Memory

don't forget to take out the garbage!

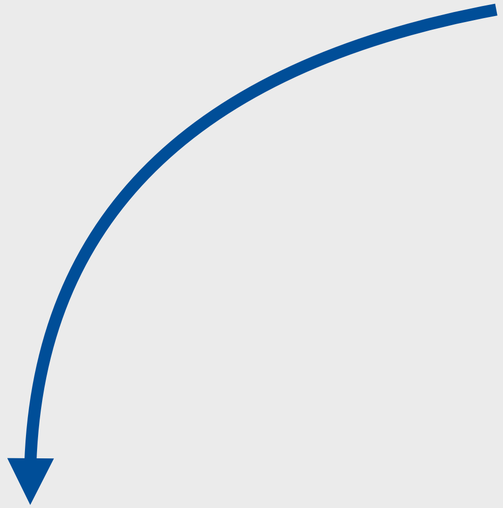
katie fenn

Memory

don't forget to take out the garbage!

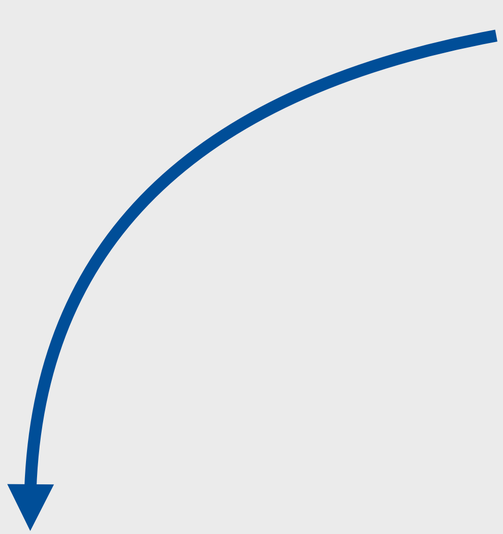
katie fenn

find me on
twitter

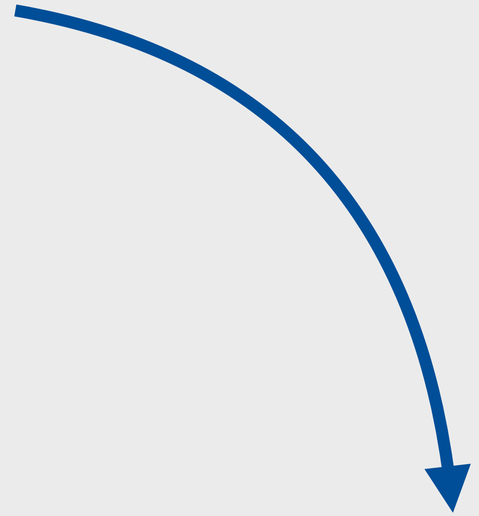


@katie_fenn

find me on twitter

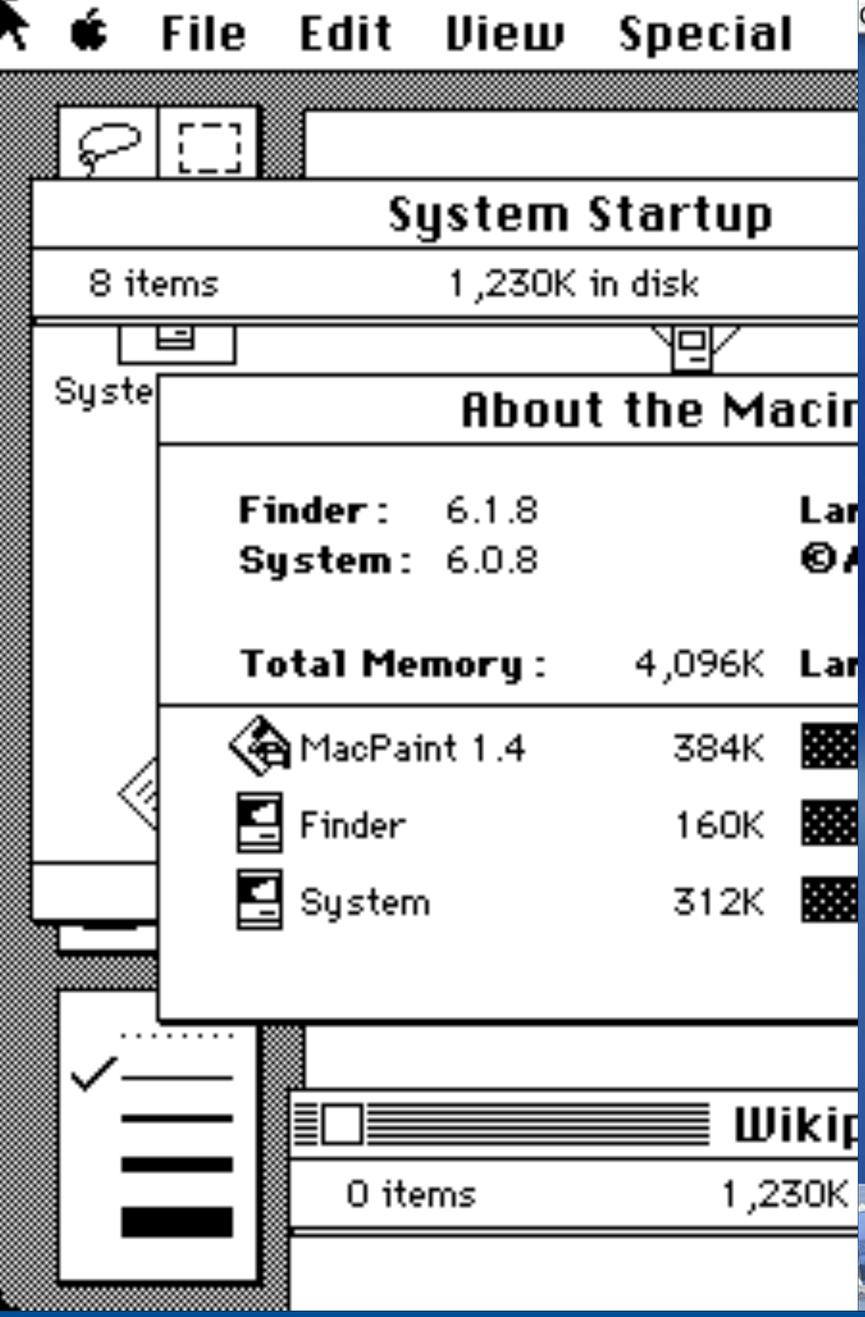


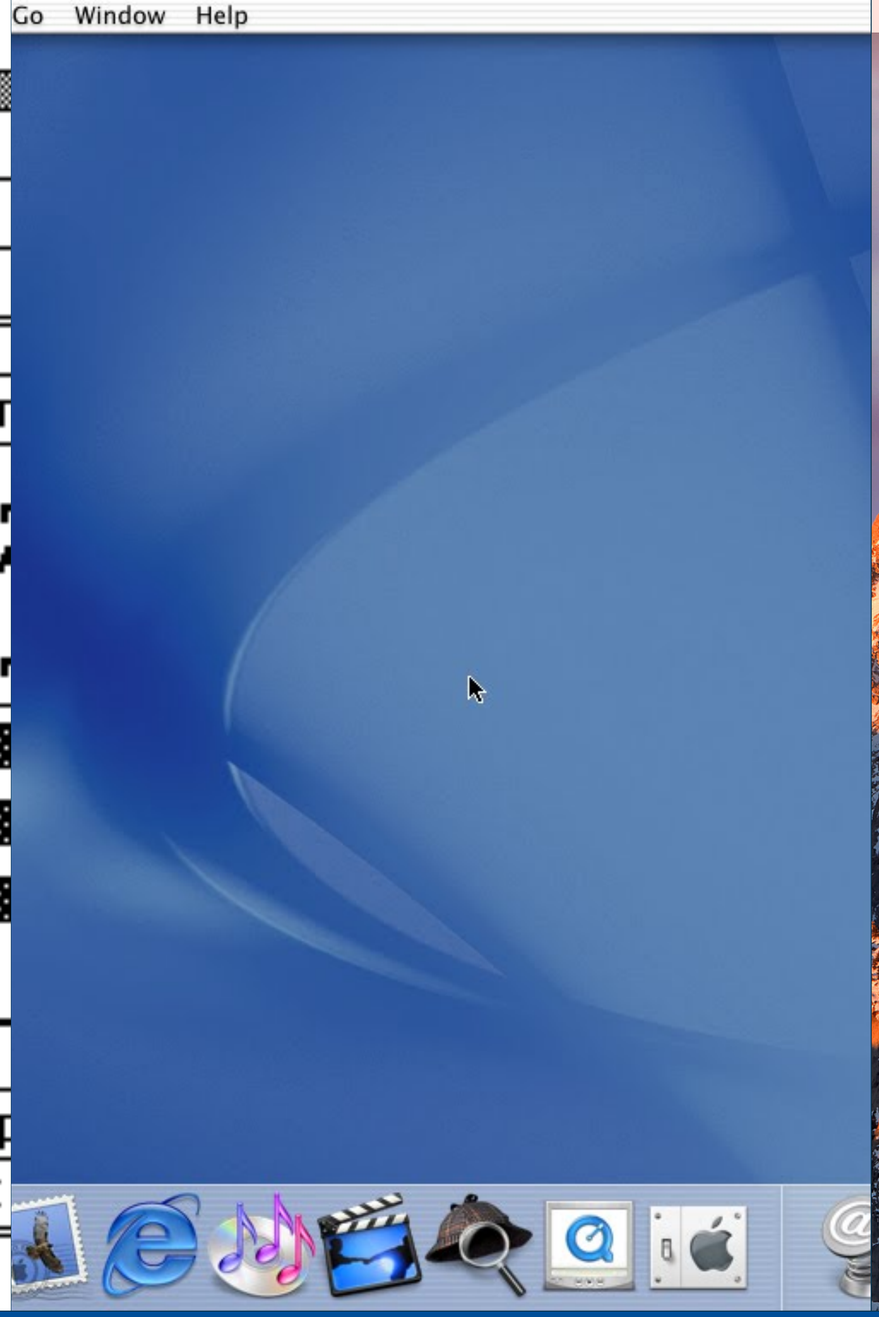
content warnings



@katie_fenn

cw: large animation


File Edit View Special
System Startup
8 items
1,230K in disk
About the Macir
Finder: 6.1.8
System: 6.0.8
Total Memory: 4,096K
MacPaint 1.4 384K
Finder 160K
System 312K
Wikip
0 items
1,230K

