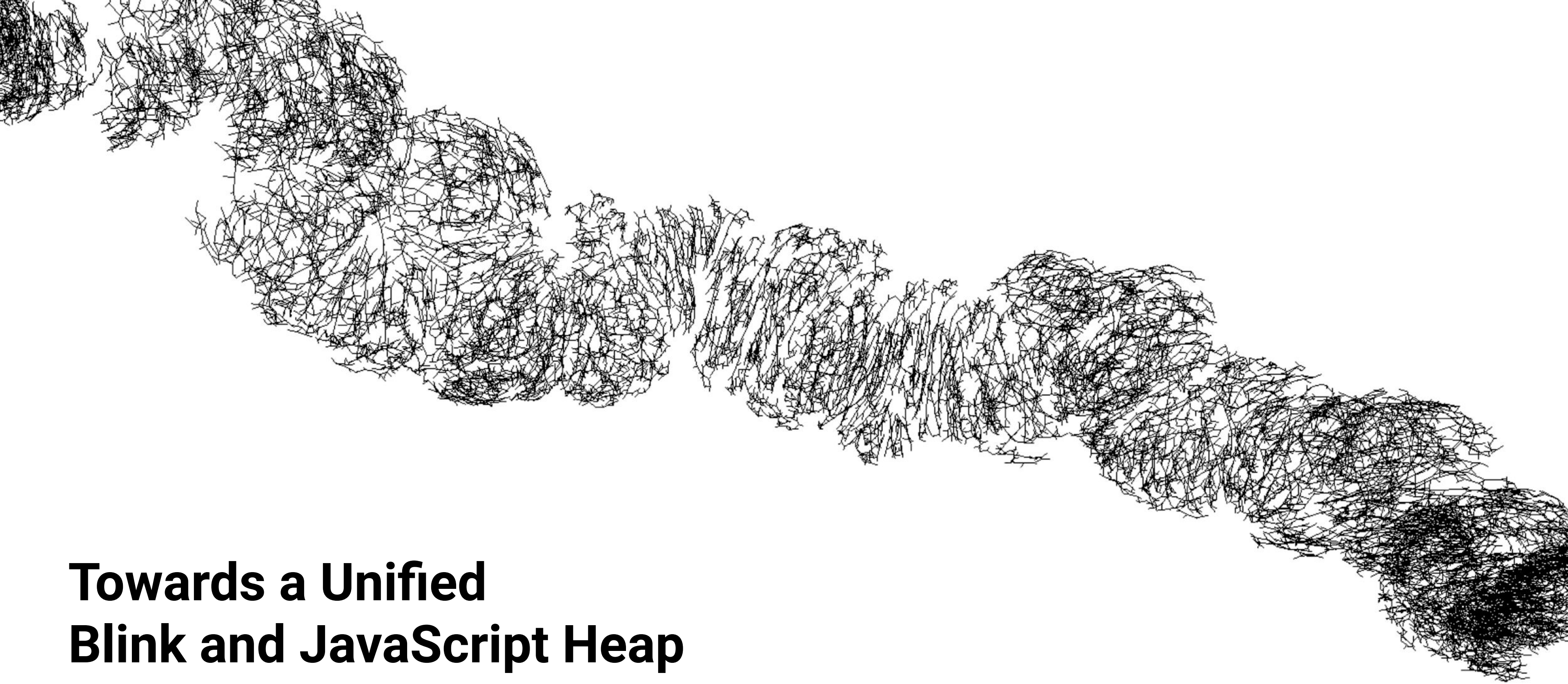


# Towards a Unified Blink and JavaScript Heap

mlippautz@
BlinkOn 9, Sunnyvale, Apr 2018

# Memory management across Blink and V8

***now*** and in ***future***

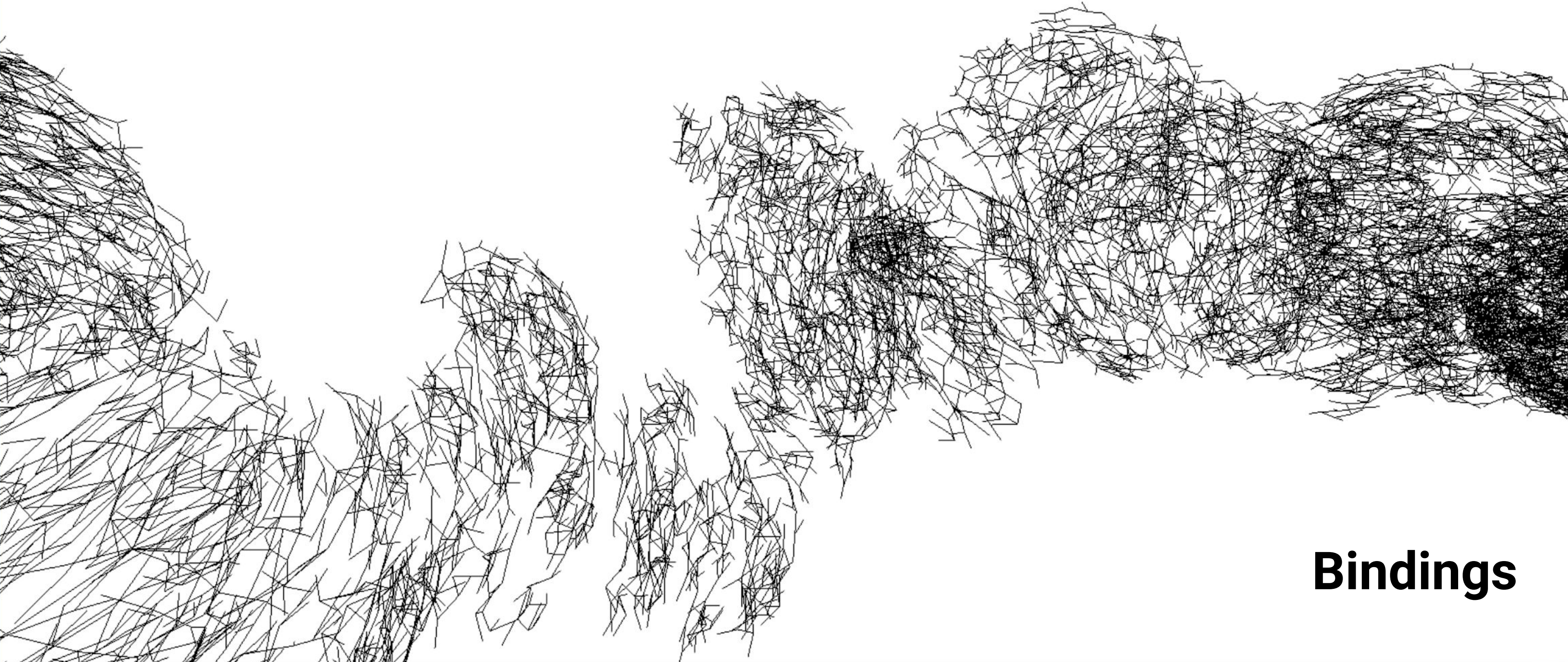
# V8 & Blink

- JavaScript ⇔ DOM

***V8***

***Blink***

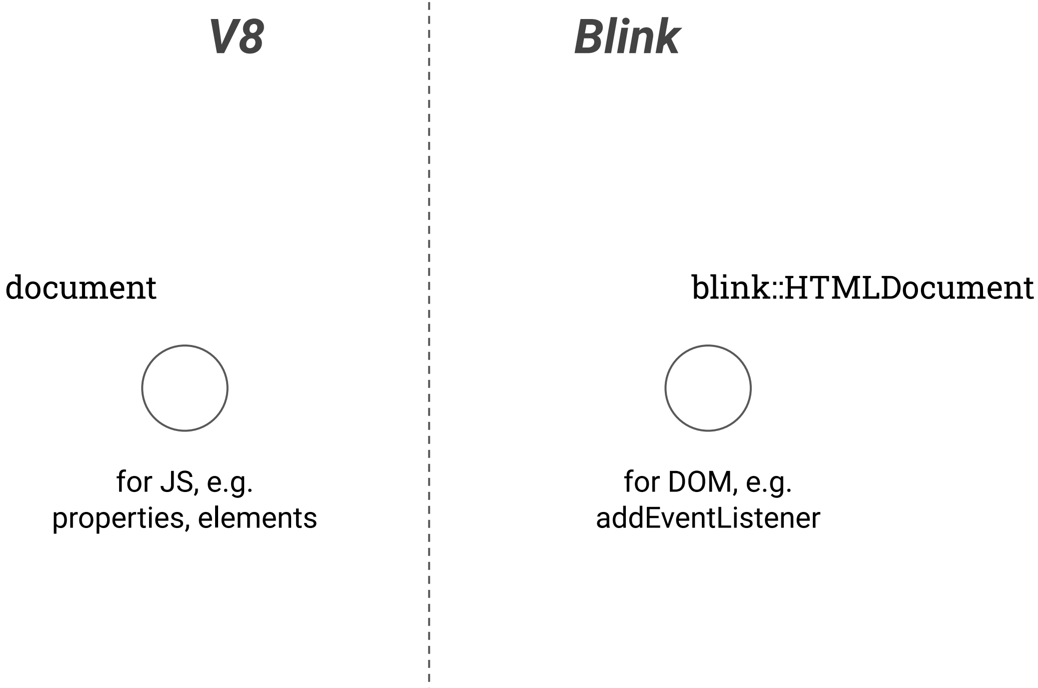
```
<script>
 document;
 document.a;
 document.addEventListener(...);
</script>
```



# Bindings

# V8 & Blink

- JavaScript ⇔ DOM
- Objects come in halves



***V8***

***Blink***

document

for JS, e.g.
properties, elements

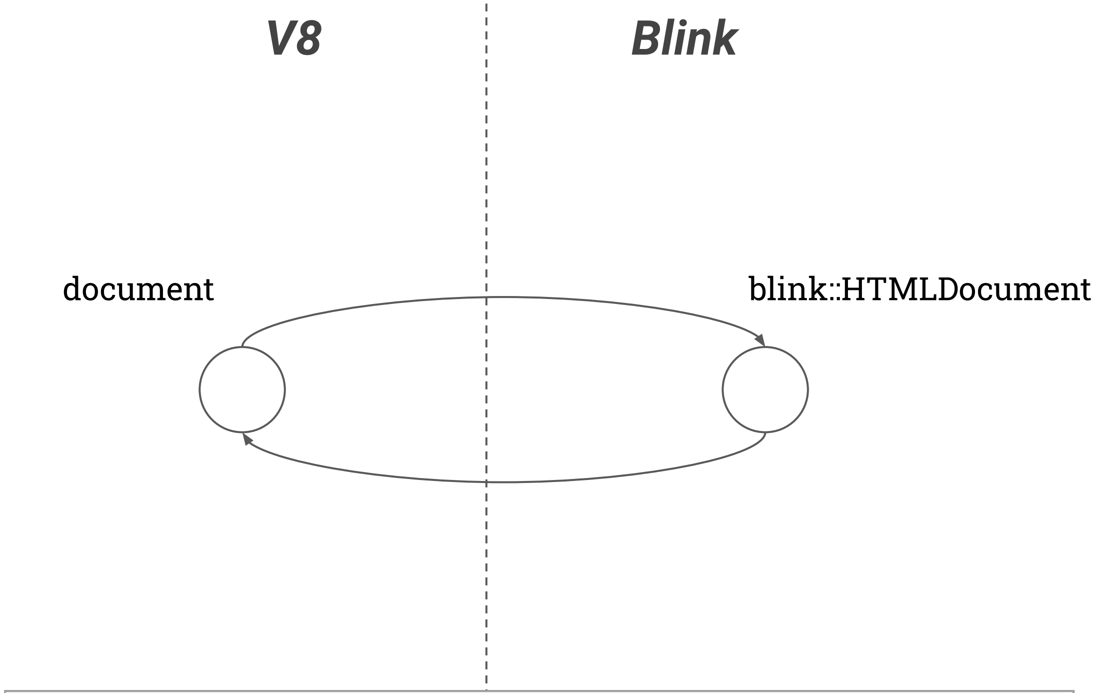
blink::HTMLDocument

for DOM, e.g.
addEventListener

```
<script>
 document;
 document.a;
 document.addEventListener(...);
</script>
```

# V8 & Blink

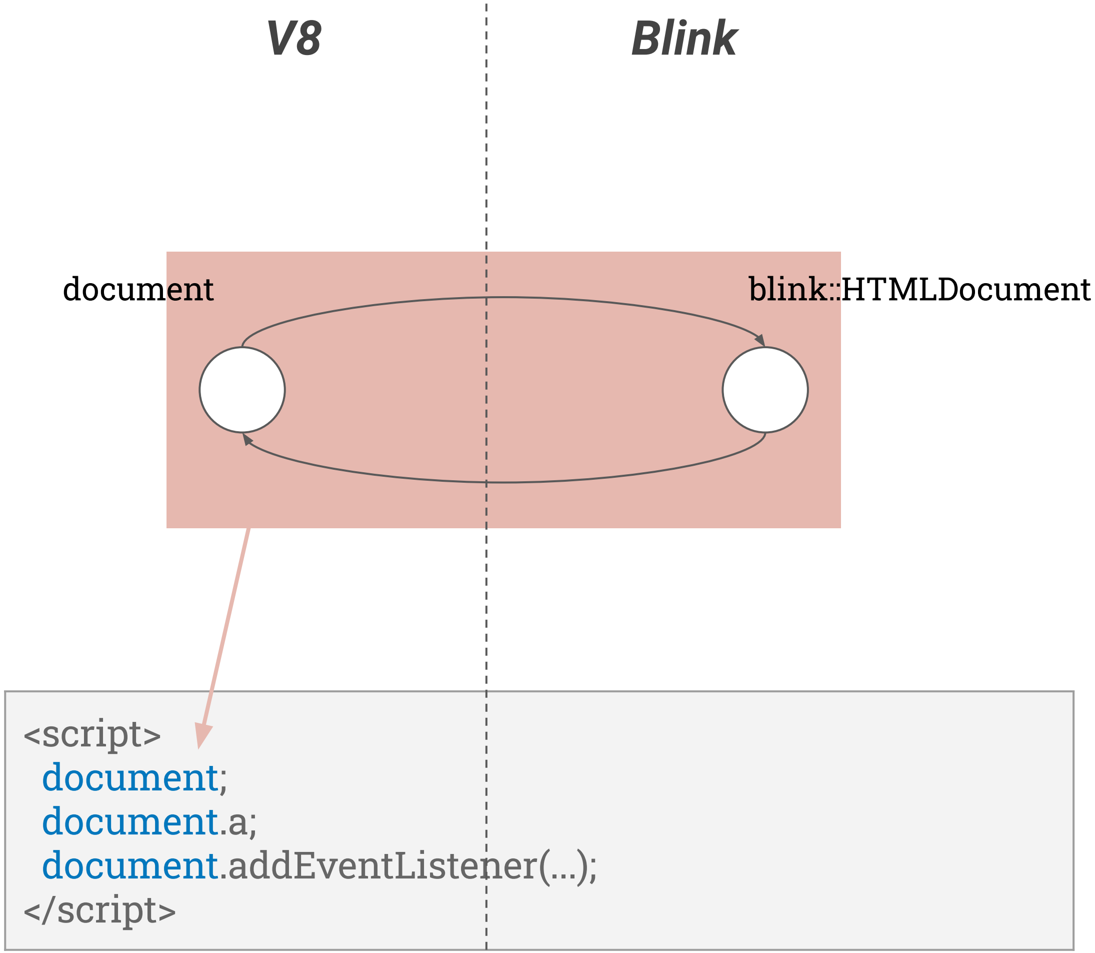
- JavaScript ⇔ DOM
- Objects come in halves
- Reference each other



