```
  |> 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
```

```
 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")
ticon.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
```

```
    |> hash_input
    |> pick_color
  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

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

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

 181, 112]}
)> recompile
ling 1 file (.ex)

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

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

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

```elixir
defmodule Identicon do
  def main(input) do
    input
    |> hash_input
    |> pick_color
  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

    %Identicon.Image{hex: hex}
  end
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

```elixir
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
```

```
5, 46, 200,
78, 206,
228, 165,
6, 141,
90, 181,
]
```

Enum.chunk(3) →

```
[
  [145, 46, 200],
  [3, 178, 206],
  [73, 228, 165],
  [65, 6, 141],
  [73, 90, 181]
]
```

Mirror rows →

```
[
  [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]
]
```

```
  |> 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
```

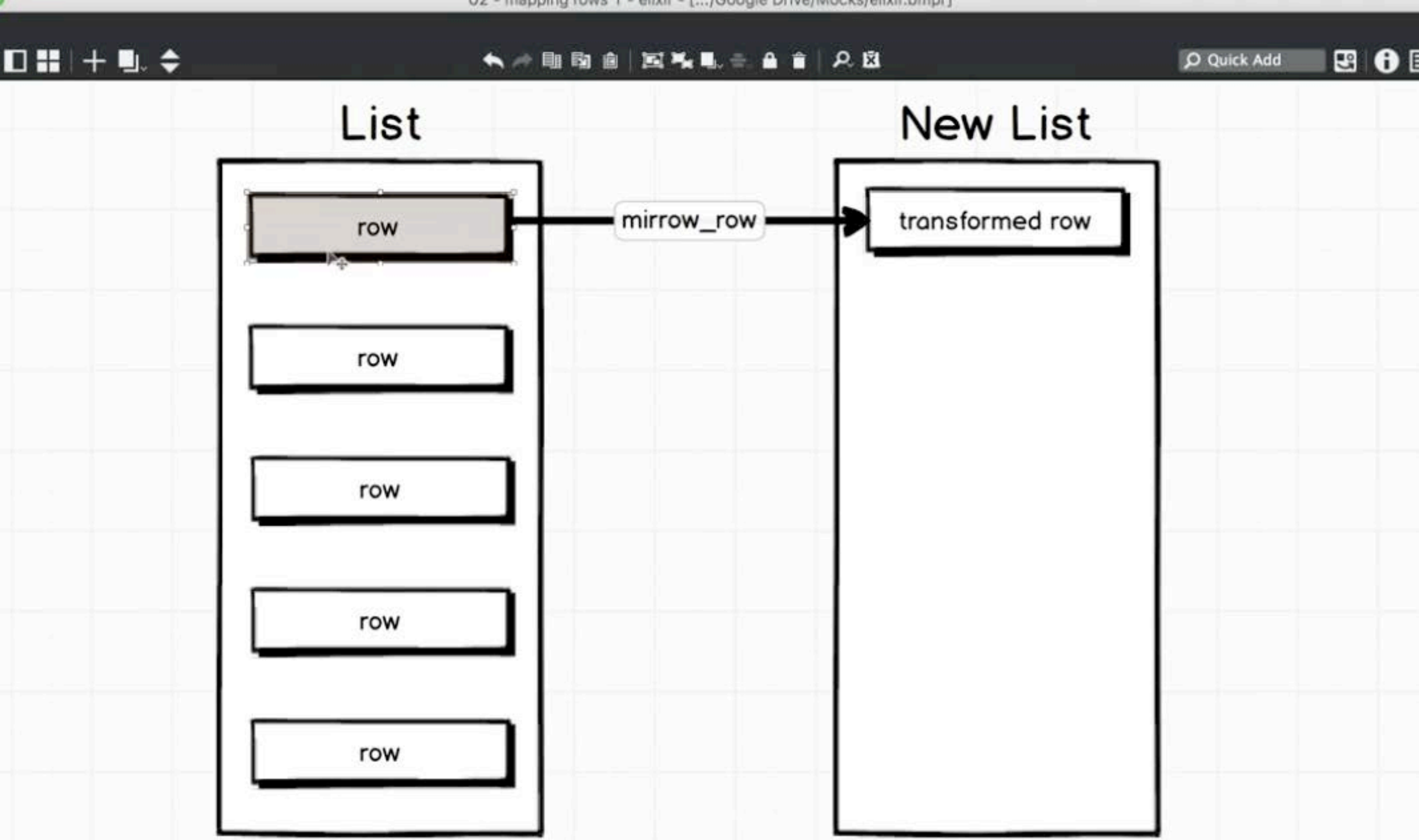
```
x(9)> recompile
mpiling 1 file (.ex)
rning: variable image is unused
lib/identicon.ex:9

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

```elixir
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

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

```elixir
def build_grid(%Identicon.Image{he
  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{he
  %Identicon.Image{image | color:
end
```



```
[[145, 46, 200], [3, 178, 206
 [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
mpiling 1 file (.ex)
rning: variable image is unused
lib/identicon.ex:9

k
x(15)> 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]]
```

identicon.ex — /Users/stephengrider/wi

identicon.ex × image.ex ×

```
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. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 ..rod/identicon (z... ⌘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 | image.ex

```
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      |> 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. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 | ..rod/identicon (z... ⌘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 | image.ex

```
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      |> 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. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 ..rod/identicon (z... ⌘2