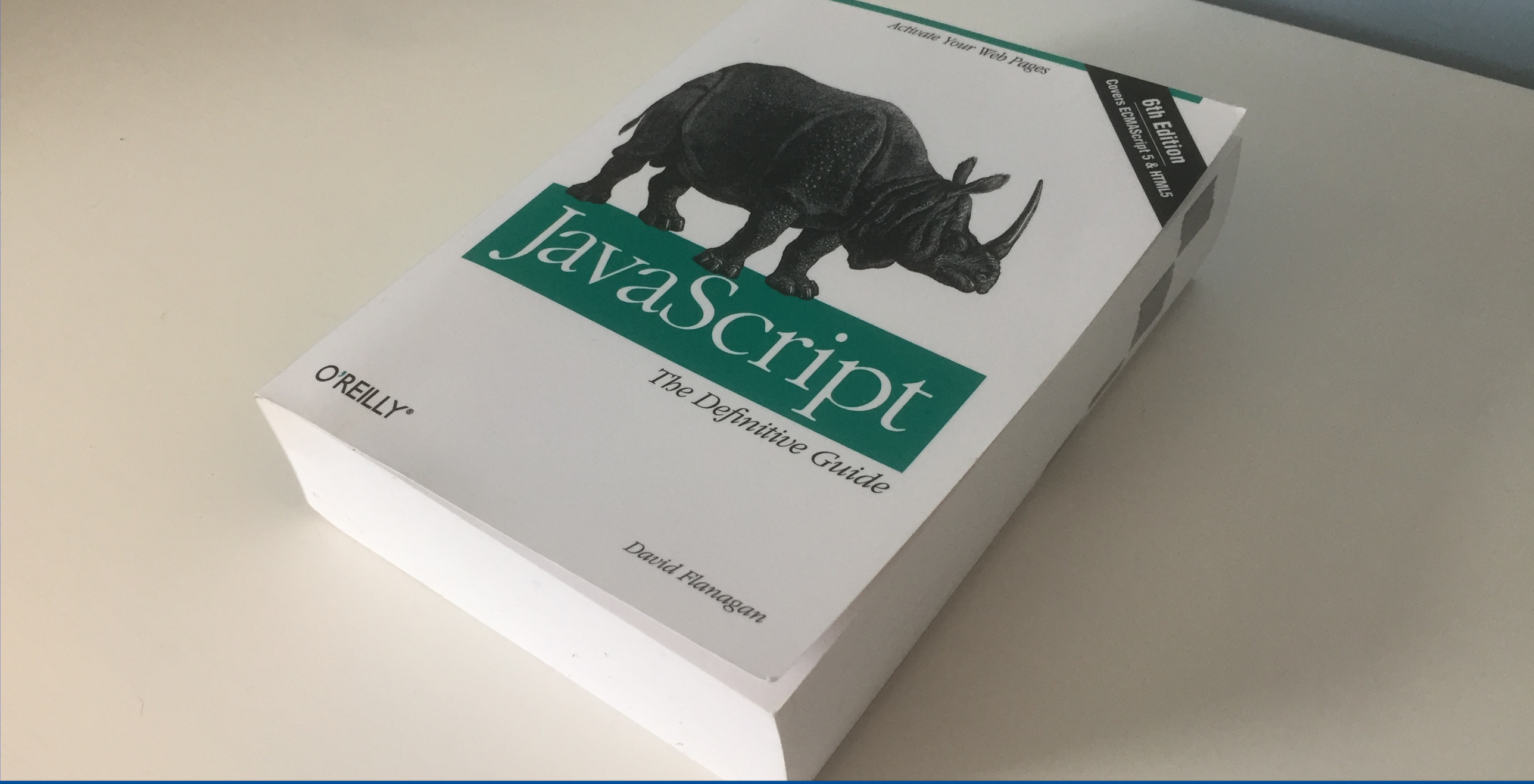

Go Window Help


19%
Sat 23:51

@katie_fenn

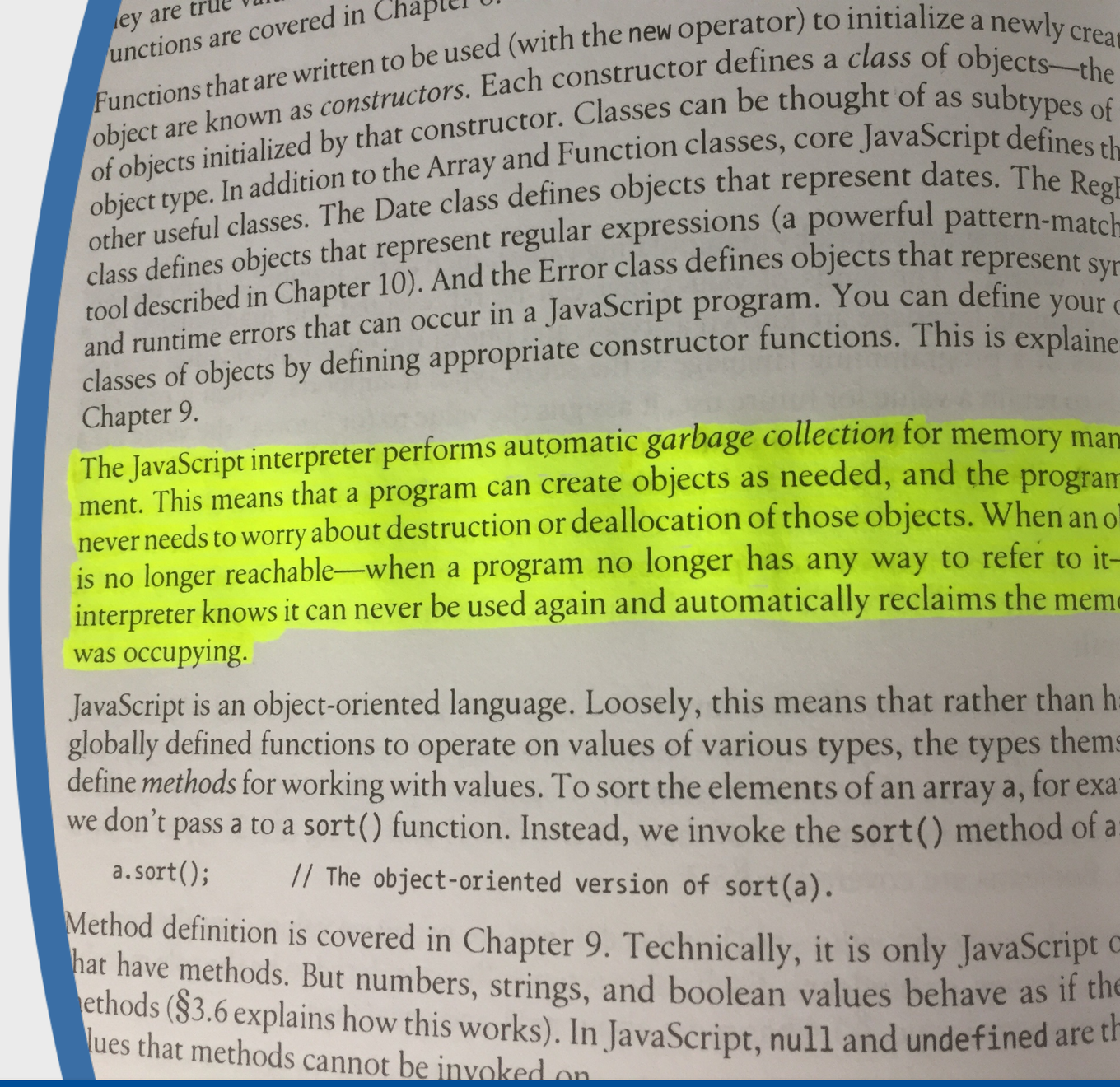
@katie_fenn

https://flic.kr/p/VdFhKW


Activate Your Web Pages
6th Edition
Covers ECMAScript 5 & HTML5
JavaScript
The Definitive Guide
O'REILLY®
David Flanagan

@katie_fenn

the programmer
never needs
to worry
about deallocation



ey are true va
unctions are covered in Chapter 8.

Functions that are written to be used (with the new operator) to initialize a newly crea
object are known as *constructors*. Each constructor defines a *class* of objects—the
of objects initialized by that constructor. Classes can be thought of as subtypes of
object type. In addition to the Array and Function classes, core JavaScript defines th
other useful classes. The Date class defines objects that represent dates. The Regl
class defines objects that represent regular expressions (a powerful pattern-match
tool described in Chapter 10). And the Error class defines objects that represent syn
and runtime errors that can occur in a JavaScript program. You can define your o
classes of objects by defining appropriate constructor functions. This is explaine
Chapter 9.

The JavaScript interpreter performs automatic *garbage collection* for memory man
ment. This means that a program can create objects as needed, and the program
never needs to worry about destruction or deallocation of those objects. When an o
is no longer reachable—when a program no longer has any way to refer to it—
interpreter knows it can never be used again and automatically reclaims the memo
was occupying.

JavaScript is an object-oriented language. Loosely, this means that rather than h
globally defined functions to operate on values of various types, the types thems
define *methods* for working with values. To sort the elements of an array a, for exa
we don't pass a to a sort() function. Instead, we invoke the sort() method of a

```
a.sort();       // The object-oriented version of sort(a).
```

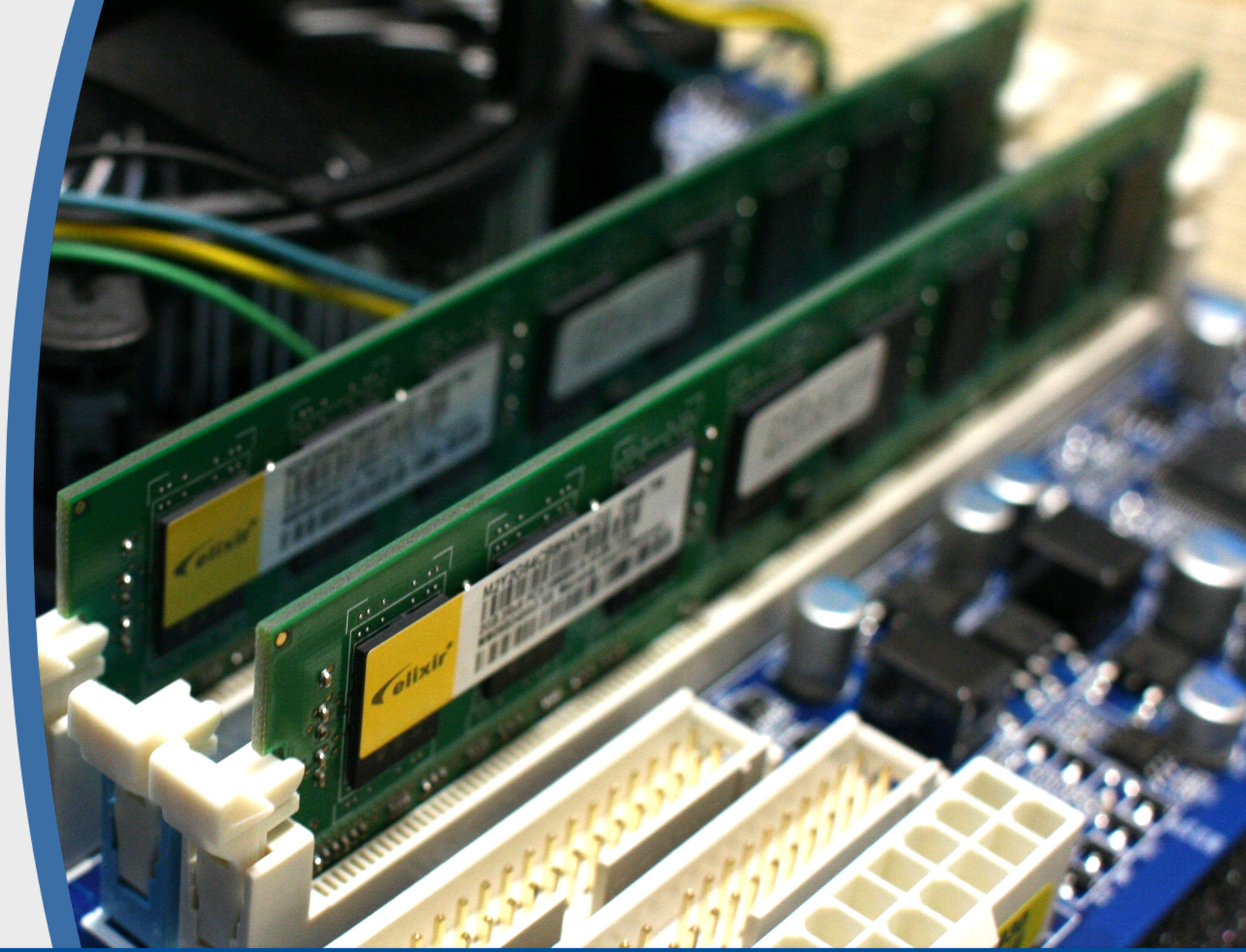
Method definition is covered in Chapter 9. Technically, it is only JavaScript o
hat have methods. But numbers, strings, and boolean values behave as if the
ethods (§3.6 explains how this works). In JavaScript, null and undefined are th
lues that methods cannot be invoked on

@katie_fenn

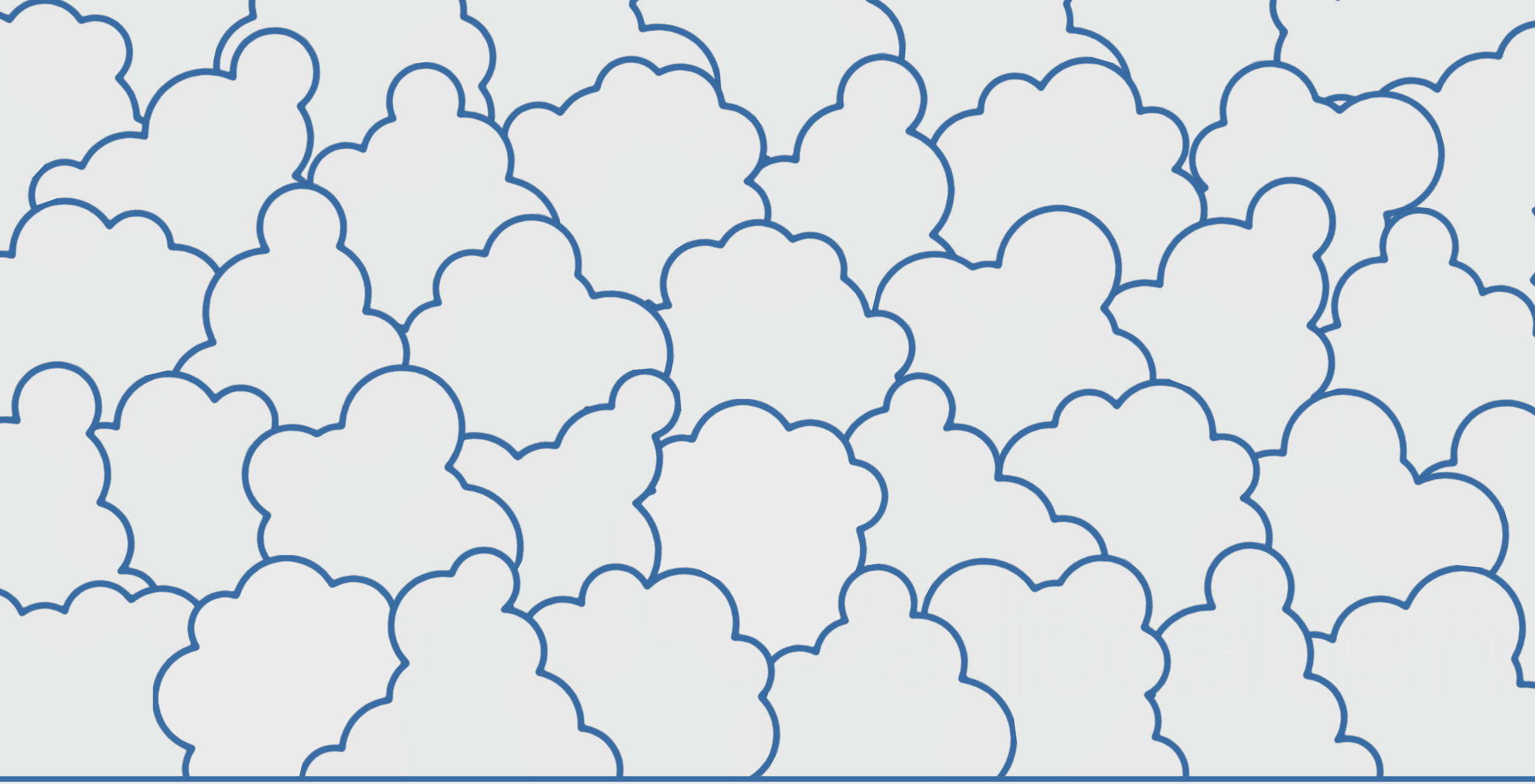
what is memory?

