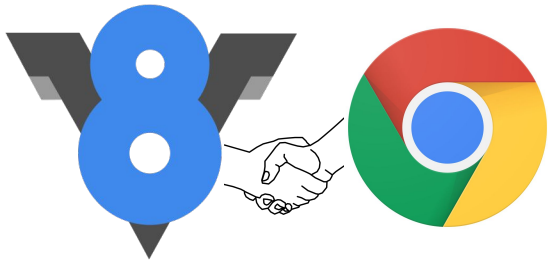
Garbage Collection as a Joint Venture



Michael Lippautz

Ulan Degenbaev, Jochen Eisinger, Kentaro Hara, Marcel Hlopko, Hannes Payer
Google Inc

MoreVMs Workshop, Brussels, 2017

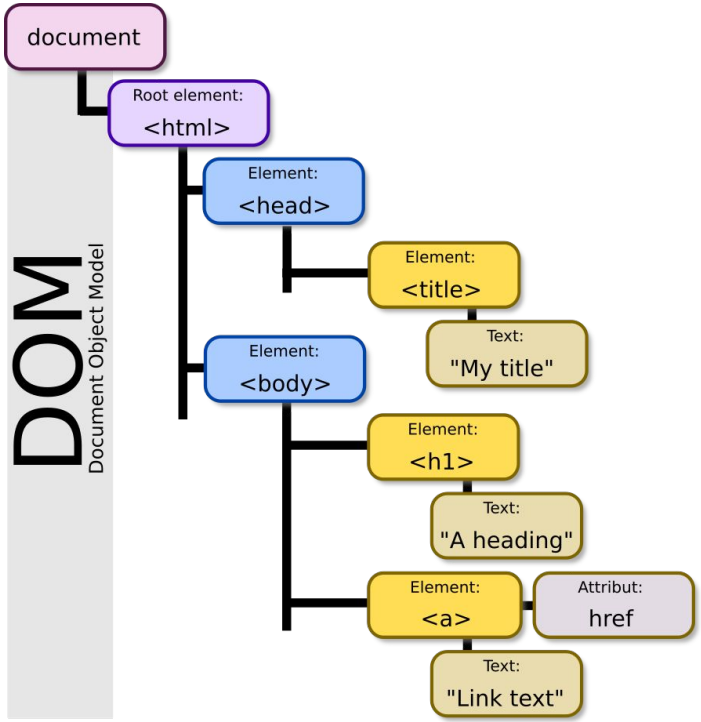
Contents

Contents

- SPOILER: Enabled in Chrome M57 -

The Web, JavaScript, and Chrome

Web - JavaScript - Chrome

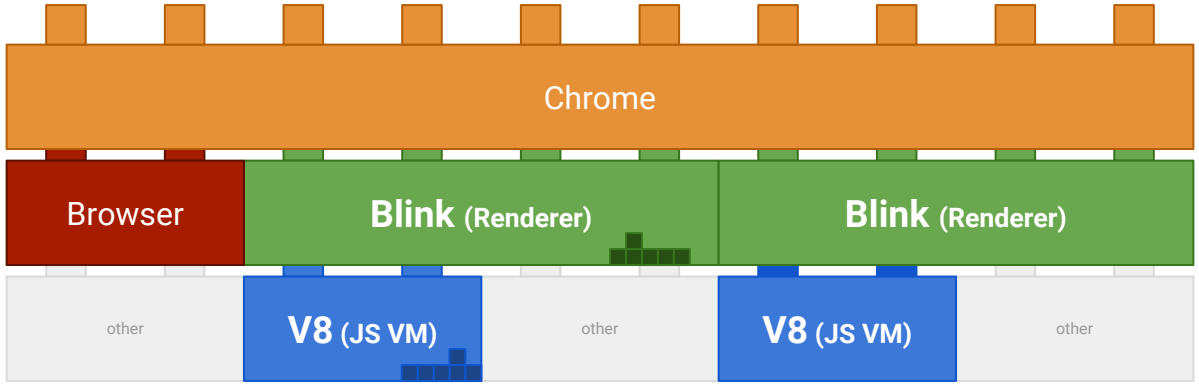


- Document Object Model (DOM)
- Cross-platform language-independent representation of HTML
- Web Interface Definition Language (IDL)
- Traditionally encoded as objects in the rendering engine (e.g. Blink)

Web - JavaScript - Chrome

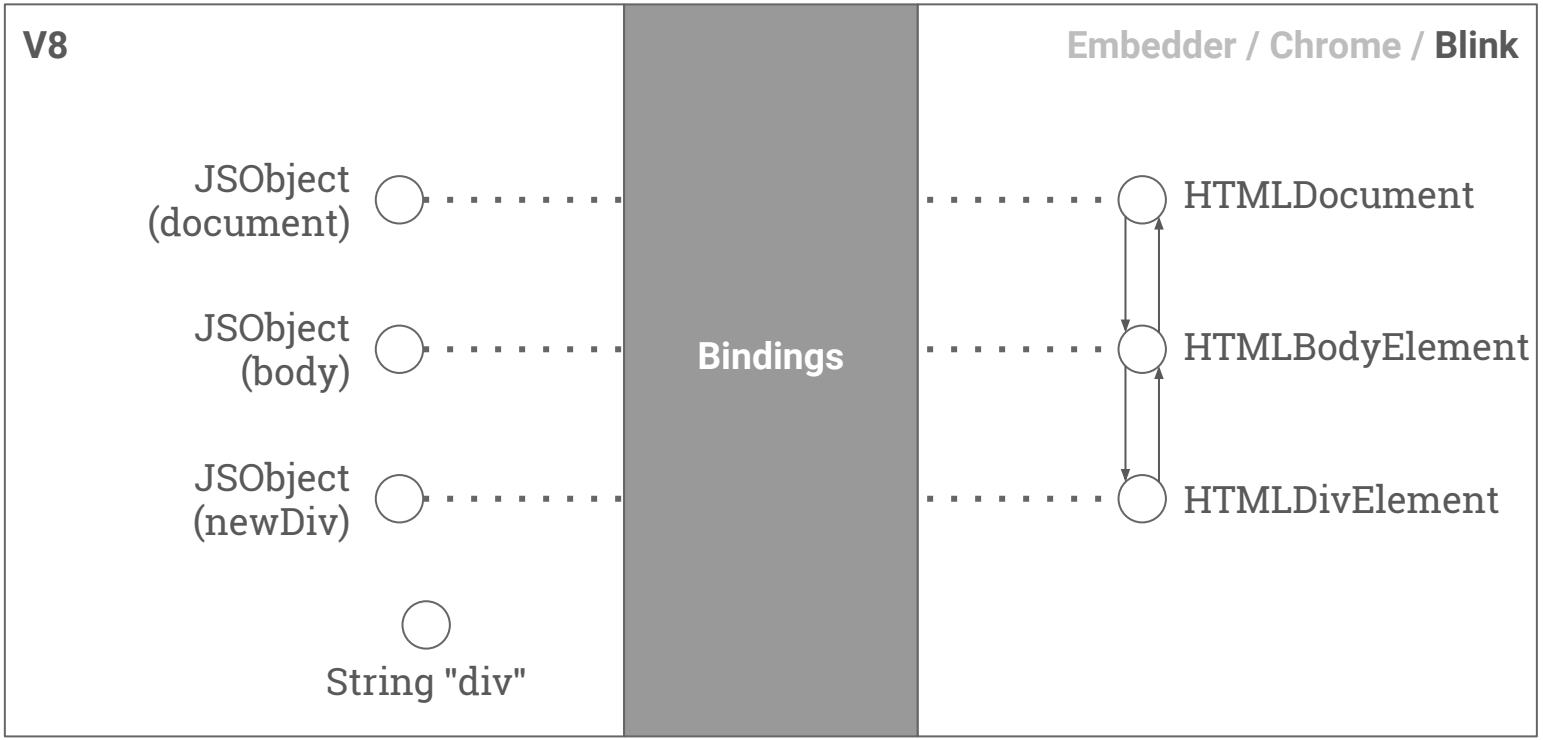
```
<script type="text/javascript">
  var newDiv;
  function createDiv() {
    newDiv = document.createElement("div");
    document.body.appendChild(newDiv);
  }

  window.onload = createDiv;
</script>
```

Web - JavaScript - Chrome at runtime



Idea: Abstraction of the browser into components
(and most of the time subcomponents)

Web - JavaScript - Chrome



Bindings: Gluing together V8 and Blink

V8

- Managed heap for JavaScript objects
- Dynamic object layout (shape changes)
- Precise GC
- Mark-Sweep-Compact Collector
  - Incremental marking
  - Concurrent sweeping
  - Parallel compaction

Blink

- Managed heap for (most of) the DOM
- Static object layout
- Precise and conservative GC (until Q1’16 ref counting)
- Stop-the-world Mark-Sweep(-Compact) Collector
  - Incremental sweeping
  - (Compaction on backing stores of collections)

Bindings: Gluing together V8 and Blink

V8

- Managed heap for JavaScript objects
- Dynamic object layout (shape changes)
- Precise GC
- Mark-
  -
  -
  - Parallel compaction

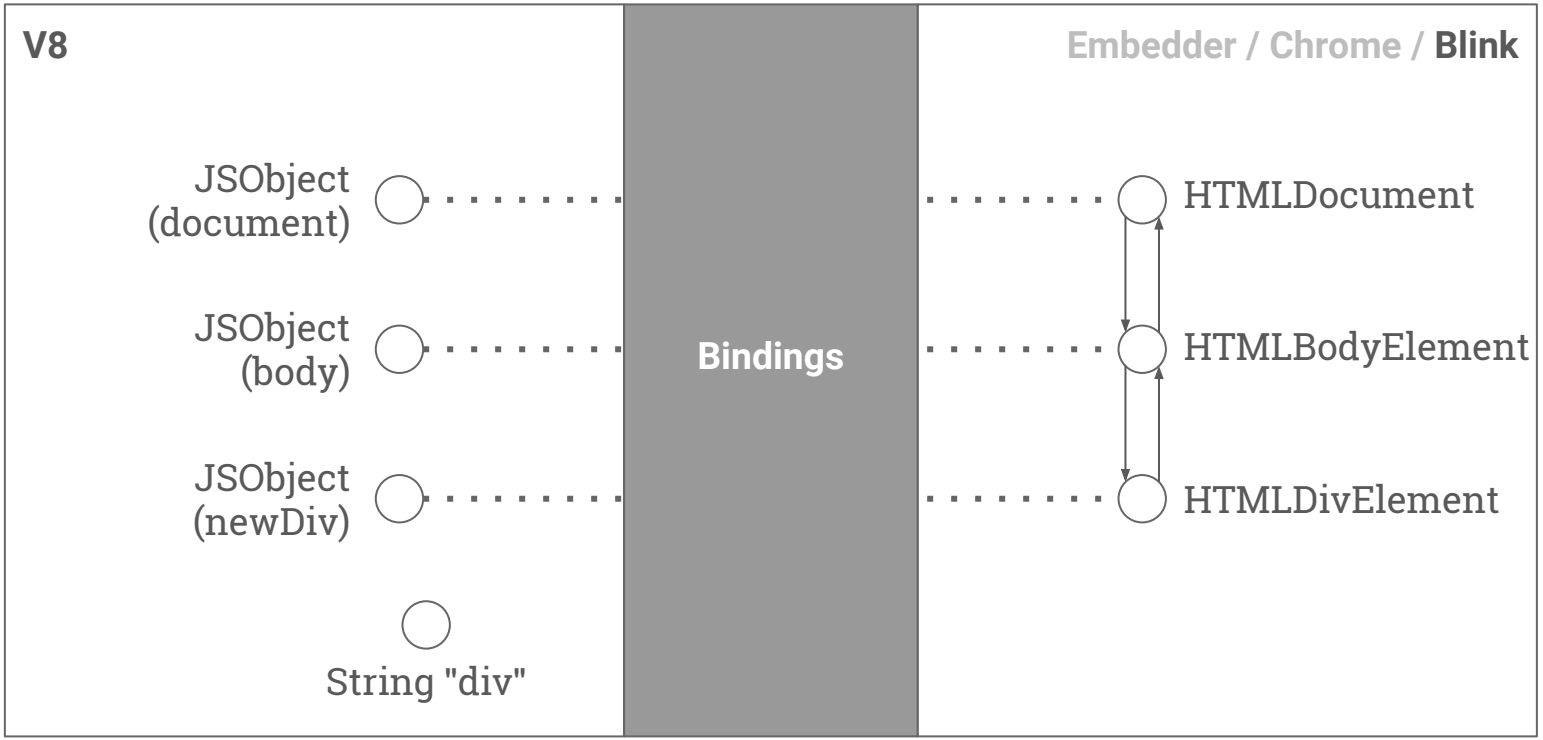
Blink

- Managed heap for (most of) the DOM
- Static object layout
- Precise and conservative GC (until Q1’16 ref
  - npact)
  - Incremental sweeping
  - (Compaction on backing stores of collections)

