Google's JavaScript Engine: V8

Thomas Hütter (1120239)
Mario Preishuber (1120643)

Fachbereich Computerwissenschaften
Naturwissenschaftliche Fakultät

UNIVERSITÄT SALZBURG

24. Februar 2014

Inhalt

Was ist JavaScript?

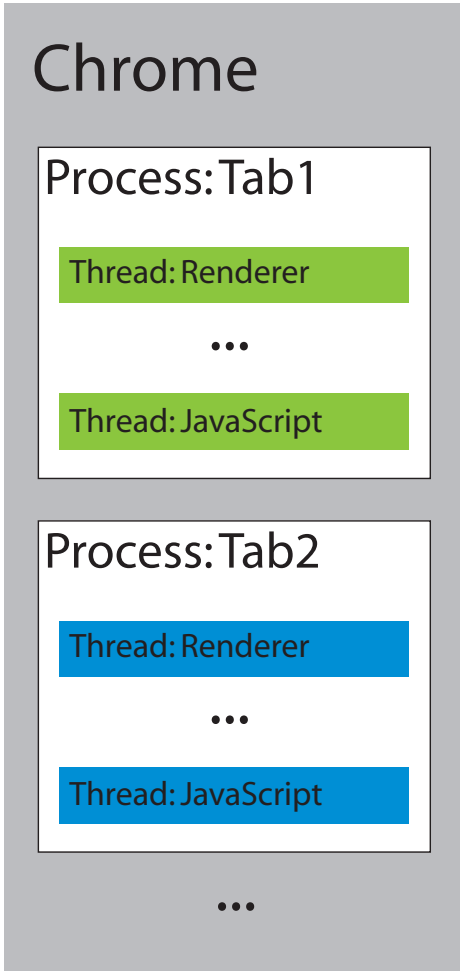
- Web Browser
- Client-seitig
- Dynamische Gestaltung von Webseiten
- JavaScript $\neq$ Java

Eigenschaften

- JavaScript ist extrem dynamisch
- Beispiel Objekte
  - Attribute können zur Laufzeit hinzgefügt oder entfernt werden
  - Sind Hash Maps

Was ist eine JavaScript Engine?

- Browser wie eigenständiges Betriebssystem
- Chrome
  - Jeder Tab ist ein eigener Prozess
  - Jeder Prozess verfügt über einen
- Beispiele:
  - V8 (Google Chrome)
  - Spidermonkey (Mozilla Firefox)
  - Chakra (Microsoft IE9)
  - JavaScriptCore (Apple Safari)



Google's

Hidden classes

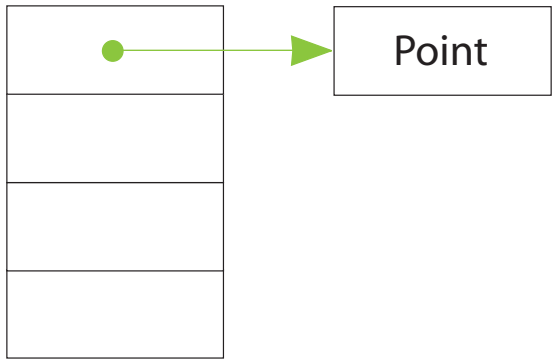
- Keine Typen zur Compiletime
- Ohne Typ Information sind Optimierungen schwer zu realisieren
- V8 Lösung: Hidden classes
  - Zur Laufzeit erzeugt
  - Objekte mit derselben hidden class können optimiert werden

Beispiel - hidden classes

```
1 function Point(x,y) {
2       this.x = x;
3       this.y = y;
4 }
5
6 var p1 = new Point(1,2);
7 var p2 = new Point(3,4);
8
9 p2.z = 5;
```

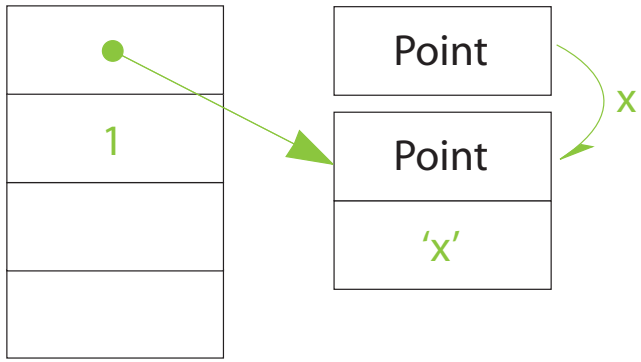
BEISPIEL - HIDDEN CLASSES

```
1  function Point(x,y) {
2        this.x = x;
3        this.y = y;
4  }
5
6  var p1 = new Point(1,2);
7  var p2 = new Point(3,4);
8
9  p2.z = 5;
```