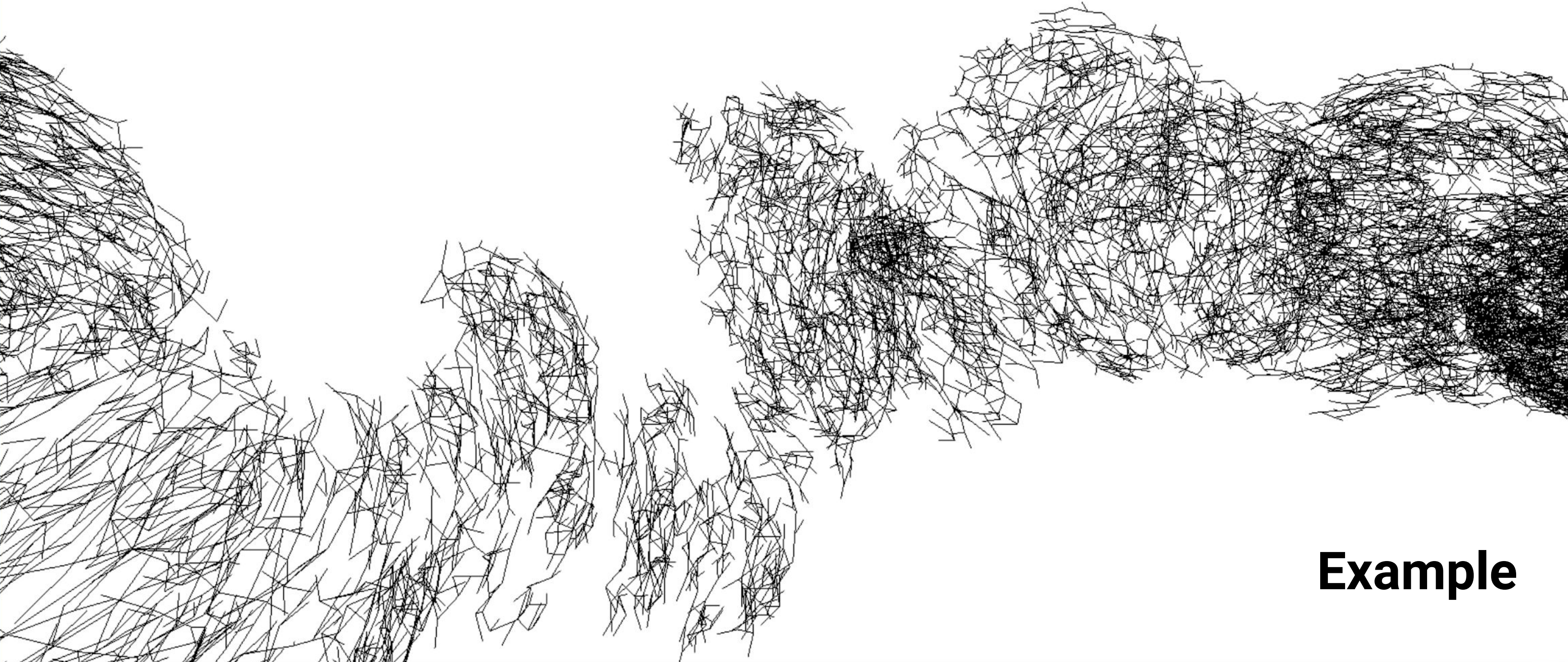
```
<script>
 document;
 document.a;
 document.addEventListener(...);
</script>
```

# V8 & Blink

- JavaScript ⇔ DOM
- Objects come in halves
- Reference each other



```
<script>
 document;
 document.a;
 document.addEventListener(...);
</script>
```



Example

# Example

*“Everything should be made as simple as possible, but no simpler”*

```
<!DOCTYPE html>
<head><script>
 function createDiv() {
  let newDiv =
    document.createElement("div");
  document.body
    .appendChild(newDiv);
 }

 document.addEventListener(
   "DOMContentLoaded", createDiv);
</script></head>
<body>
 <span></span>
</body>
</html>
```

# Example

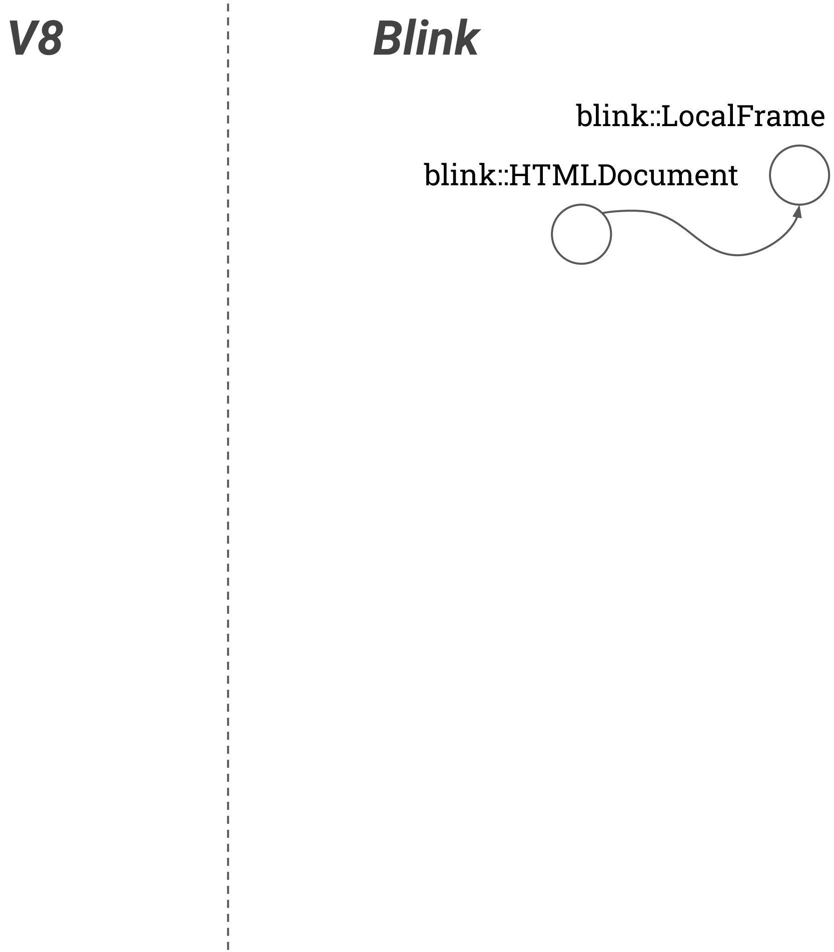
```
<!DOCTYPE html>
<head><script>
 function createDiv() {
  let newDiv =
    document.createElement("div");
   document.body
     .appendChild(newDiv);
 }

 document.addEventListener(
   "DOMContentLoaded", createDiv);
</script></head>
<body>
 <span></span>
</body>
</html>
```

***V8***

***Blink***

# Example

```
<!DOCTYPE html>
<head><script>
 function createDiv() {
  let newDiv =
     document.createElement("div");
   document.body
     .appendChild(newDiv);
 }

 document.addEventListener(
   "DOMContentLoaded", createDiv);
</script></head>
<body>
 <span></span>
</body>
</html>
```



V8

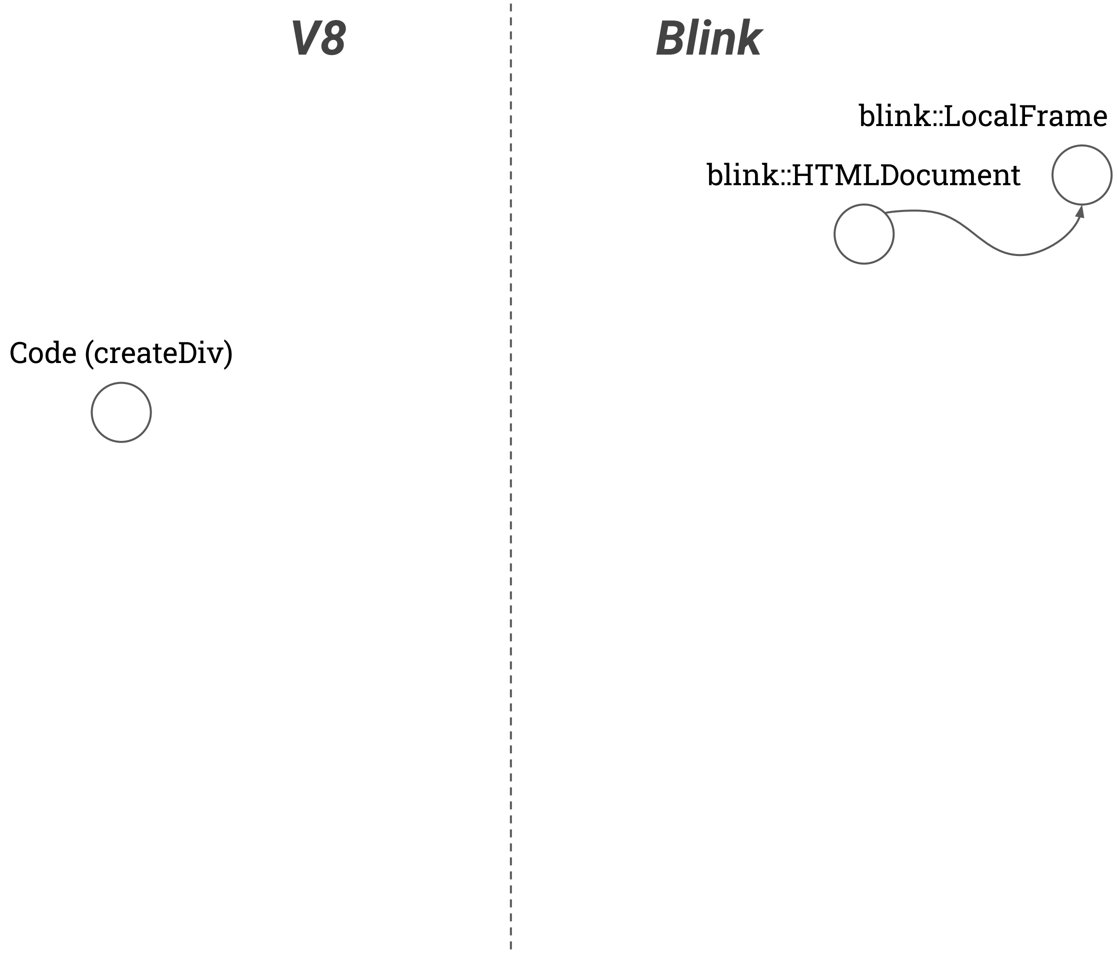
Blink

blink::LocalFrame

blink::HTMLDocument

# Example

```
<!DOCTYPE html>
<head><script>
 function createDiv() {
  let newDiv =
      document.createElement("div");
   document.body
       .appendChild(newDiv);
 }

 document.addEventListener(
    "DOMContentLoaded", createDiv);
</script></head>
<body>
 <span></span>
</body>
</html>
```



V8

Blink

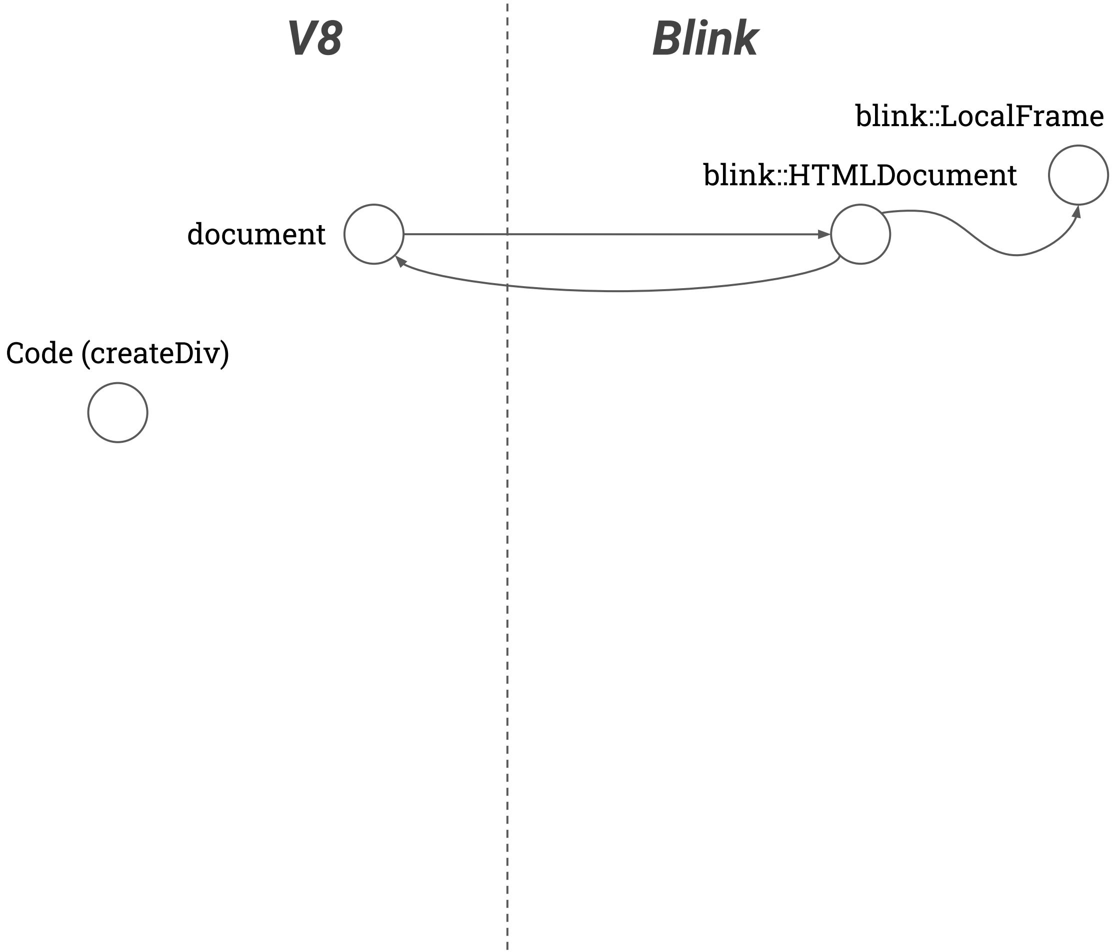
blink::LocalFrame

blink::HTMLDocument

Code (createDiv)

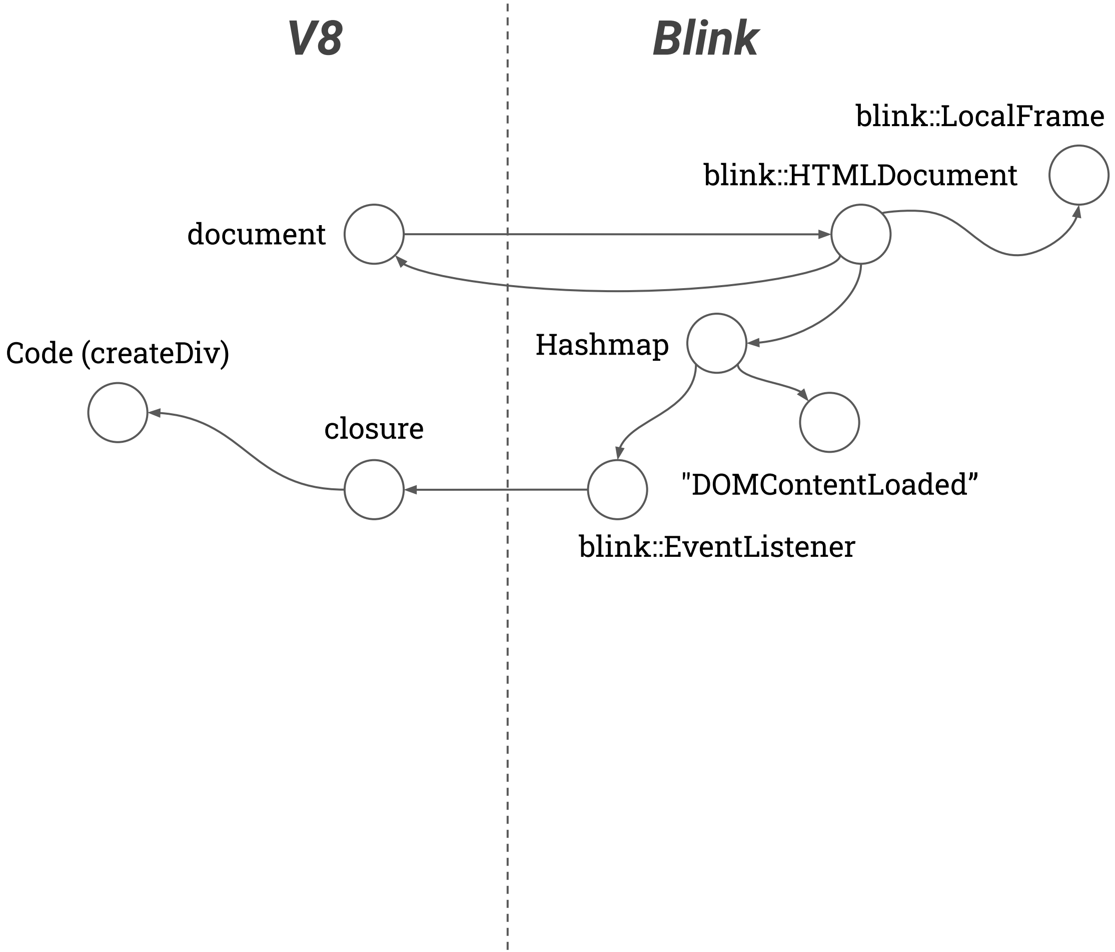
# Example

```
<!DOCTYPE html>
<head><script>
 function createDiv() {
  let newDiv =
     document.createElement("div");
   document.body
      .appendChild(newDiv);
 }

 document.addEventListener(
    "DOMContentLoaded", createDiv);
</script></head>
<body>
 <span></span>
</body>
</html>
```