```
 [73, 90, 181, 90, 73]]
iex(16)> recompile
Compiling 1 file (.ex)
warning: variable image is unus
ed
  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 | image.ex

```
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    |> 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
```


iTerm

1. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 ..rod/identicon (z... ⌘2

```
 [73, 90, 181, 90, 73]]
iex(16)> recompile
Compiling 1 file (.ex)
warning: variable image is unus
ed
  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 × image.ex ×

```
6     |> build_grid
7   end
8
9   def build_grid(%Identicon.I
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] =
19
20    row ++ [second, first]
21  end
22
```

File 0 Project 0 ✔ No issues lib/identicon.ex 14:23

1. iex -S mix (beam.smp)

iex (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)>
```

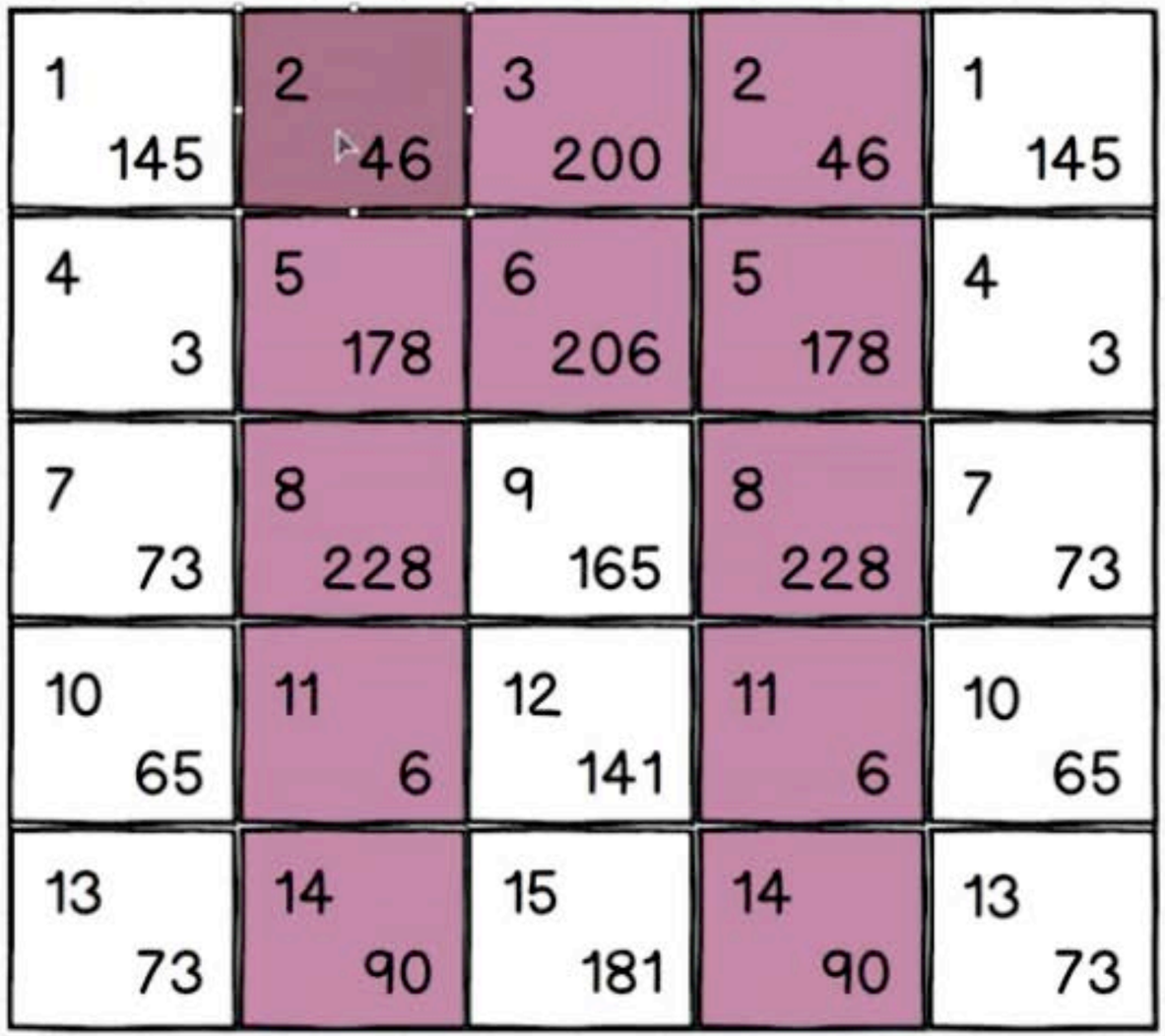
[145, 46, 200,

3, 178, 206,

73, 228, 165,

65, 6, 141,

73, 90, 181,



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

```
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 × image.ex ×

```
6       |> build_grid
7     end
8
9     def build_grid(%Identicon.I
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]
```

File 0 Project 0 ✔ No Issues lib/identicon.ex 16:1

1. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 ..rod/identicon (z... ⌘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
```

```
iTerm2 Shell Edit View Profiles Toolbelt Window Help                Tue 5:20 PM
identicon.ex — /Users/                    1. iex -S mix (beam.smp)
identicon.ex ×  image.ex ×                iex (beam.smp) ⌘1  ..rod/identicon (z... ⌘2
```

```
6       |> build_grid
7     end
8
9     def build_grid(%Identicon.I
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]
```

```
File 0  Project 0  ✔ No Issues  lib/identicon.ex  16:1
```

```
 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)>
```