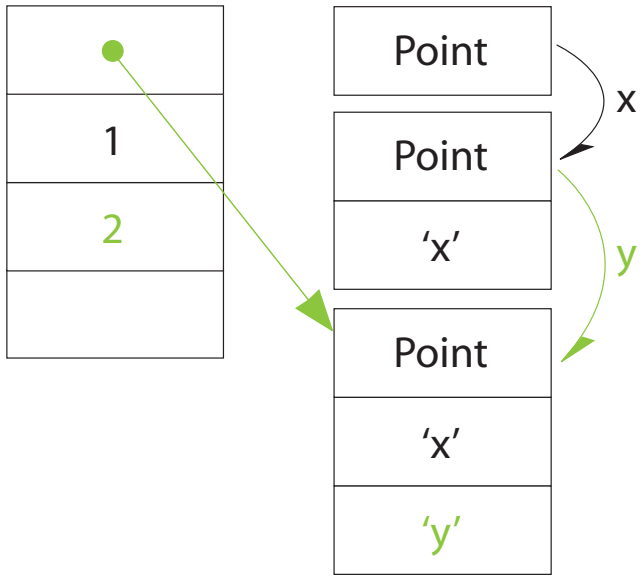
BEISPIEL - HIDDEN CLASSES

```
1  function Point(x,y) {
2        this.x = x;
3        this.y = y;
4  }
5
6  var p1 = new Point(1,2);
7  var p2 = new Point(3,4);
8
9  p2.z = 5;
```



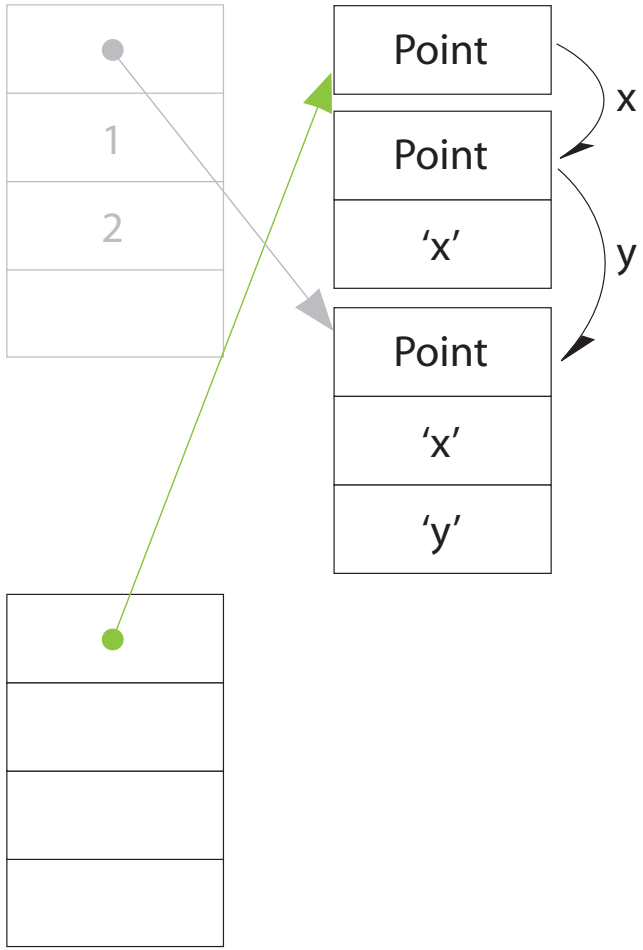
BEISPIEL - HIDDEN CLASSES

```
1  function Point(x,y) {
2        this.x = x;
3        this.y = y;
4  }
5
6  var p1 = new Point(1,2);
7  var p2 = new Point(3,4);
8
9  p2.z = 5;
```