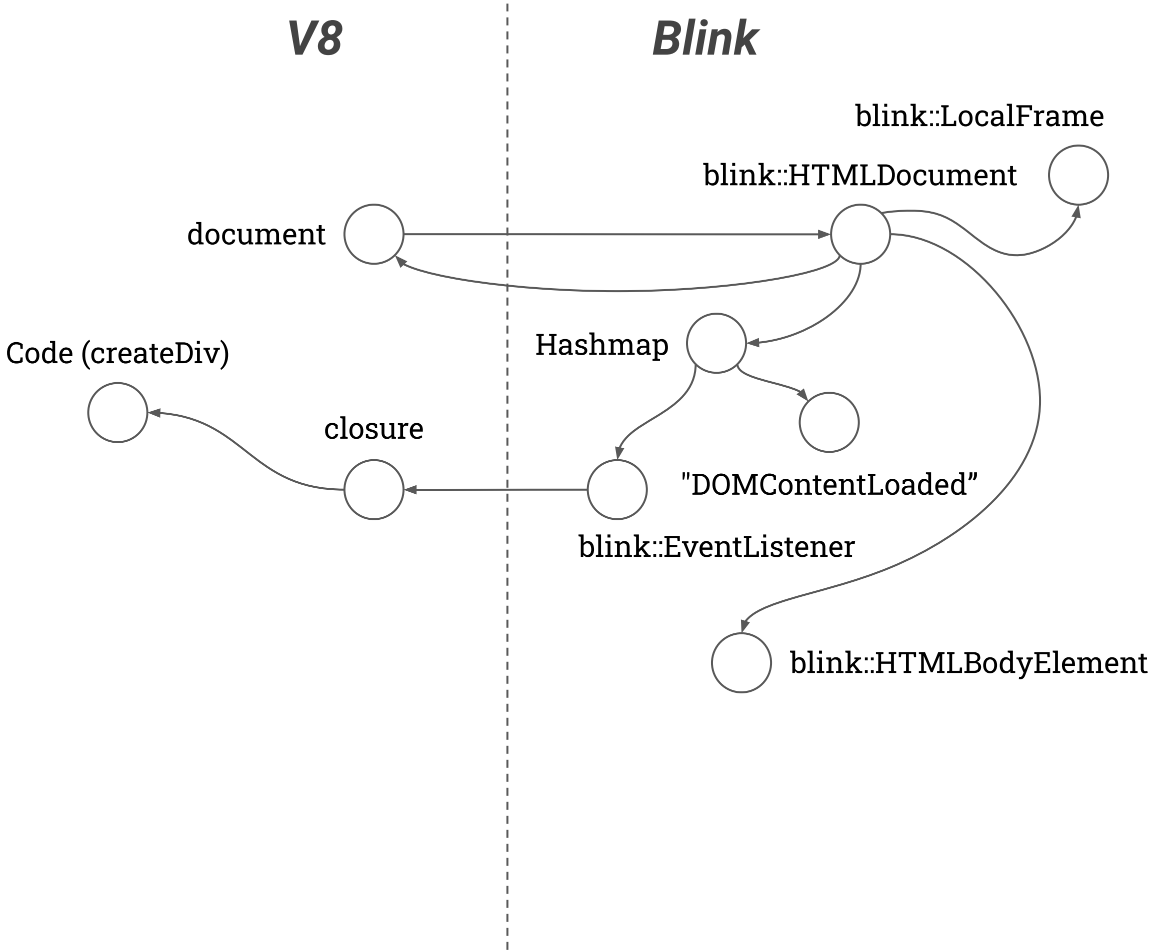
# Example

```
<!DOCTYPE html>
<head><script>
 function createDiv() {
  let newDiv =
     document.createElement("div");
    document.body
      .appendChild(newDiv);
 }

 document.addEventListener(
    "DOMContentLoaded", createDiv);
</script></head>
<body>
 <span></span>
</body>
</html>
```



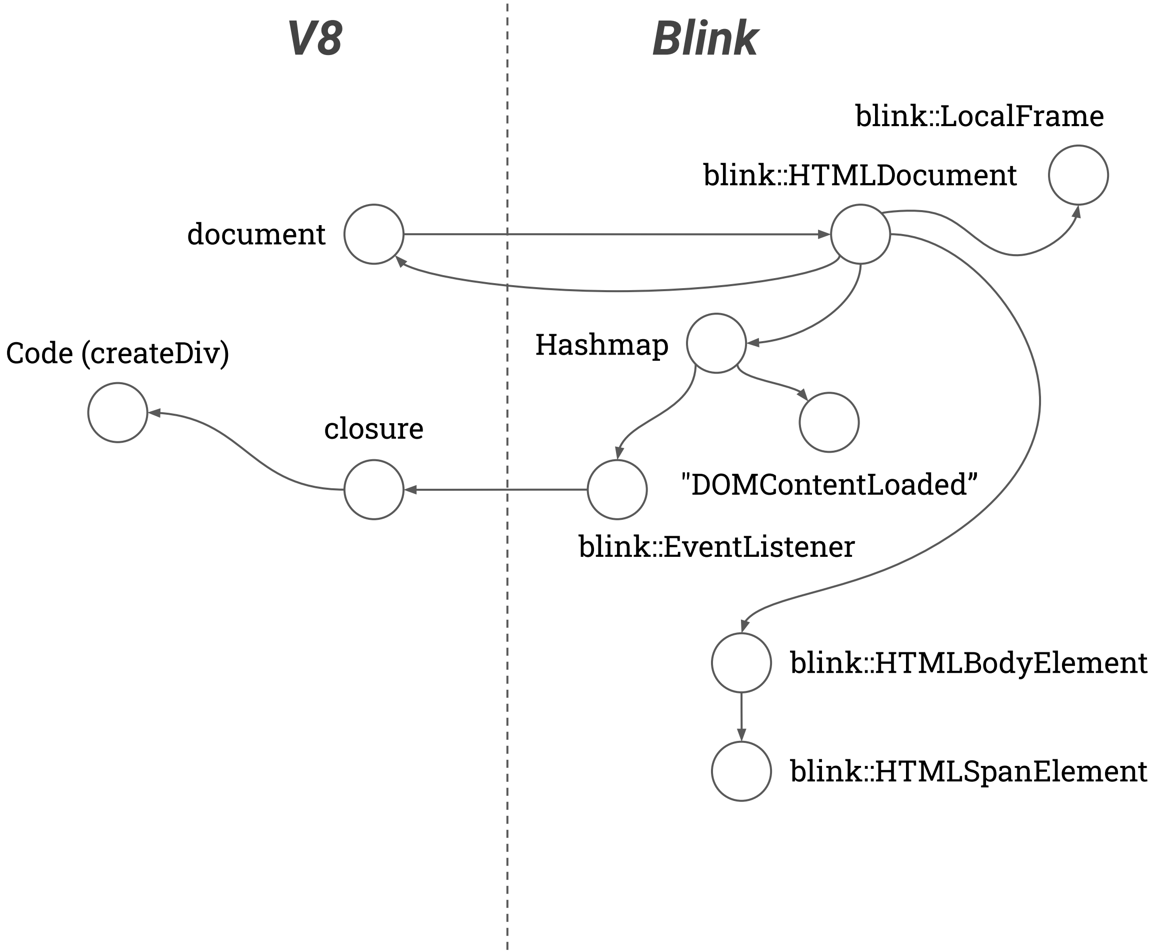
# Example

```
<!DOCTYPE html>
<head><script>
 function createDiv() {
  let newDiv =
    document.createElement("div");
   document.body
     .appendChild(newDiv);
 }

 document.addEventListener(
   "DOMContentLoaded", createDiv);
</script></head>
<body>
 <span></span>
</body>
</html>
```



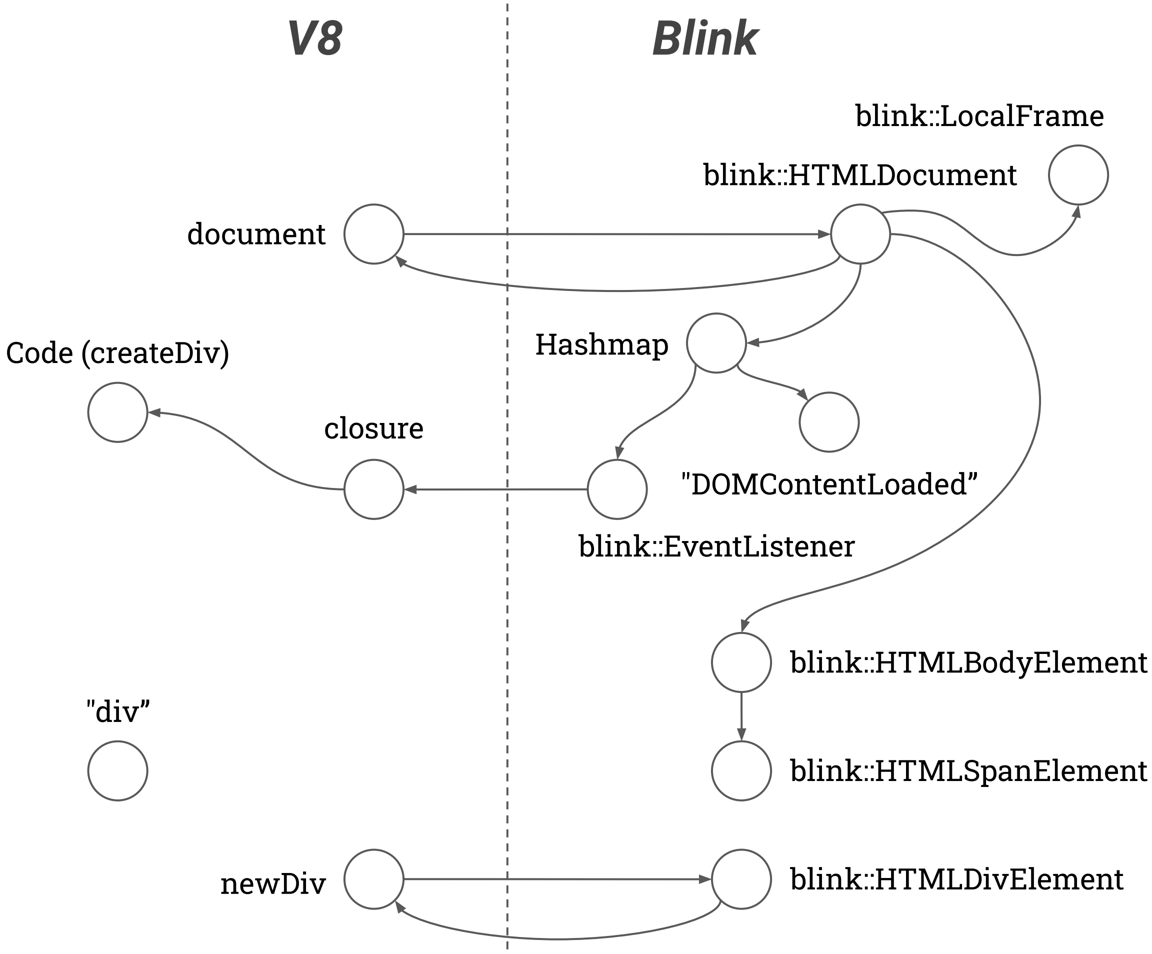
# Example

```
<!DOCTYPE html>
<head><script>
 function createDiv() {
  let newDiv =
      document.createElement("div");
    document.body
      .appendChild(newDiv);
 }

 document.addEventListener(
    "DOMContentLoaded", createDiv);
</script></head>
<body>
 <span></span>
</body>
</html>
```



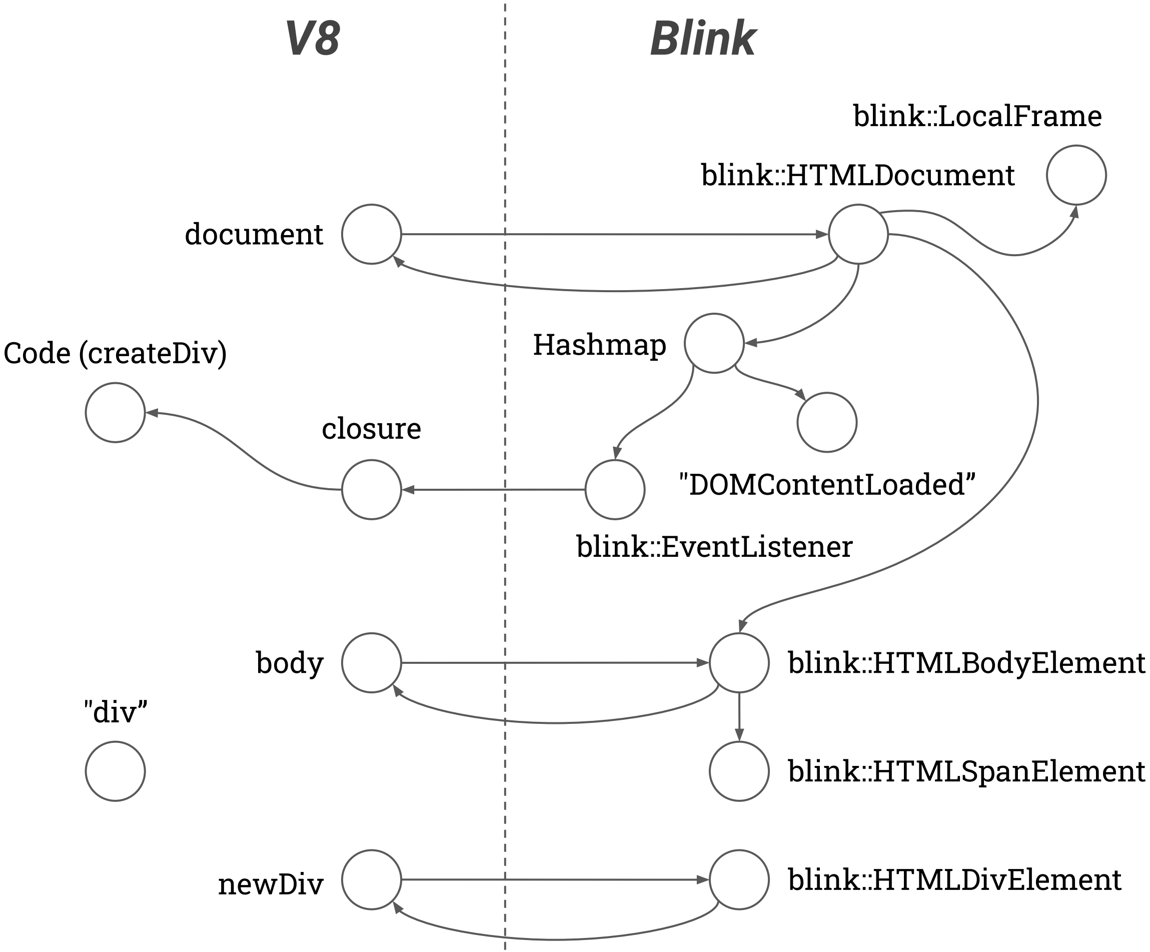
# Example

```
<!DOCTYPE html>
<head><script>
 function createDiv() {
  let newDiv =
    document.createElement("div");
   document.body
     .appendChild(newDiv);
 }

 document.addEventListener(
   "DOMContentLoaded", createDiv);
</script></head>
<body>
 <span></span>
</body>
</html>
```