[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

02 - grid squar

elixir

[145, 46, 200,

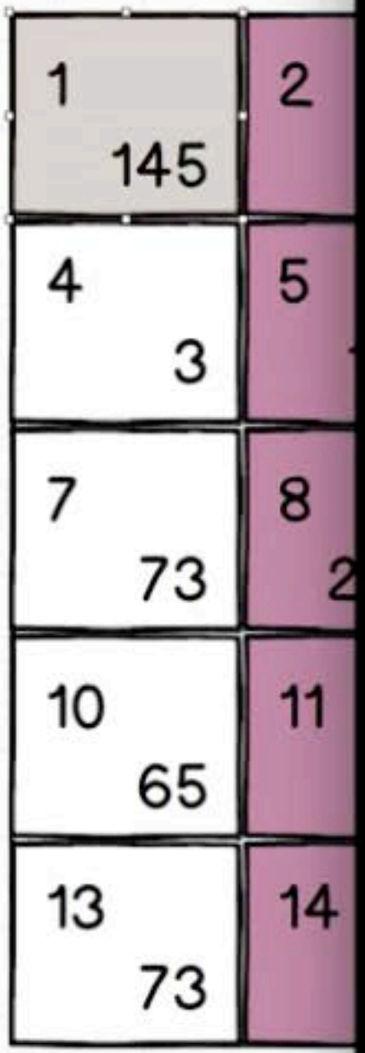
3, 178, 206,

73, 228, 165,

65, 6, 141,

73, 90, 181,

112]



1. iex -S mix (beam.smp)

iex (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)>
```

```elixir
defmodule Identicon do
  def main(input) do
    input
    |> hash_input
    |> pick_color
    |> build_grid
    |>
  end

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

    %Identicon.Image{image | grid: grid}
```

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.

** (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.

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

  def filter_odd_squares(%Identicon.Image{grid: grid} = image) do
    grid = Enum.filter grid, fn({code, _index}) ->
      rem(code, 2) == 0
    end

    %Identicon.Image{image | grid: grid}
  end

  def build_grid(%Identicon.Image{hex: hex} = image) do
    grid =
      hex
```

identicon.ex — /Users/

identicon.ex × image.ex ×

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


1. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 ..rod/identicon (zsh) ⌘2

```
 {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)>
```

```elixir
defmodule Identicon do
  def main(input) do
    input
    |> hash_input
    |> pick_color
    |> build_grid
    |> filter_odd_squares
  end

  def filter_odd_squares(%Identicon.Image{grid: grid} = image) do
    grid = Enum.filter grid, fn({code, _index}) ->
      rem(code, 2) == 0
    end

    %Identicon.Image{image | grid: grid}
  end

  def build_grid(%Identicon.Image{hex: hex} = image) do
```

egd

MODULE

egd

MODULE SUMMARY

egd - erlang graphical drawer.

DESCRIPTION

egd - erlang graphical drawer

DATA TYPES

color()

egd_image()

font()

point() = {integer(), integer()}

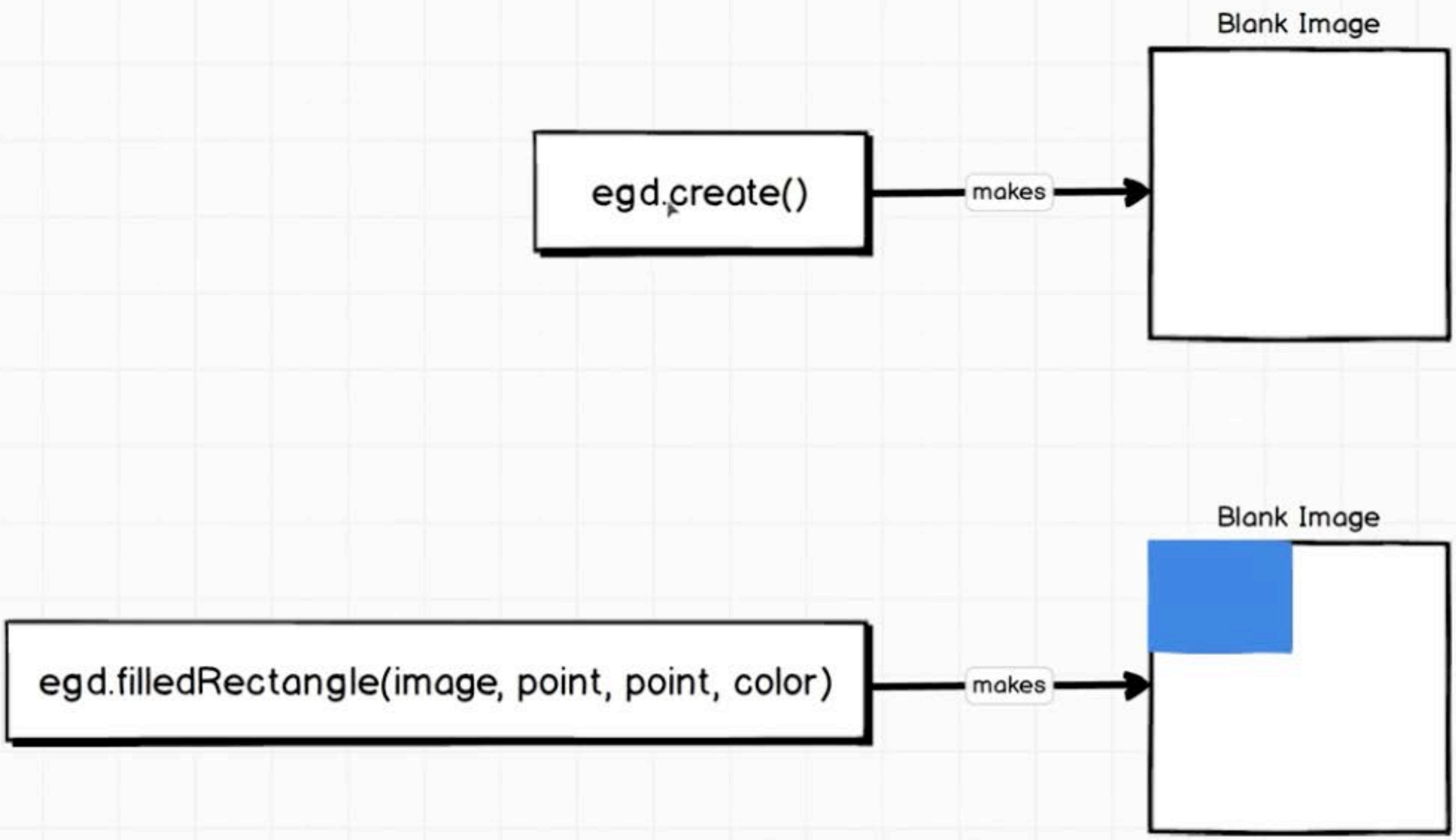
render_option() = {render_engine, opaque} | {render_engine, alpha}