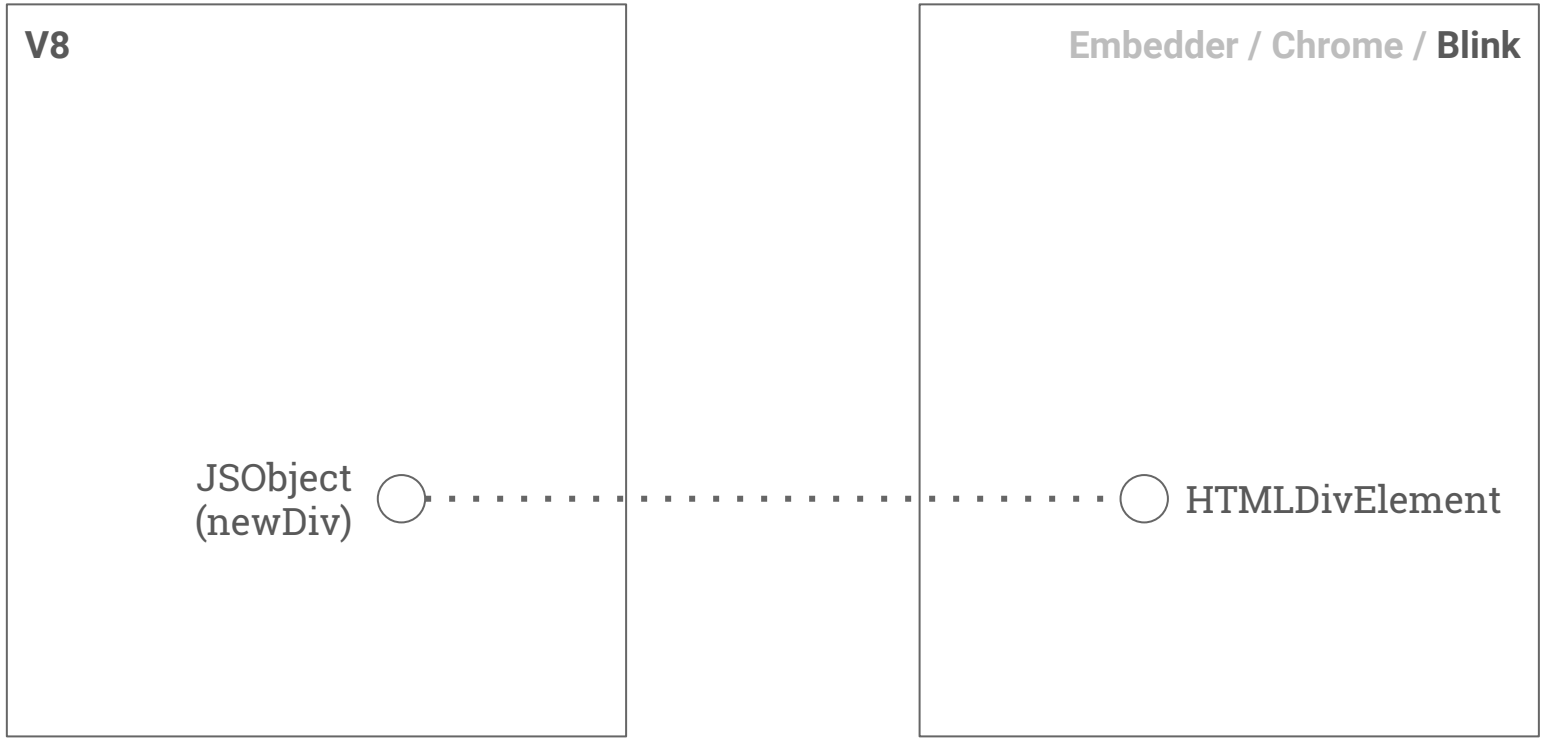
Two very different systems communicating with each other!