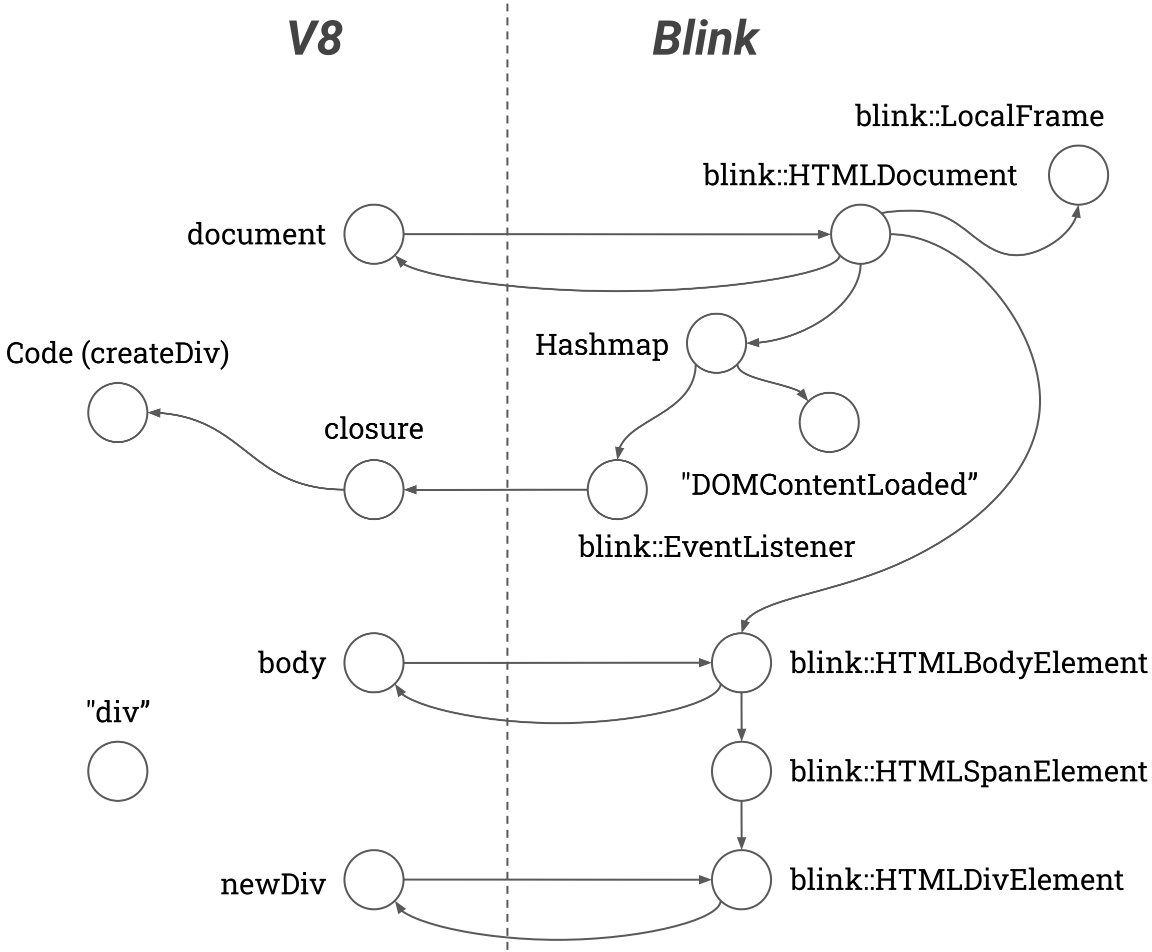
# Example

```
<!DOCTYPE html>
<head><script>
 function createDiv() {
  let newDiv =
    document.createElement("div");
   document.body
     .appendChild(newDiv);
 }

 document.addEventListener(
   "DOMContentLoaded", createDiv);
</script></head>
<body>
 <span></span>
</body>
</html>
```



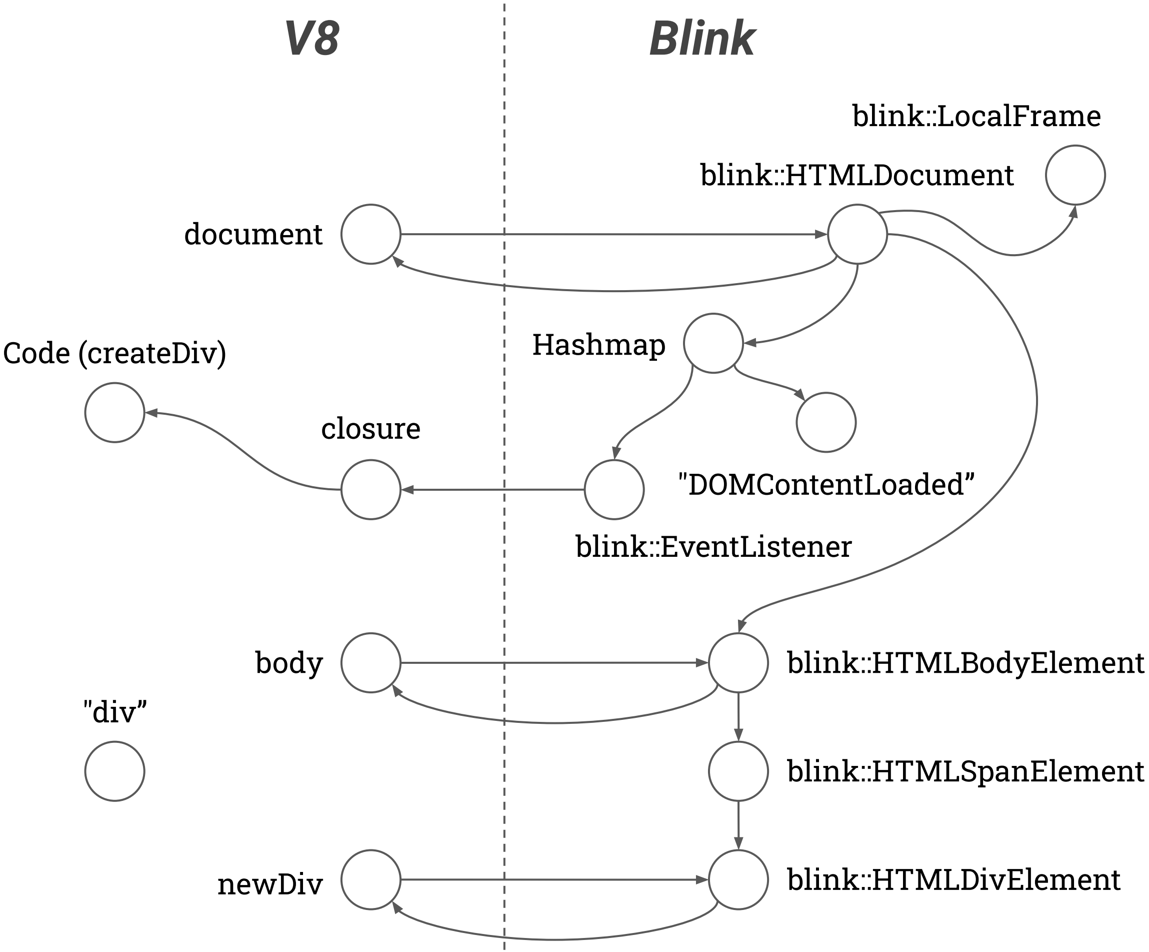
# Example

```
<!DOCTYPE html>
<head><script>
 function createDiv() {
  let newDiv =
     document.createElement("div");
    document.body
      .appendChild(newDiv);
 }

 document.addEventListener(
    "DOMContentLoaded", createDiv);
</script></head>
<body>
 <span></span>
</body>
</html>
```



# Example

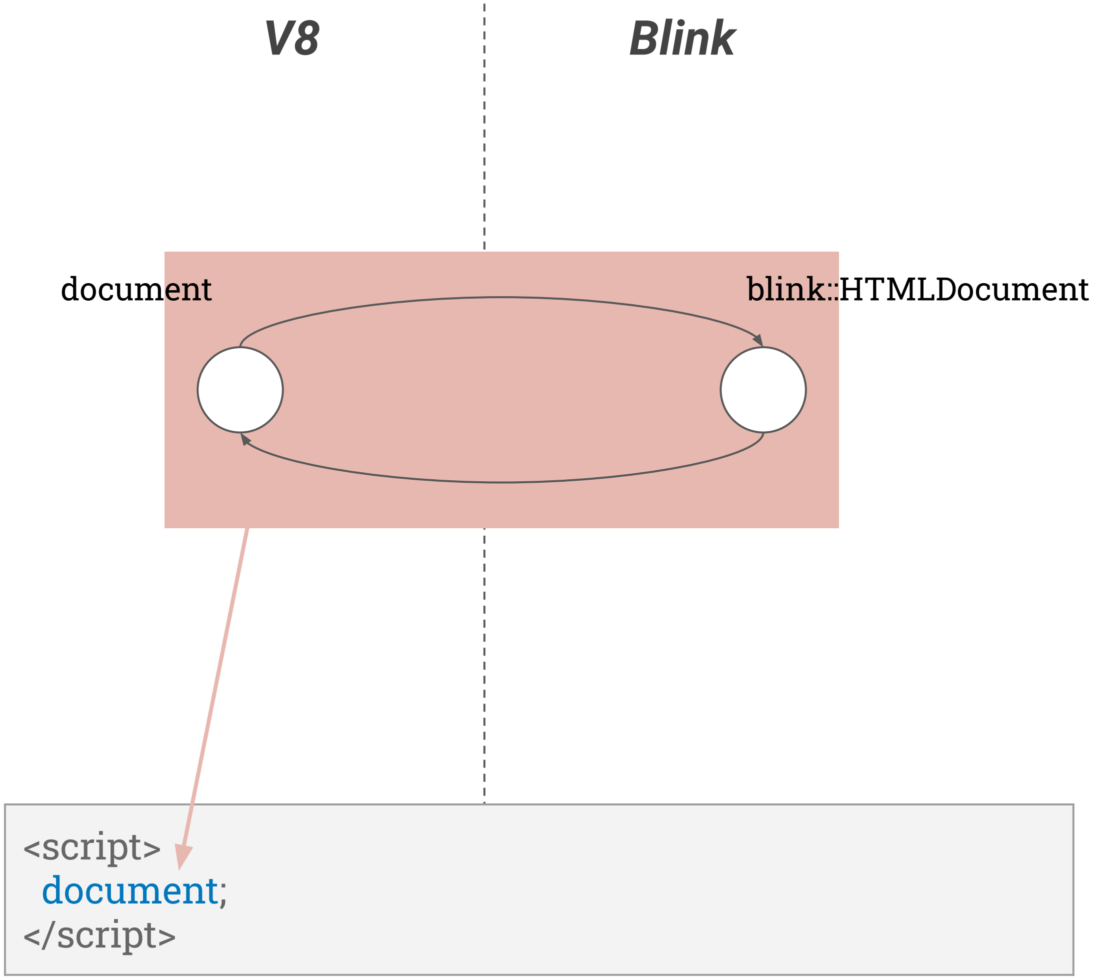
```
<!DOCTYPE html>
<head><script>
 function createDiv() {
  let newDiv =
      document.createElement("div");
    document.body
       .appendChild(newDiv);
 }

 document.addEventListener(
    "DOMContentLoaded", createDiv);
</script></head>
<body>
 <span></span>
</body>
</html>
```



Take a deep breath.

## Where is the problem?

# V8 & Blink

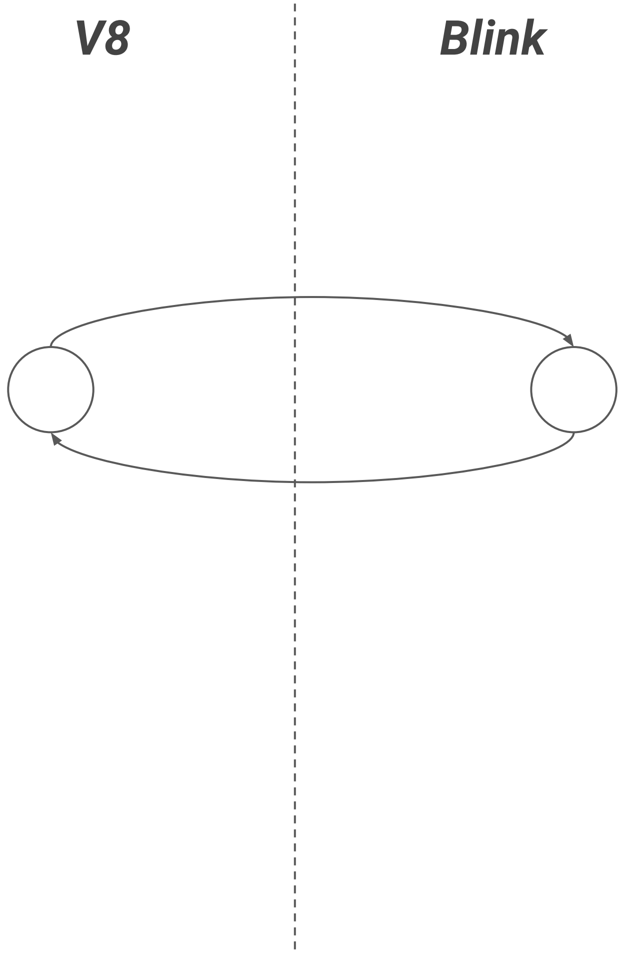


```
<script>
 document;
</script>
```

# V8 & Blink

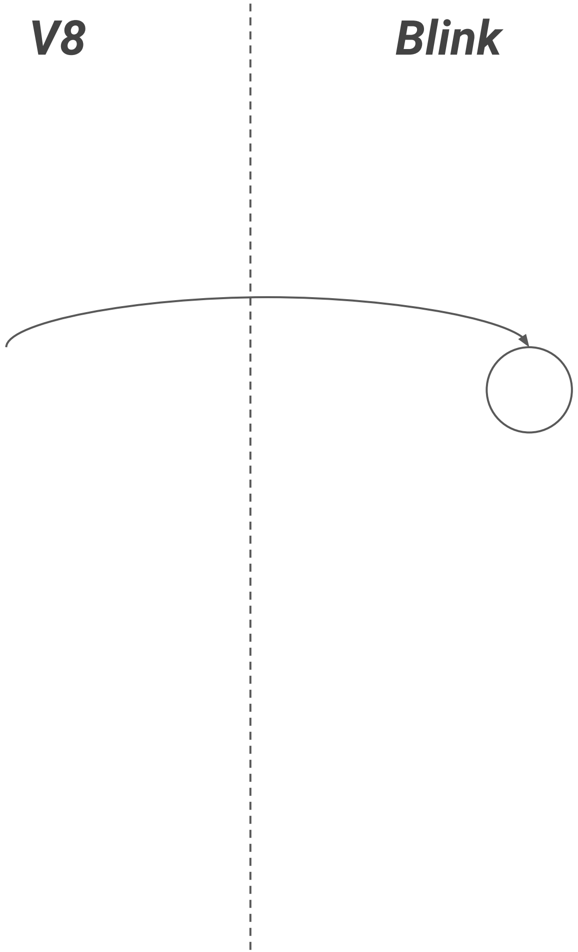
- Two components with separate managed heaps

