90, 23}, {73, 24}]
iex(20)> recompile
Compiling 2 files (.ex)
:ok
iex(21)> recompile
```



```
identicon.ex — /Users/
identicon.ex x image.ex x
6 |> build_grid
7 end
8
9 def build_grid(%Identicon.Image)
10   grid =
11     hex
12     |> Enum.chunk(3)
13     |> Enum.map(&mirror_row)
14     |> List.flatten
15     |> Enum.with_index
16
17   %Identicon.Image{image | }
18 end
19
20 def mirror_row(row) do
21   [first, second | _tail] =
22
23   row ++ [second, first]
```

```
1. lex -S mix (beam.smp)
x lex (beam.smp) 1 x ..rod/identicon (z... 2
200},
grid: [{145, 0}, {46, 1}, {200, 2},
{46, 3}, {145, 4}, {3, 5},
{178, 6}, {206, 7}, {178, 8},
{3, 9}, {73, 10}, {228, 11},
{165, 12}, {228, 13}, {73, 14},
{65, 15}, {6, 16}, {141, 17},
{6, 18}, {65, 19}, {73, 20},
{90, 21}, {181, 22}, {90, 23},
{73, 24}],
hex: [145, 46, 200, 3, 178, 206,
73, 228, 165, 65, 6, 141, 73, 90,
181, 112]}
iex(22)>
```

73, 90, 181,

1 145	2 46	3 200	2 46	1 145
4 3	5 178	6 206	5 178	4 3
7 73	8 228	9 165	8 228	7 73
10 65	11 6	12 141	11 6	10 65
13 73	14 90	15 181	14 90	13 73



elixir

[145, 46, 200,

3, 178, 206,

73, 228, 165,

65, 6, 141,

73, 90, 181,

1401

1 145	2
4 3	5
7 73	8 2
10 65	11
13 73	14

02 - grid squan

1. lex -S mix (beam.smp)

lex (beam.smp) 1

rod/identicon (zsh) 2

200},  
grid: [{145, 0}, {46, 1}, {200, 2},  
{46, 3}, {145, 4}, {3, 5},  
{178, 6}, {206, 7}, {178, 8},  
{3, 9}, {73, 10}, {228, 11},  
{165, 12}, {228, 13}, {73, 14},  
{65, 15}, {6, 16}, {141, 17},  
{6, 18}, {65, 19}, {73, 20},  
{90, 21}, {181, 22}, {90, 23},  
{73, 24}],  
hex: [145, 46, 200, 3, 178, 206,  
73, 228, 165, 65, 6, 141, 73, 90,  
181, 112]}  
iex(22)>