The Cycle Problem

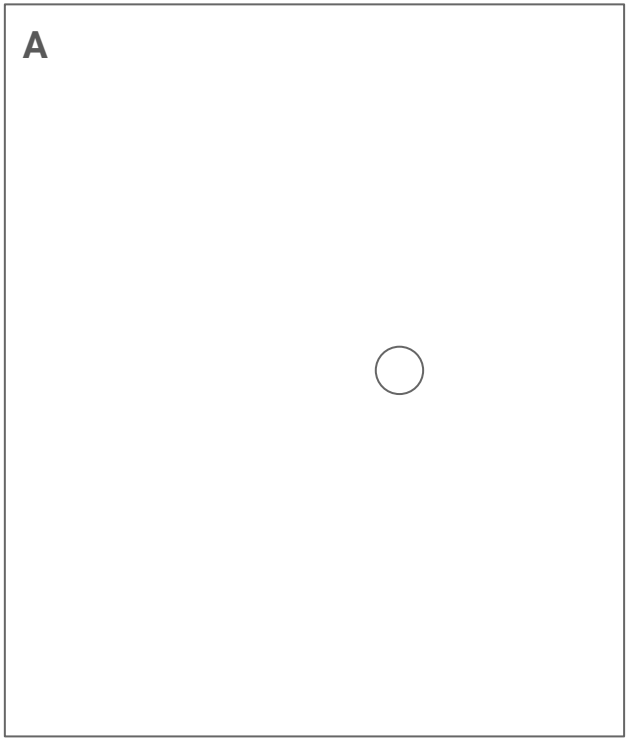
The Cycle Problem

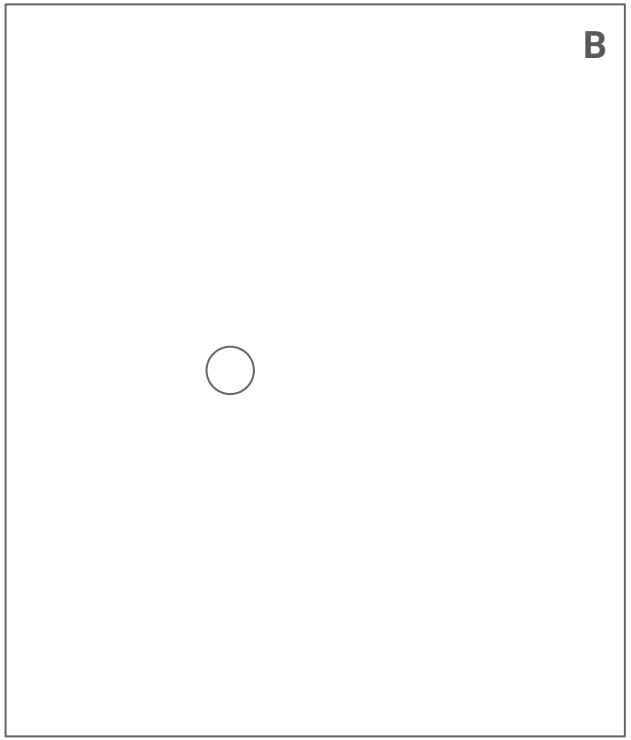


The Cycle Problem

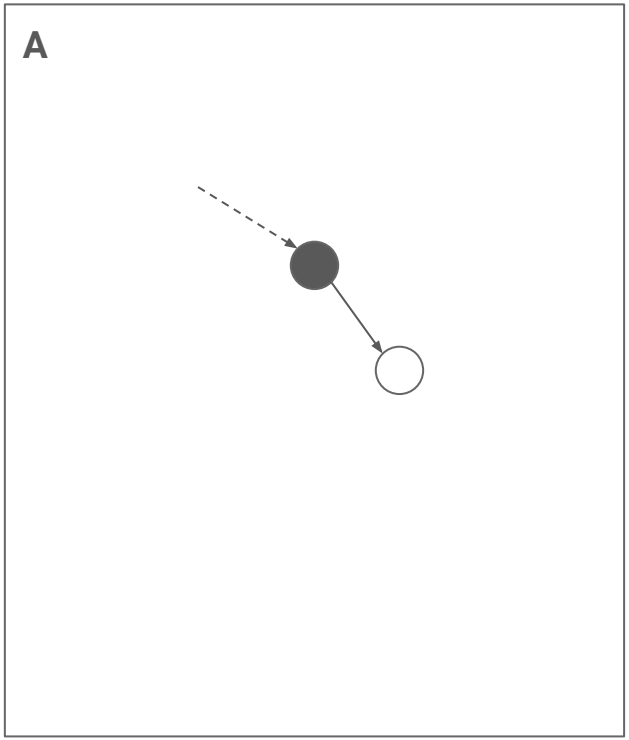


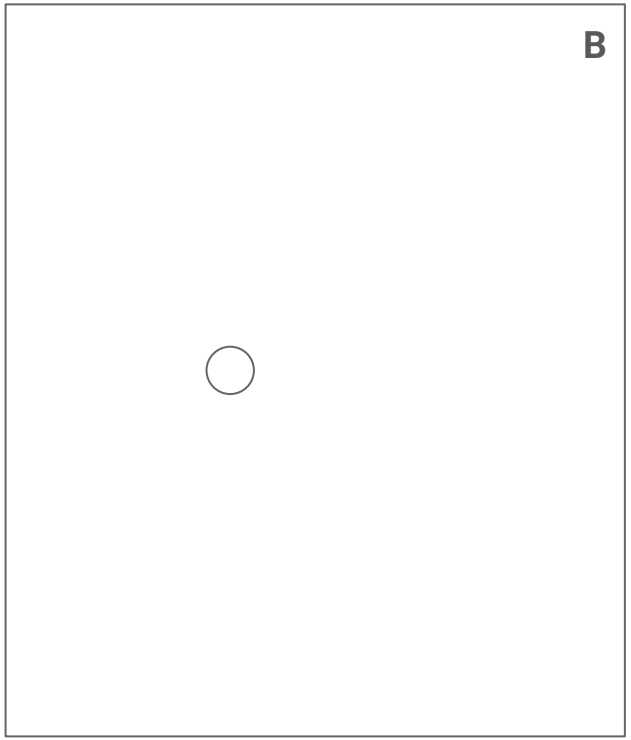
The Cycle Problem



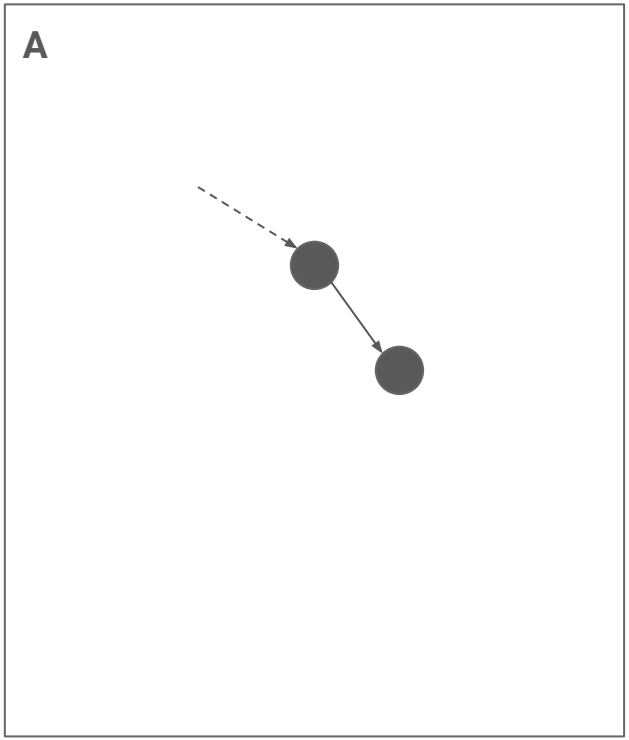


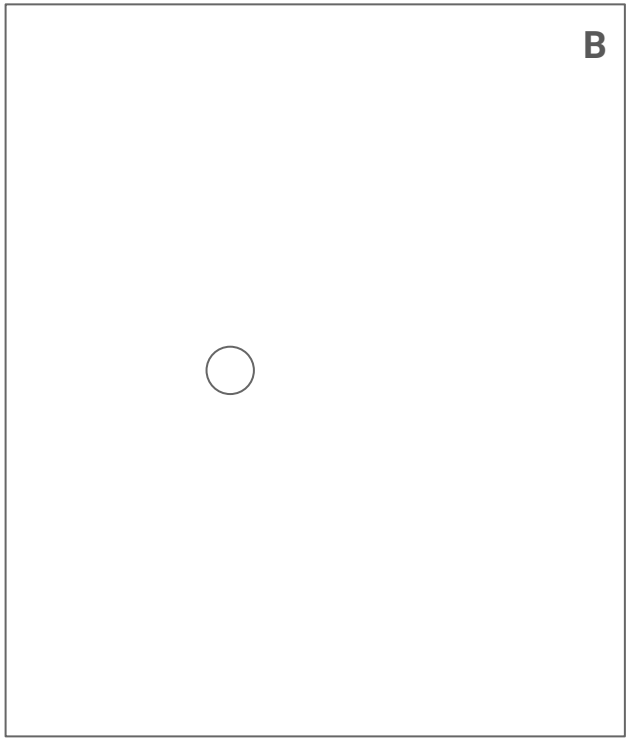
The Cycle Problem



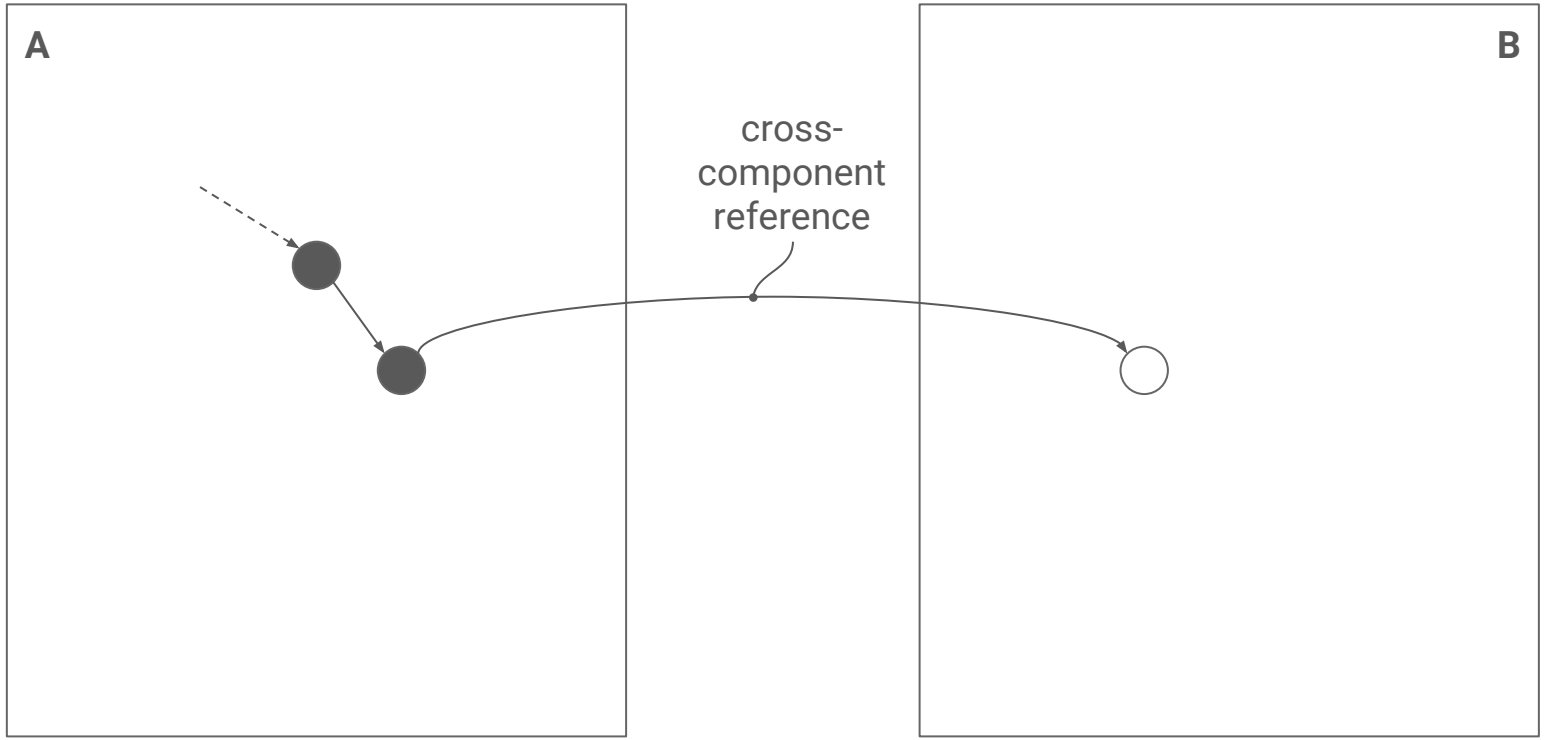


The Cycle Problem