*Conceptually, **cannot** be root references because that would form cycles*



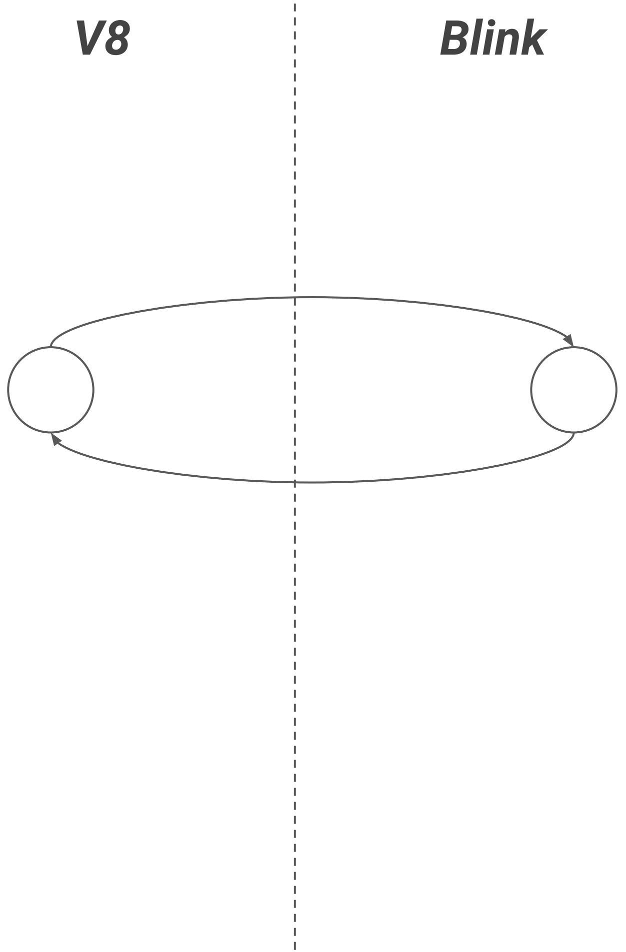
# Blink’s view

Treats incoming references as roots

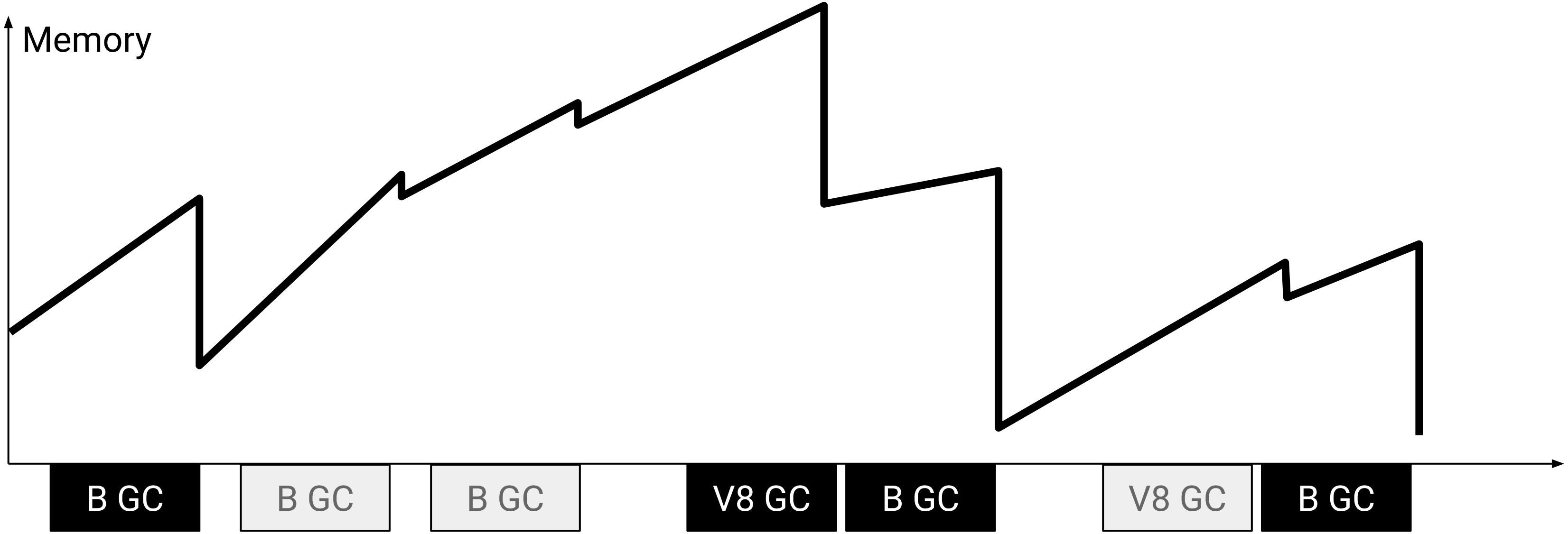


# V8’s view

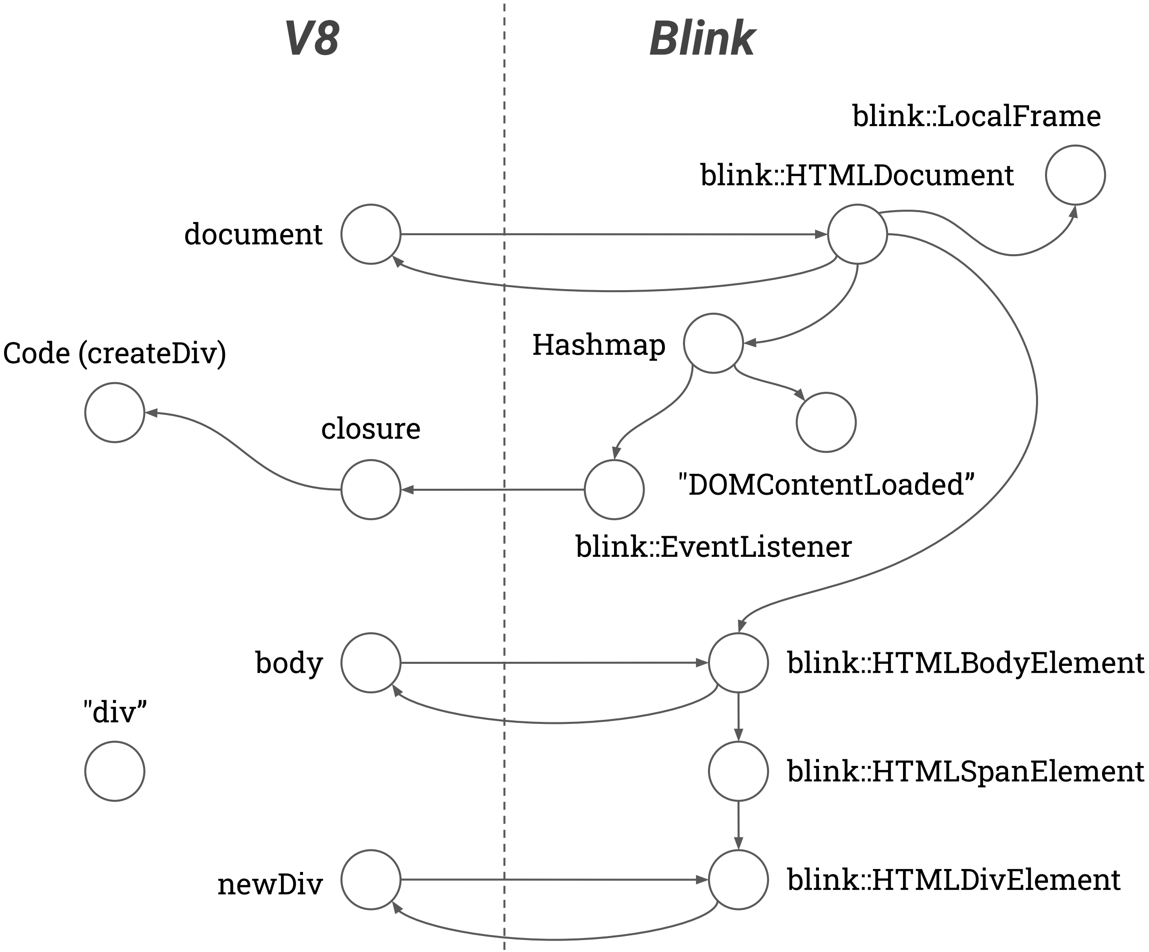
Trace *relevant* transitive closure (wrapper tracing)



# Scheduling problem



# Alternative world

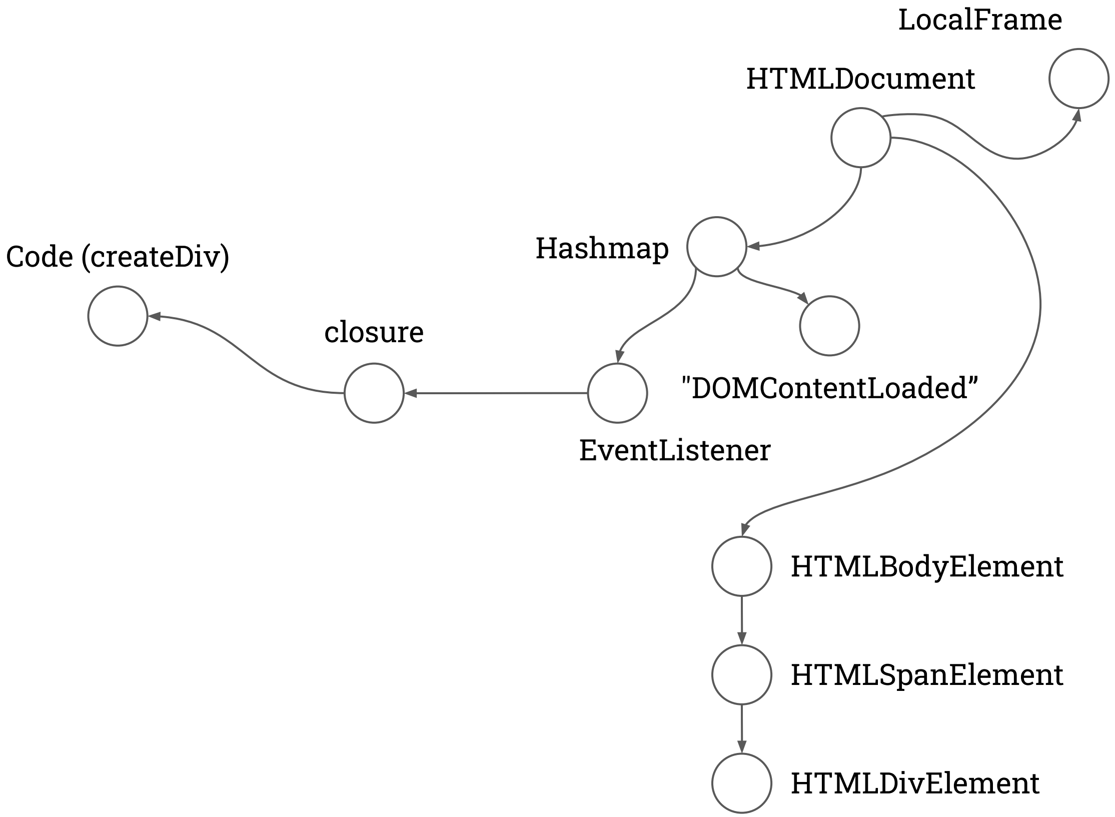


# Alternative world

Only have a single managed heap for all renderer memory

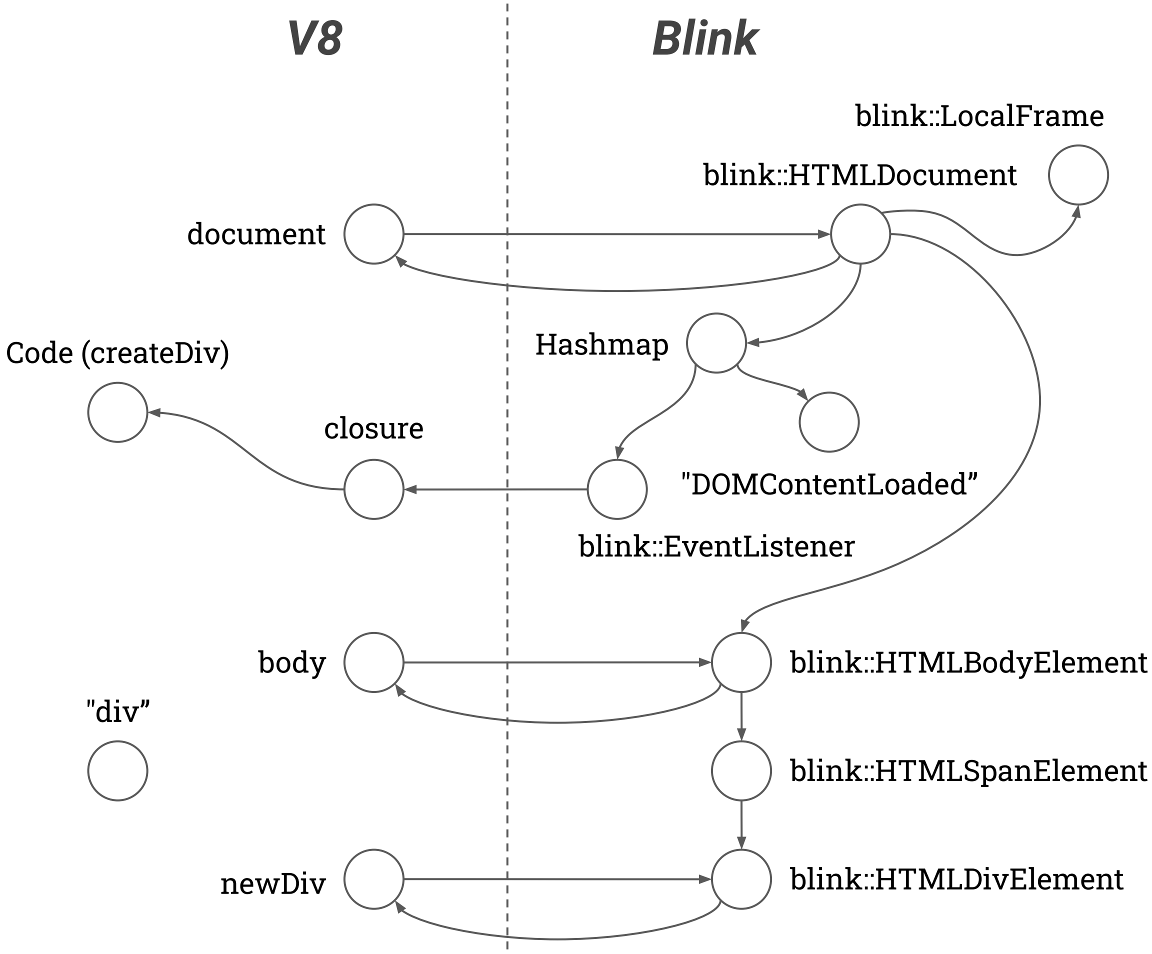
(Blink in JS)

## *Renderer heap*

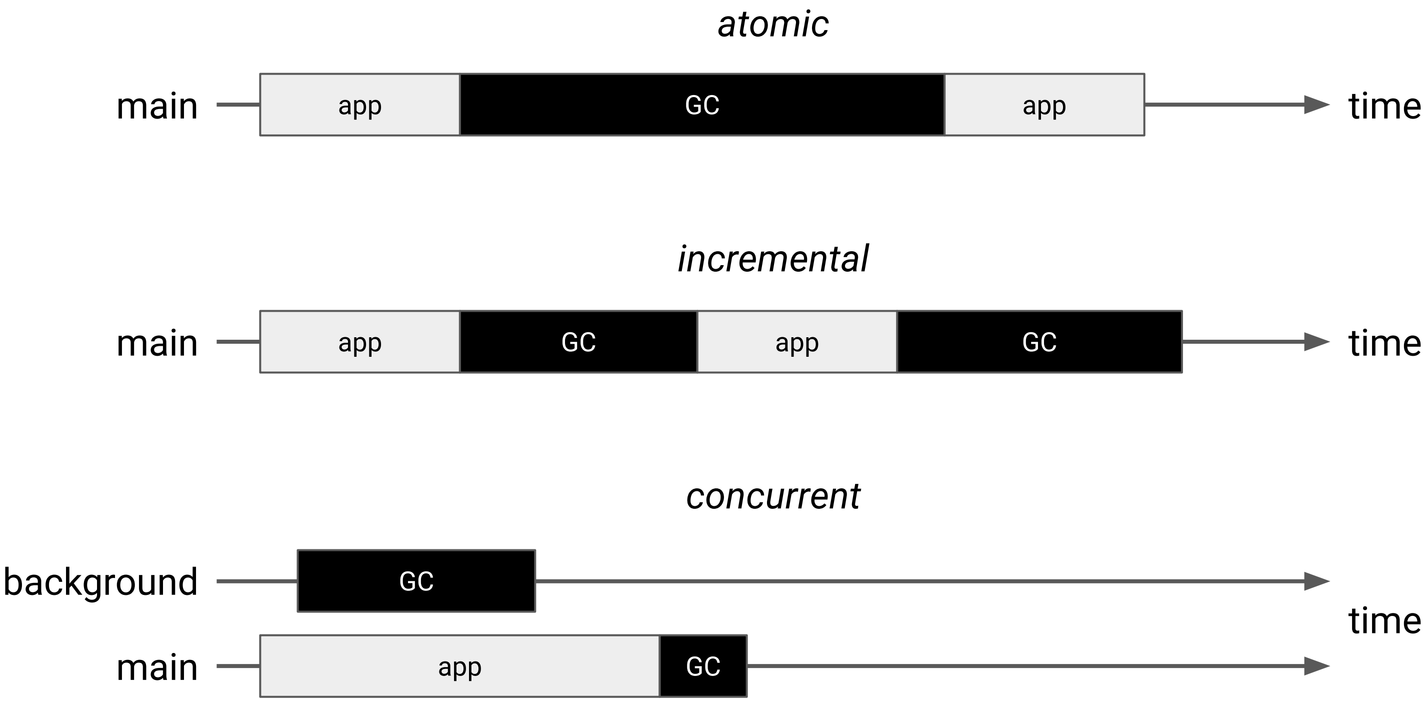


# Unified heap

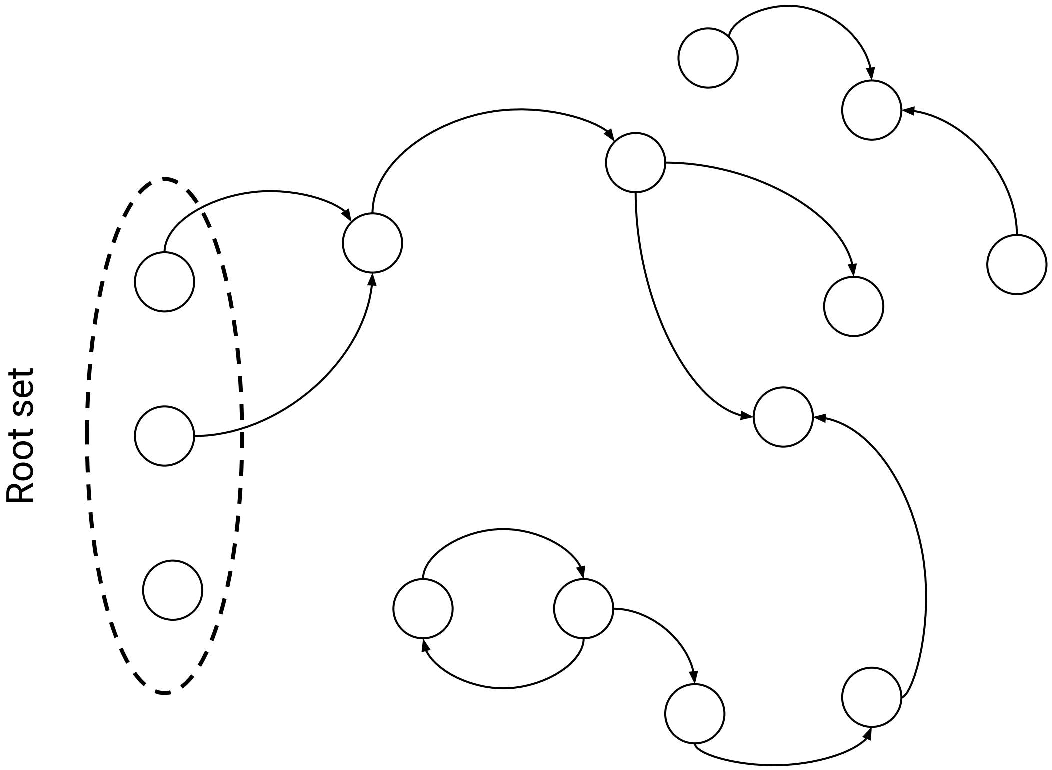
- Compromise
- Keep separate heap implementations
    - Blink: Oilpan
    - V8: Orinoco
- Allow **full** garbage collections across those component boundaries
- Deprecate wrapper tracing



