Welcome to our Presentation

Oliver Jessner, Andreas Scheicher
University of Salzburg, department of Computer Sciences
Garbage Collection in V8
What is V8 and what is a Garbage Collector?

A Garbage Collector is a program that produces dangling pointers.

V8 is the JavaScript Engine of Node.js, in all variants of Chromium and Opera.
What is V8 and what is a Garbage Collector?
A Garbage Collector is a program that produces dangling pointers.

- V8 is the JavaScript Engine of Node.js, in all variants of Chromium and Opera.
- V8 is developed by Google and the Open Source Community.
What is V8 and what is a Garbage Collector?

A Garbage Collector is a program that produces dangling pointers.

- V8 is the JavaScript Engine of Node.js, in all variants of Chromium and Opera.
- V8 is developed by Google and the Open Source Community.
- A Garbage Collector is a program that provides automatic memory management.
Cost of execution time circa 5%
Modern Garbage Collectors

- Cost of execution time circa 5%
- Multithreading
Cost of execution time circa 5%
Multithreading
Mixed Strategies
How does a Object Graph look like?

```
root
  ↓
obj1
  ↓
obj3
  ↓
obj4
  ↓
obj5
  ↓
obj2
```
Now a dereference takes place

```
root

obj1

obj2

obj3

obj4

obj5
```

\[ \text{X} \]
The Objects can not be reached from the root node

- **root**
- **obj1**
- **obj2**
- **obj3**
- **obj4**
- **obj5**
How does a Simple Garbage Collector work?

```javascript
var new = function (ref){
    ref = allocate();

    if(ref === null){
        mark();
        sweep();
        ref = allocate();
    }

    if(ref === null)
        throw new outOfMemoryException();
}
```
```javascript
var mark = function (){
  var ref;
  for(var obj : heap){
    ref = obj.address;
    if(ref && ref.Unmarked){
      ref.mark();
      recursiveMark(ref);
    }
  }
}
```
var sweep = function (){
    foreach (var obj : heap){
        if(obj.isMarked)
            obj.UnMark();
        else
            free(obj);
    }
}
What kind of Garbage Collector are existing?

- Tracing Garbage Collection
What kind of Garbage Collector are existing?

- Tracing Garbage Collection
  - Mark and Sweep
  - Generational Garbage Collection
What kind of Garbage Collector are existing?

- Tracing Garbage Collection
  - 1. Mark and Sweep
  - 2. Generational Garbage Collection
- Reference counting
Which techniques are used in modern languages?

- Tracing Garbage Collection (JavaScript, Python, Ruby, Rust, C#)
  - 1. Mark and Sweep
  - 2. Generational Garbage Collection

- Reference counting (C++, PHP, Perl, Vala)
State of the art Generational Garbage Collection
Divides the heap into more heaps (V8 uses two generations)
Generational Garbage Collection
What is the difference?

- Divides the heap into more heaps (V8 uses two generations)
- More generations lead to less interruption times
Generational Garbage Collection
What is the difference?

- Divides the heap into more heaps (V8 uses two generations)
- More generations lead to less interruption times
- Mark and Copy/Semi Space
Generational Garbage Collection

What is the difference?

- Divides the heap into more heaps (V8 uses two generations)
- More generations lead to less interruption times
- Mark and Copy/Semi Space
- Mark and Compact
More generations less Stop-the-World interrupts

80% die young

20% survive
What are the differences between the Old- and Young-generation?

Young Generation

- Fast collection
What are the differences between the Old- and Young-generation?

Young Generation

- Fast collection
- Frequent collection
What are the differences between the Old- and Young-generation?

Young Generation

- Fast collection
- Frequent collection
- Needs more space
Why is the collection of the Young Generation faster?
Keyword, Mark and Copy

To Space

From Space
Why is the collection of the Young Generation faster?
Keyword, Mark and Copy

To Space

From Space
Why is the collection of the Young Generation faster?
Keyword, Mark and Copy

To Space

From Space
Why is the collection of the Young Generation faster?
Keyword, Mark and Copy

To Space

From Space
Why is the collection of the Young Generation faster?
Keyword, Mark and Copy

To Space

From Space
Why is the collection of the Young Generation faster?

Keyword, Mark and Copy

To Space

From Space
Now the mark phase starts
Keyword, Mark and Copy

To Space

From Space
Now the mark phase starts
Keyword, Mark and Copy
Delete toSpace
Keyword, Mark and Copy

To Space

From Space
Delete fromSpace
Keyword, Mark and Copy
What are the differences between the Old- and Young-generation?

Old Generation
- Slower collection
What are the differences between the Old- and Young-generation?

**Old Generation**

- Slower collection
- Infrequent collection
What are the differences between the Old- and Young-generation?

Old Generation
- Slower collection
- Infrequent collection
- Needs less space
Keyword Mark and Compact

free

live
Keyword Mark and Compact

\[ \text{free} \quad \text{live} \]
Keyword Mark and Compact

free live
There is one big flaw with Generations

Intergenerational References

Young Generation

yObj

Old Generation

array[... , yObj]
Now the Young Generation gets collected
We have a null reference in the Old Generation

Young Generation
null

Old Generation
array[..., yObj]
There is one big flaw with Generations

Intergenerational References

Young Generation

null

Old Generation

array[..., yObj], yObj
Any further questions?