@katie_fenn

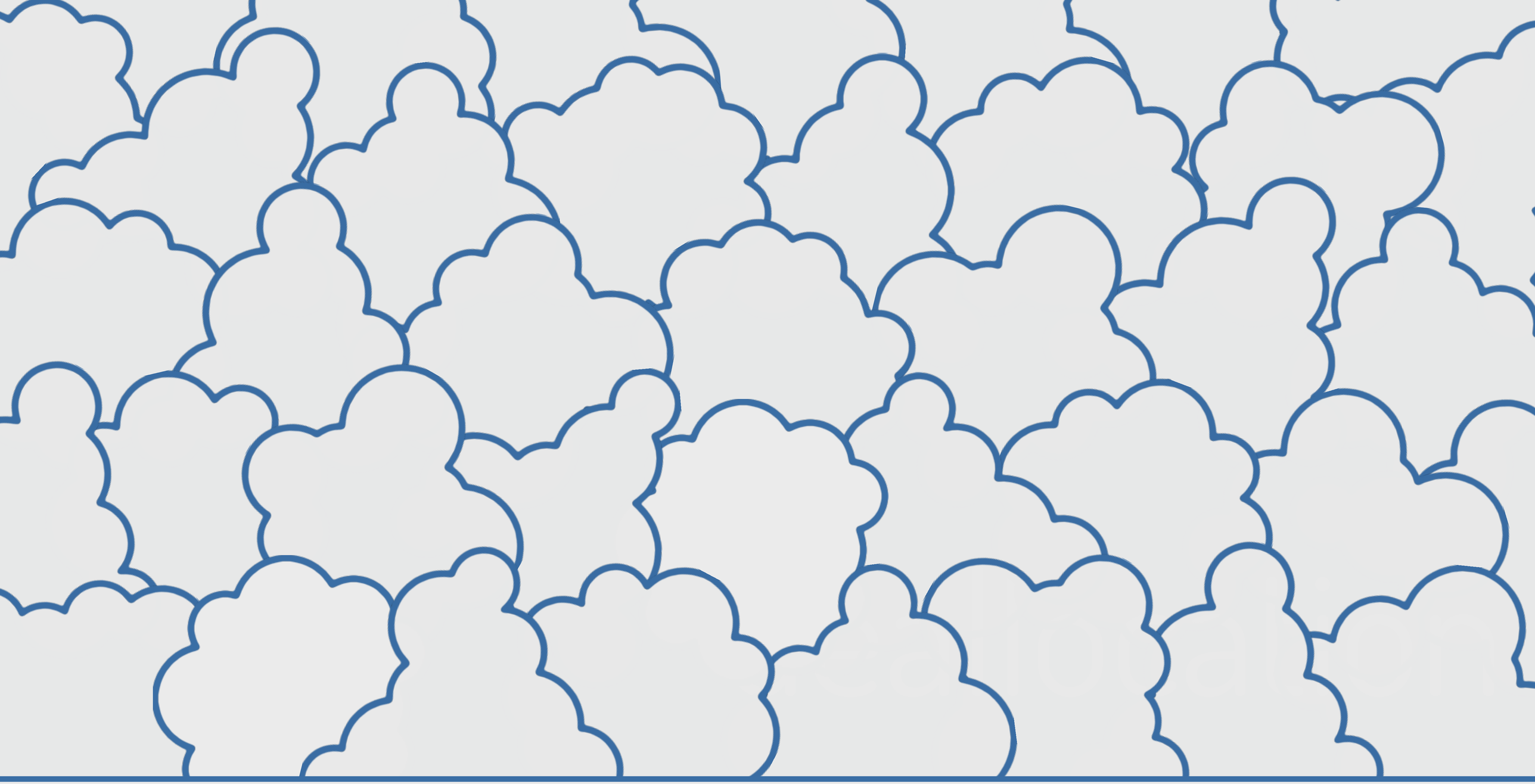
ram



heaps



@katie_fenn



@katie_fenn

allocation

```
var variable = 'value'
```

allocation

```
var variable = 'value'
```

allocation

+=

<<=

^=

*=

-=

|=

```
var variable = 'value'
```

>>=

&=

**=

/=

>>>=

%=

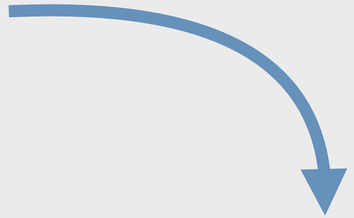
allocation

assigns a value
from the right...

```
var variable = 'value'
```

allocation

assigns a value
from the right...



```
var variable = 'value'
```

...to a
memory address
on the left

deallocation

```
function () {
  var a = 1
  function () {
    var b = 2




  }()
}()
```

deallocation

```
function () {
  var a = 1
  function () {
    var b = 2
    function () {
      var c = a + b                    a: 1
    }()
  }()
}()
```

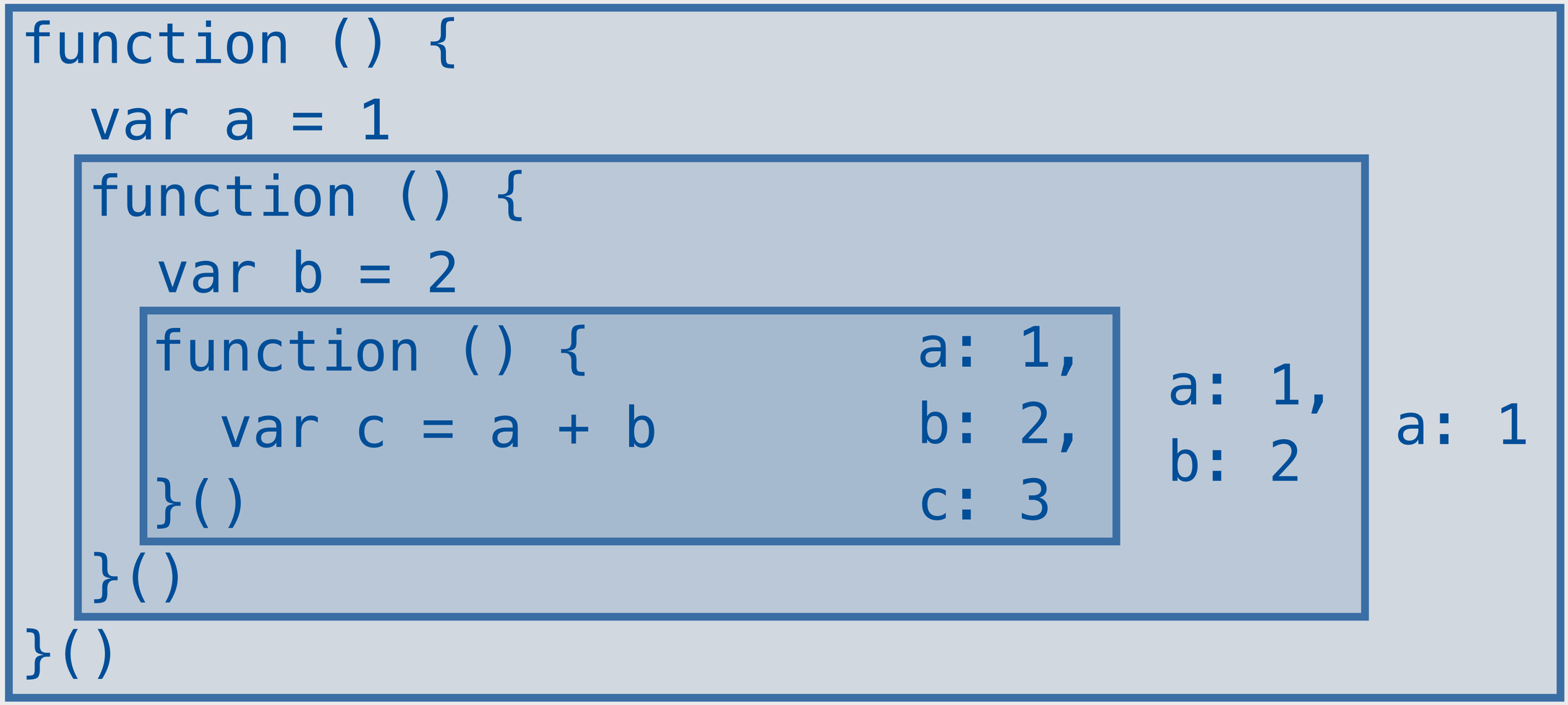
deallocation

```
function () {
  var a = 1
  function () {
    var b = 2
    function () {
      var c = a + b
    }()
  }()
}()
```

a: 1,
b: 2

a: 1

deallocation



deallocation

```
function () {
  var a = 1
  function () {
    var b = 2
    function () {
      var c = a + b
    }()
  }()
}()
```

```
a: 1,
b: 2,
c: 3
```

```
a: 1,
b: 2
```

```
a: 1
```

deallocation

```
function () {
  var a = 1
  function () {
    var b = 2
    function () {
      var c = a + b
    }()
  }()
}()
```

```
a: 1,
b: 2,
c: 3
```

```
a: 1,
b: 2
```

```
a: 1
```

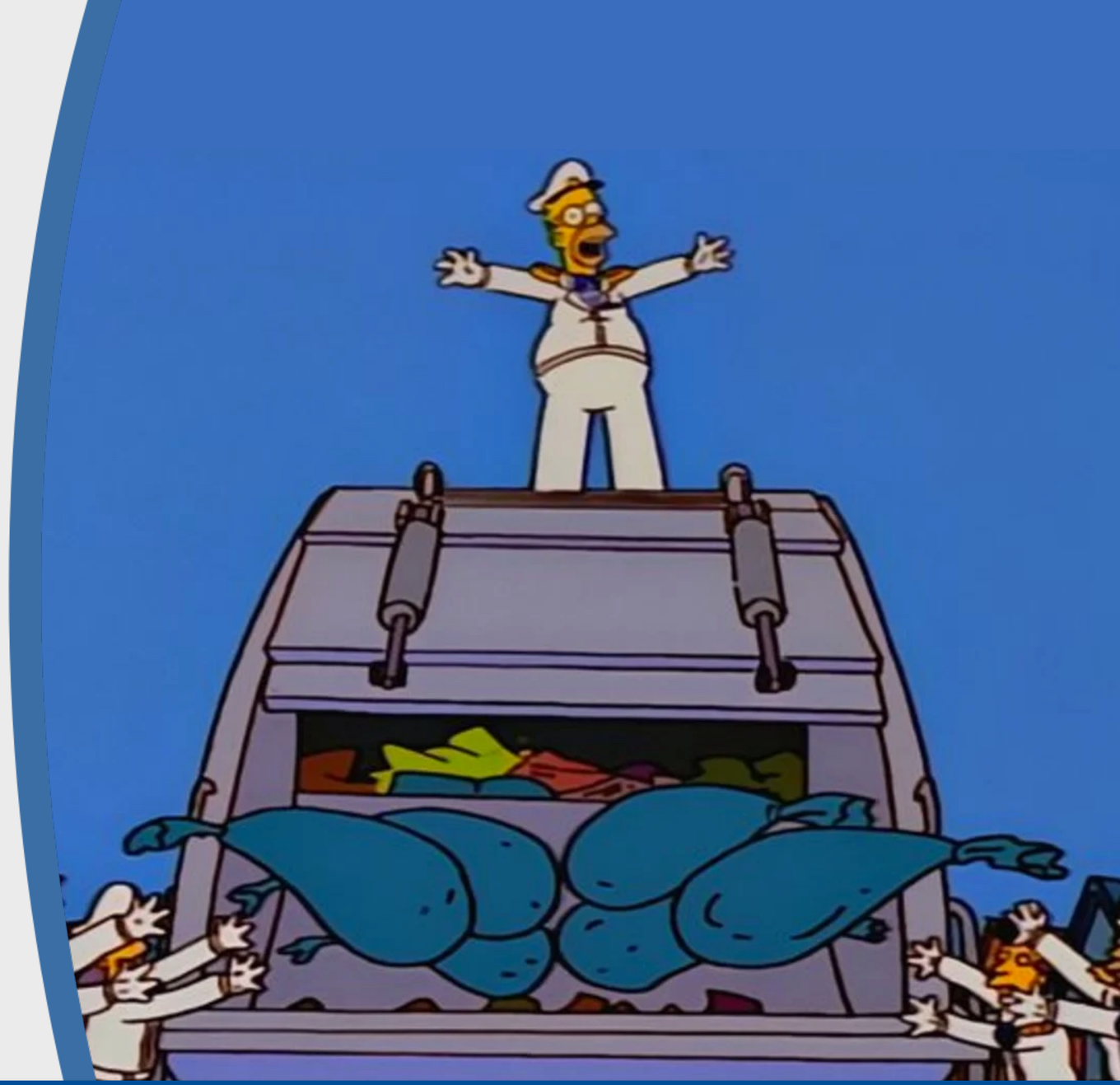
deallocation

```
function () {
  var a = 1
  function () {
    var b = 2
    function () {
      var c = a + b
    }()
  }()
}()
```