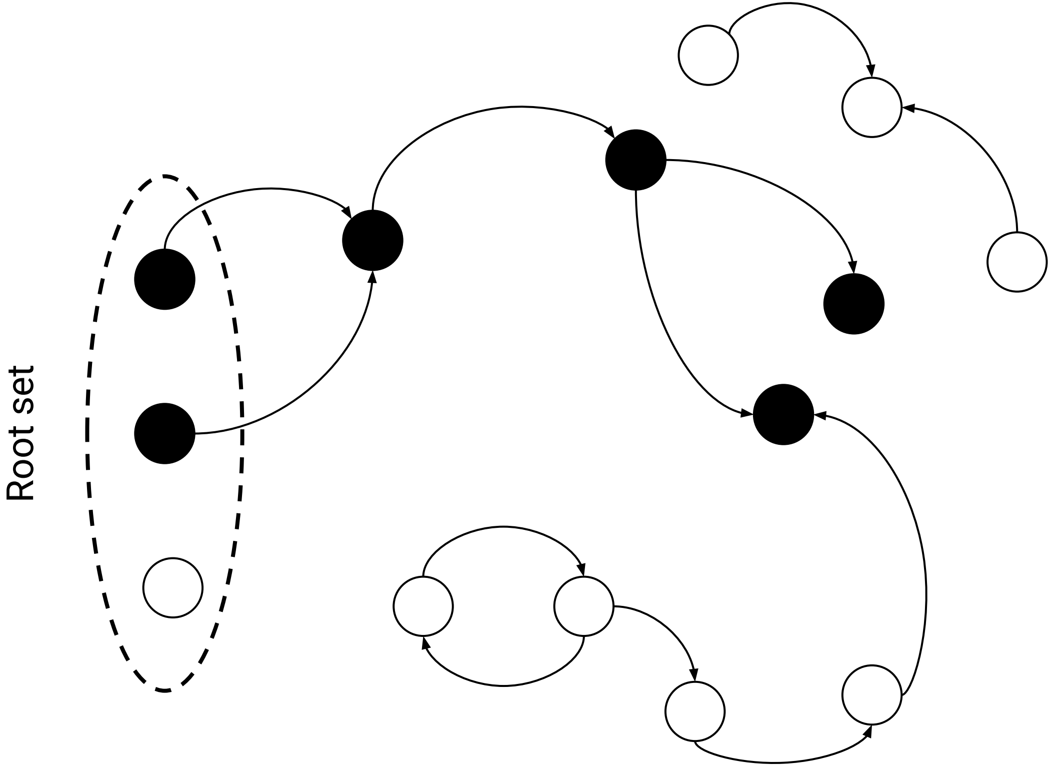
# Interlude: Garbage Collection



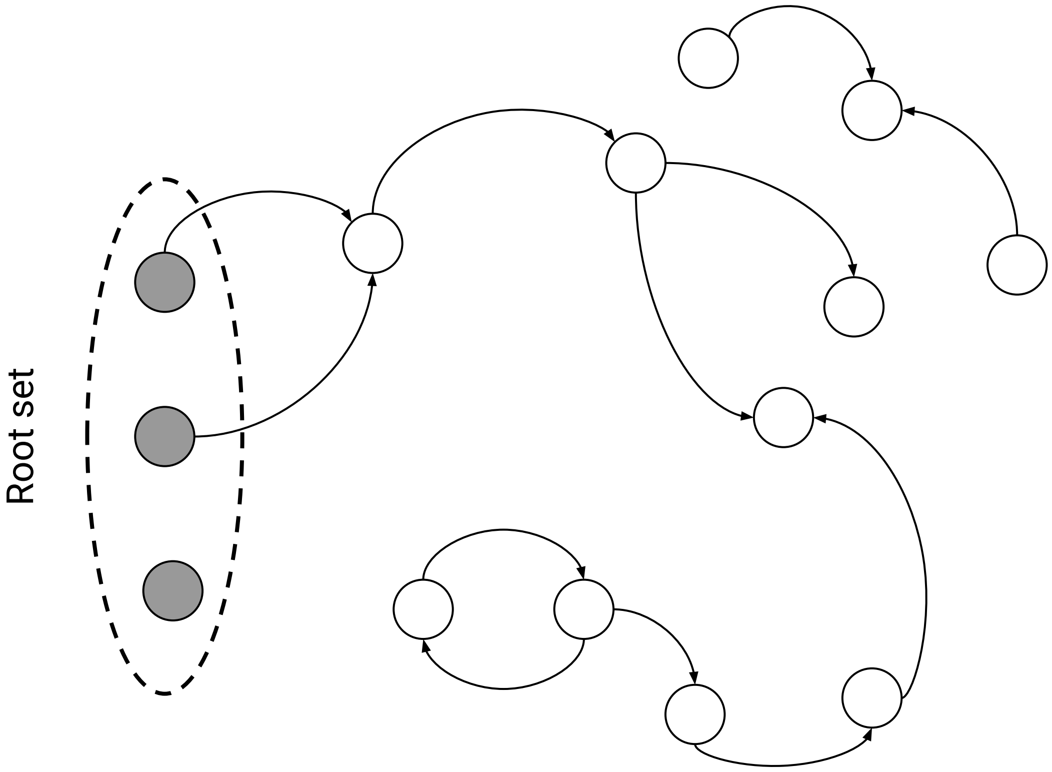
# Interlude: Marking



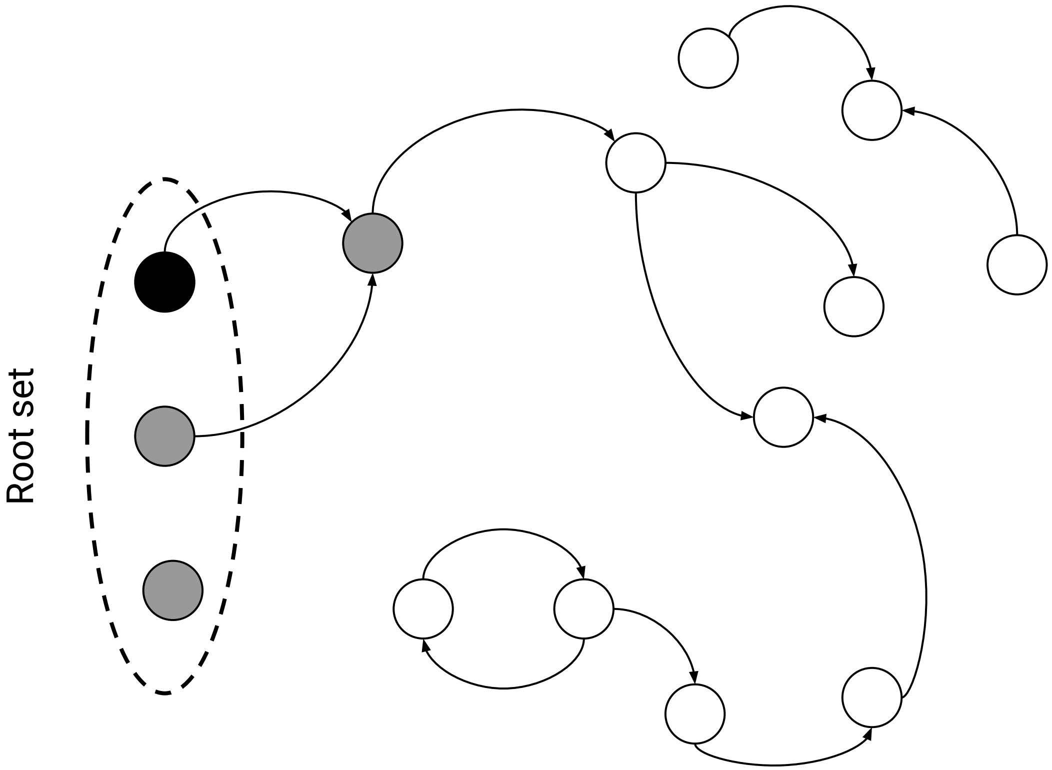
# Interlude: (Atomic) Marking



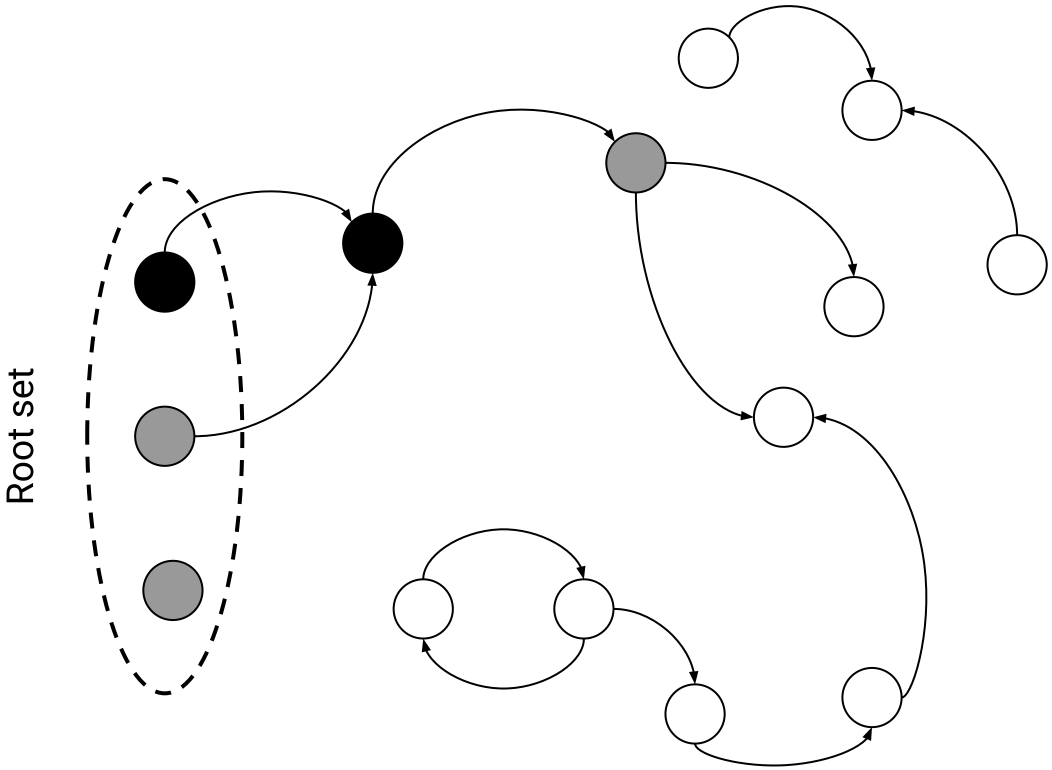
# Interlude: Incremental Marking



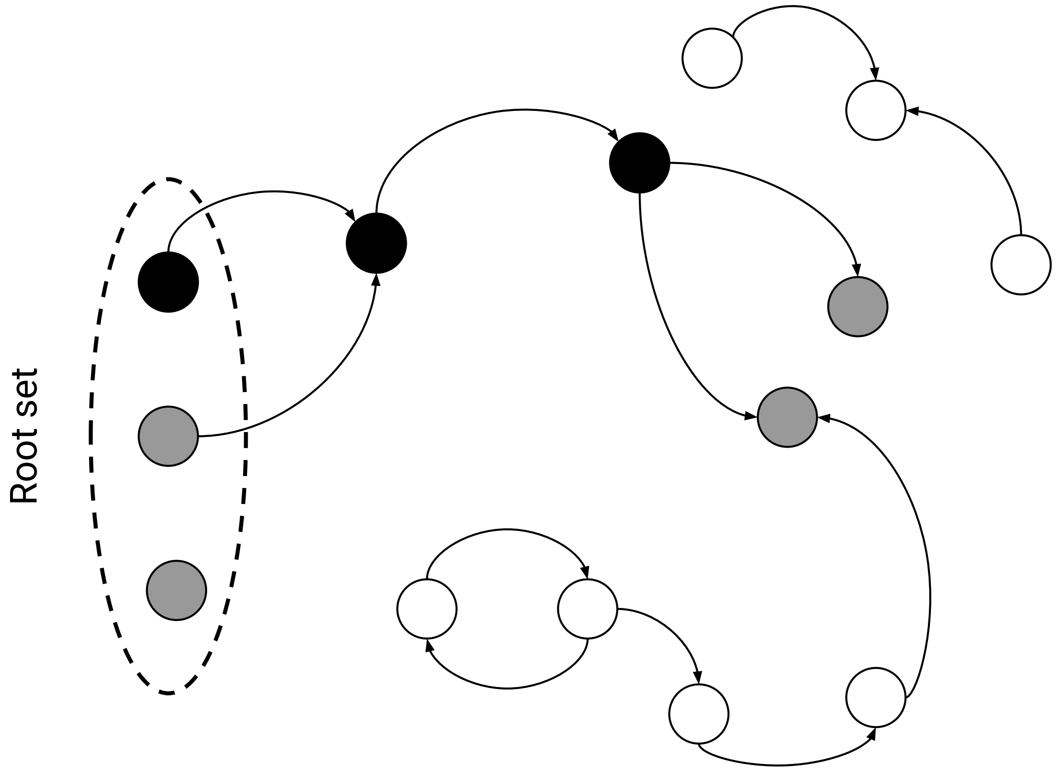
# Interlude: Incremental Marking



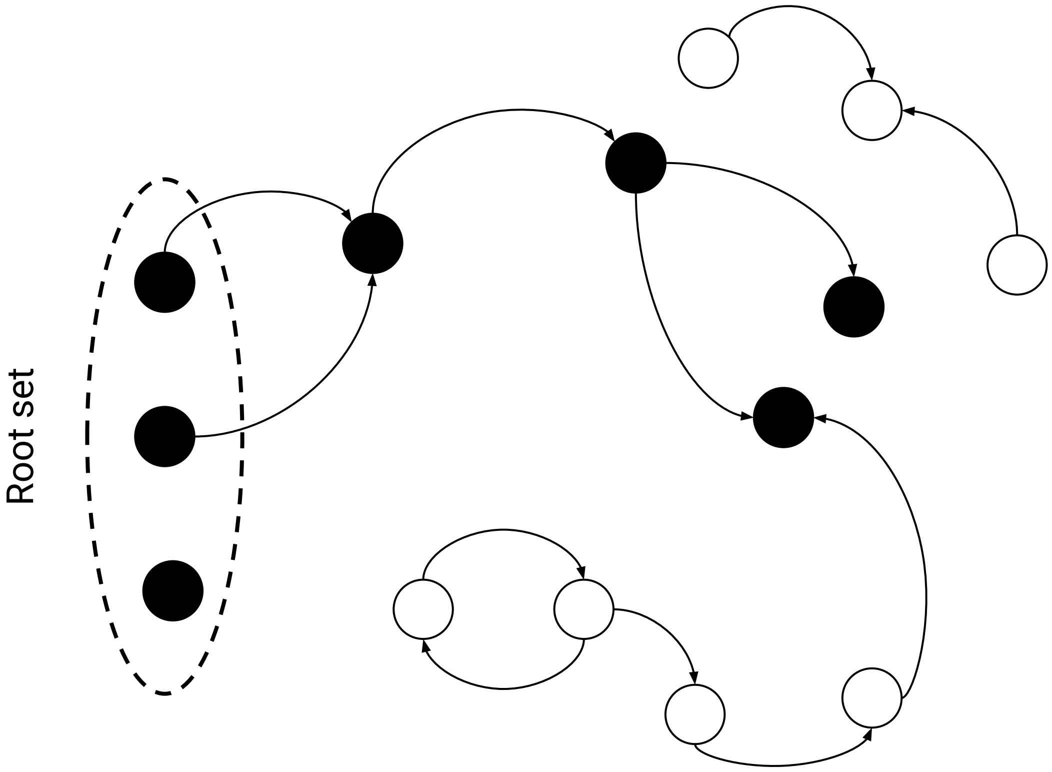
# Interlude: Incremental Marking



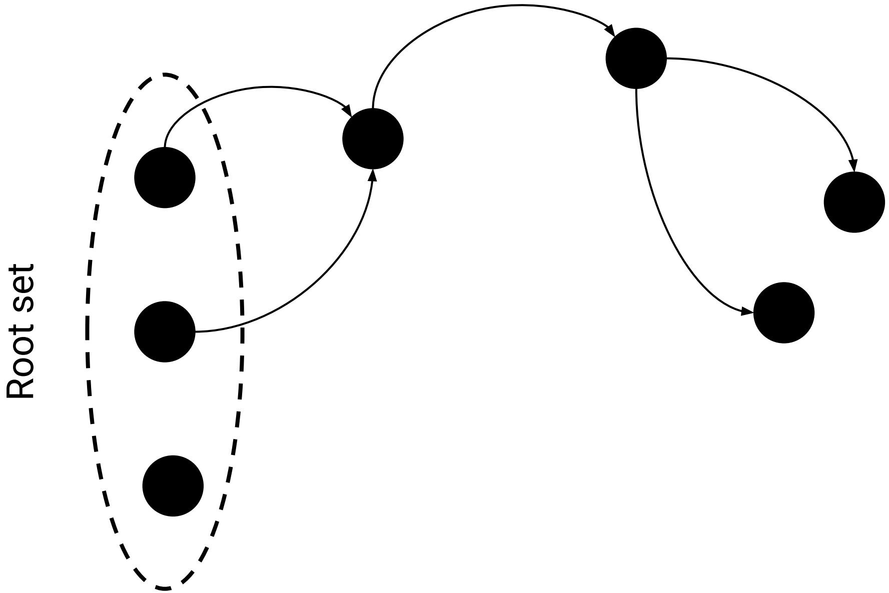
# Interlude: Incremental Marking



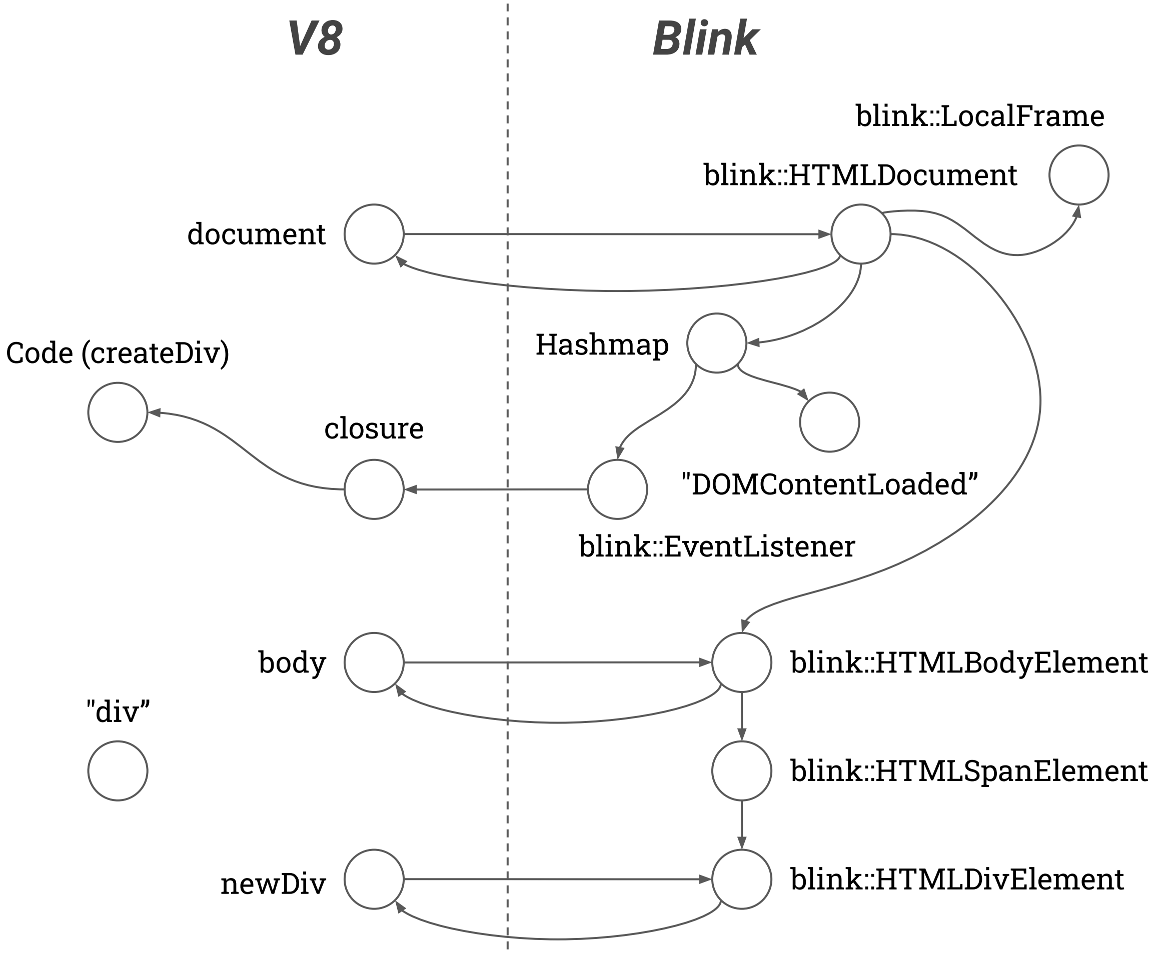
# Interlude: Incremental Marking



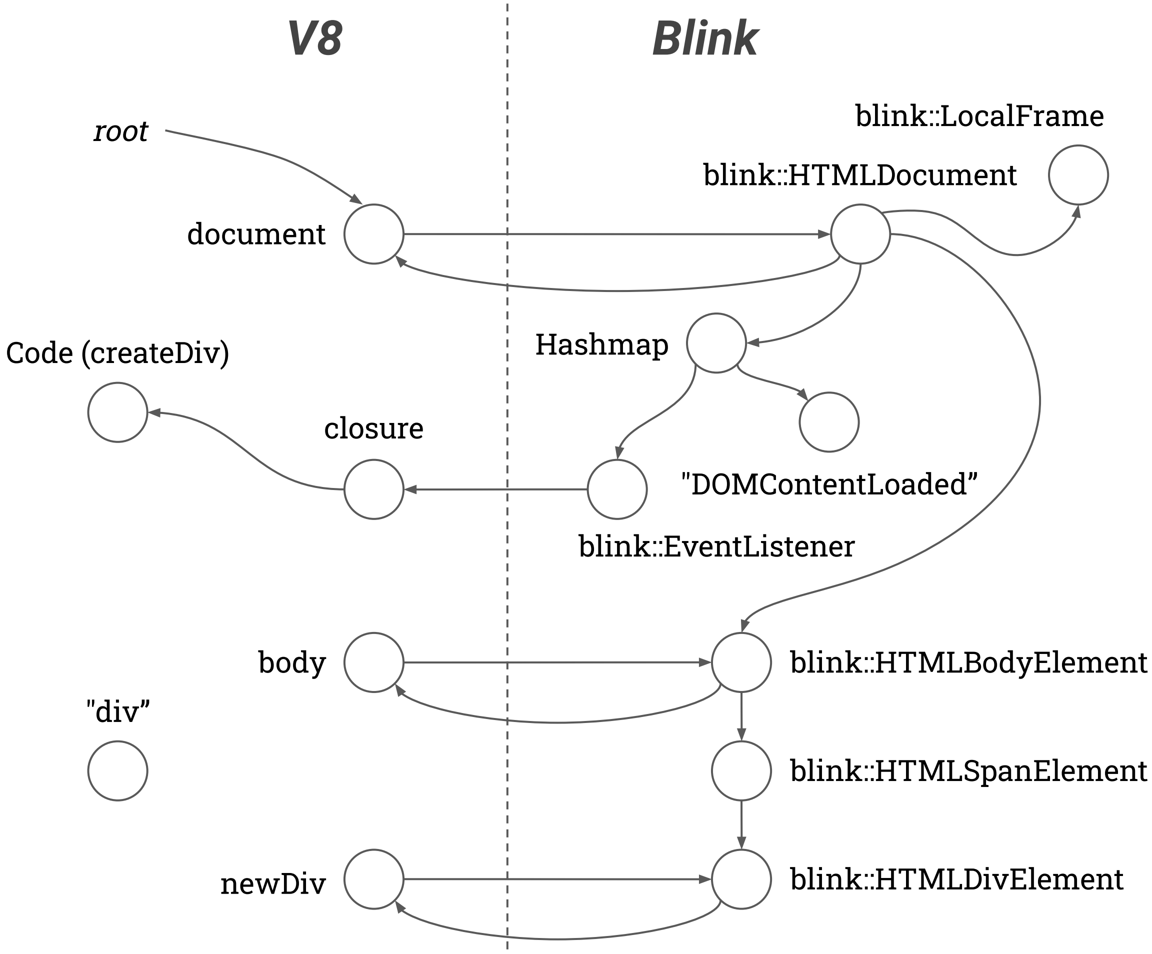