EXPORTS

```
color(Color::Value | Name) -> color()
```

Types:

```
Value = {byte(), byte(), byte()} | {byte(), byte(), byte(), byte()}
```

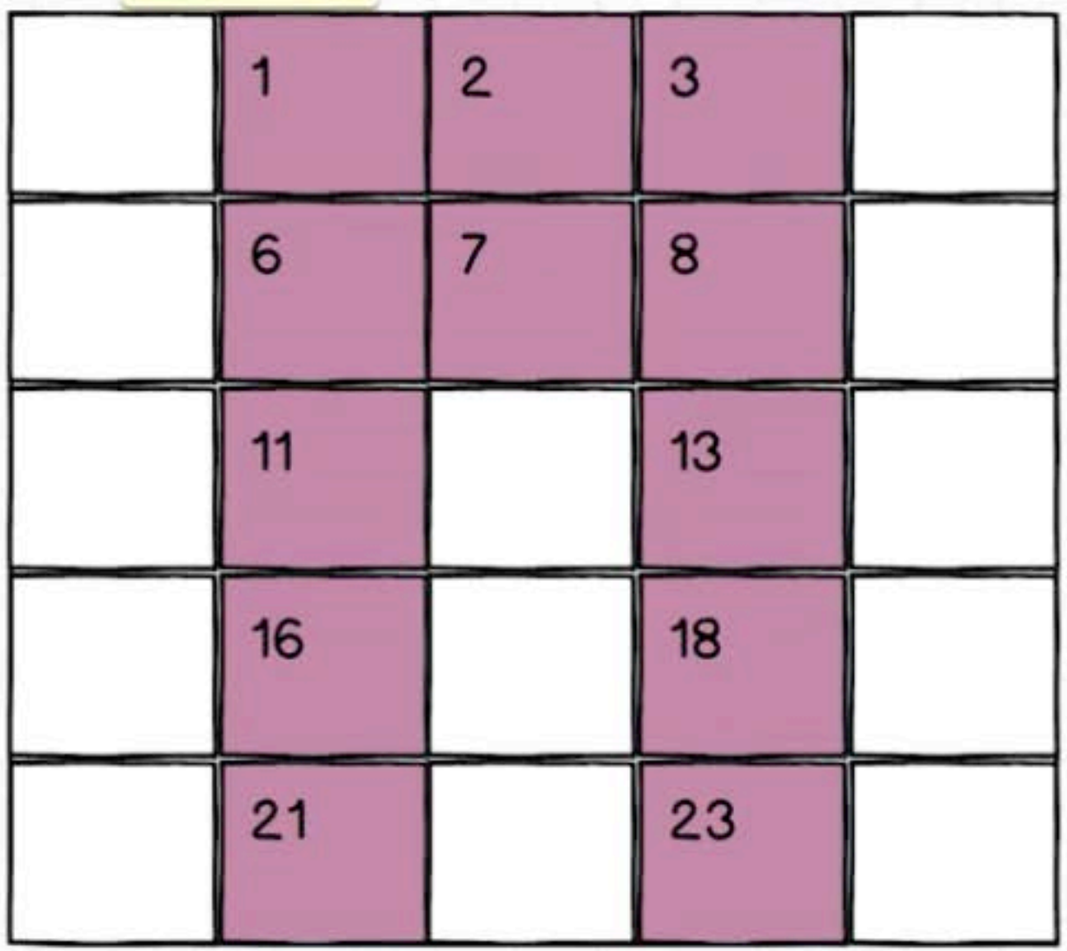


```
1  defmodule Identicon do
2    def main(input) do
3      input
4      |> hash_input
5      |> pick_color
6      |> build_grid
7      |> filter_odd_squares
8      |> build_pixel_map
9    end
10
11   def build_pixel_map() do
12
13   end
14
15   def filter_odd_squares(%Identicon.Image{grid: grid} =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

rem(index, 5) * 50 = 100

double-click to edit

div(index, 5) * 50 = 50

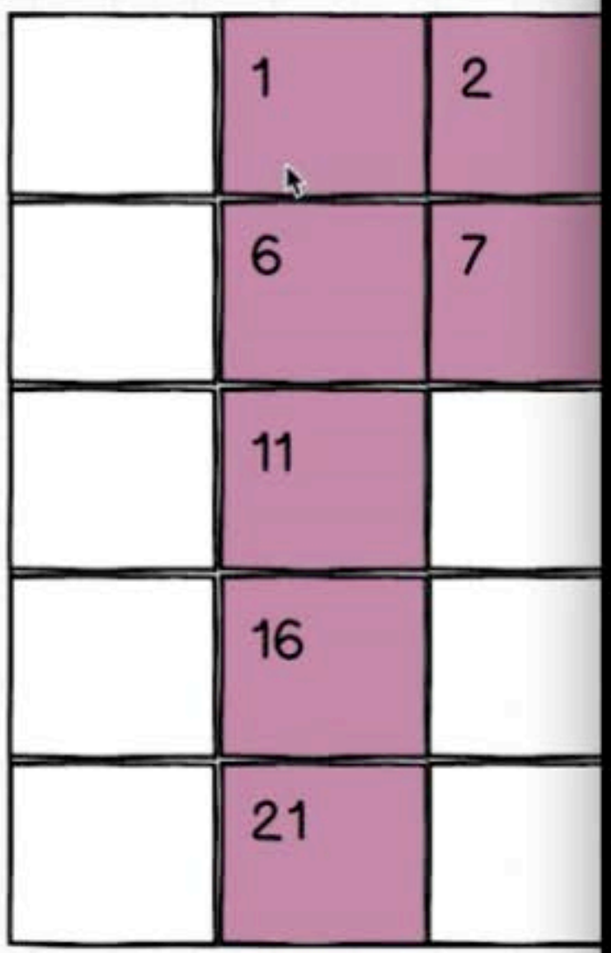


02 - pixel map points - elixir - [...]

elixir

rem(index, 5) * 50 = 100

div(index, 5) * 50 = 50



1. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 ..rod/identicon (z... ⌘2

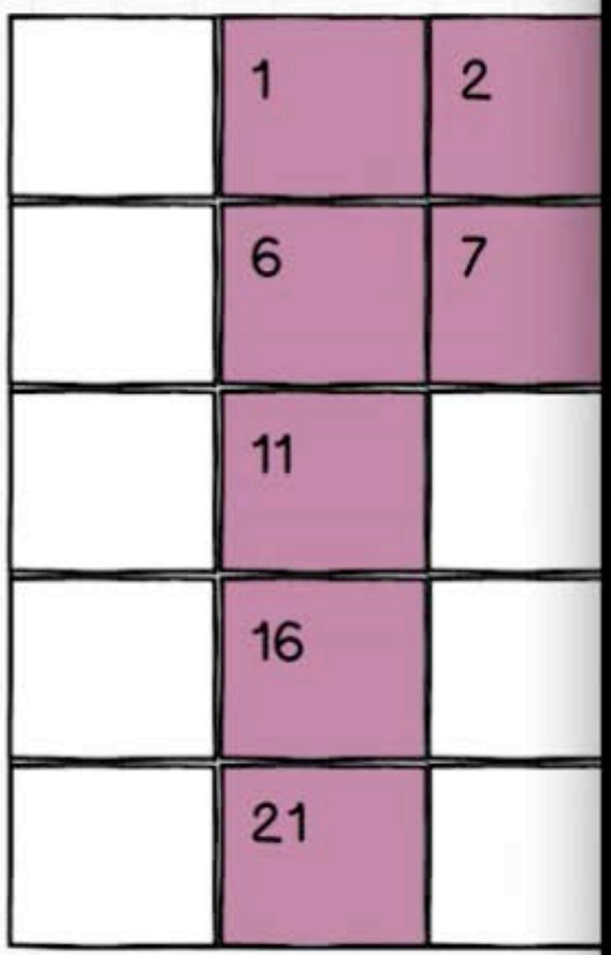
```
iex(35)> Identicon.main("asdf")
%Identicon.Image{color: {145, 46
  200},
 grid: [{46, 1}, {200, 2}, {46,
  {178, 6}, {206, 7}, {178, 8},
  {228, 11}, {228, 13}, {6, 16},
  {6, 18}, {90, 21}, {90, 23}],
 hex: [145, 46, 200, 3, 178, 206