```
identicon.ex  image.ex x
1  defmodule Identicon do
2    def main(input) do
3      input
4      |> hash_input
5      |> pick_color
6      |> build_grid
7      |> |
8    end
9
10   def build_grid(%Identicon.Image{hex: hex} = image) do
11     grid =
12       hex
13       |> Enum.chunk(3)
14       |> Enum.map(&mirror_row/1)
15       |> List.flatten
16       |> Enum.with_index
17
18     %Identicon.Image{image | grid: grid}
```



Elixir  
v1.3.4



search

PAGES

MODULES

EXCEPTIONS

PROTOCOLS

Access

Agent

Application

Atom

Base

Behaviour

Bitwise

Calendar

Calendar.ISO

Code

Date

DateTime

Dict

Enum

## Functions

**append(tuple, value)**

*Inserts an element at the end of a tuple*

**delete\_at(tuple, index)**

*Removes an element from a tuple*

**duplicate(data, size)**

*Creates a new tuple*

**insert\_at(tuple, index, value)**

*Inserts an element into a tuple*

**to\_list(tuple)**

*Converts a tuple to a list*

## Functions

**append(tuple, value)**

`append(tuple, term) :: tuple`



Inserts an element at the end of a tuple.

Returns a new tuple with the element appended at the end, and contains the elements in `tuple` followed by `value` as the last element.



Chrome

File

Edit

View

History

Bookmarks

People

Window

Help

Enum - Elixir v1.3.4

x

Stephen

elixir-lang.org/docs/stable/elixir/Enum.html#filter/2

🔍

☆

🔔


📄

🌐

⋮

Elixir

v1.3.4



🔍

search

PAGES

MODULES

EXCEPTIONS

PROTOCOLS

Dict

Enum

Top

Summary

+ Types

+ Functions

Exception

File

File.Stat

File.Stream

xx (Enum.OutOfBoundsError) out of bounds error

filter enumerable, fun

</>

filter(t, (element -> as\_boolean(term))) :: list

Filters the enumerable, i.e. returns only those elements for which fun returns a truthy value.

See also reject/2.

Examples

iex> Enum.filter([1, 2, 3], fn(x) -> rem(x, 2) == 0 end)

[2]

filter\_map enumerable, filter, mapper

</>

filter\_map(t, (element -> as\_boolean(term)), (element -> element)) :: list

Filters the enumerable and maps its elements in one pass.

AtomFile Edit View Selection Find Packages Window Help

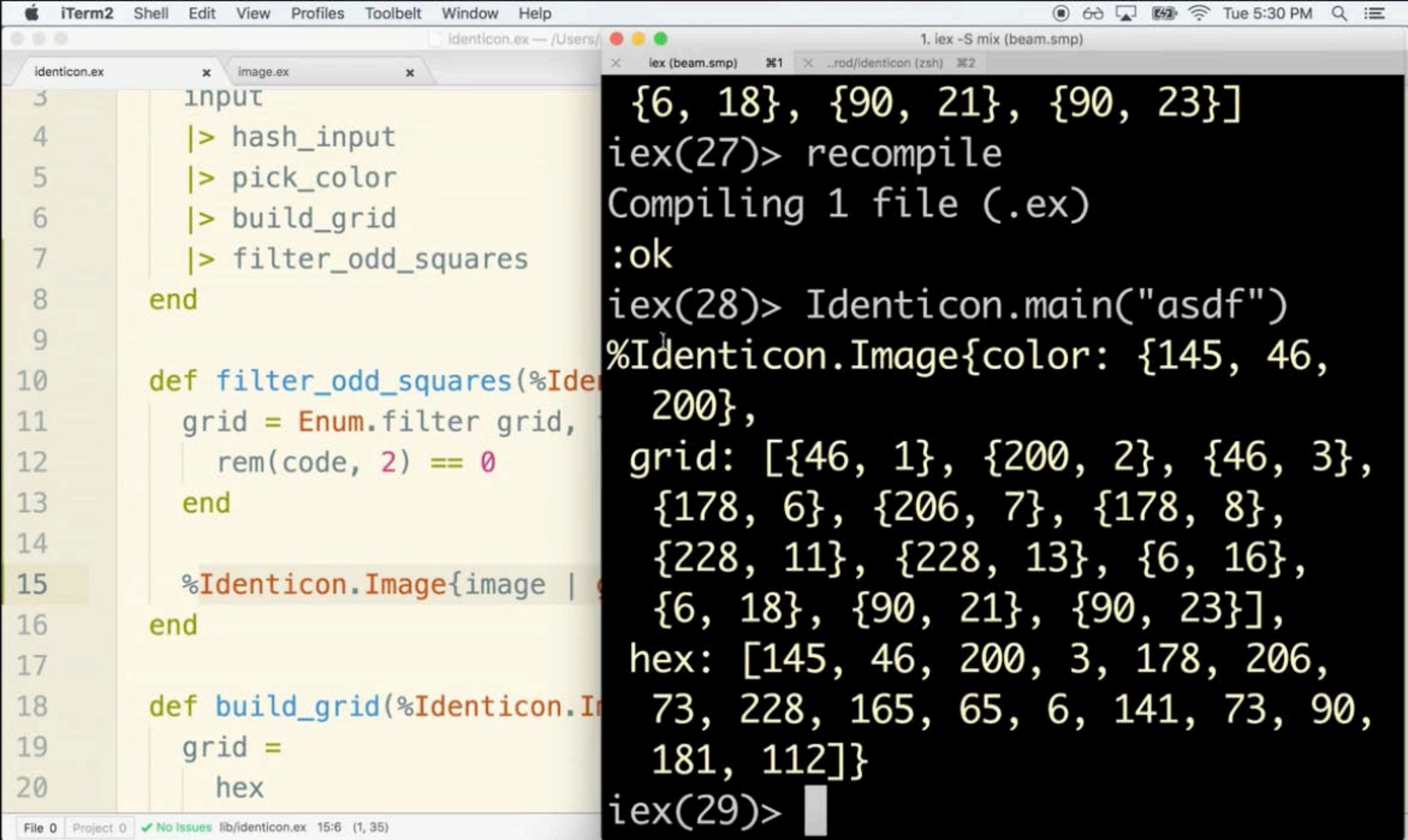
identicon.ex — /Users/stephengrider/workspace/ElixirWorkspace/prod/identicon

identicon.eximage.ex

```
3 input
4   |> hash_input
5   |> pick_color
6   |> build_grid
7   |> filter_odd_squares
8 end
9
10 def filter_odd_squares(%Identicon.Image{grid: grid} = image) do
11   grid = Enum.filter grid, fn({code, _index}) ->
12     rem(code, 2) == 0
13   end
14
15   %Identicon.Image{image | grid: grid}
16 end
17
18 def build_grid(%Identicon.Image{hex: hex} = image) do
19   grid =
20     hex
```

File 0Project 0No Issues lib/identicon.ex\* 11:12LF UTF-8 Elixir 062-grid-structure +7 7 updates





```
input
|> hash_input
|> pick_color
|> build_grid
|> filter_odd_squares
```

```
end
```

```
def filter_odd_squares(%Identicon.Image{color: {color, grid}})
  grid = Enum.filter grid,
    rem(code, 2) == 0
end
```

```
%Identicon.Image{image |
end
```

```
def build_grid(%Identicon.Image{color: {color, grid}})
  grid =
  hex
```

```
{6, 18}, {90, 21}, {90, 23}]
iex(27)> recompile
Compiling 1 file (.ex)
:ok
iex(28)> Identicon.main("asdf")
%Identicon.Image{color: {145, 46,
200},
grid: [{46, 1}, {200, 2}, {46, 3},
{178, 6}, {206, 7}, {178, 8},
{228, 11}, {228, 13}, {6, 16},
{6, 18}, {90, 21}, {90, 23}],
hex: [145, 46, 200, 3, 178, 206,
73, 228, 165, 65, 6, 141, 73, 90,
181, 112]}
iex(29)>
```