# Interlude: Sweeping



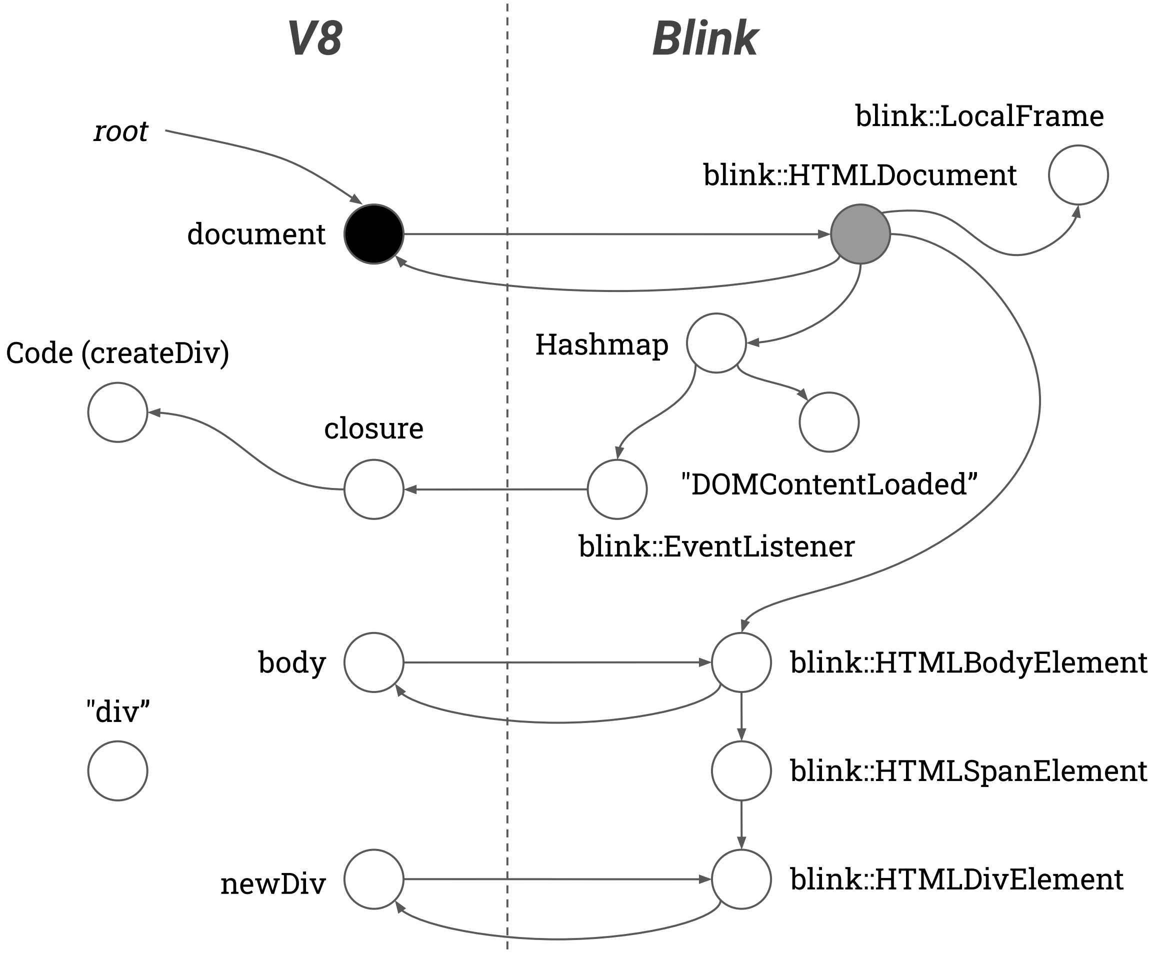
# Unified heap



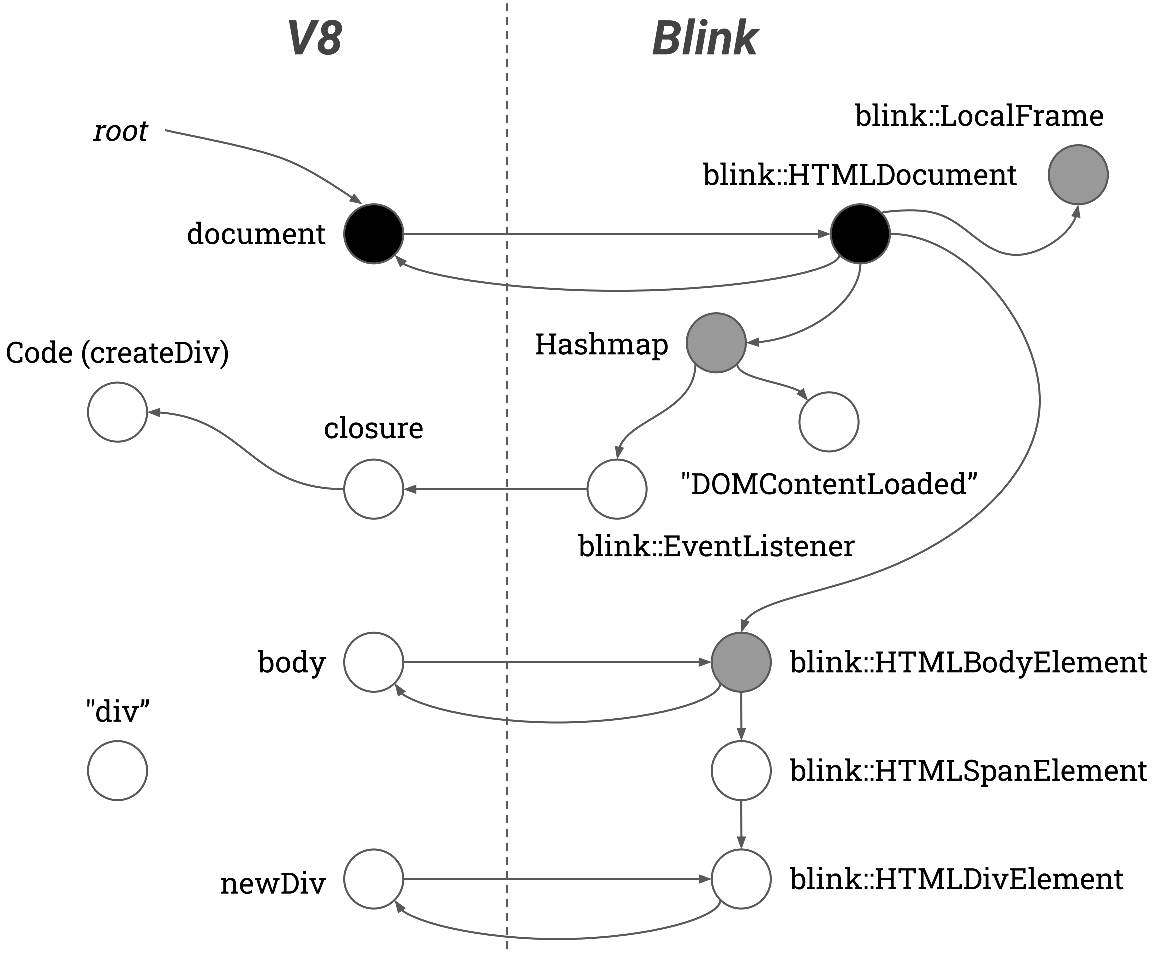
# Unified heap



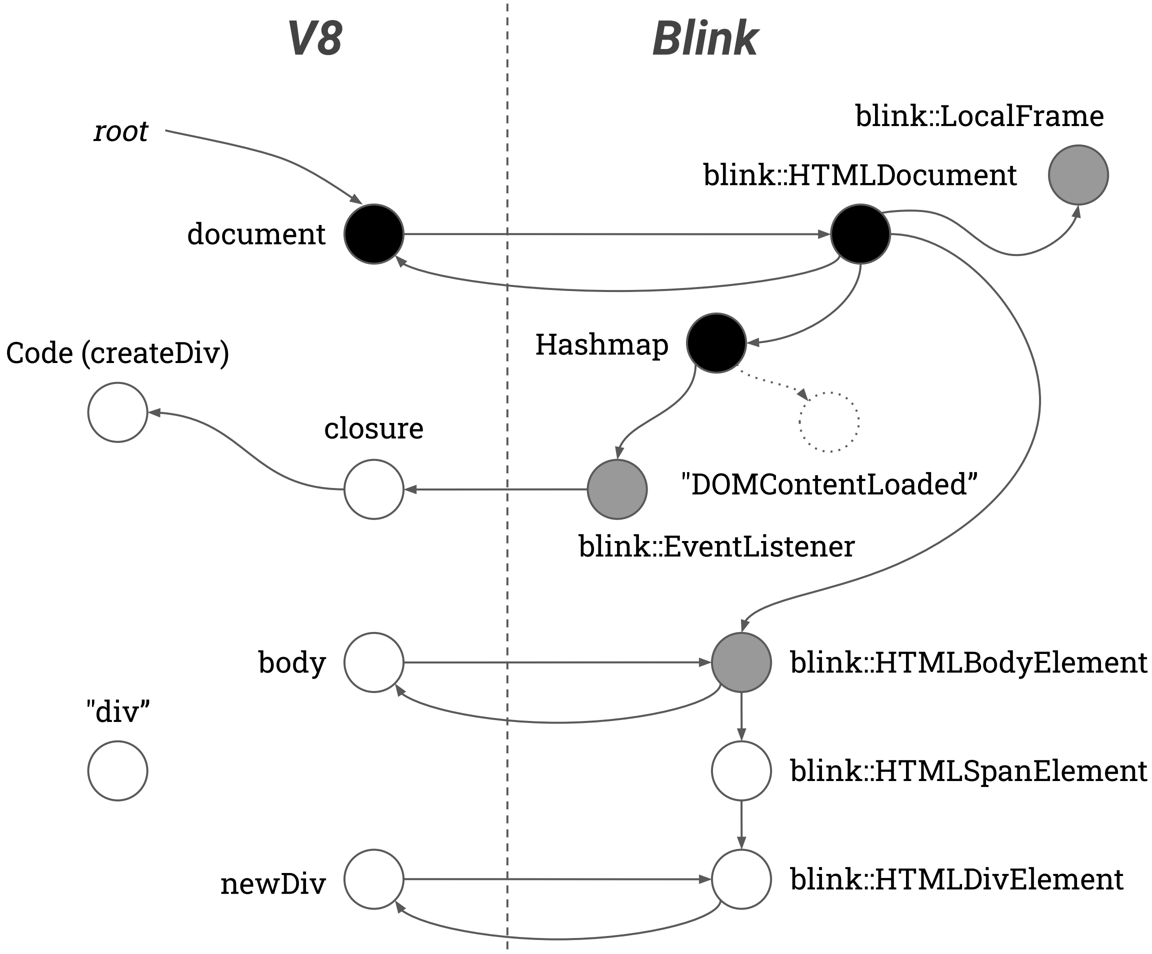
# V8


V8
Blink
root
blink::LocalFrame
blink::HTMLDocument
document
Hashmap
Code (createDiv)
closure
"DOMContentLoaded”
blink::EventListener
body
blink::HTMLBodyElement
"div”
blink::HTMLSpanElement
newDiv
blink::HTMLDivElement

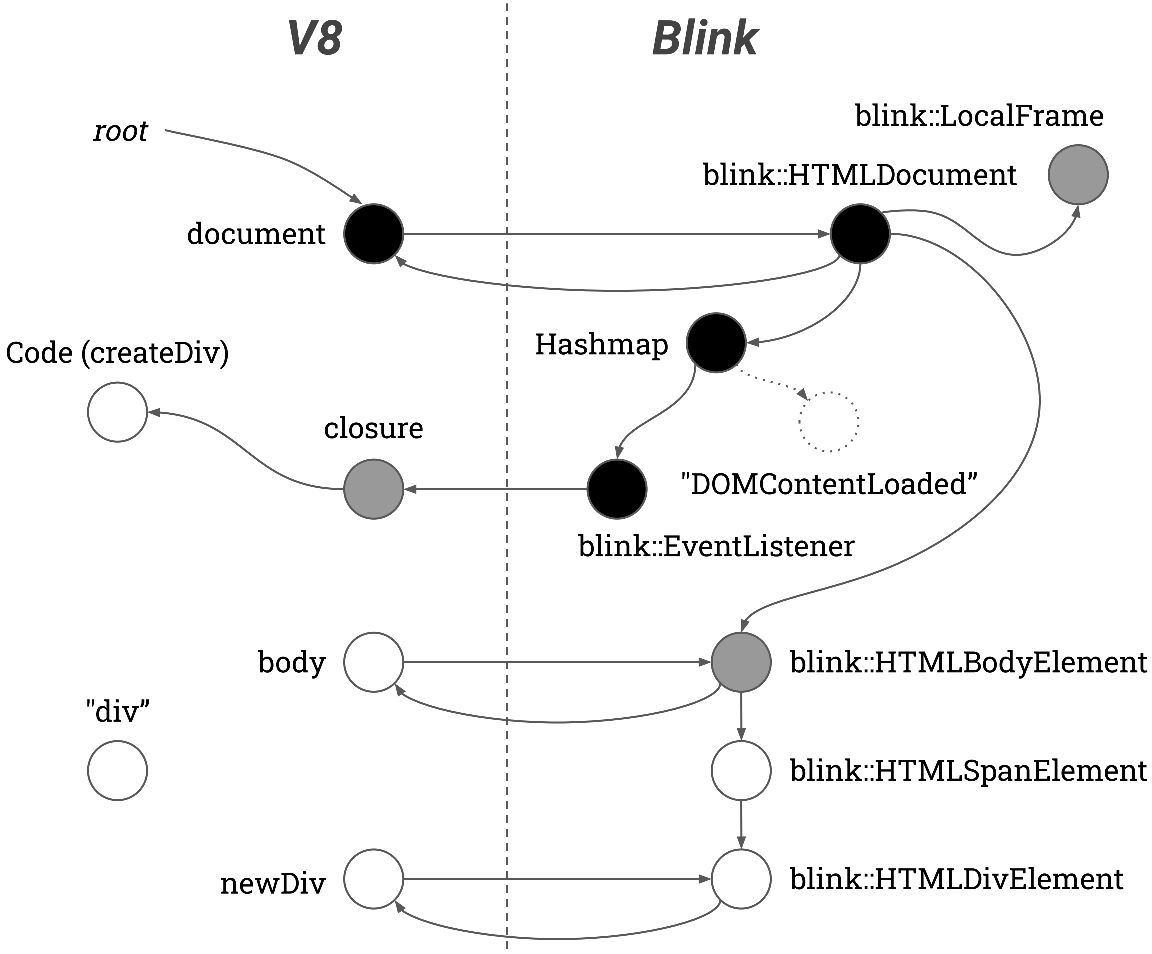
# Blink


V8
Blink
root
blink::LocalFrame
blink::HTMLDocument
document
Code (createDiv)
Hashmap
closure
"DOMContentLoaded"
blink::EventListener
body