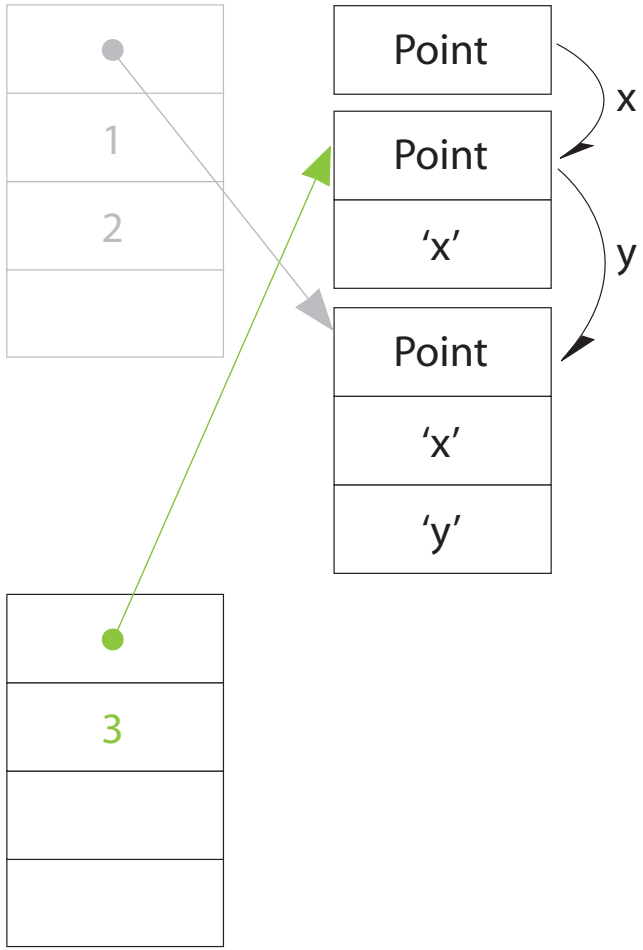
BEISPIEL - HIDDEN CLASSES

```
1  function Point(x,y) {
2        this.x = x;
3        this.y = y;
4  }
5
6  var p1 = new Point(1,2);
7  var p2 = new Point(3,4);
8
9  p2.z = 5;
```



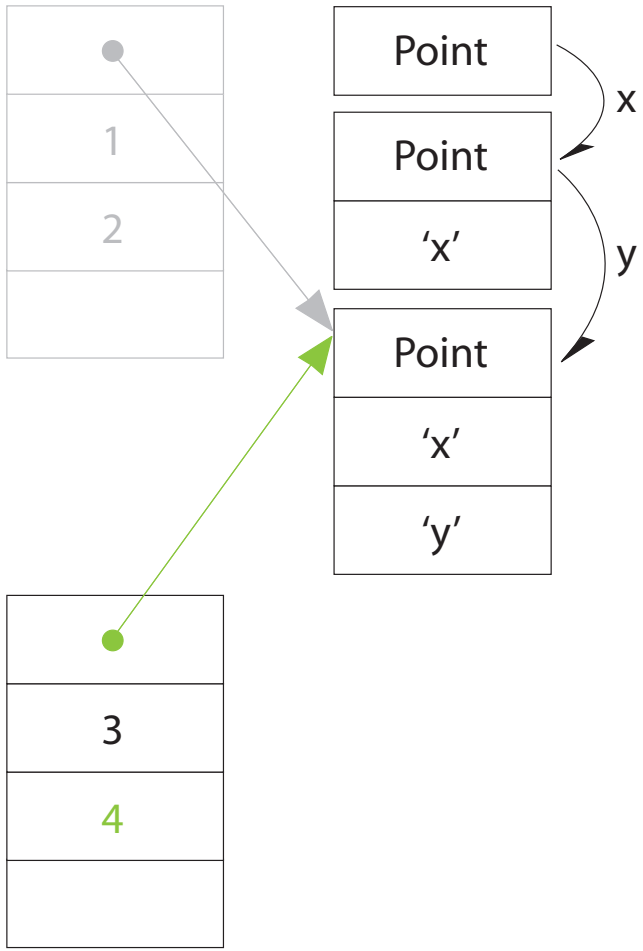
BEISPIEL - HIDDEN CLASSES

```
1  function Point(x,y) {
2        this.x = x;
3        this.y = y;
4  }
5
6  var p1 = new Point(1,2);
7  var p2 = new Point(3,4);
8
9  p2.z = 5;
```