  73, 228, 165, 65, 6, 141, 73,
  181, 112]}
iex(36)> rem(1, 5) * 50
50
iex(37)> div(1, 5) * 50
0
iex(38)>
```

02 - pixel map points - elixir - [...]

elixir

rem(index, 5) * 50 = 100

div(index, 5) * 50 = 50



1. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 ..rod/identicon (z... ⌘2

```
  {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,
  181, 112]}
iex(36)> rem(1, 5) * 50
50
iex(37)> div(1, 5) * 50
0
iex(38)> rem(2, 5) * 50
100
iex(39)> div(2, 5) * 50
0
iex(40)>
```

identicon.ex — /Users/stephengrider/w

identicon.ex | image.ex

```
7       |> filter_odd_squares
8       |> build_pixel_map
9     end
10
11    def build_pixel_map(%Identicon.Ima
12      Enum.map grid, fn({_code, index}
13        horizontal = rem(index, 5) * 5
14        vertical = div(index, 5) * 50
15
16        top_left = {horizontal, vertic
17        bottom_right = {horizontal + 5
18
19        {top_left, bottom_right}
20      end
21    end
22
23    def filter_odd_squares(%Identicon.
```

File 0 Project 0 ✔ No Issues lib/identicon.ex 19:15

1. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 ..rod/identicon (z... ⌘2