variables are retained

as long as their parent scope exists

garbage collection



garbage collection

@katie_fenn

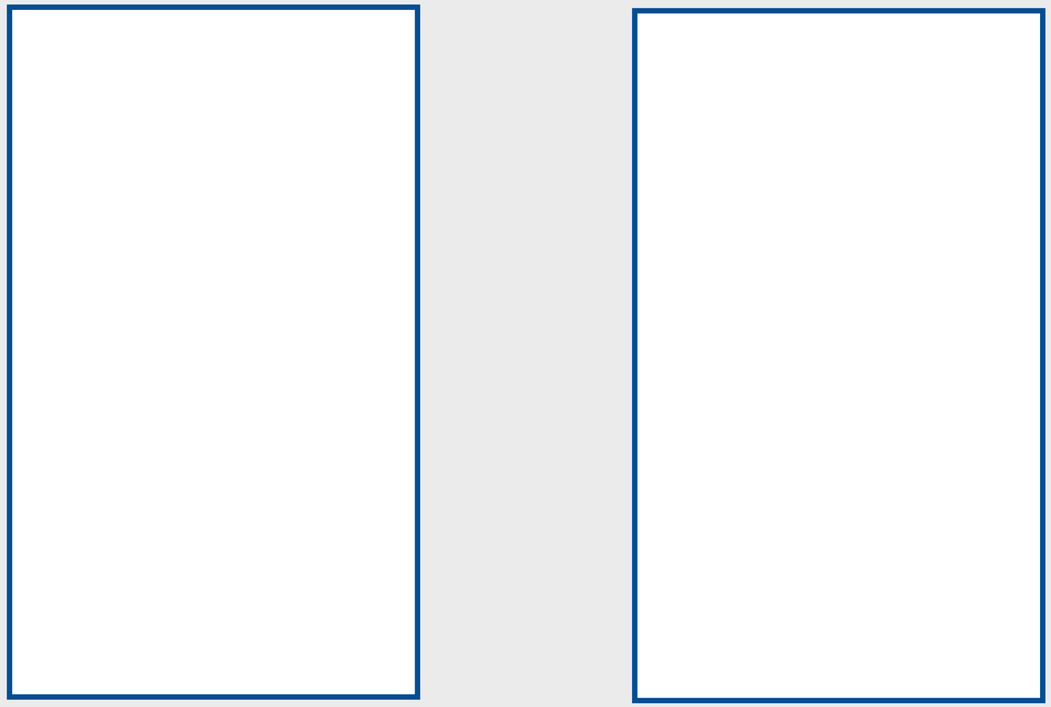
```
int *array = malloc(10 * sizeof(int));
```

```
int *array = malloc(10 * sizeof(int));
```

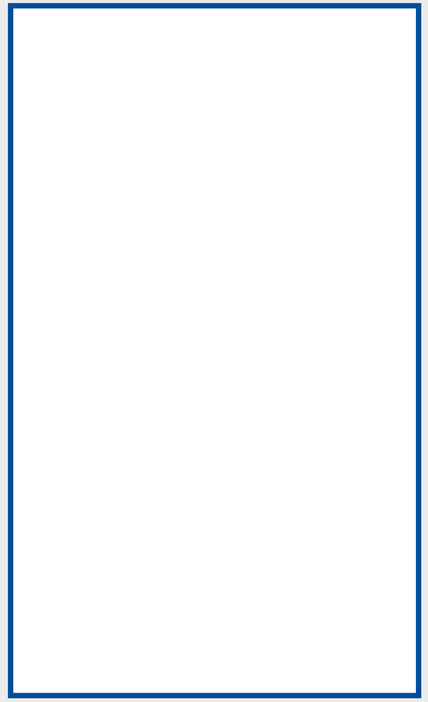
garbage collector:

```
free(array);
```