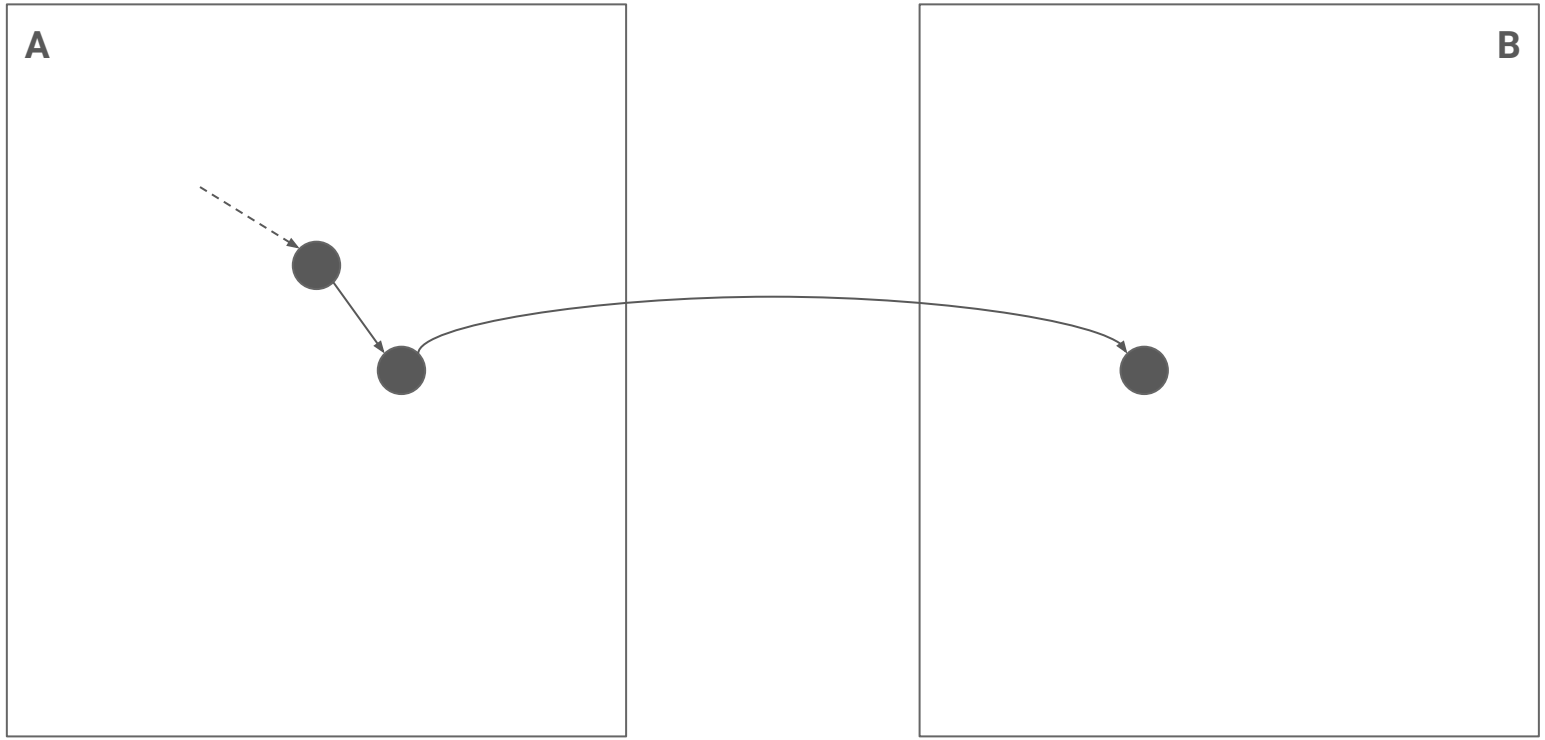




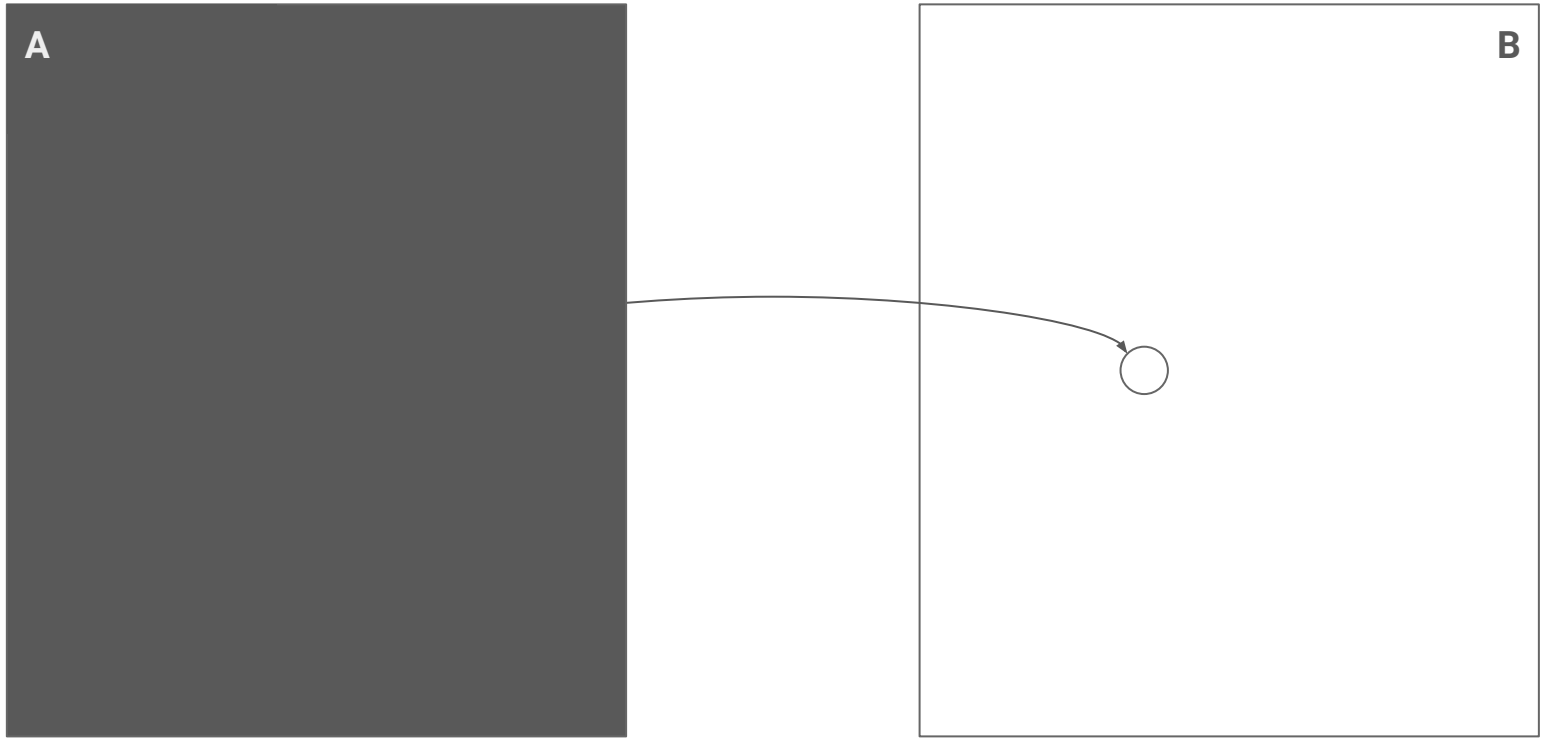
The Cycle Problem



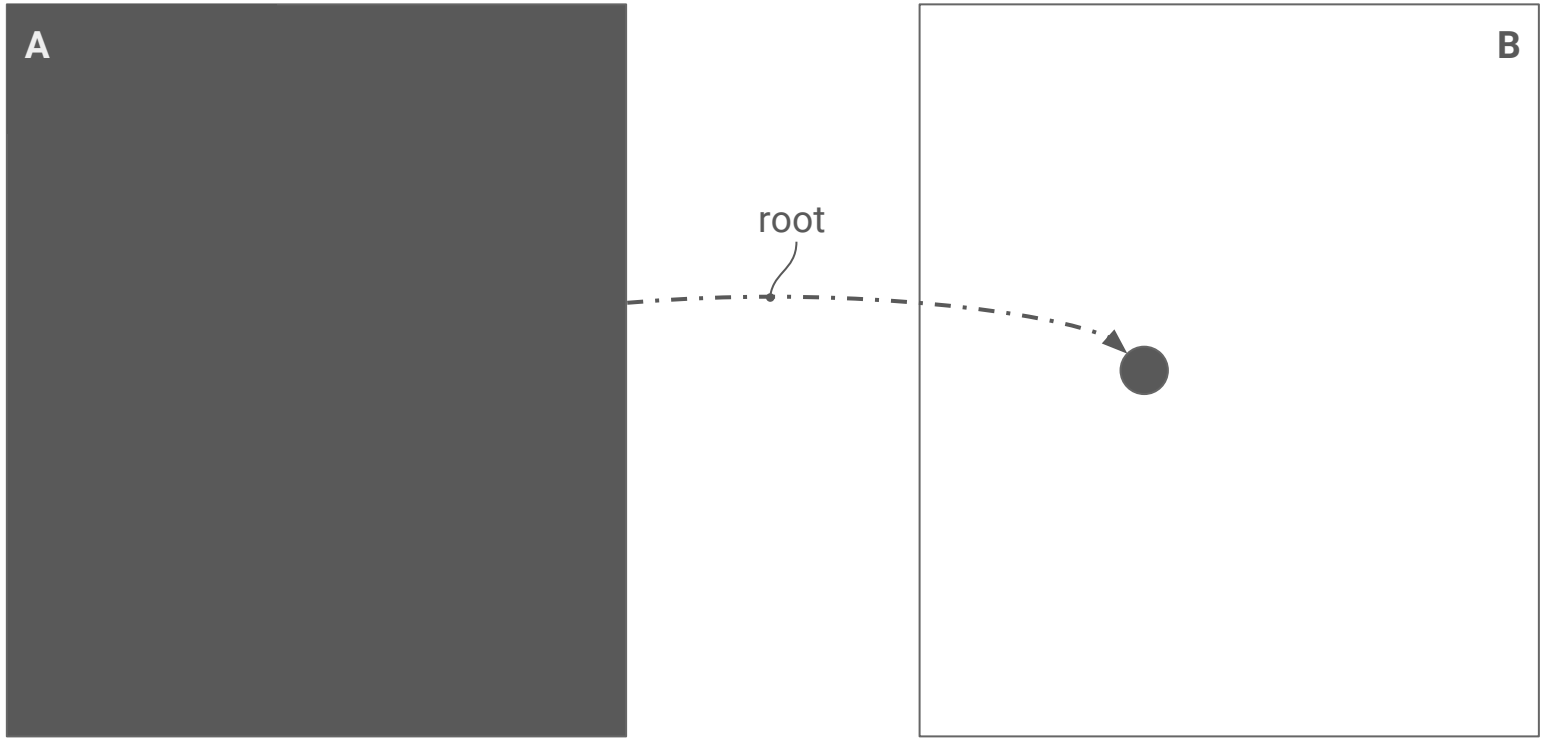
The Cycle Problem



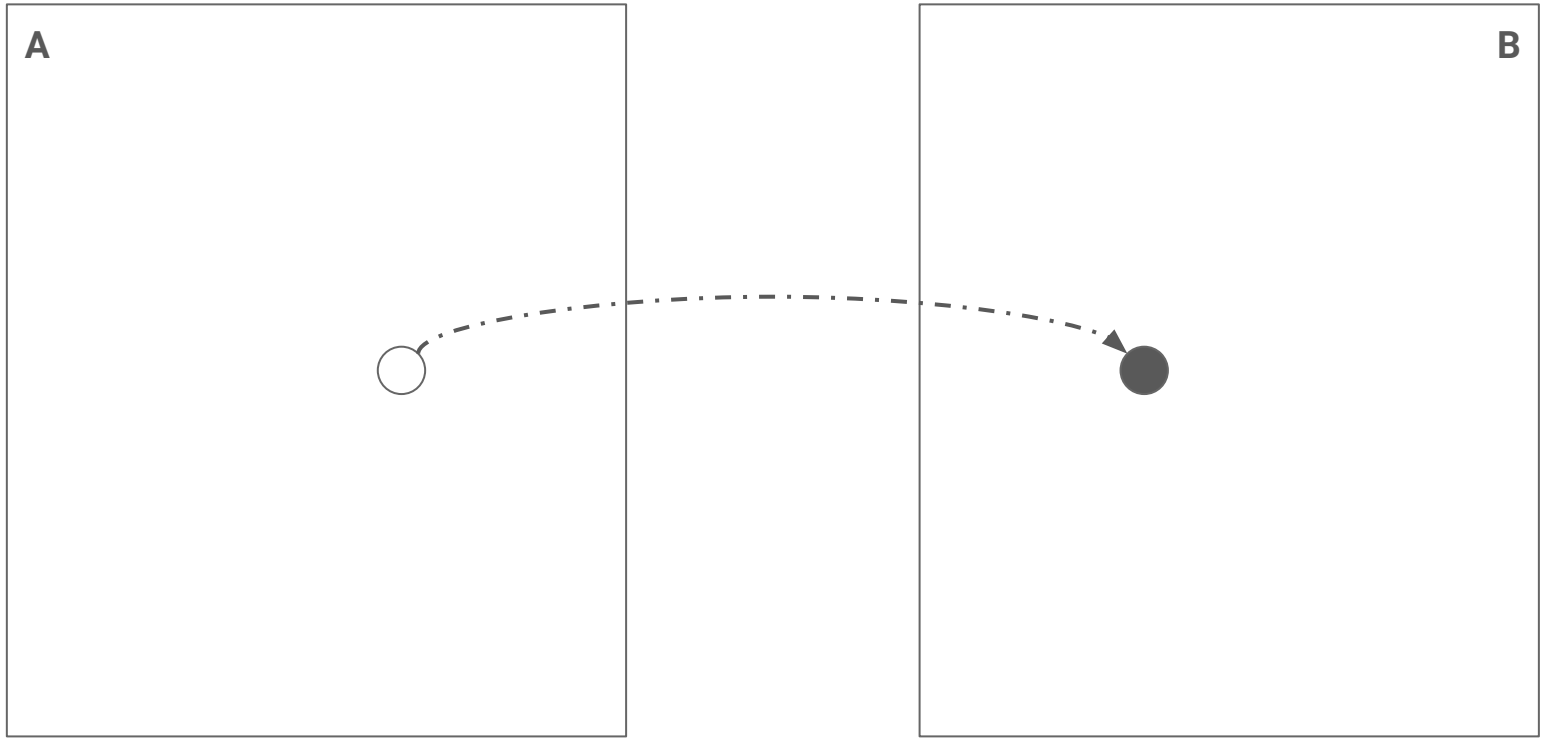
The Cycle Problem



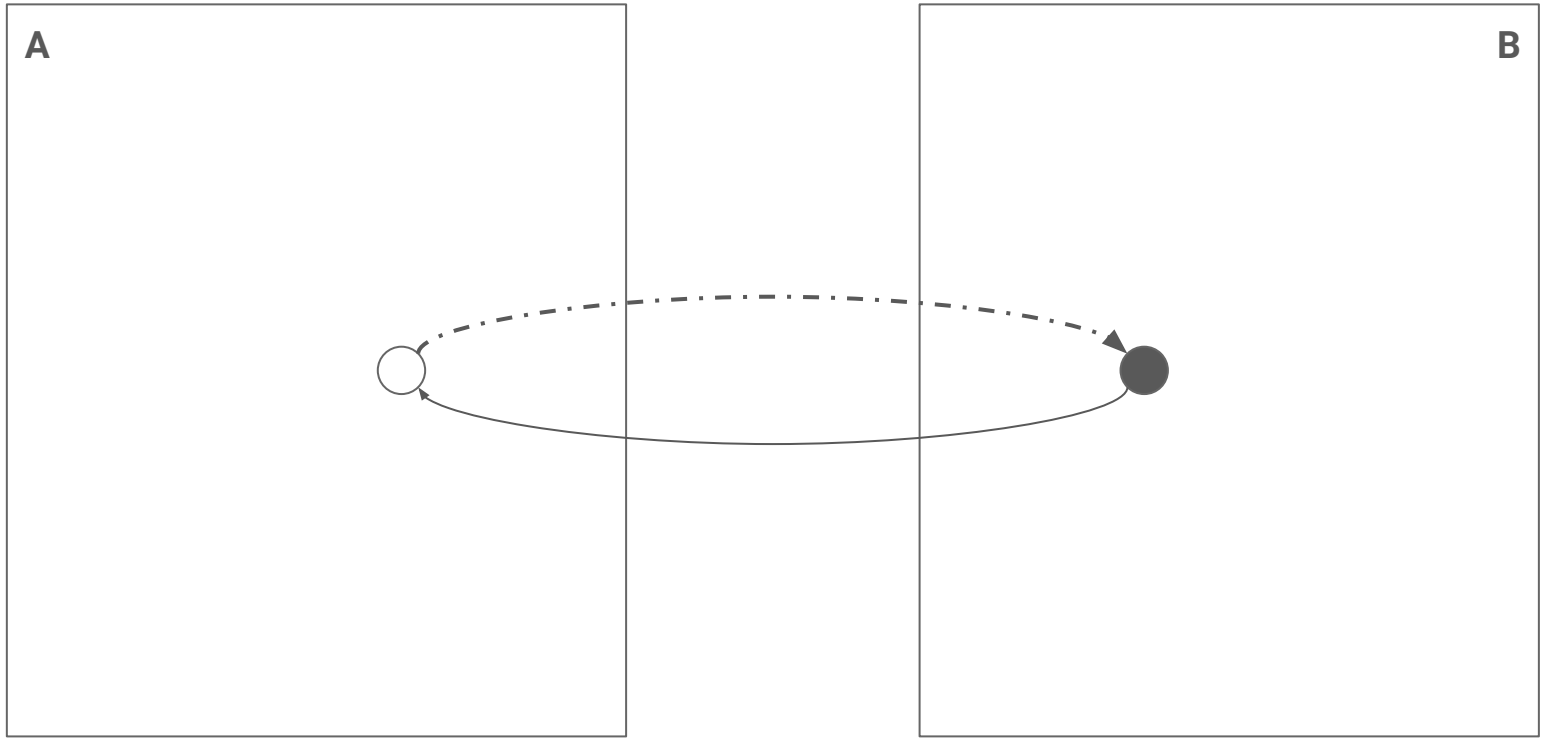
The Cycle Problem



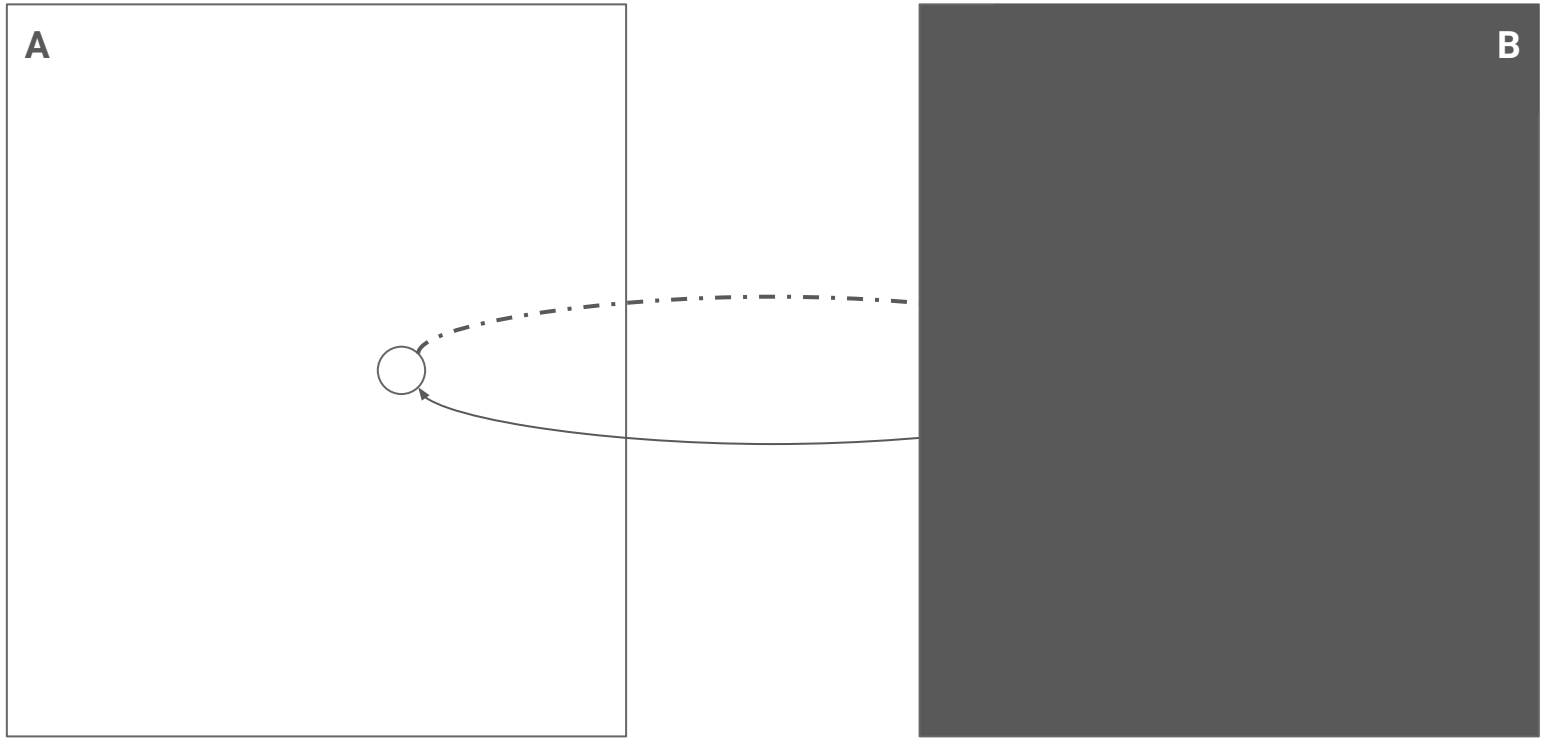
The Cycle Problem



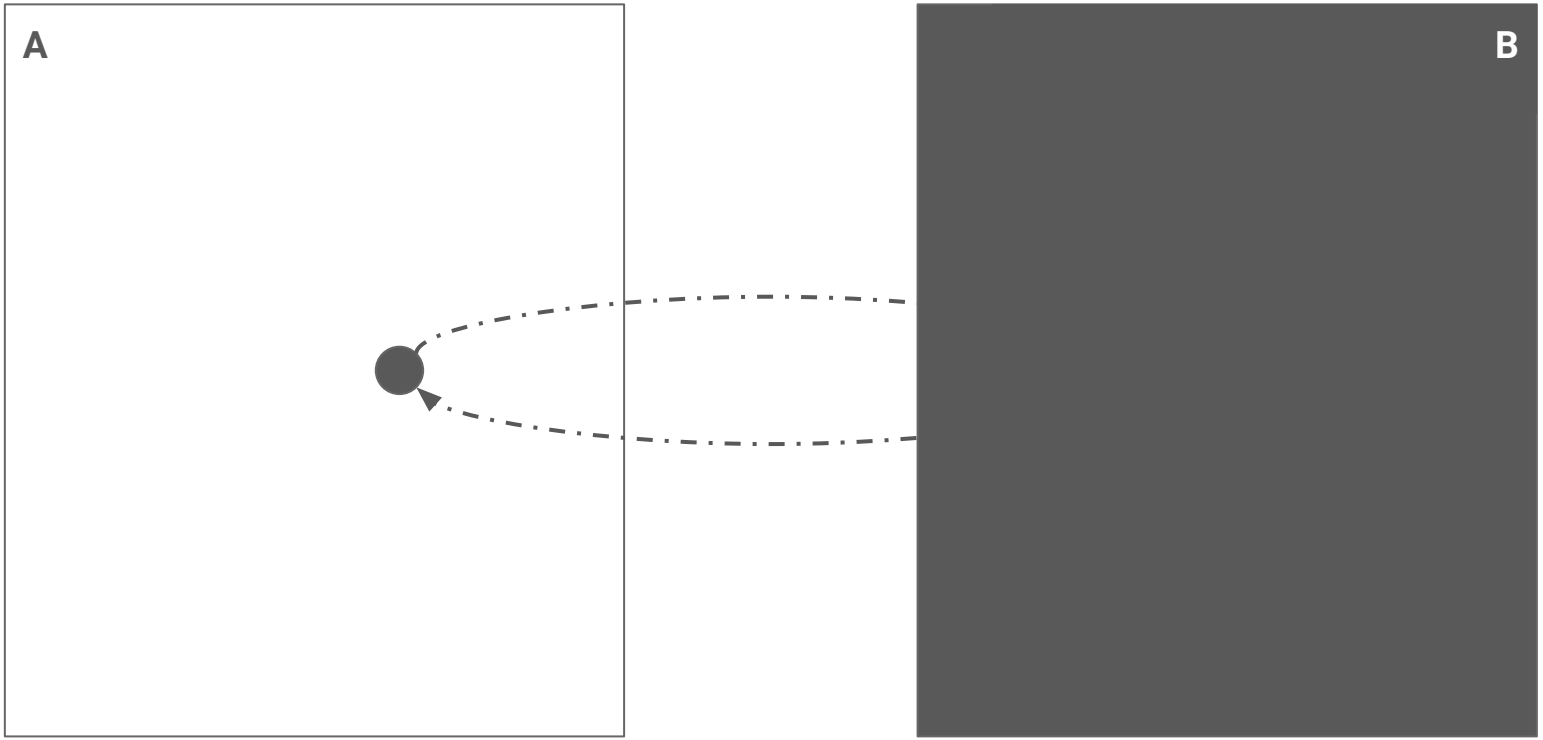