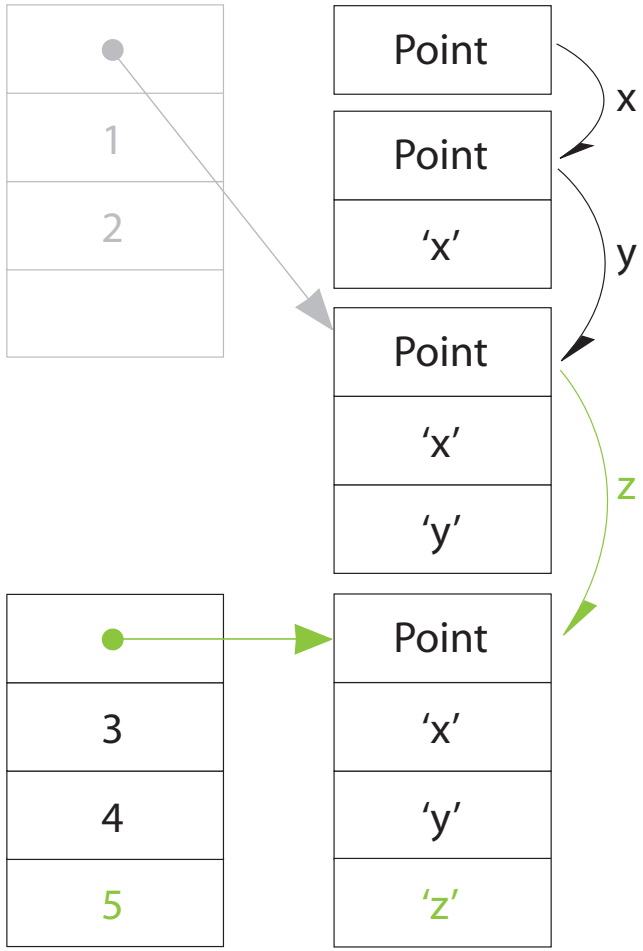
Beispiel - Hidden Classes

```
1  function Point(x,y) {
2        this.x = x;
3        this.y = y;
4  }
5
6  var p1 = new Point(1,2);
7  var p2 = new Point(3,4);
8
9  p2.z = 5;
```



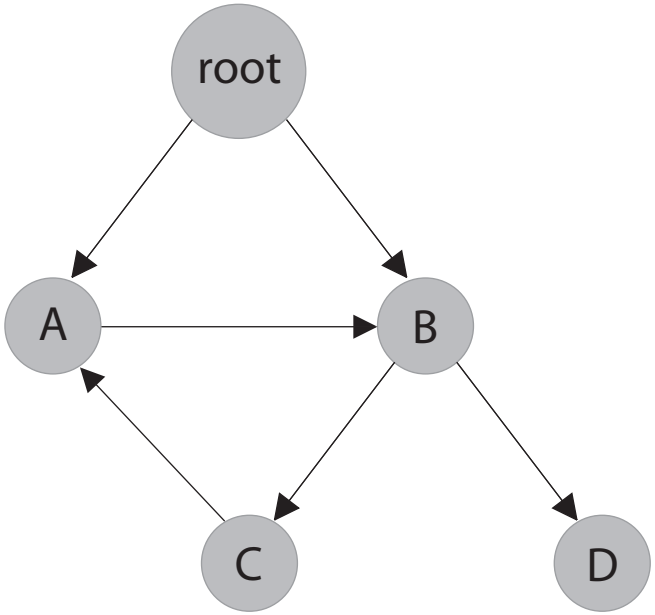
BEISPIEL - HIDDEN CLASSES

```
1  function Point(x,y) {
2        this.x = x;
3        this.y = y;
4  }
5
6  var p1 = new Point(1,2);
7  var p2 = new Point(3,4);
8
9  p2.z = 5;
```