```
2
iex(35)> Identicon.main("asdf")
%Identicon.Image{color: {145, 46
  200},
 grid: [{46, 1}, {200, 2}, {46,
  {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,
  181, 112]}
iex(36)> rem(1, 5) * 50
50
iex(37)> div(1, 5) * 50
0
```

identicon.ex | image.ex

```
7       |> filter_odd_squa
8       |> build_pixel_map
9     end
10
11    def build_pixel_map(
12      Enum.map grid, fn(
13        horizontal = rem(
14        vertical = div(i
15
16        top_left = {horiz
17        bottom_right = {h
18
19        {top_left, bottom
20      end
21    end
22
23    def filter_odd_square
```

File 0 Project 0 ✔ No Issues lib/identicon.ex 19:15

1. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 | ..rod/identicon (z... ⌘2

```
  lib/identicon.ex:11


:ok
iex(41)> Identicon.main("asdf")
[{{50, 0}, {100, 50}}, {{100, 0}, {150, 50
 {{150, 0}, {200, 50}}, {{50, 50}, {100, 1
 {{100, 50}, {150, 100}}, {{150, 50}, {200

 {{50, 100}, {100, 150}},
 {{150, 100}, {200, 150}},
 {{50, 150}, {100, 200}},
 {{150, 150}, {200, 200}},
 {{50, 200}, {100, 250}},
 {{150, 200}, {200, 250}}]
iex(42)>
```

```
8       |> build_pixel_map
9     end
10
11    def build_pixel_map(%Identicon.Image{grid: grid} = image) do
12      pixel_map = Enum.map grid, fn({_code, index}) ->
13        horizontal = rem(index, 5) * 50
14        vertical = div(index, 5) * 50
15
16        top_left = {horizontal, vertical}
17        bottom_right = {horizontal + 50, vertical + 50}
18
19        {top_left, bottom_right}
20      end
21
22      %Identicon.Image{image | pixel_map: pixel_map}
23    end
24
25    def filter_odd_squares(%Identicon.Image{grid: grid} = image) do
```

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

identicon.ex | image.ex

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

identicon.ex × image.ex ×

```
11    def build_pixel_ma
12      pixel_map = Enum
13        horizontal = r
14        vertical = div
15
16        top_left = {ho
17        bottom_right =
18
19        {top_left, bot
20      end
21
22      %Identicon.Image
23    end
24
25    def filter_odd_squ
26      grid = Enum.filt
27        rem(code, 2) =
28      end
```

File 0 Project 0 ✔ No Issues lib/identicon.ex 22:24

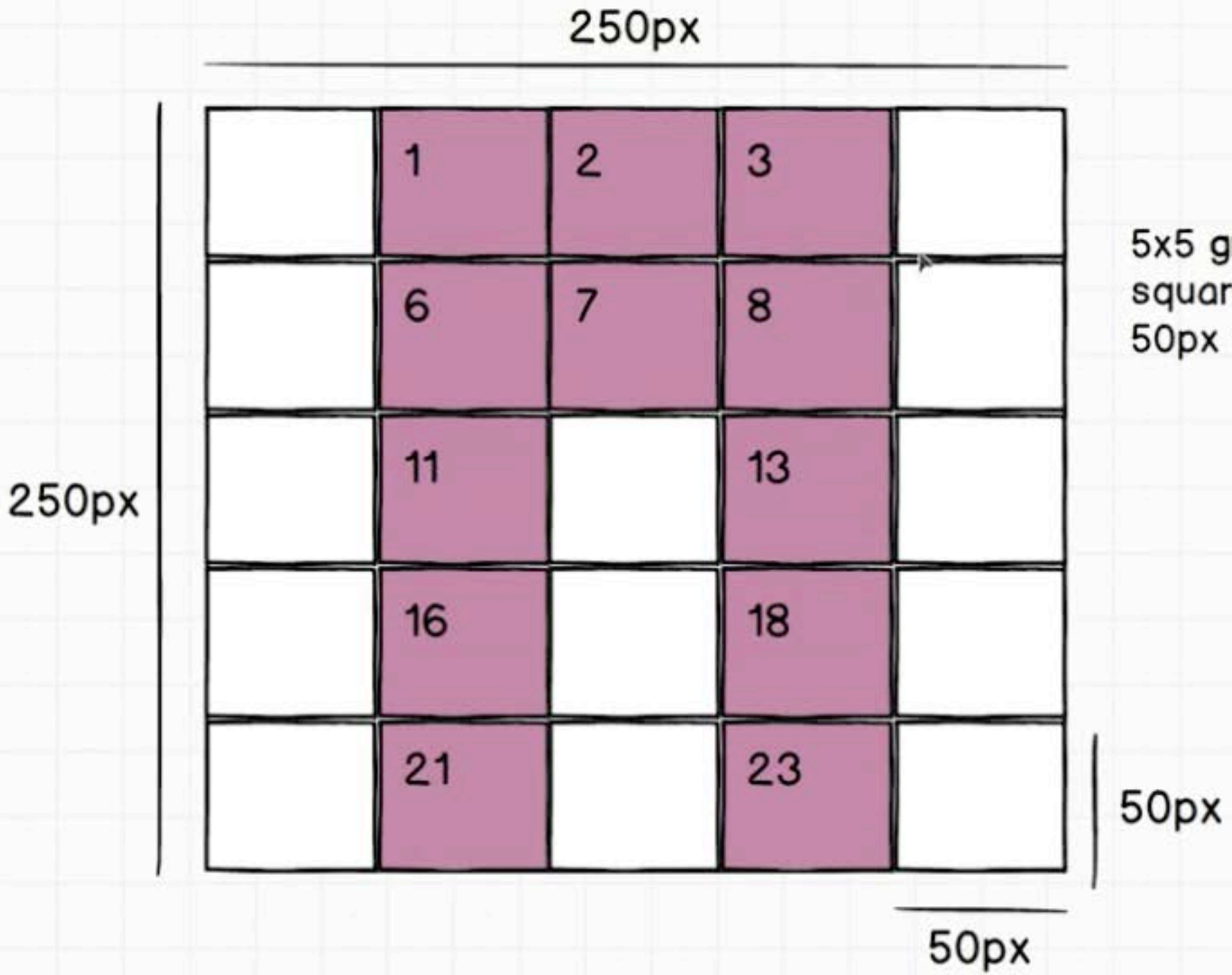
1. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 ...rod/identicon (z... ⌘2