The Cycle Problem



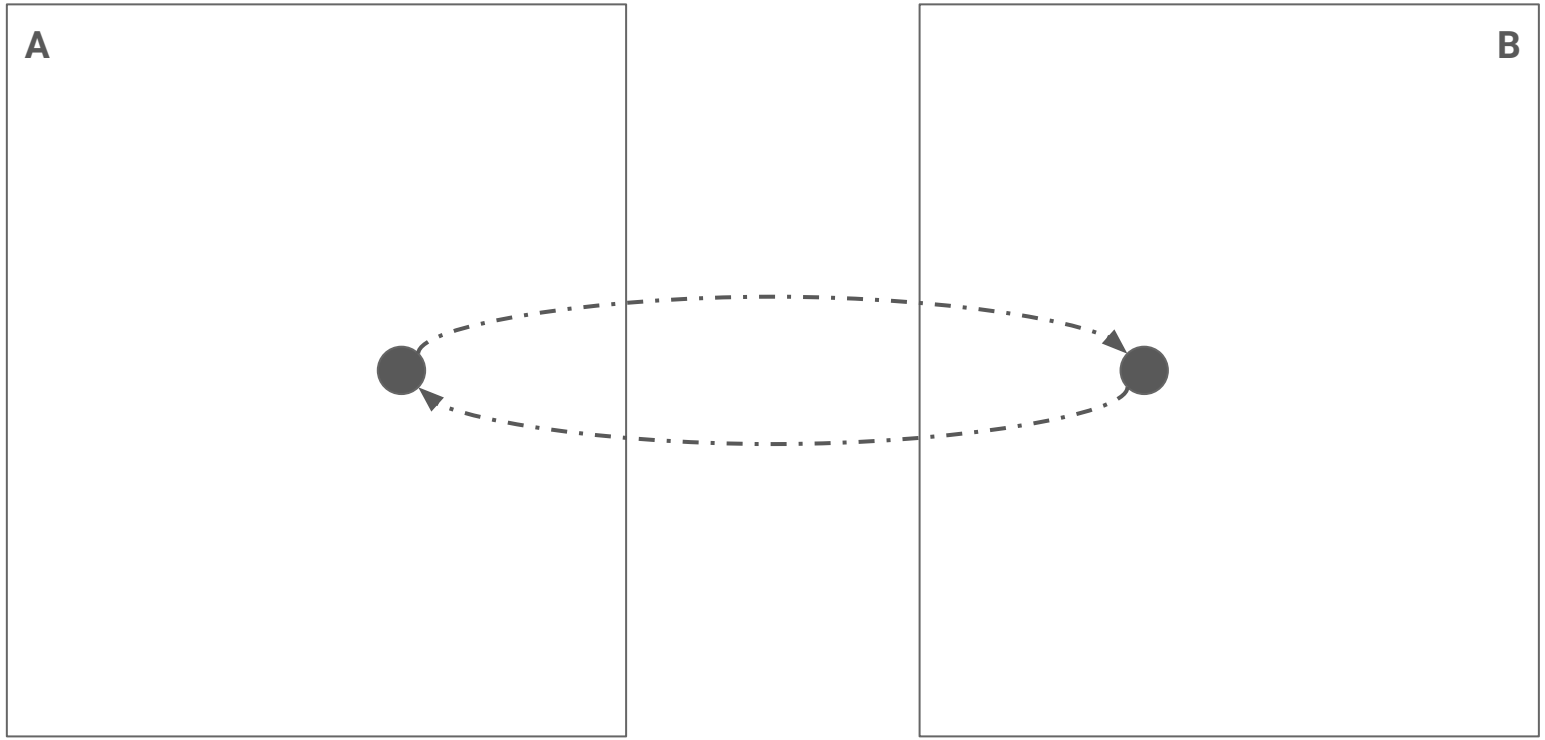
The Cycle Problem



The Cycle Problem



The Cycle Problem



The Cycle Problem

- Manual breaking of cycles using weak references
  - Not possible when cross-component reference requires object to stay alive

Observations

- Similar to cycle problem when using reference counts
  - Lots of algorithms and literature on how to deal with cycles