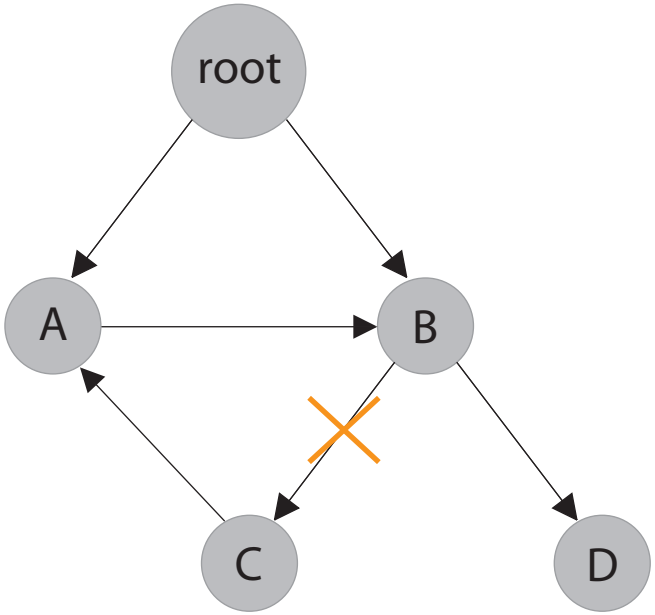
Garbage Collector

- Zwischen live Memory und dead Memory unterscheiden
- dead Memory deallozieren



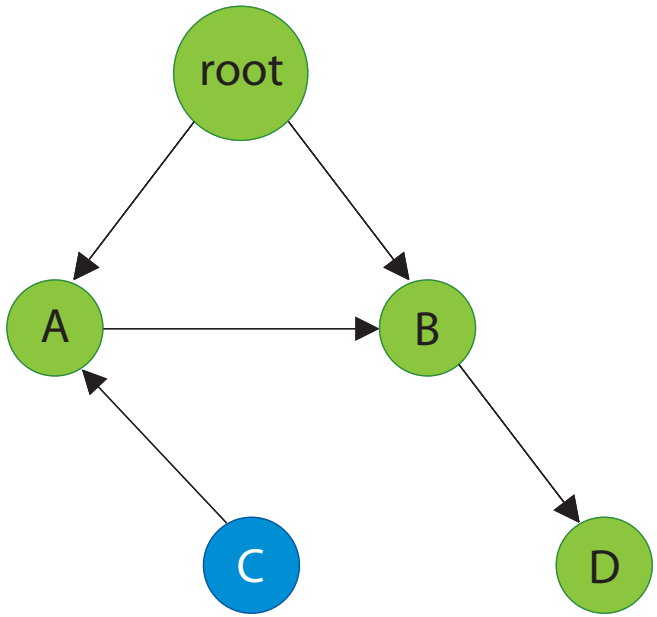
Garbage Collector

- Zwischen live Memory und dead Memory unterscheiden