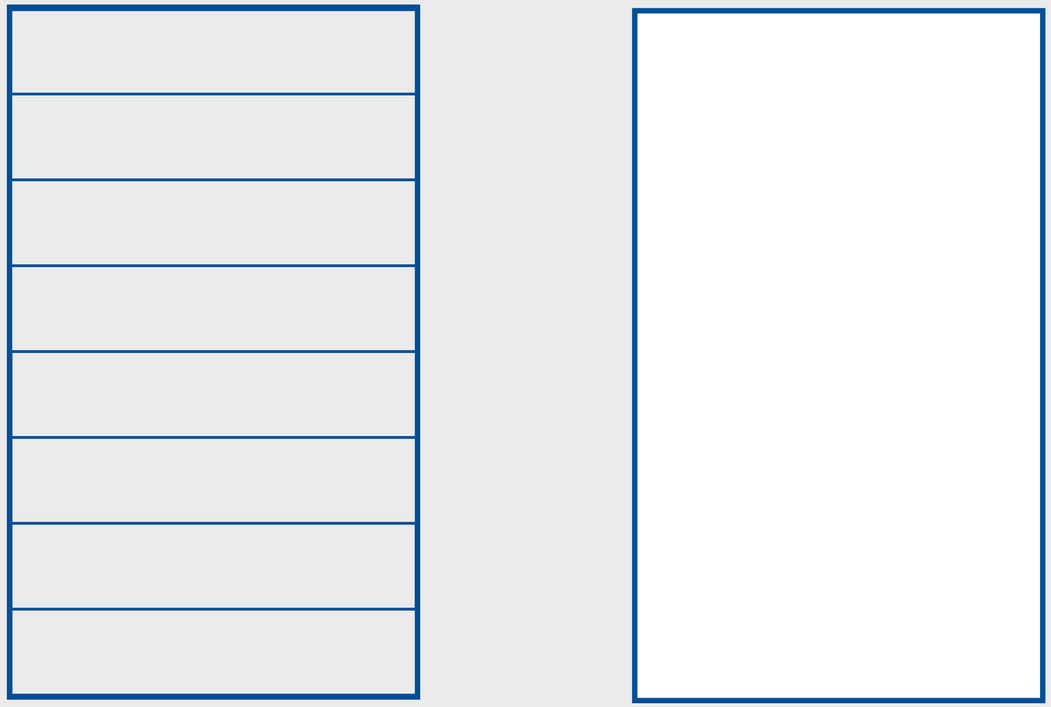
young generation



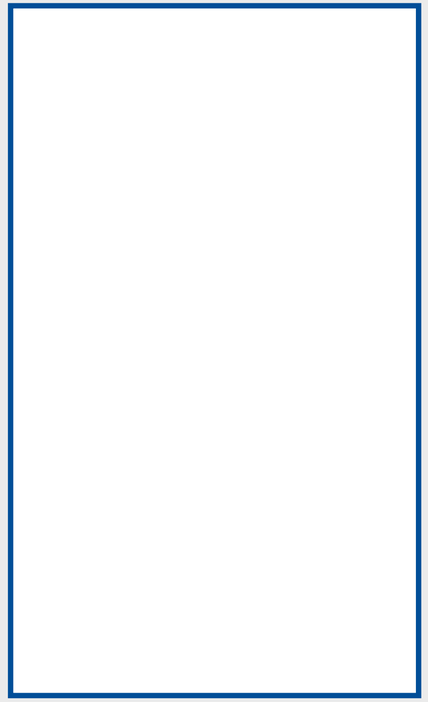
old generation

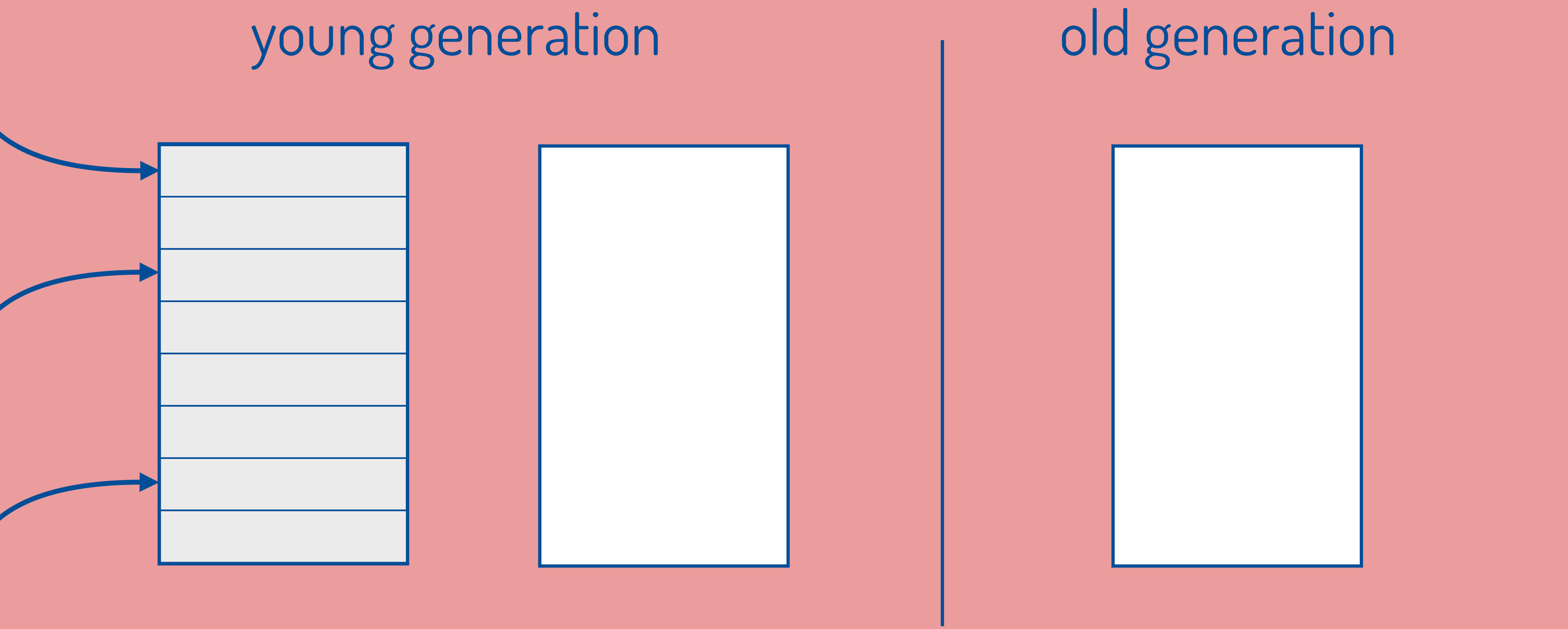


young generation



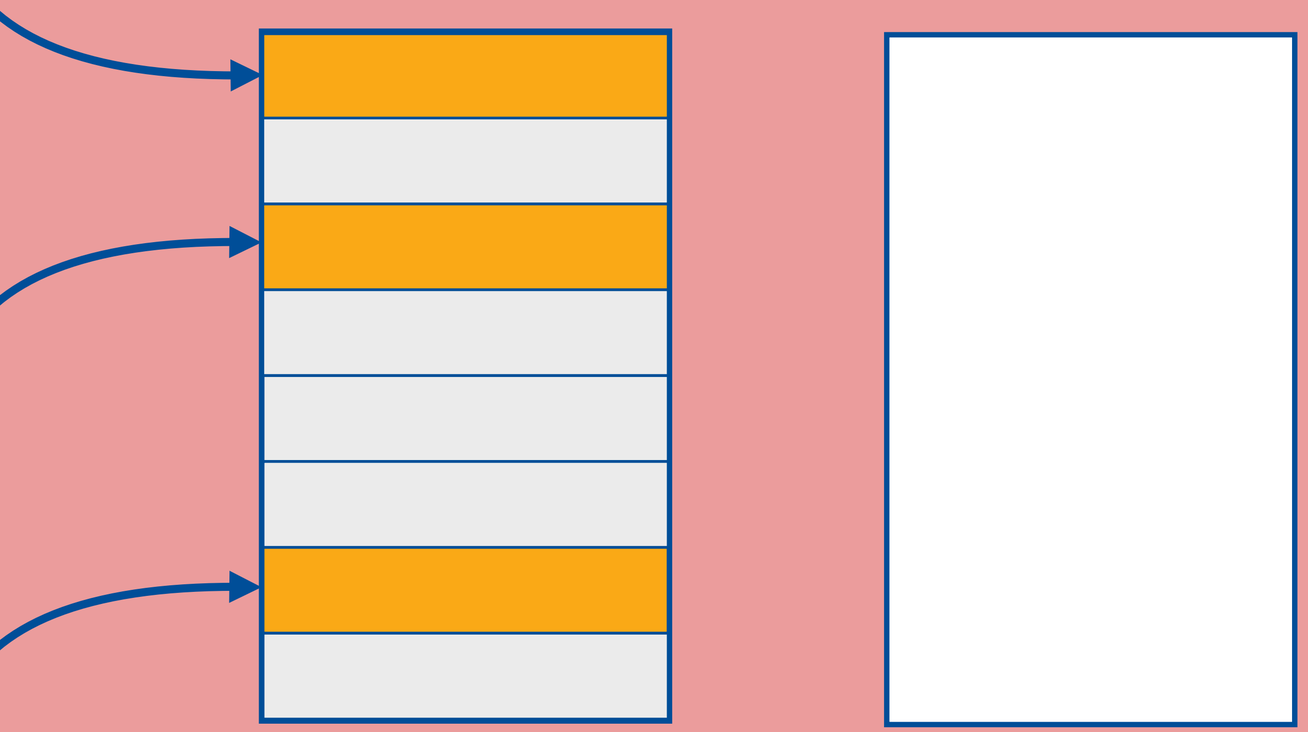
old generation



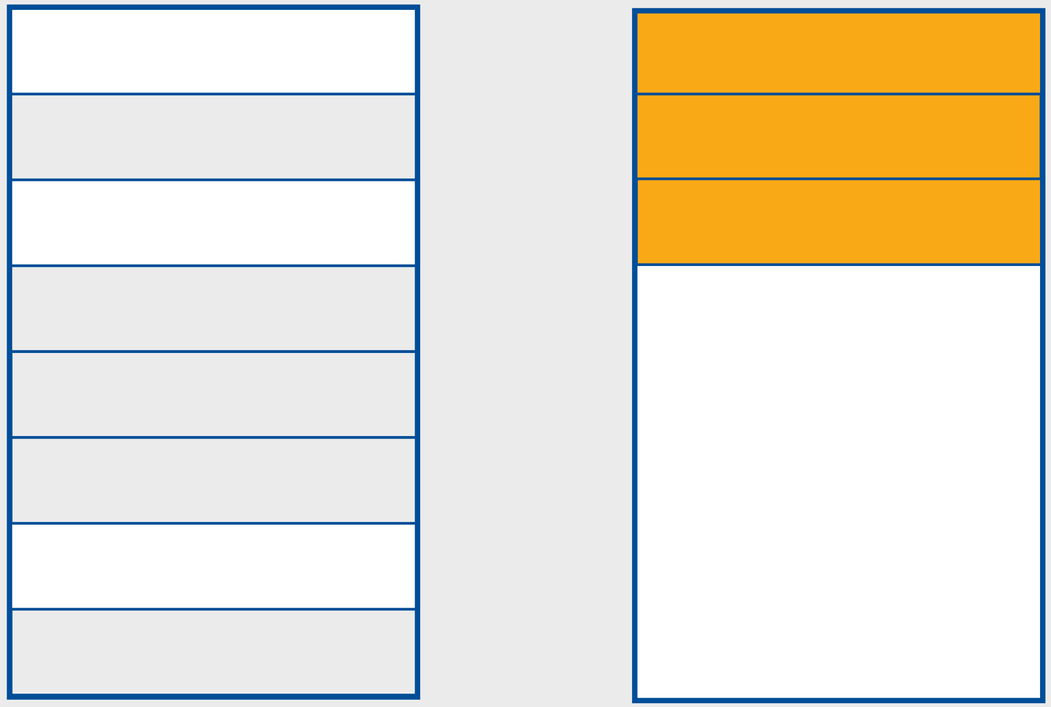

young generation
old generation

young generation

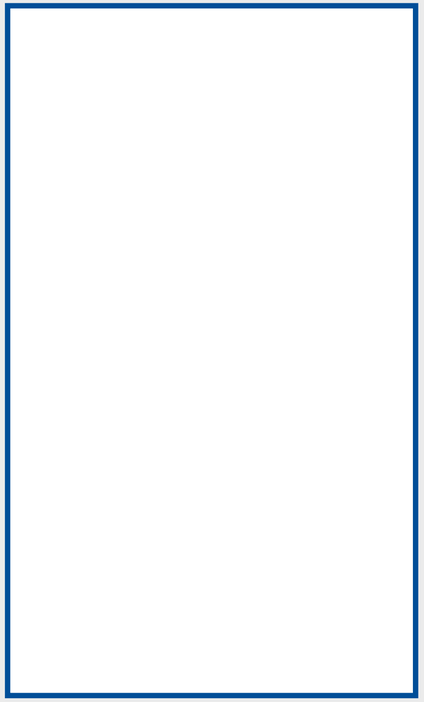
old generation



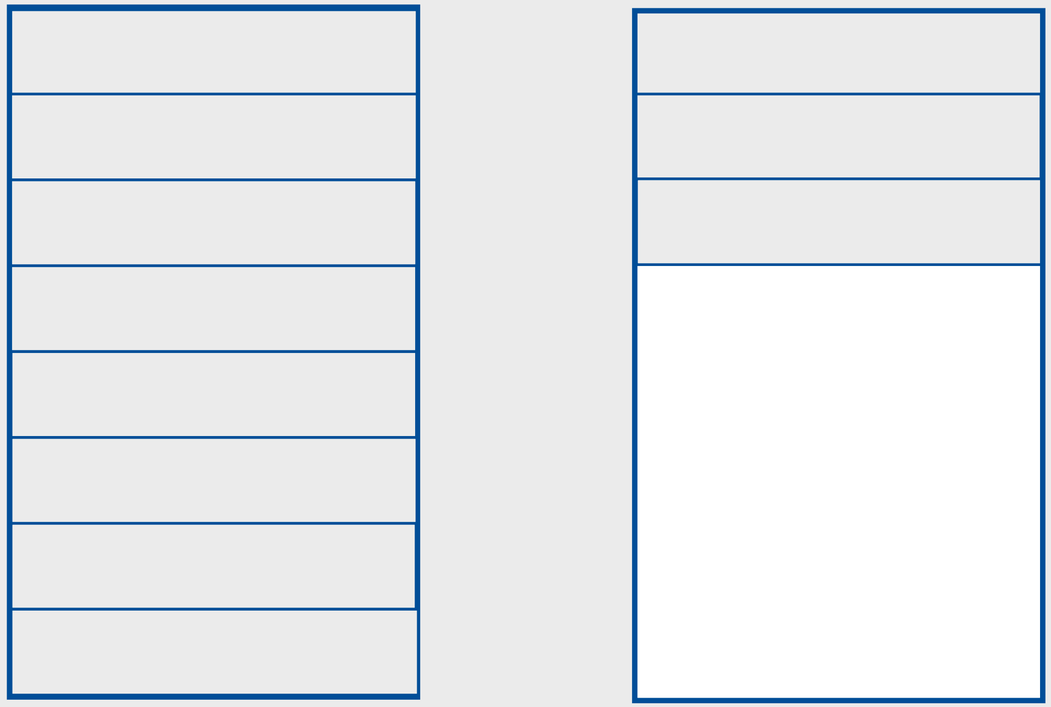