- Cross-component information required to break cycles automatically

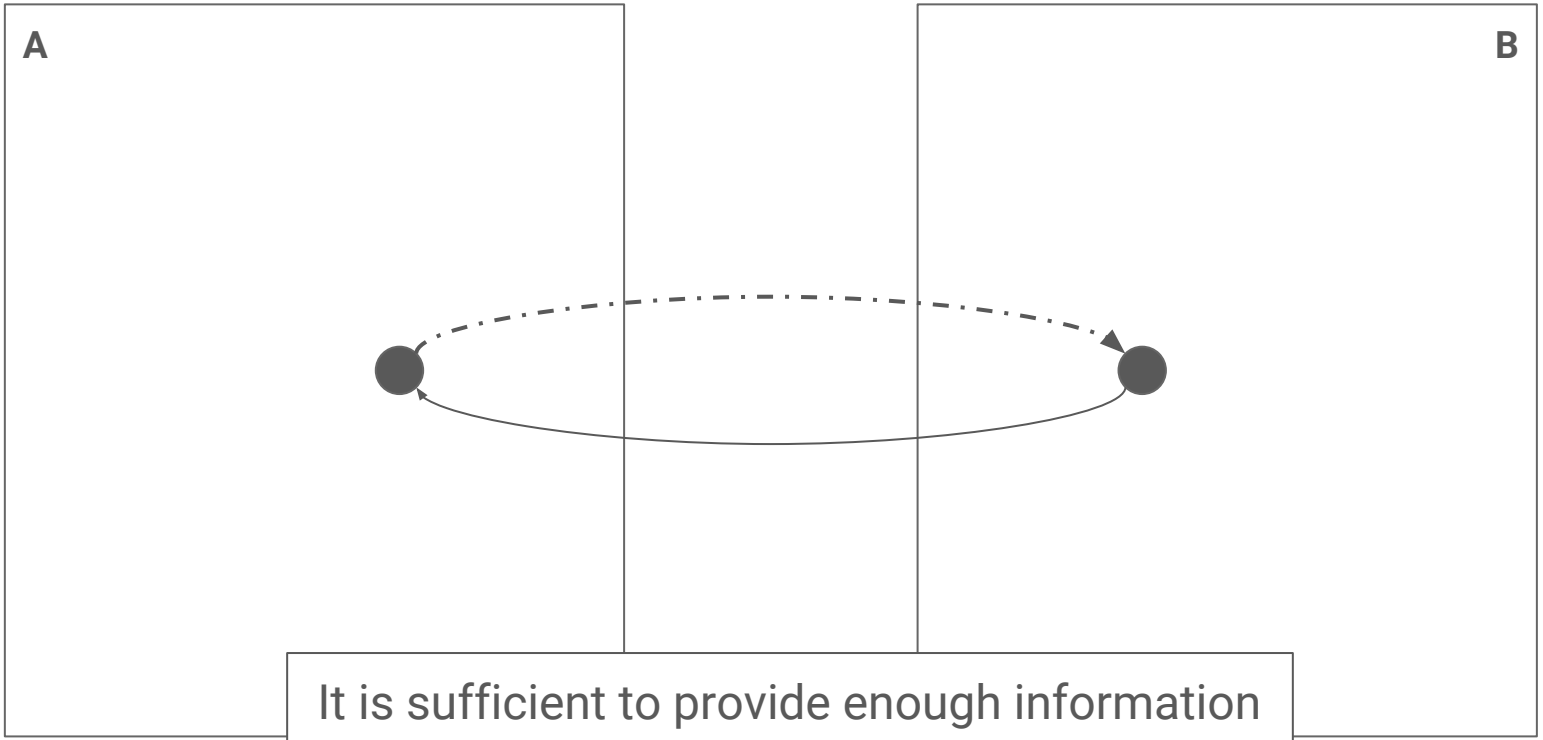
Cross-Component Garbage Collection

Cross-Component Garbage Collection

Provide and process liveness information

across components boundaries

Cross-Component Garbage Collection



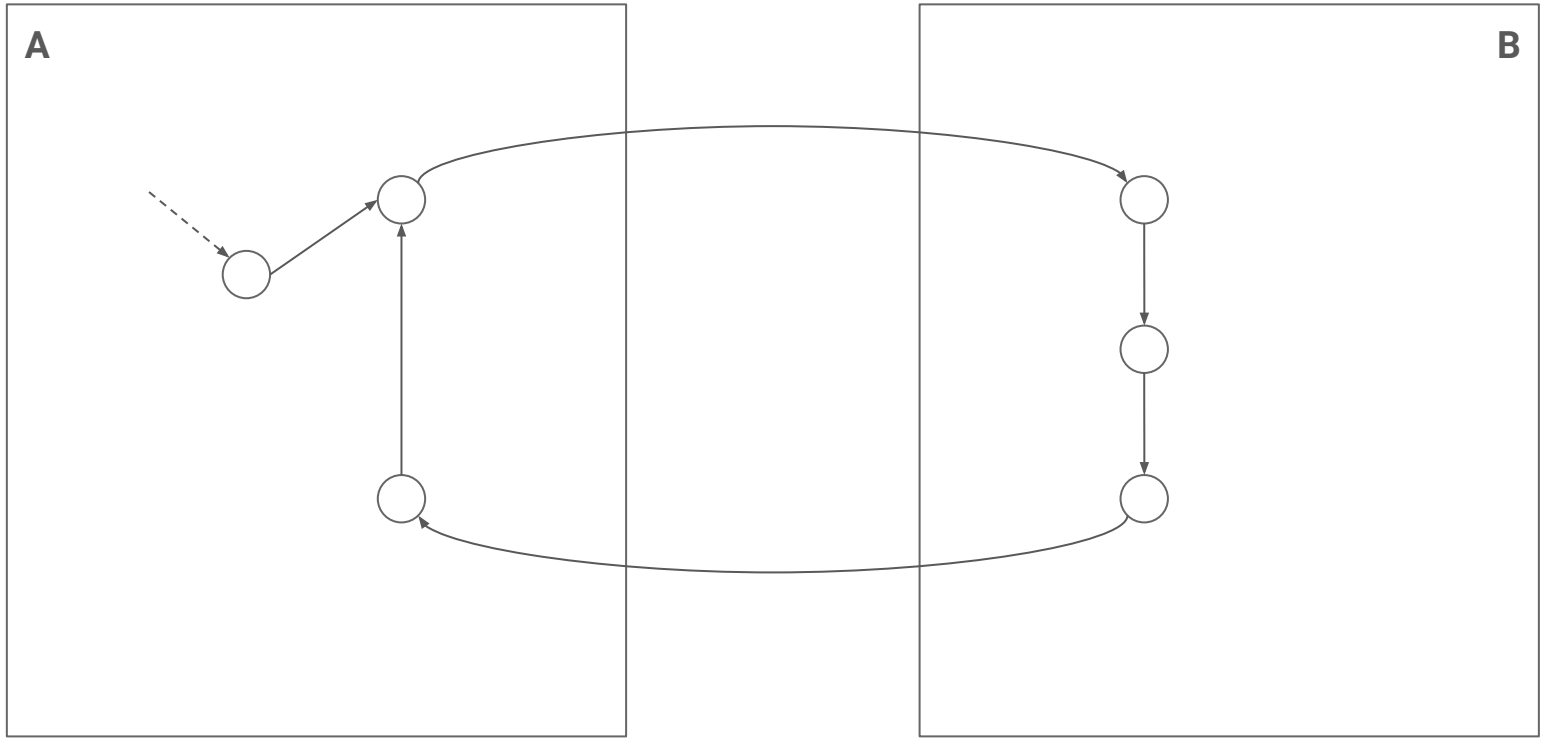
It is sufficient to provide enough information to break one link of the cycle