```
hex: [145, 46, 200, 3, 178, 206, 73, 228,
 165, 65, 6, 141, 73, 90, 181, 112],
pixel_map: [{{50, 0}, {100, 50}},
 {{100, 0}, {150, 50}},
 {{150, 0}, {200, 50}},
 {{50, 50}, {100, 100}},
 {{100, 50}, {150, 100}},
 {{150, 50}, {200, 100}},
 {{50, 100}, {100, 150}},
 {{150, 100}, {200, 150}},
 {{50, 150}, {100, 200}},
 {{150, 150}, {200, 200}},
 {{50, 200}, {100, 250}},
 {{150, 200}, {200, 250}}]}
iex(44)>
```

```
12  def draw_image(%Identicon.Image{color: color, pixel_map: pixel_map}) do
13
14  end
15
16  def build_pixel_map(%Identicon.Image{grid: grid} = image) do
17    pixel_map = Enum.map grid, fn({_code, index}) ->
18      horizontal = rem(index, 5) * 50
19      vertical = div(index, 5) * 50
20
21      top_left = {horizontal, vertical}
22      bottom_right = {horizontal + 50, vertical + 50}
23
24      {top_left, bottom_right}
25    end
26
27    %Identicon.Image{image | pixel_map: pixel_map}
28  end
29
```

```
    |> pick_color
    |> build_grid
    |> filter_odd_squares
    |> build_pixel_map
    |> draw_image
  end

  def draw_image(%Identicon.Image{color: color, pixel_map: pixel_map}) do

  end

  def build_pixel_map(%Identicon.Image{grid: grid} = image) do
    pixel_map = Enum.map grid, fn({_code, index}) ->
      horizontal = rem(index, 5) * 50
      vertical = div(index, 5) * 50

      top_left = {horizontal, vertical}
      bottom_right = {horizontal + 50, vertical + 50}
```



5x5 grid of
squares, each
50px tall and wide

Creates a filled rectangle object.

```
line(Image::egd_image(), P1::point(), P2::point(), Color::color()) -
> ok
```

Creates a line object from P1 to P2 in the image.

```
rectangle(Image::egd_image(), P1::point(), P2::point(),
Color::color()) -> ok
```

Creates a rectangle object.

```
render(Image::egd_image()) -> binary()
```

Equivalent to `render(Image, png, [{render_engine, opaque}])`.

```
render(Image::egd_image(), Type::png | raw_bitmap) -> binary()
```

Equivalent to `render(Image, Type, [{render_engine, opaque}])`.

```
render(Image::egd_image(), Type::png | raw_bitmap, Options::
[render_option()]) -> binary()
```

Renders a binary from the primitives specified by egd_image(). The binary can either be a raw bitmap with rgb tripplets or a binary in png format.

```
save(Binary::binary(), Filename::string()) -> ok
```

Saves the binary to file.

```
8     |> build_pixel_map
9     |> draw_image
10  end
11
12  def draw_image(%Identicon.Image{color: color, pixel_map: pixel_map}) do
13    image = :egd.create(250, 250)
14    fill = :egd.color(color)
15
16    Enum.each pixel_map, fn({start, stop}) ->
17      :egd.filledRectangle(image, start, stop, fill)
18    end
19
20    :egd.render(image)
21  end
22
23  def build_pixel_map(%Identicon.Image{grid: grid} = image) do
24    pixel_map = Enum.map grid, fn({_code, index}) ->
```

```
    |> pick_color
    |> build_grid
    |> filter_odd_squares
    |> build_pixel_map
    |> draw_image
    |> save_image(input)
  end

  def save_image(image, input) do
    File.write("#{input}.png", image)
  end

  def draw_image(%Identicon.Image{color: color, pixel_map: pixel_map}) do
    image = :egd.create(250, 250)
    fill = :egd.color(color)

    Enum.each pixel_map, fn({start, stop}) ->
      :egd.filledRectangle(image, start, stop, fill)
```

identicon.ex × image.ex

```
6       |> buil
7       |> filt
8       |> buil
9       |> draw
10      |> save
11    end
12
13    def save_
14      File.wr
15    end
16
17    def draw_
18      image =
19      fill =
20
21      Enum.ea
22        :egd.
```

File 0 Project 0 ✔ No Issues lib/identicon.

1. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 ..rod/identicon (z... ⌘2