- dead Memory deallozieren

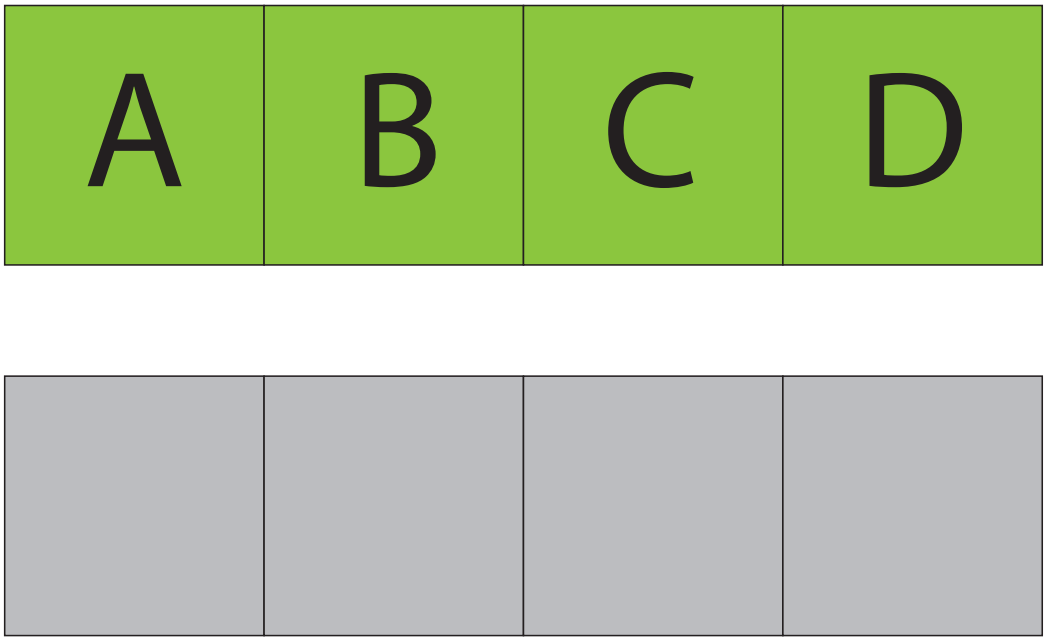


Garbage Collector

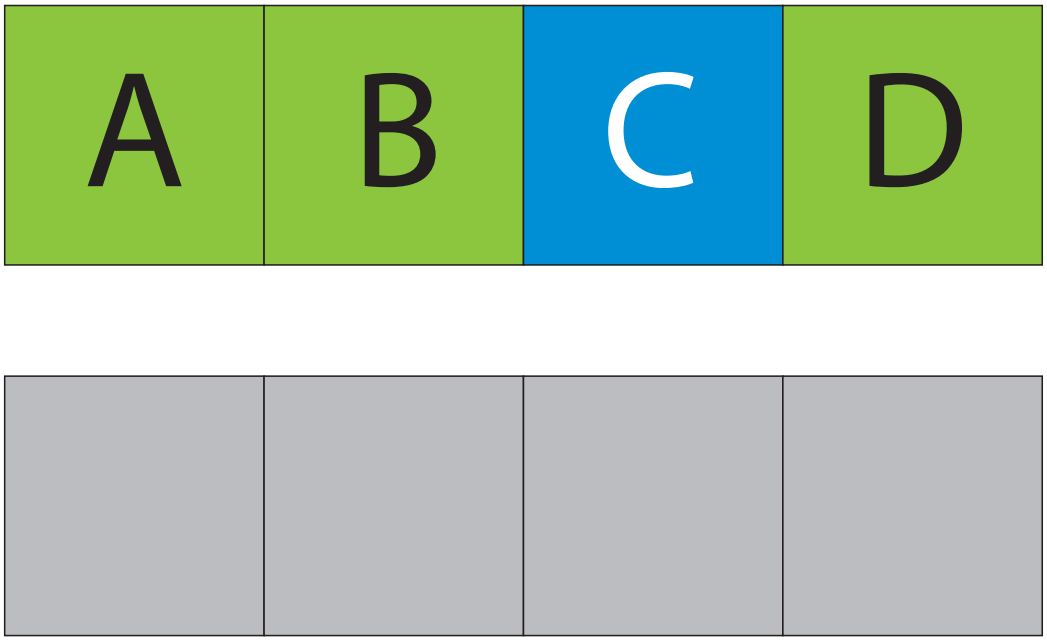
- Zwischen live Memory und dead Memory unterscheiden
- dead Memory deallozieren