Tracing across component boundaries

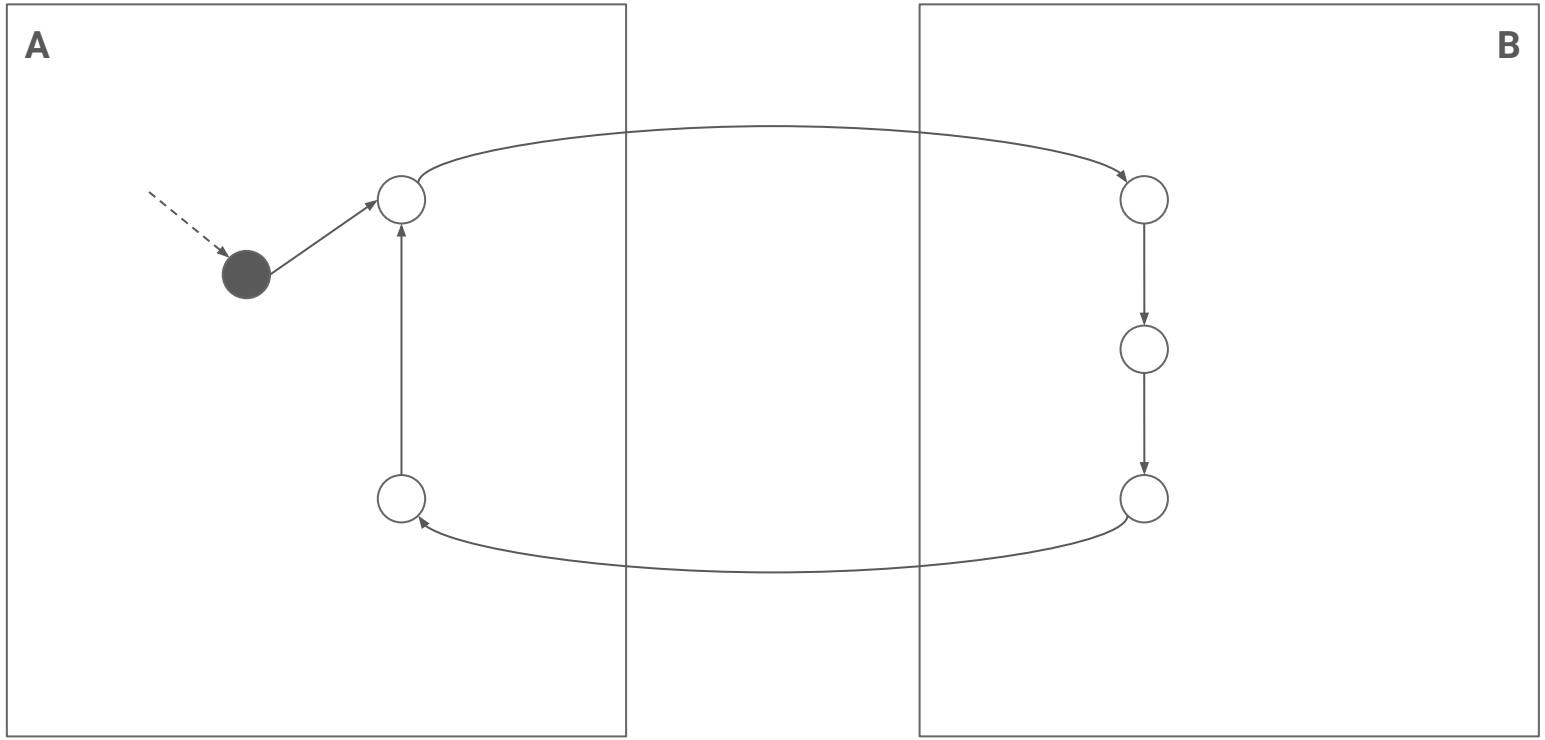
trace :: Object -> [Objects]