```
ernel.ParallelCompiler.spawn_compilers/3
        (mix) lib/mix/compilers/elixir.ex:157: Mix.
Compilers.Elixir.compile_manifest/7
        (mix) lib/mix/compilers/elixir.ex:78: Mix.C
ompilers.Elixir.compile/5
        (mix) lib/mix/task.ex:296: Mix.Task.run_tas
k/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
iex(44)> recompile
Compiling 1 file (.ex)
:ok
iex(45)>
```

identicon.ex × image.ex

```
6       |> buil
7       |> filt
8       |> buil
9       |> draw
10      |> save
11    end
12
13    def save_
14      File.wr
15    end
16
17    def draw_
18      image =
19      fill =
20
21      Enum.ea
22        :egd.
```

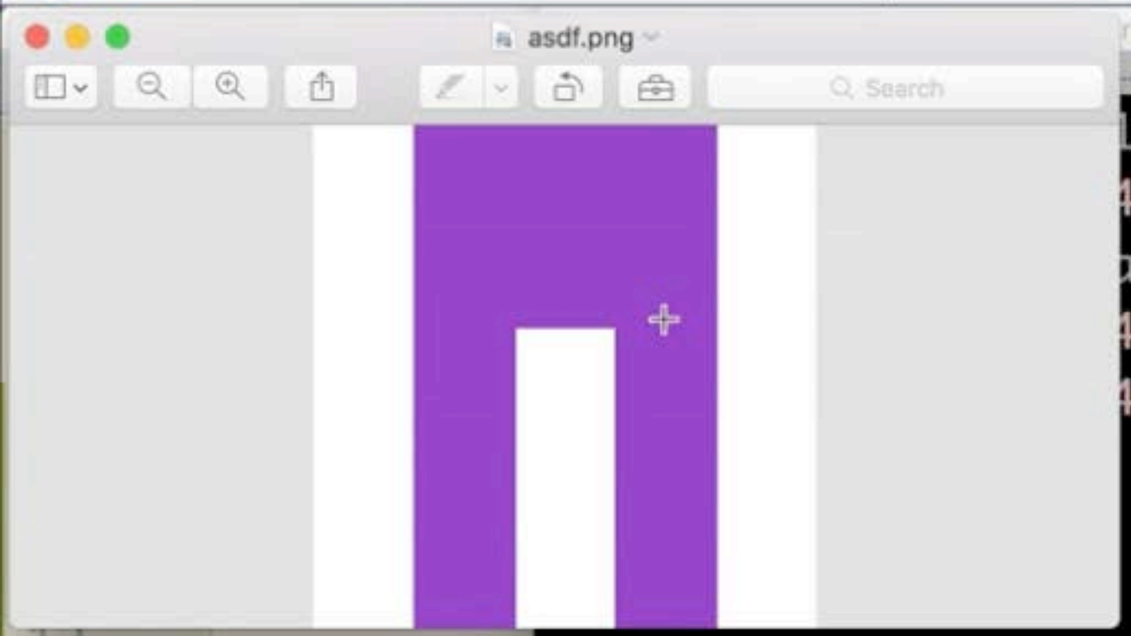

1. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 ..rod/identicon (z... ⌘2

```
Compilers.Elixir.compile_manifest/7
        (mix) lib/mix/compilers/elixir.ex:78: Mix.C
ompilers.Elixir.compile/5
        (mix) lib/mix/task.ex:296: Mix.Task.run_tas
k/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
iex(44)> recompile
Compiling 1 file (.ex)
:ok
iex(45)> Identicon.main("asdf")
:ok
iex(46)>
```

Preview File Edit View Go Tools Window Help

Wed 3:49 PM



rider@stephens-MacBook-Pro: ~/workspace/ElixirWorkspace/prod/identicon (zsh)

⌘2 ..rod/identicon (zsh) ⌘3

```
18 17:29:28 on ttys003
4-creating-the-pixel-map) ✗ ls
asdf.png  config    lib       mix.exs   test
4-creating-the-pixel-map) ✗ open asdf.png
4-creating-the-pixel-map) ✗
```

```
12
13  def save_
14    File.wr
15  end
16
17  def draw_
18    image =
19    fill =
20
21    Enum.ea
22      :egd.
```

File 0 Project 0 ✔ No Issues lib/identicon.

banana.png

```
rider@stephens-MacBook-Pro: ~/workspace/ElixirWorkspace/prod/identicon (zsh)
tsh) ⌘2  ..rod/identicon (zsh) ⌘3
18 17:29:28 on ttys003
4-creating-the-pixel-map) ✗ ls
asdf.png  config    lib       mix.exs   test
4-creating-the-pixel-map) ✗ open asdf.png
4-creating-the-pixel-map) ✗ open banana.png
4-creating-the-pixel-map) ✗
```

```
12
13  def save_
14    File.wr
15  end
16
17  def draw_
18    image =
19    fill =
20
21    Enum.ea
22      :egd.
```

identicon.ex × image.ex

```
1   defmodule I
2     def main(
3       input
4       |> hash
5       |> pick
6       |> buil
7       |> filt
8       |> buil
9       |> draw
10      |> save
11    end
12
13    def save_
14      File.wr
15    end
16
17    def draw_
18      image =
```


1. iex -S mix (beam.smp)

iex (beam.smp) ⌘1 ..rod/identicon (zsh) ⌘2 ..rod/identicon (zsh) ⌘3

```
ompilers.Elixir.compile/5
        (mix) lib/mix/task.ex:296: Mix.Task.run_tas
k/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
iex(44)> recompile
Compiling 1 file (.ex)
:ok
iex(45)> Identicon.main("asdf")
:ok
iex(46)> Identicon.main("banana")
:ok
iex(47)>
```