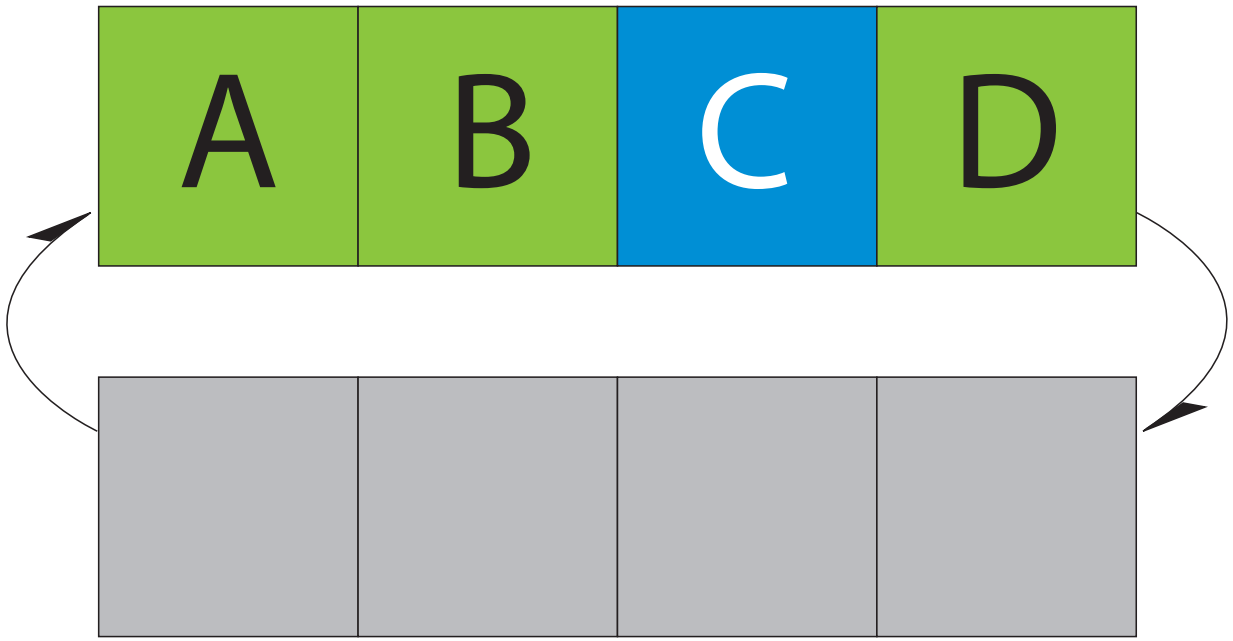
Mark-and-sweep



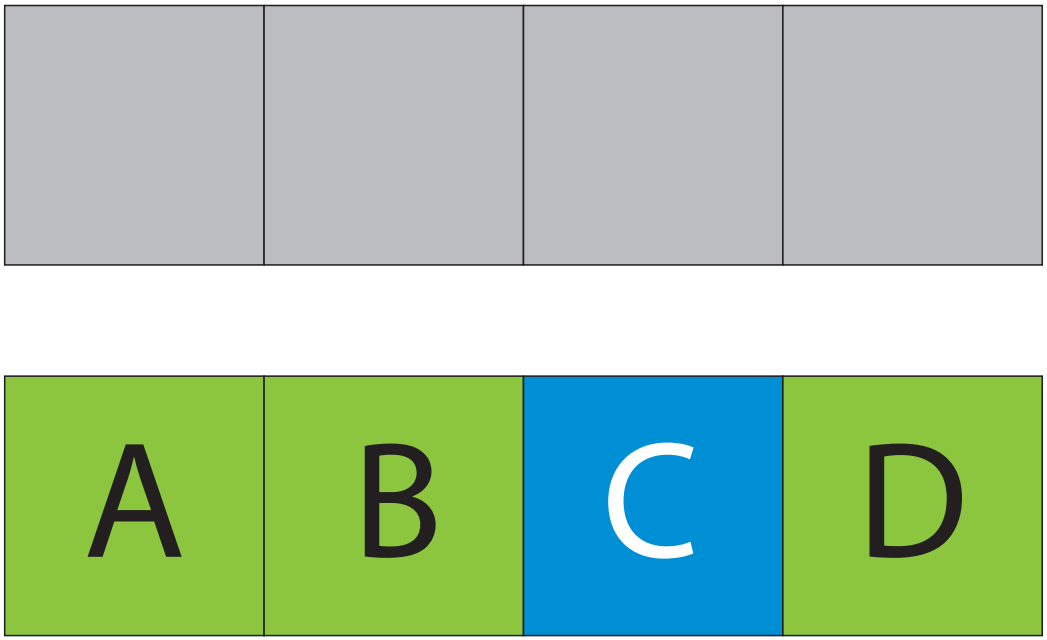
Mark-and-sweep



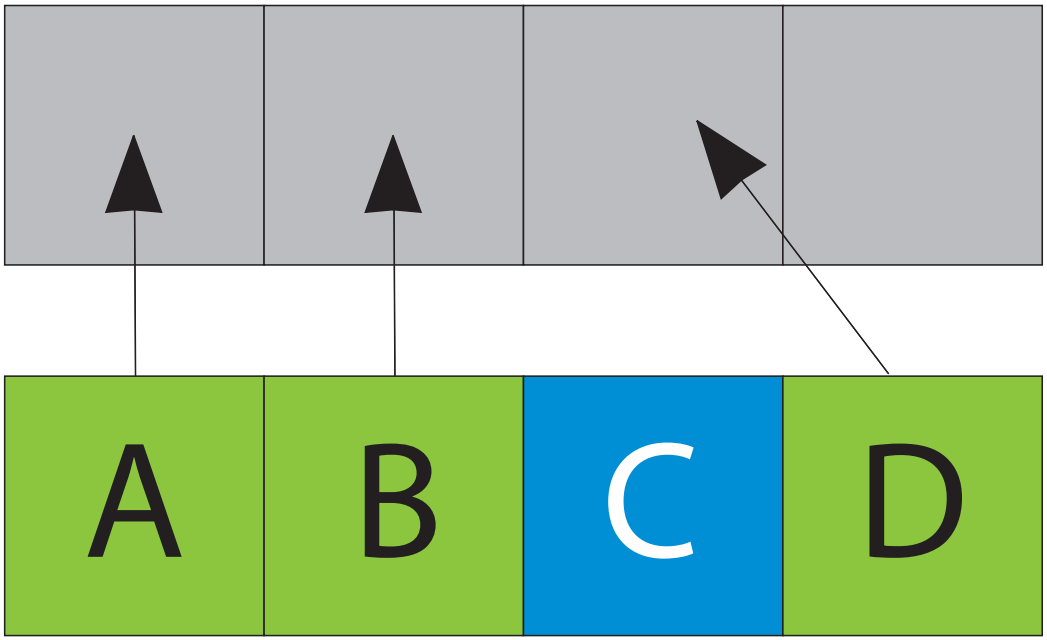
Mark-and-sweep



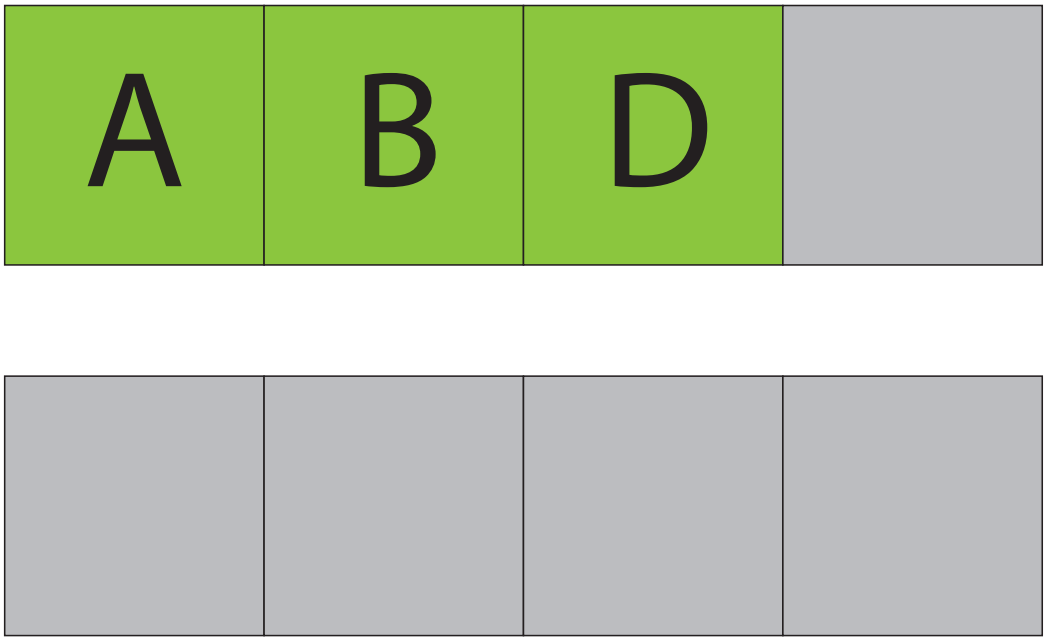
Mark-and-sweep



Mark-and-sweep



Mark-and-sweep



Generational garbage collection

- Kurzlebige Objekte
- Häufige Garbage Collection

Generational garbage collection

- Kurzlebige Objekte
- Häufige Garbage Collection

- Langlebige Objekte
- Seltene Garbage Collection

Generational garbage collection

Young generation

- Kurzlebige Objekte
- Häufige Garbage Collection

- Langlebige Objekte
- Seltene Garbage Collection

Generational garbage collection

Young generation

- Kurzlebige Objekte
- Häufige Garbage Collection

Old generation

- Langlebige Objekte
- Seltene Garbage Collection

Compiler

- V8 verfügt über 2 Compiler
  - Full Compiler: erzeugt guten Code
  - Optimizing Compiler: erzeugt wirklich guten Code

Full Compiler

- Erzeugt schnell guten Code
- Trifft zur Laufzeit Annahmen bezüglich des Typs
- Benutzt Inline Caches (ICs)
  - Liefert Typ-Informationen zur Laufzeit des Programms

Der Optimizing Compiler

- V8 compiliert Hot Functions mit dem Optimizing Compiler
- Dazu sind Typ-Informationen nötig
  - ICs liefern diese Informationen
  - Operationen werden spekulativ optimiert
- Konstrukte wie zum Beispiel `try { } catch { }` können nicht oder nur schwer optimiert werden.

Danke, für die Aufmerksamkeit!

Fragen?