Return all objects that are transitively reachable from a given object

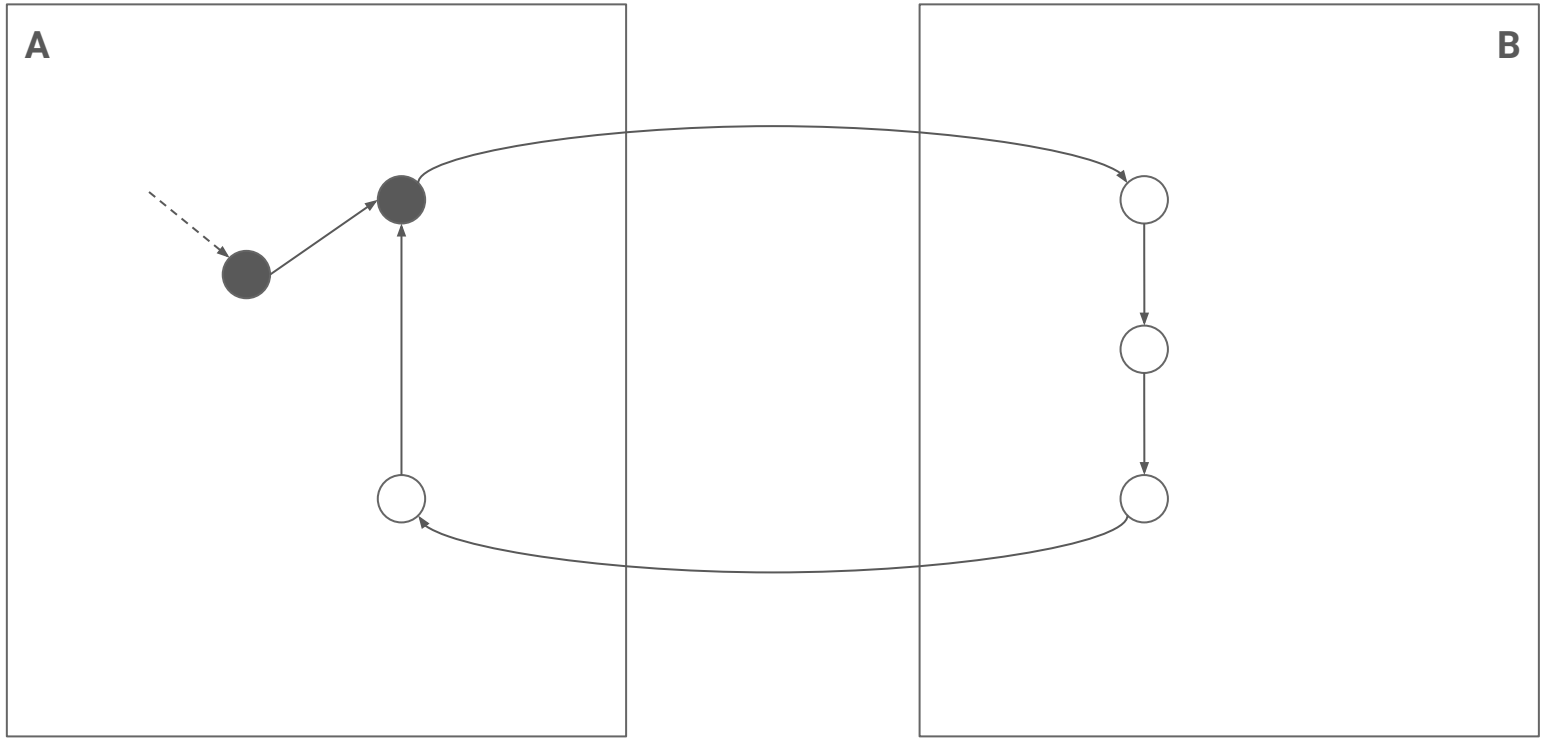
Tracing across component boundaries



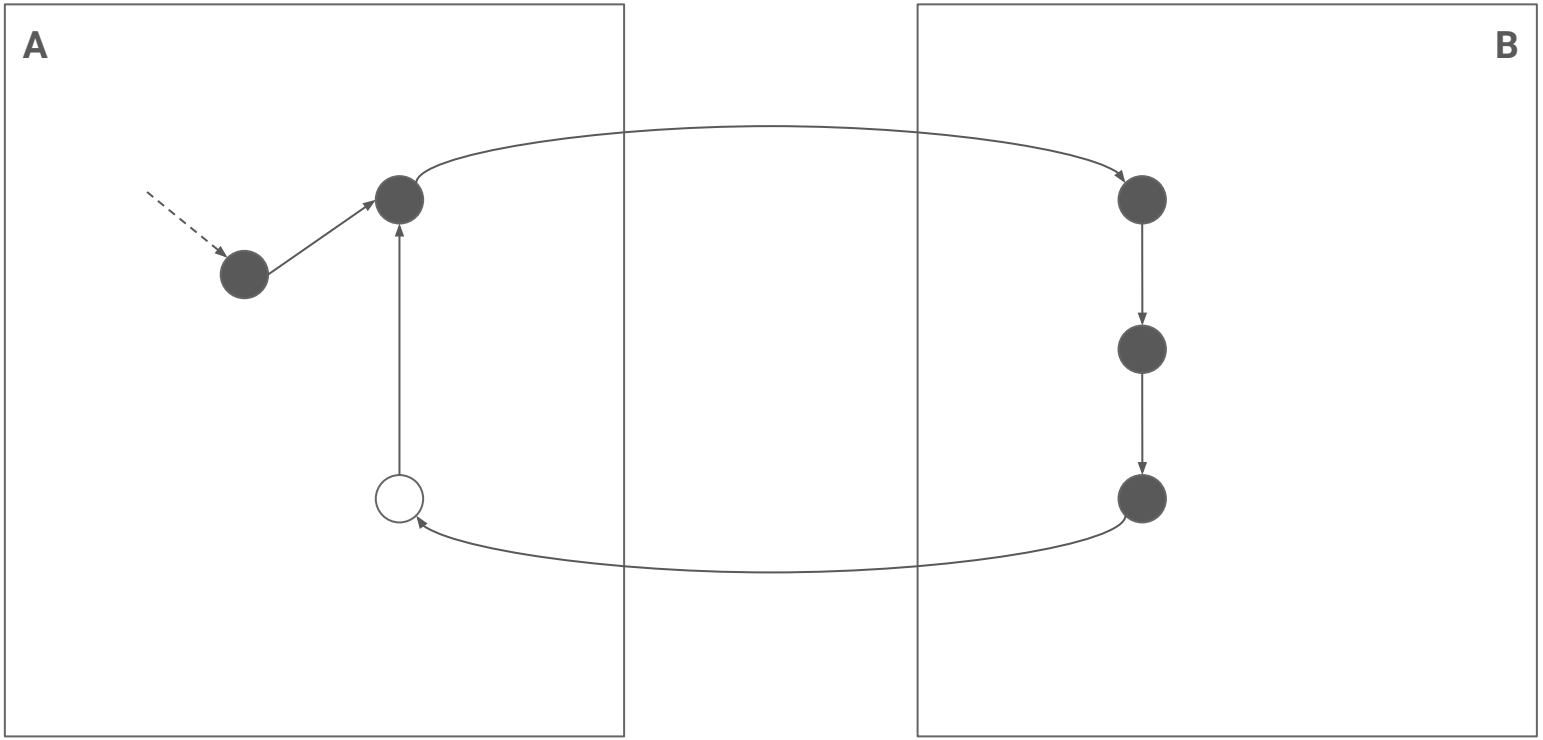
Tracing across component boundaries



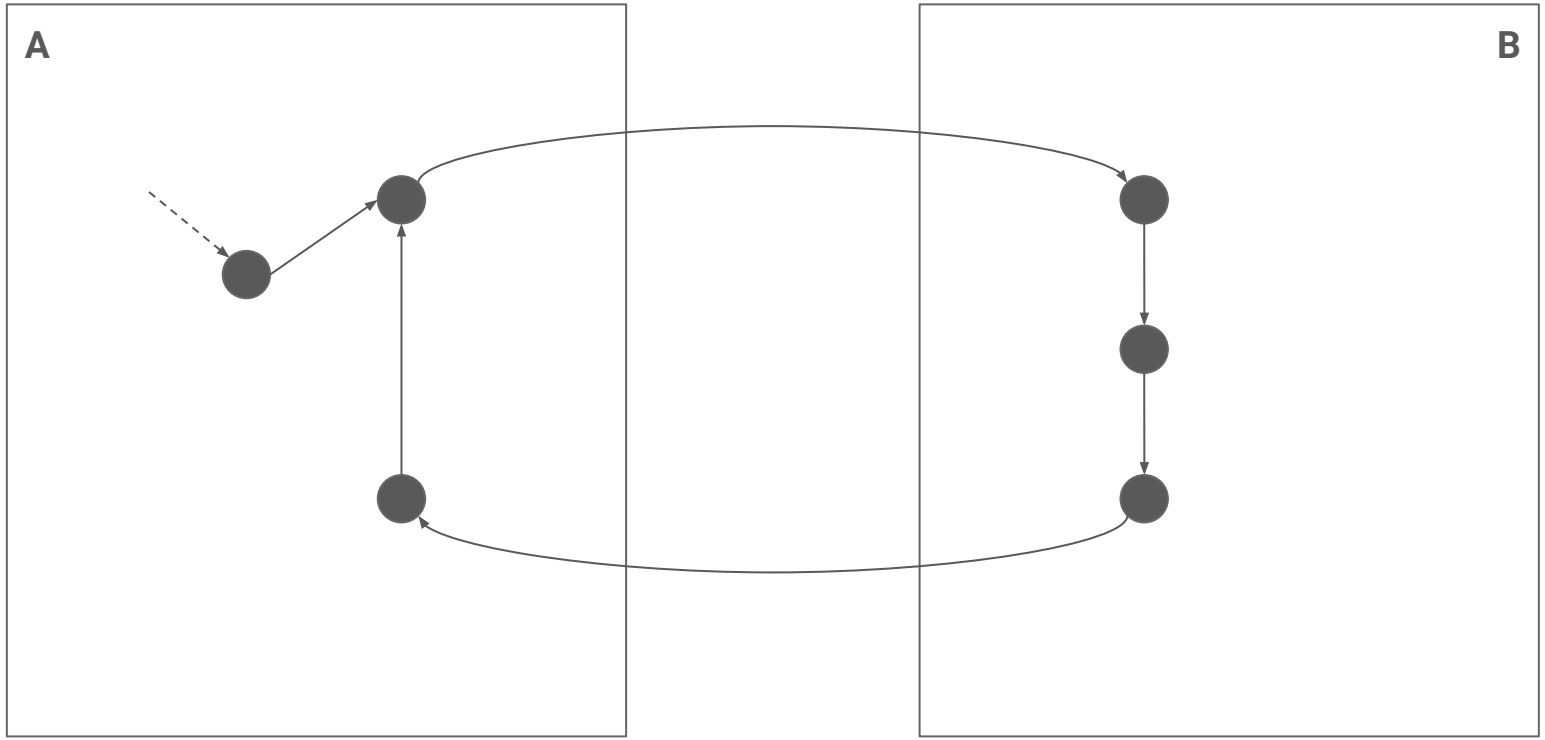
Tracing across component boundaries



Tracing across component boundaries



Tracing across component boundaries



V8: Tracing from JavaScript to C++ and back

- Batch-based communication
- Control over when to start tracing

V8 exposes an interface that embedders need to implement

RegisterReferences([Object])

Allows communicating objects that are entry points into the embedders heap

RegisterExternallyReferencedObject(Object)

Communicate that Object is reachable through some entry point object

AdvanceTracing(deadline, force_completion)

Trace objects on the embedder's heap

Interlude: Incremental Marking

- Main thread gets interrupted and marks *n* objects until all live objects are marked
- Ingredients:
  - One marking deque
  - Three colors
    - White: unknown
    - Grey: on marking deque and will be processed
    - Black: marked
  - Barriers ensuring consistency
  - A driver that ensures marking progress

Blink: Write Barrier

- Preserving strong tri-color invariant
  - No black to white edges
- Dijkstra-style write barrier

```
write(Object source, Object value):
  if (is_tracing && is_marked(source) && !is_marked(value))
    mark(value)
    marking_deque.push(value)
```

Correctness: C++ Write Barriers

Key problem: Not emitting the write barrier

- Idea: Rely on C++ type system to enforce the barriers
- GC pointers: GC-aware smart pointers
  - Back pointer to object header
  - Emit barrier on write, (copy) construction, move
- Restrict tracing methods to accepting only those GC pointers
  - Result: Compile failure if raw pointers are used

Successfully annotated ~**250** fields in the DOM that required a barrier

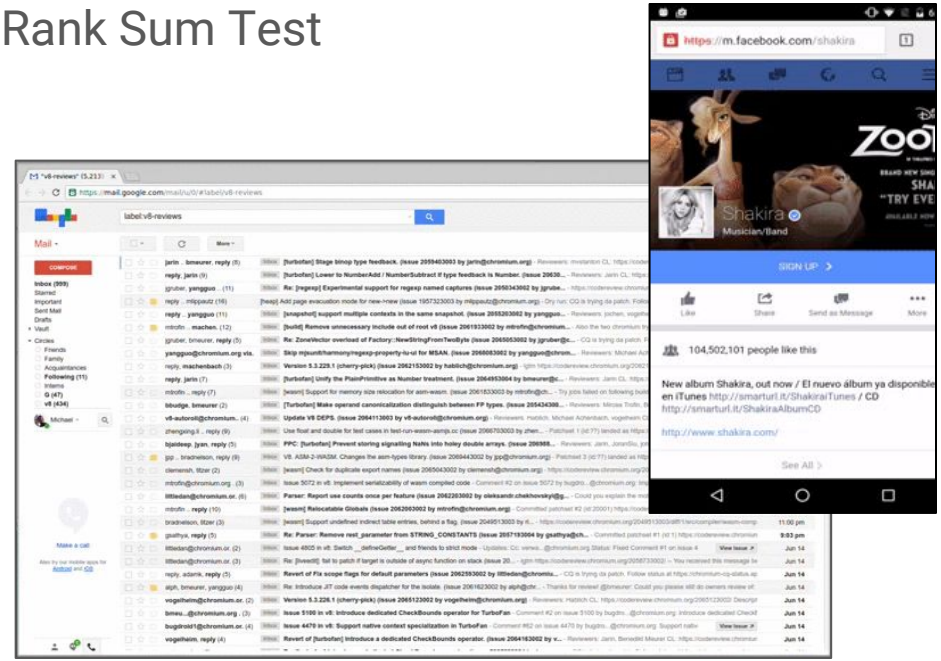
Benchmarking / Results

Baseline: Object Grouping

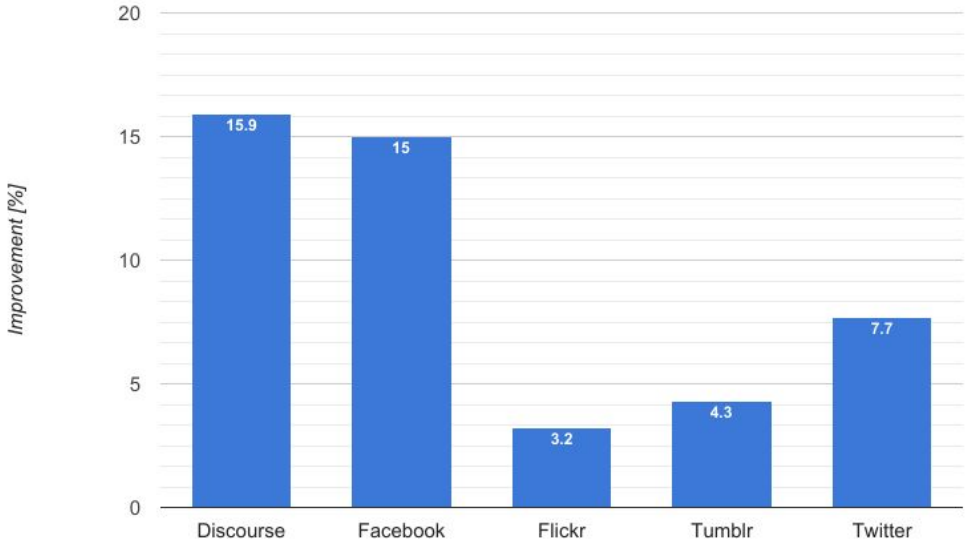
- Objects are grouped together based on rules, e.g., DOM tree
- A group is considered as live if one object of the group is alive
- Incremental over approximation possible but expensive
  - Needs to consider live and dead objects
- Prone to memory leaks in certain situations

Benchmarking

- Real-world workloads using Catapult benchmarking framework
- Trace every single GC event, e.g. marking, marking finalization, atomic pause
- Hypothesis testing using Wilcoxon Rank Sum Test



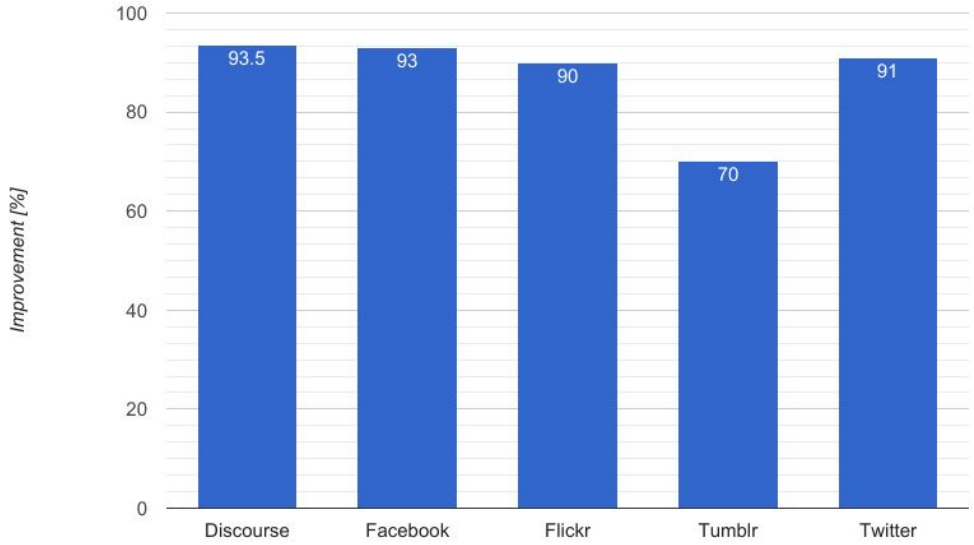
Average Latency-critical GC Pause



Wilcoxon Rank Sum Test	**significant**	**significant**	**not significant**	**not significant**	**significant**

Incremental Marking Finalization

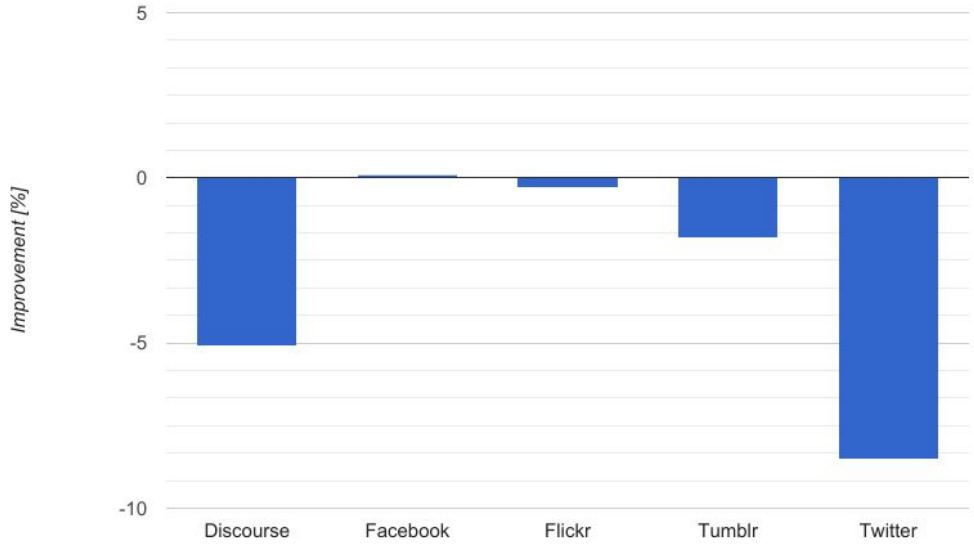
Potentially last incremental step before atomic pause



Wilcoxon Rank Sum Test	**significant**	**significant**	**significant**	**significant**	**significant**

Marking Time

the tradeoff



Wilcoxon Rank Sum Test	**significant**	**not significant**	**not significant**	**not significant**	**significant**

Google

Fin. Thank you!

chromium.org

v8project.org