young generation



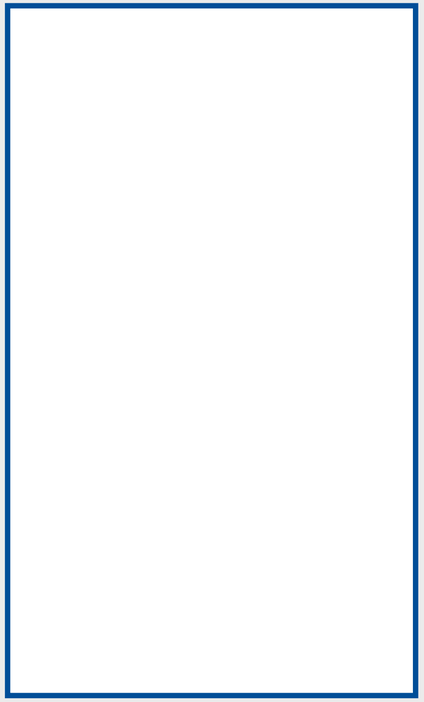
old generation



young generation

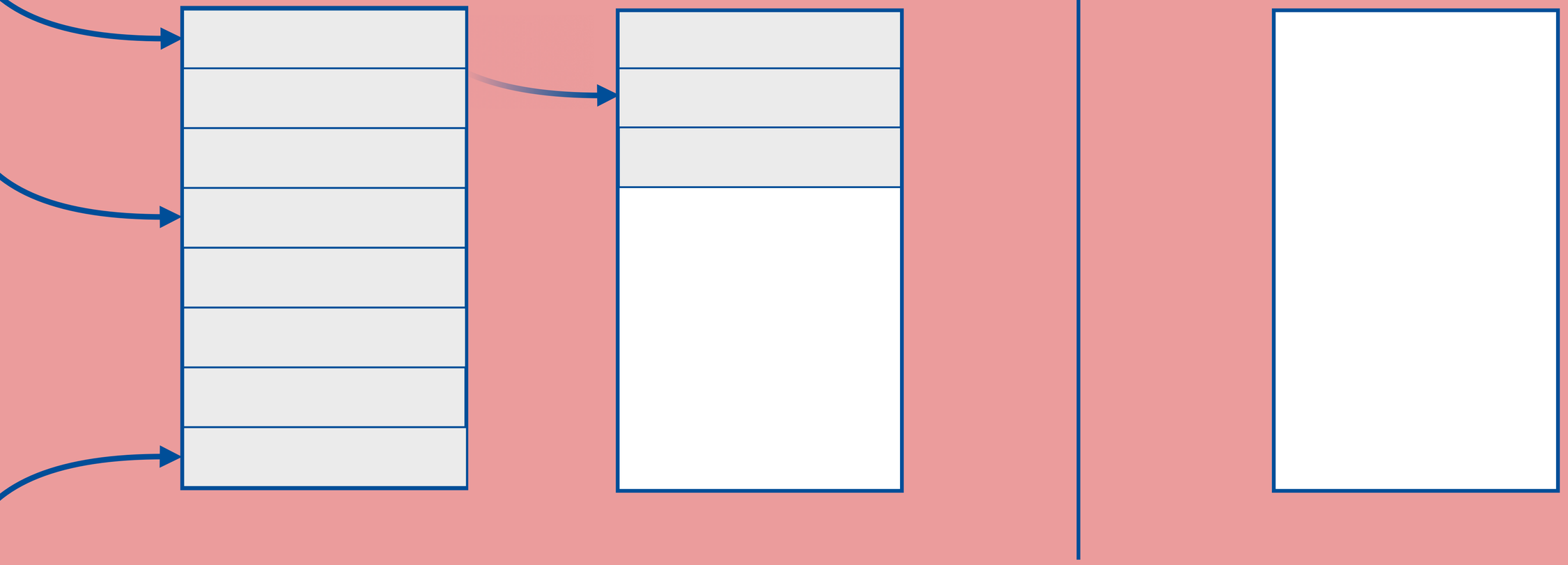


old generation



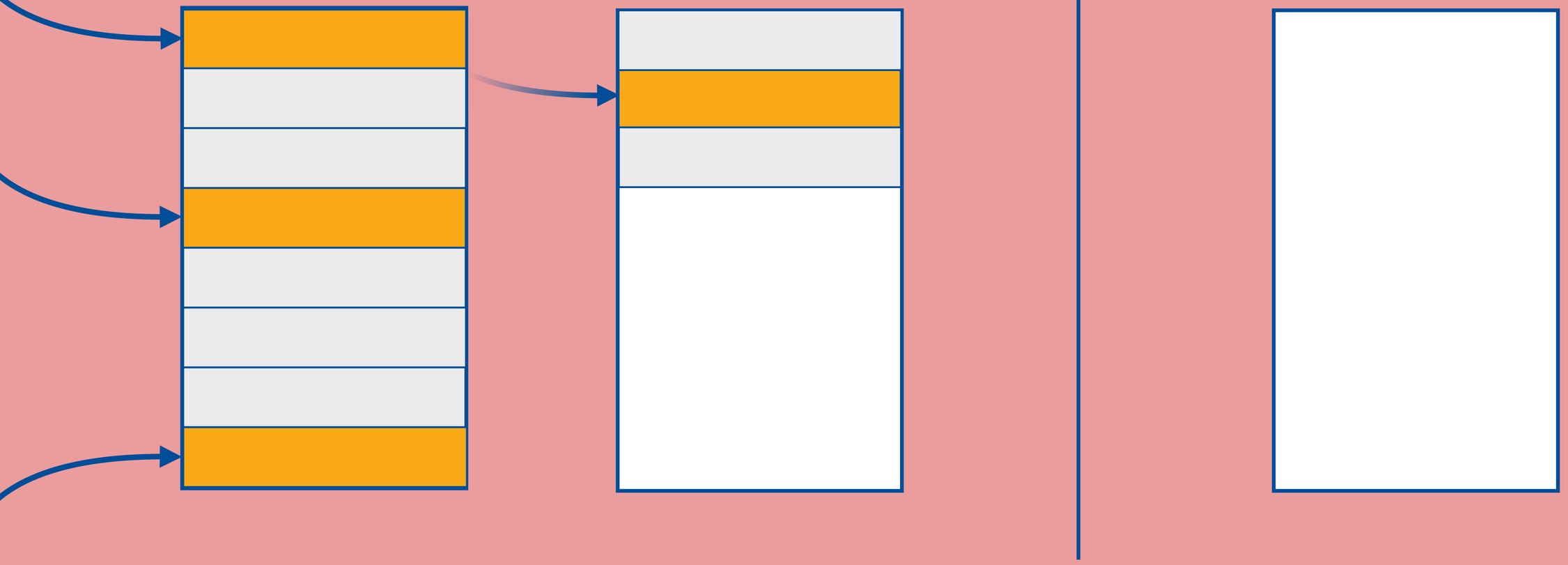
young generation

old generation

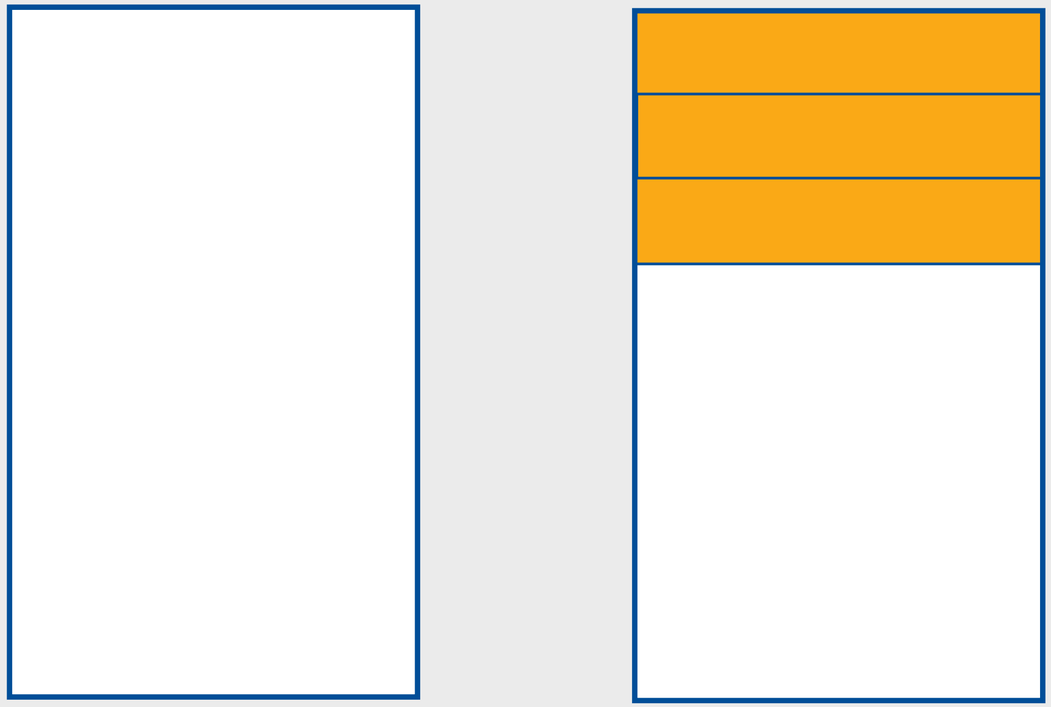


young generation

old generation



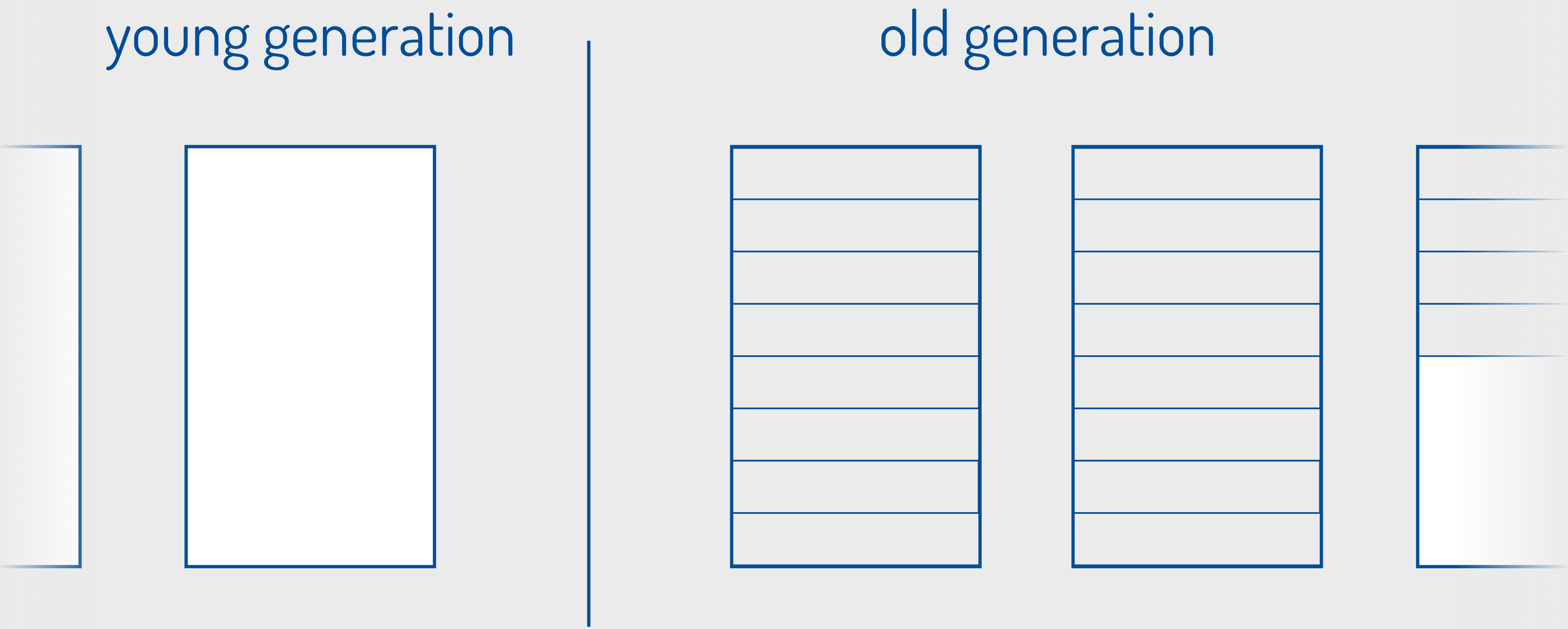
young generation



old generation

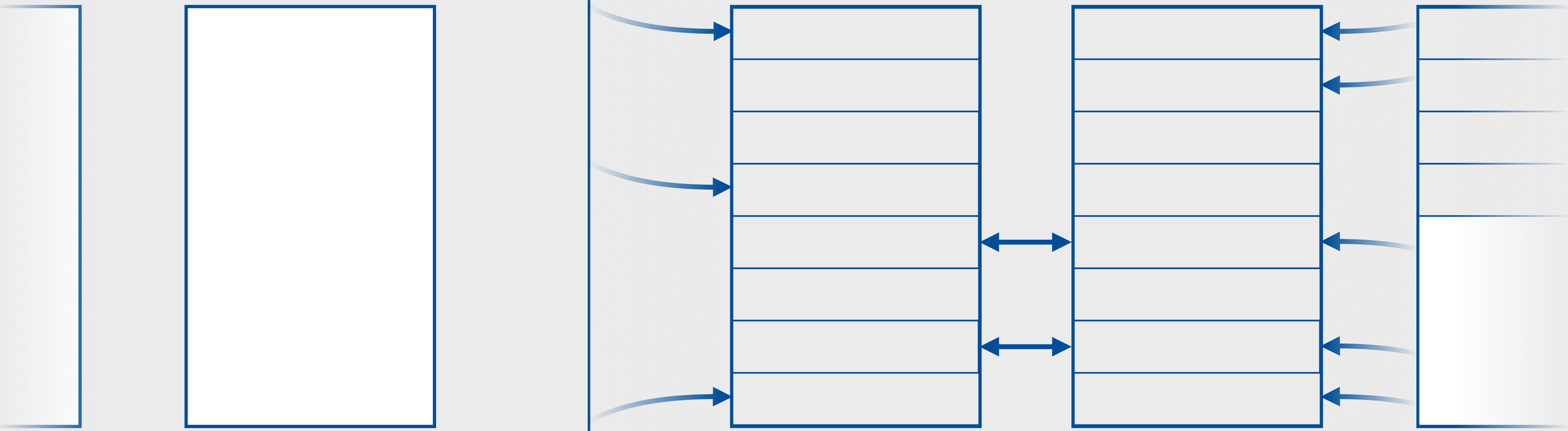
scavenging:

the minor garbage collector


young generation
old generation

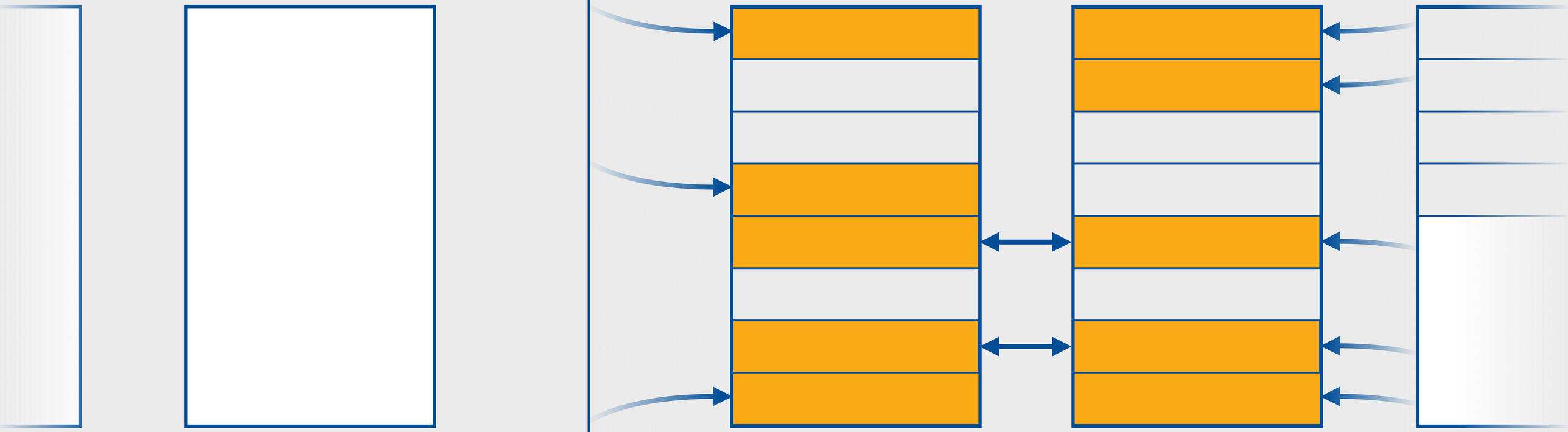
young generation

old generation



young generation

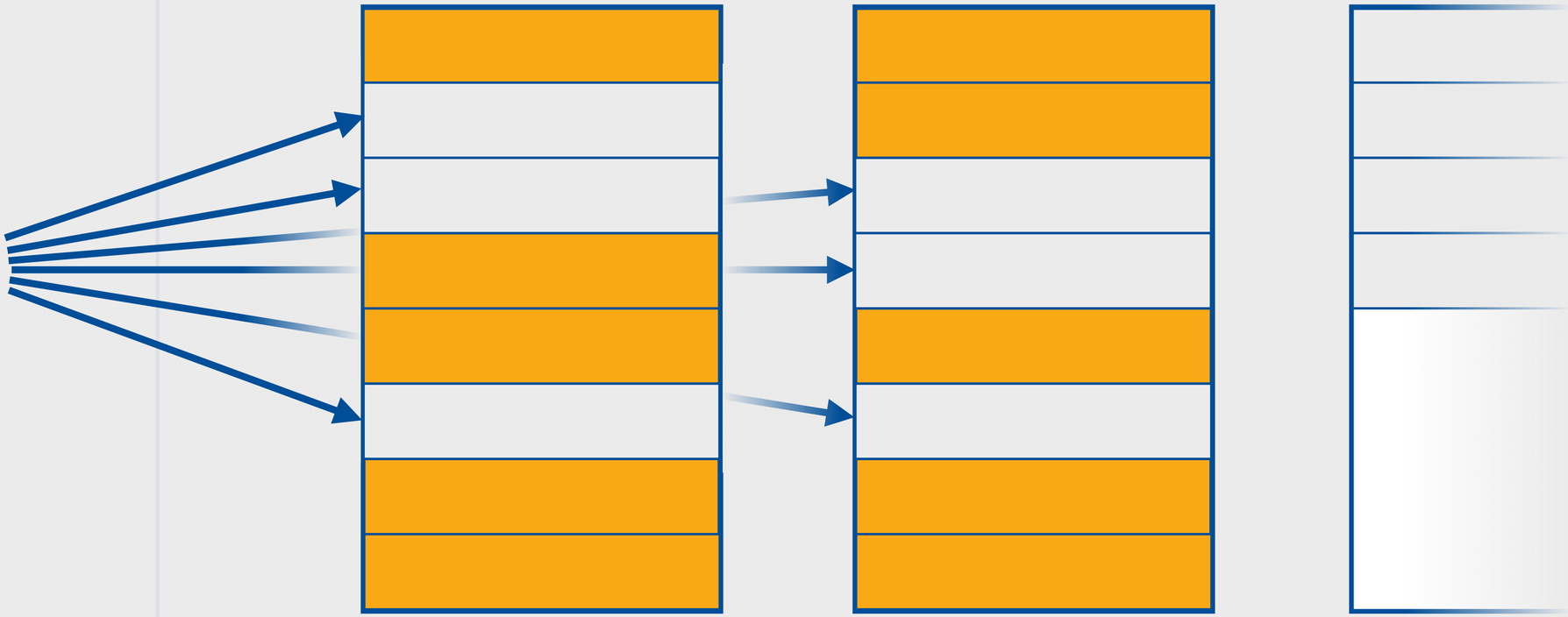
old generation




young generation
old generation
marking

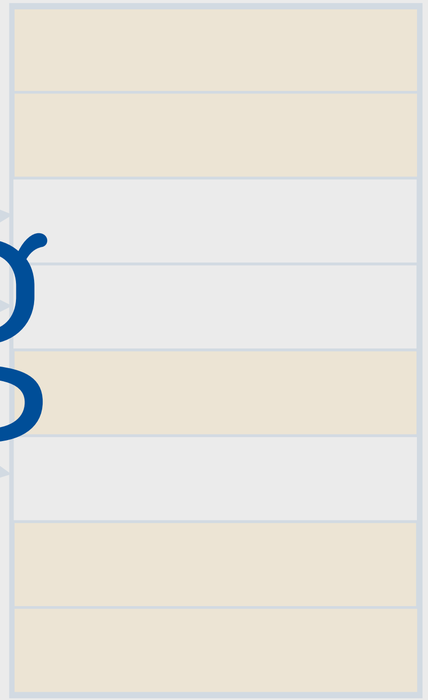
freelist

old generation

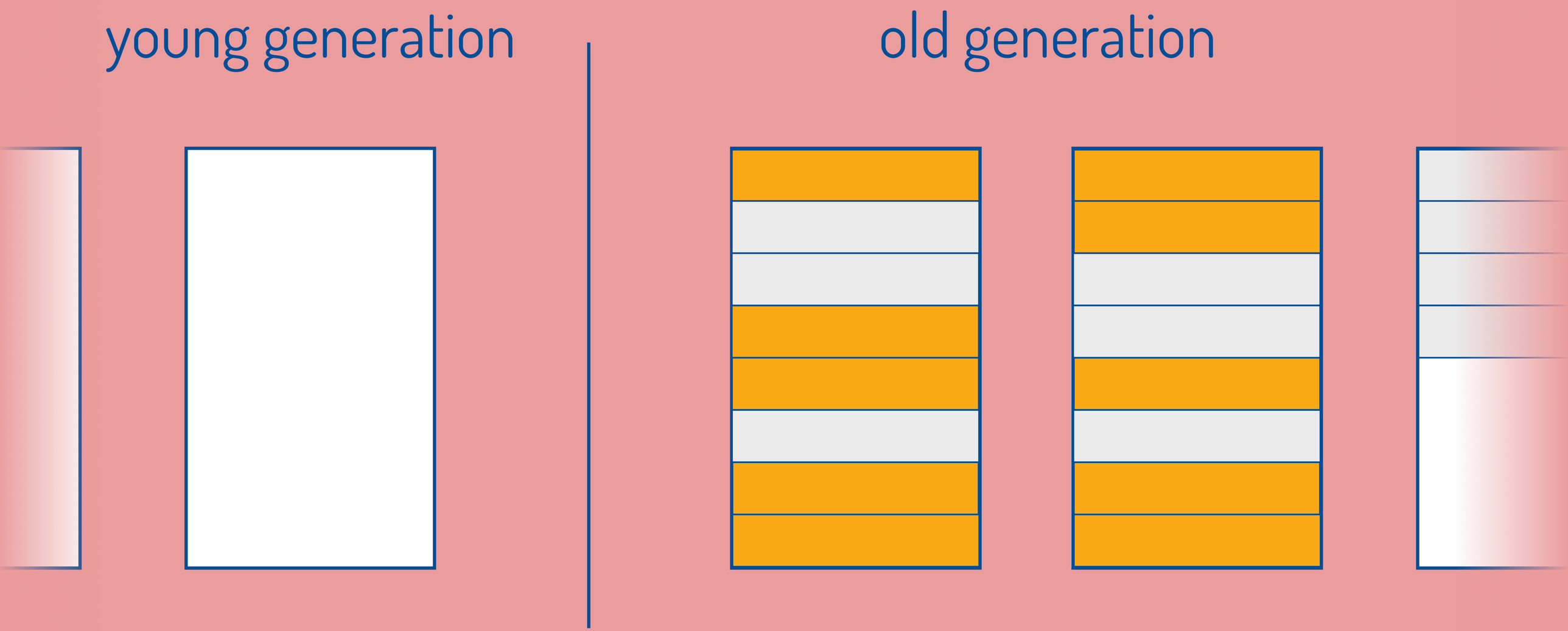


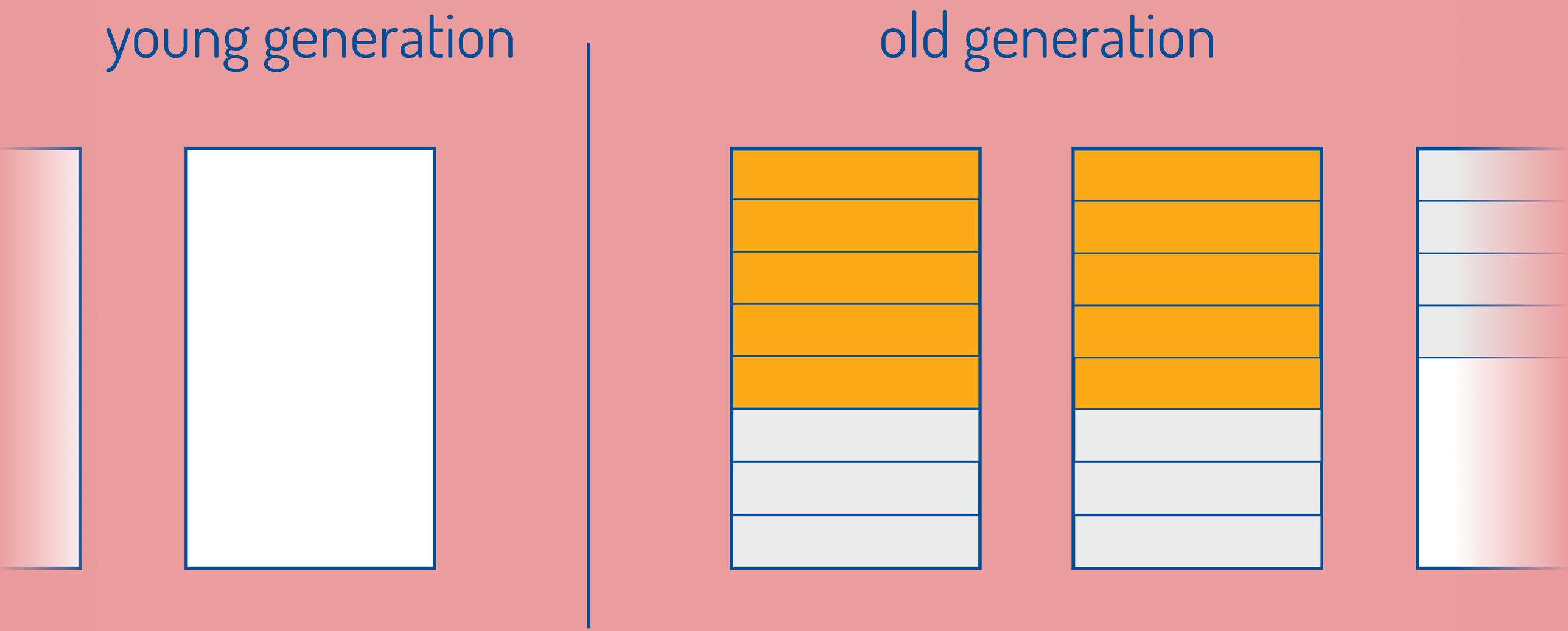
young free list tion

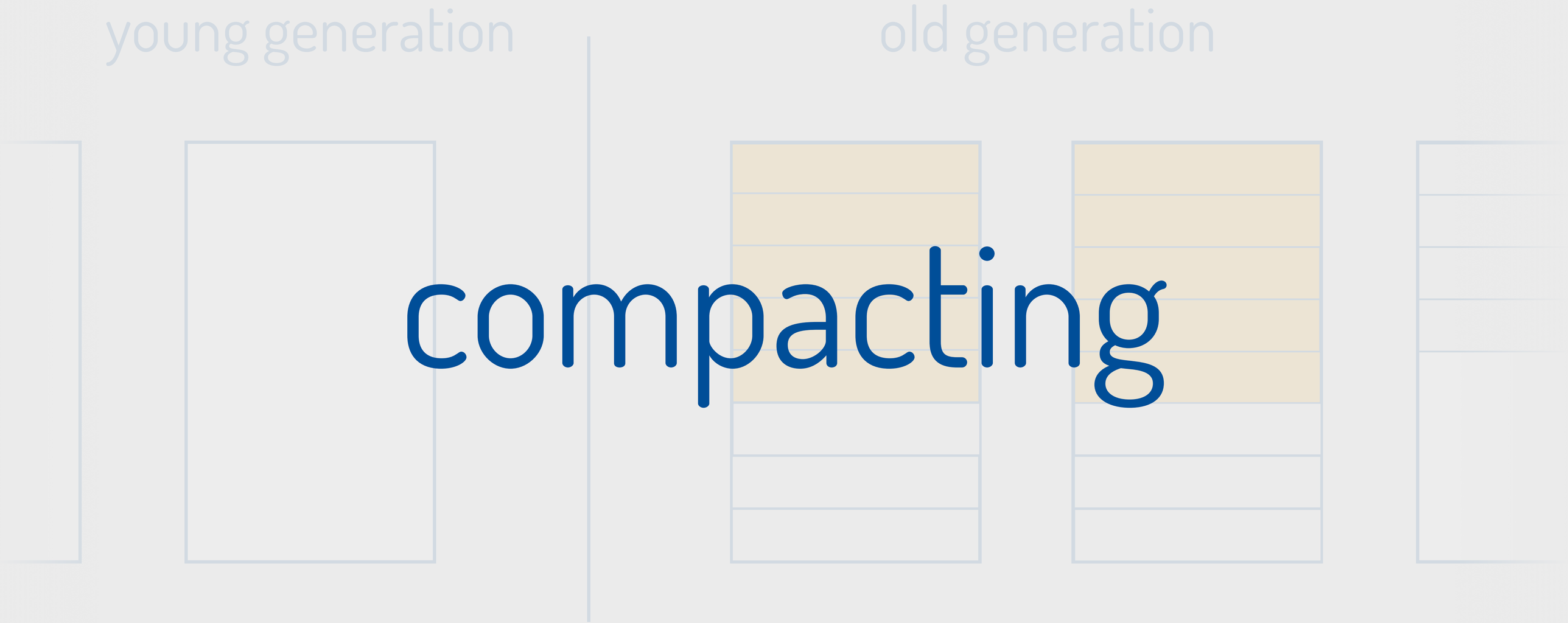
old generation



sweeping


young generation
old generation


young generation
old generation


young generation
old generation
compacting

mark and compact:

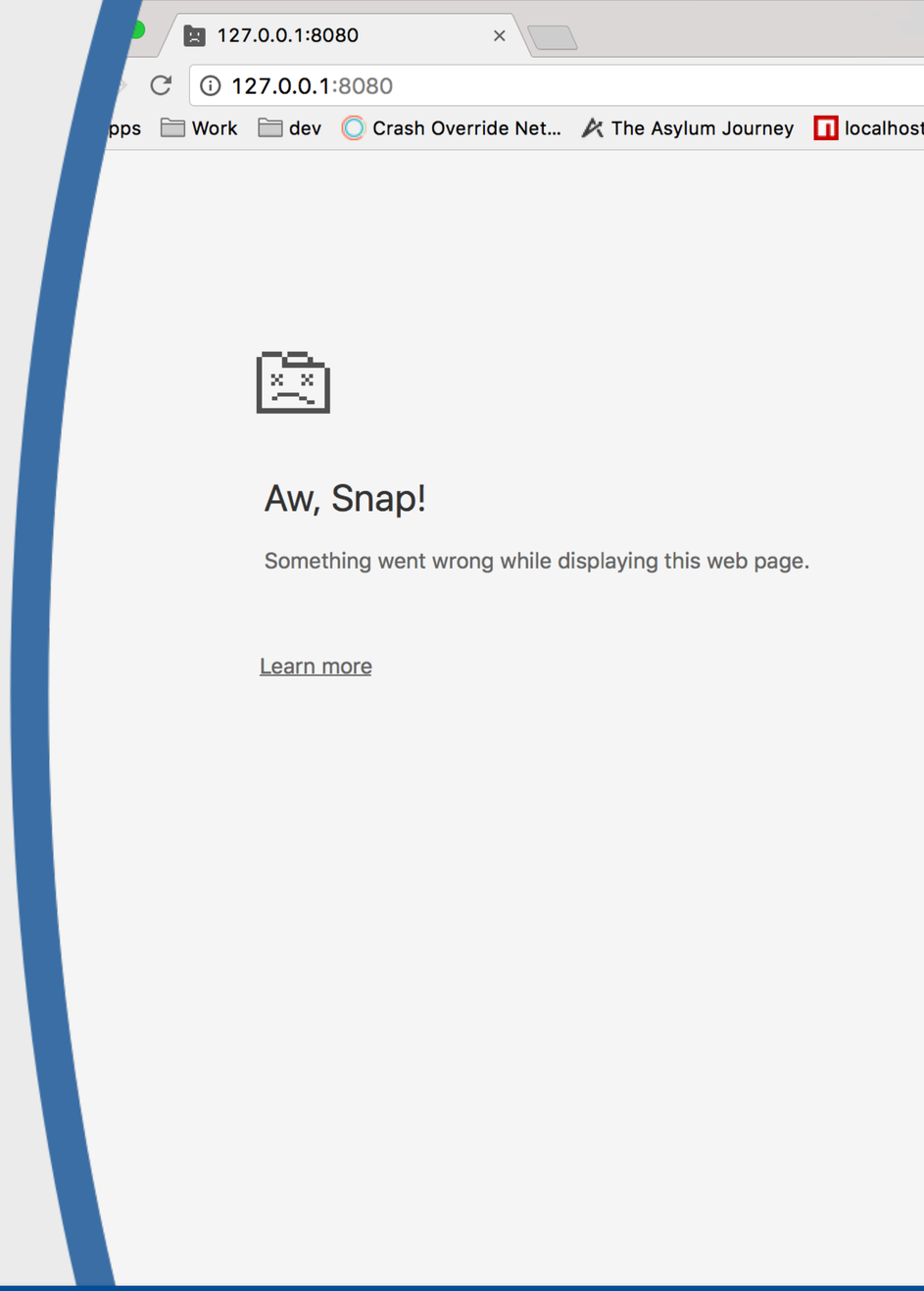
the major garbage collector

memory bloat
and
memory leaks

@katie_fenn

knowing you have a

problem

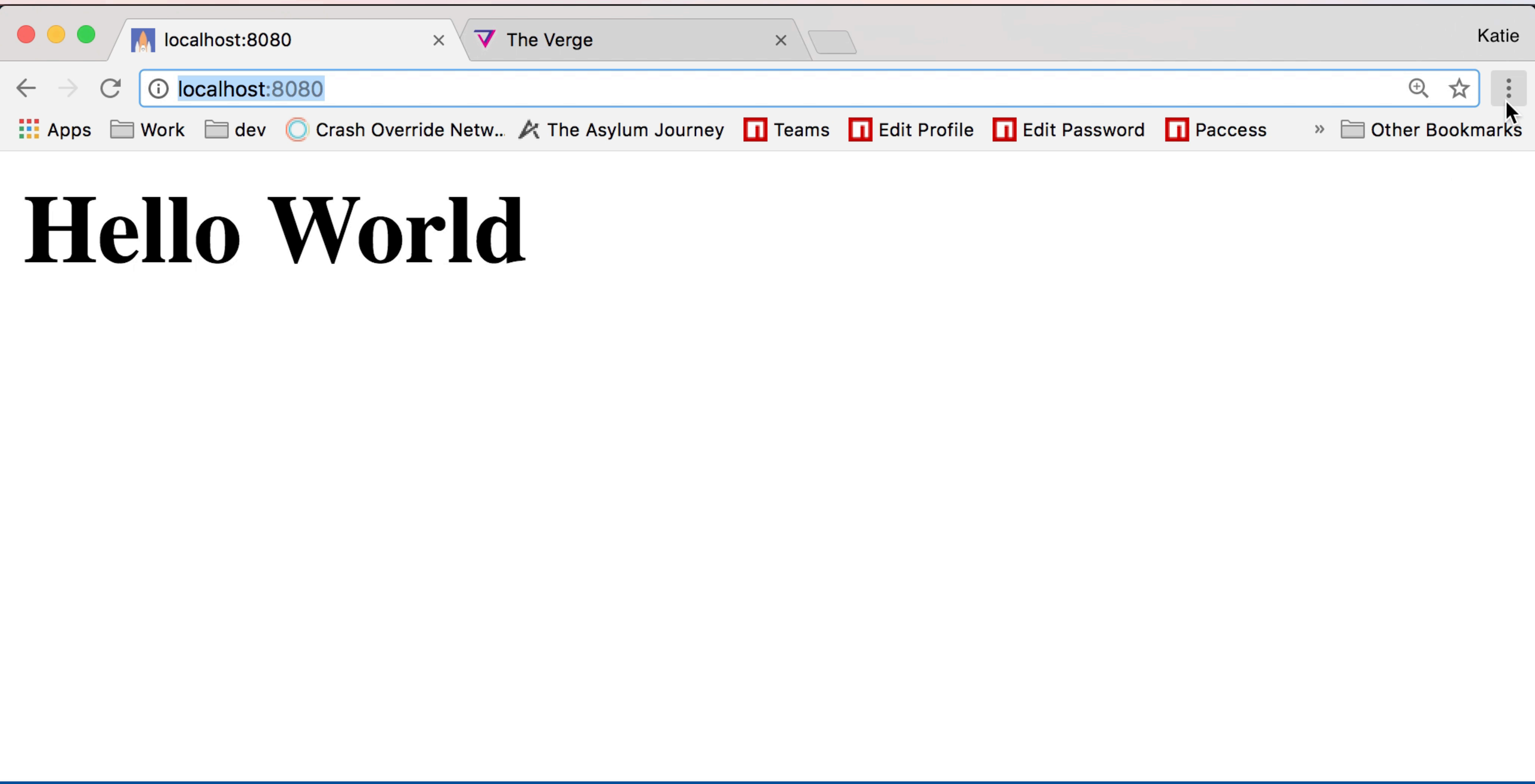


knowing you have a

problem

memory bloat:
making websites
unnecessarily large
for their purpose

cw: large animation

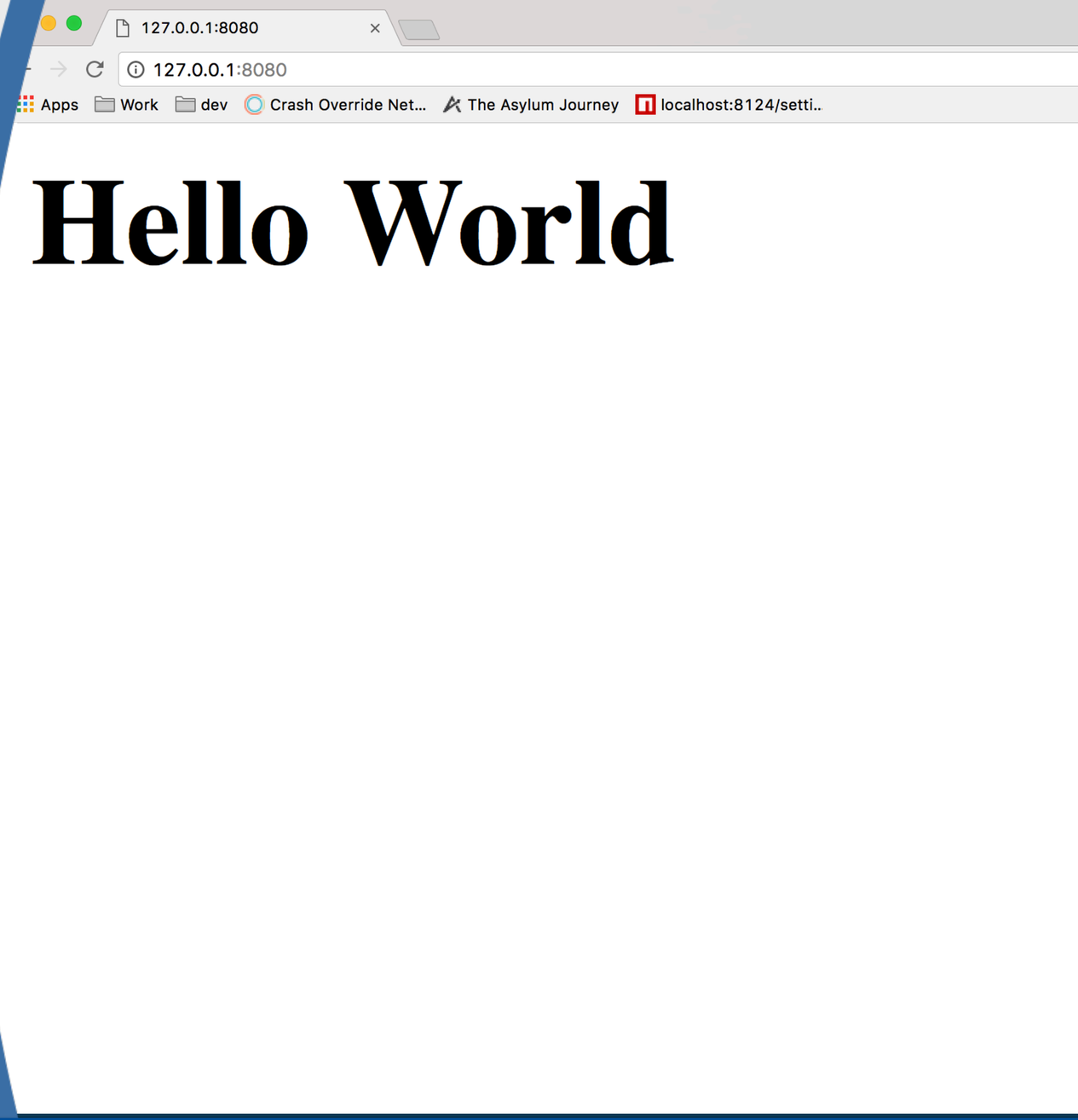

localhost:8080
The Verge
Katie
localhost:8080
Apps
Work
dev
Crash Override Netw..
The Asylum Journey
Teams
Edit Profile
Edit Password
Paccess
Other Bookmarks
Hello World

@katie_fenn

baseline

~100 MB

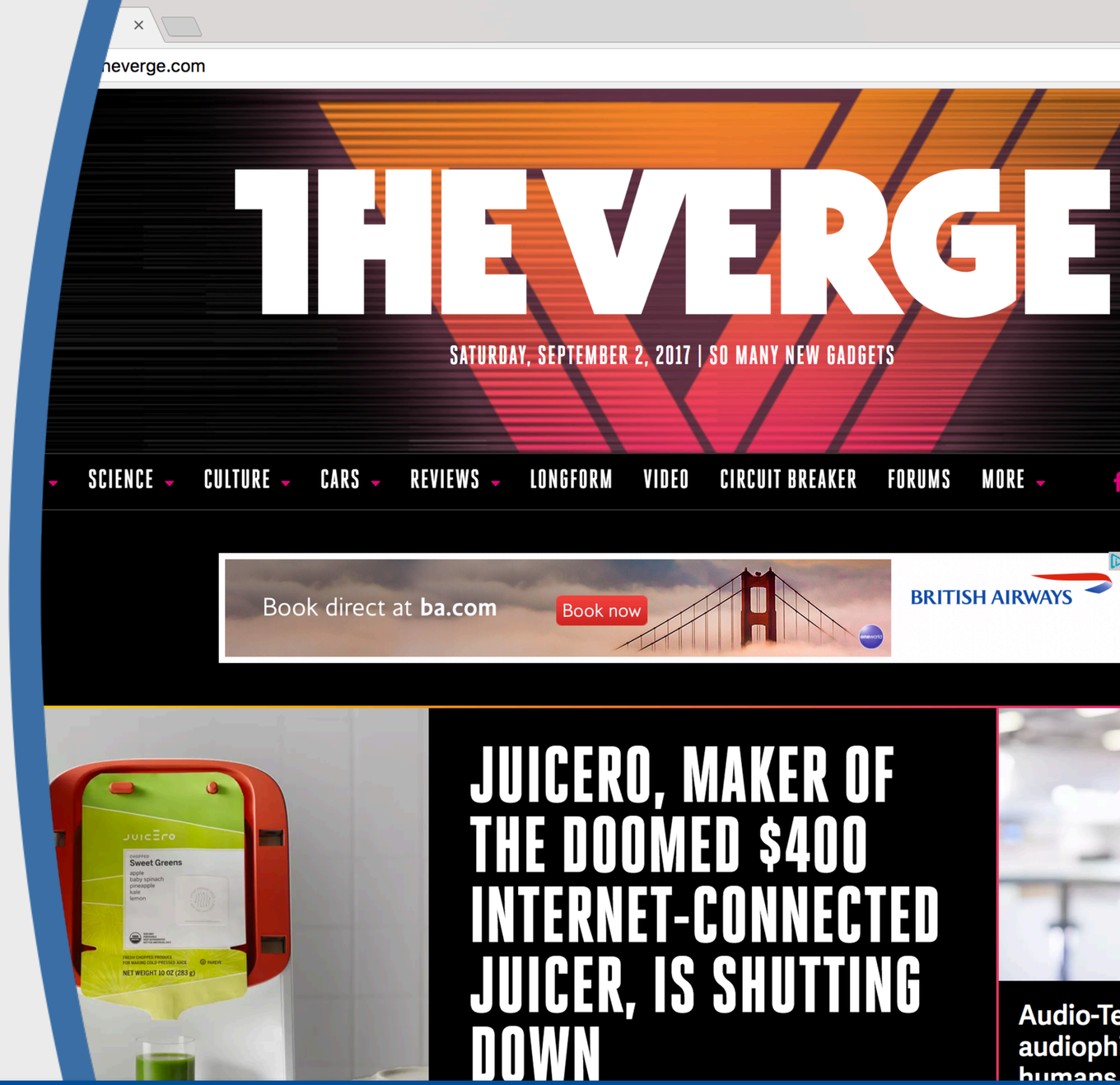
~3KB js



the verge

~200 MB

~100KB js



memory leaks:

when data does not die

cw: large animation

your
data

cw: large animation

you

live demo

summary

memory is a
finite resource

summary

javascript

manages memory

on our behalf

summary

the garbage collector

frees memory

after it is used

summary

bloat: websites that are unnecessarily large for their purpose

summary

memory leaks:
when unused memory
is not freed

summary

use devtools to
confirm problems

thank you

special thanks
@mathias
and the v8 gc team