blink::HTMLBodyElement
"div"
blink::HTMLSpanElement
newDiv
blink::HTMLDivElement

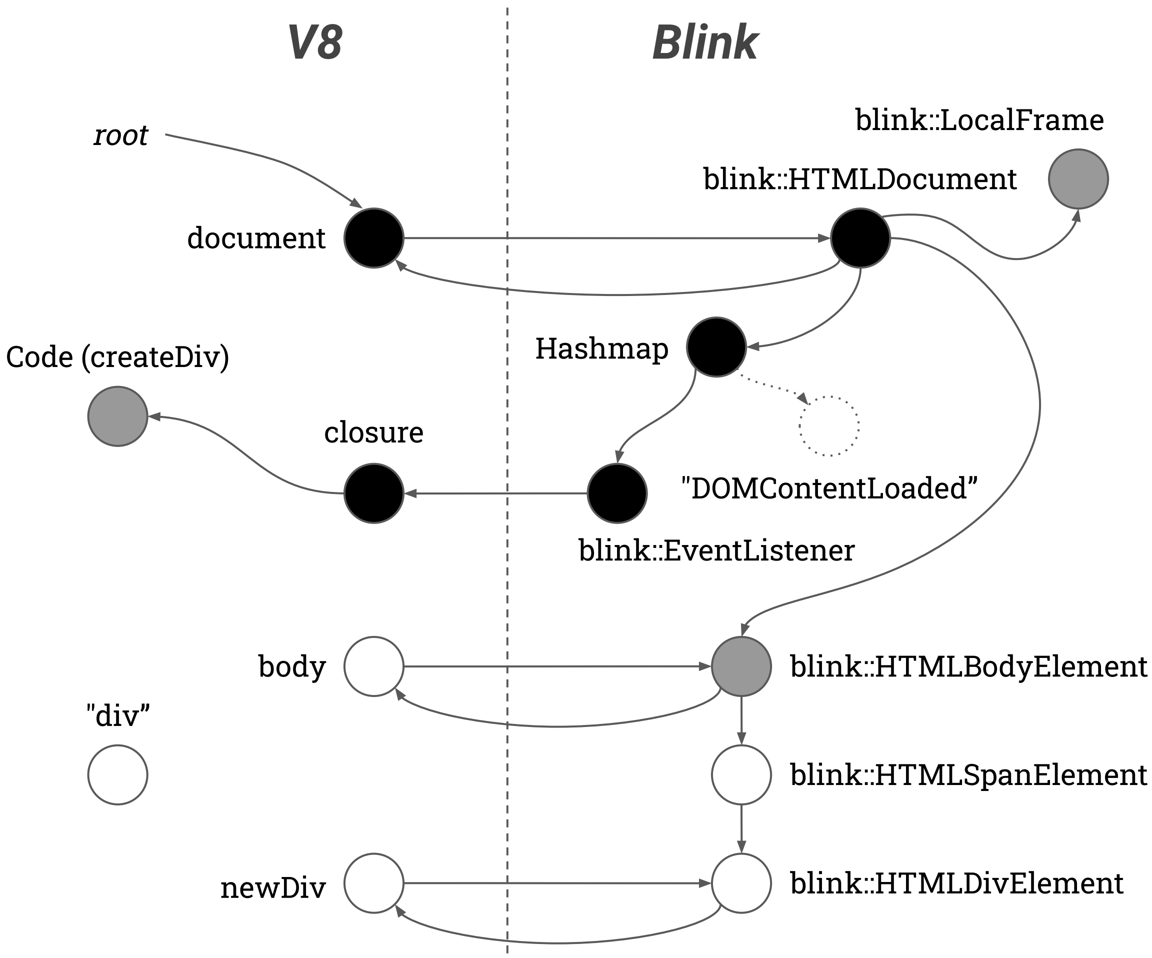
# Blink


V8
Blink
root
document
blink::LocalFrame
blink::HTMLDocument
Hashmap
Code (createDiv)
closure
"DOMContentLoaded”
blink::EventListener
body
blink::HTMLBodyElement
"div”
blink::HTMLSpanElement
newDiv
blink::HTMLDivElement

# Blink

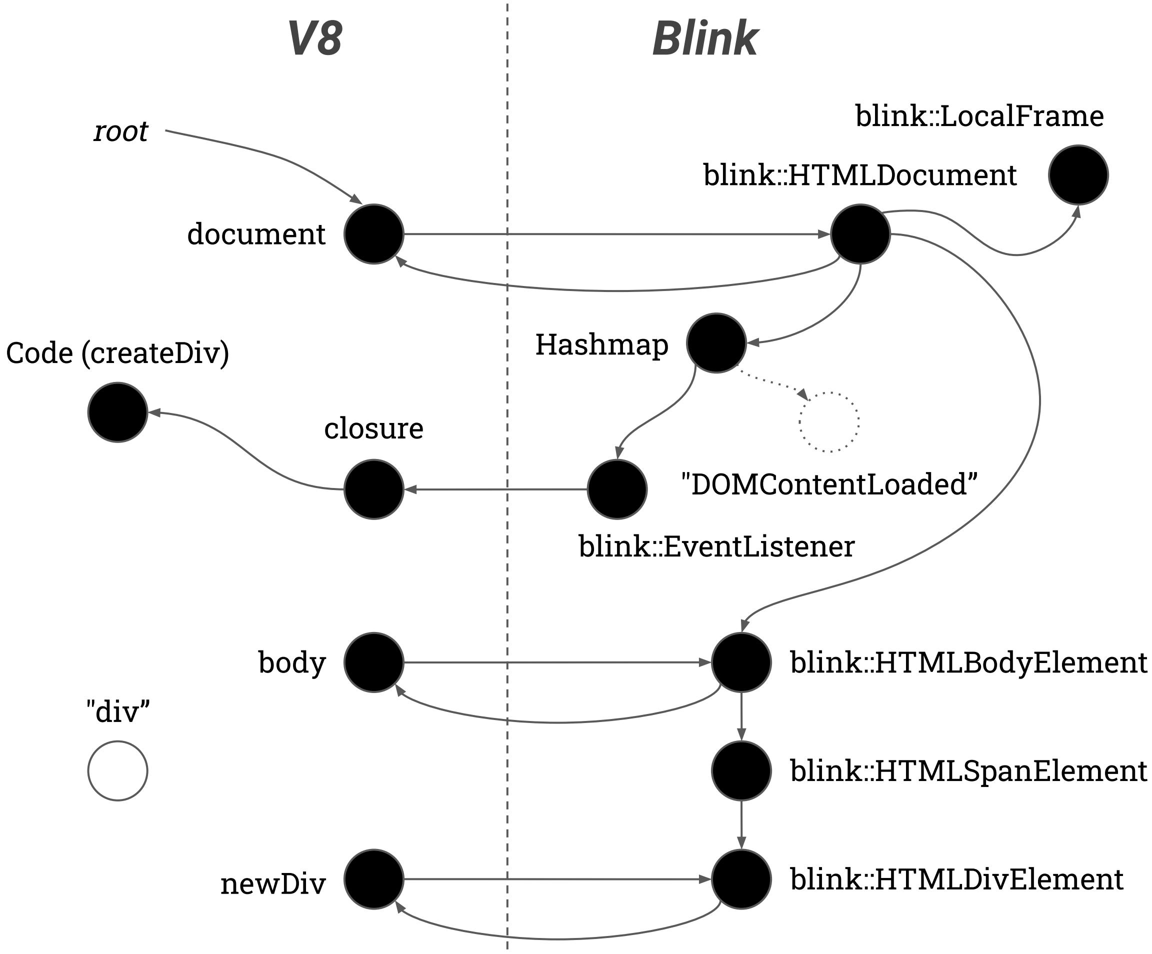

V8
Blink
root
document
blink::LocalFrame
blink::HTMLDocument
Code (createDiv)
Hashmap
closure
"DOMContentLoaded"
blink::EventListener
body
blink::HTMLBodyElement
"div"
blink::HTMLSpanElement
newDiv
blink::HTMLDivElement

# V8

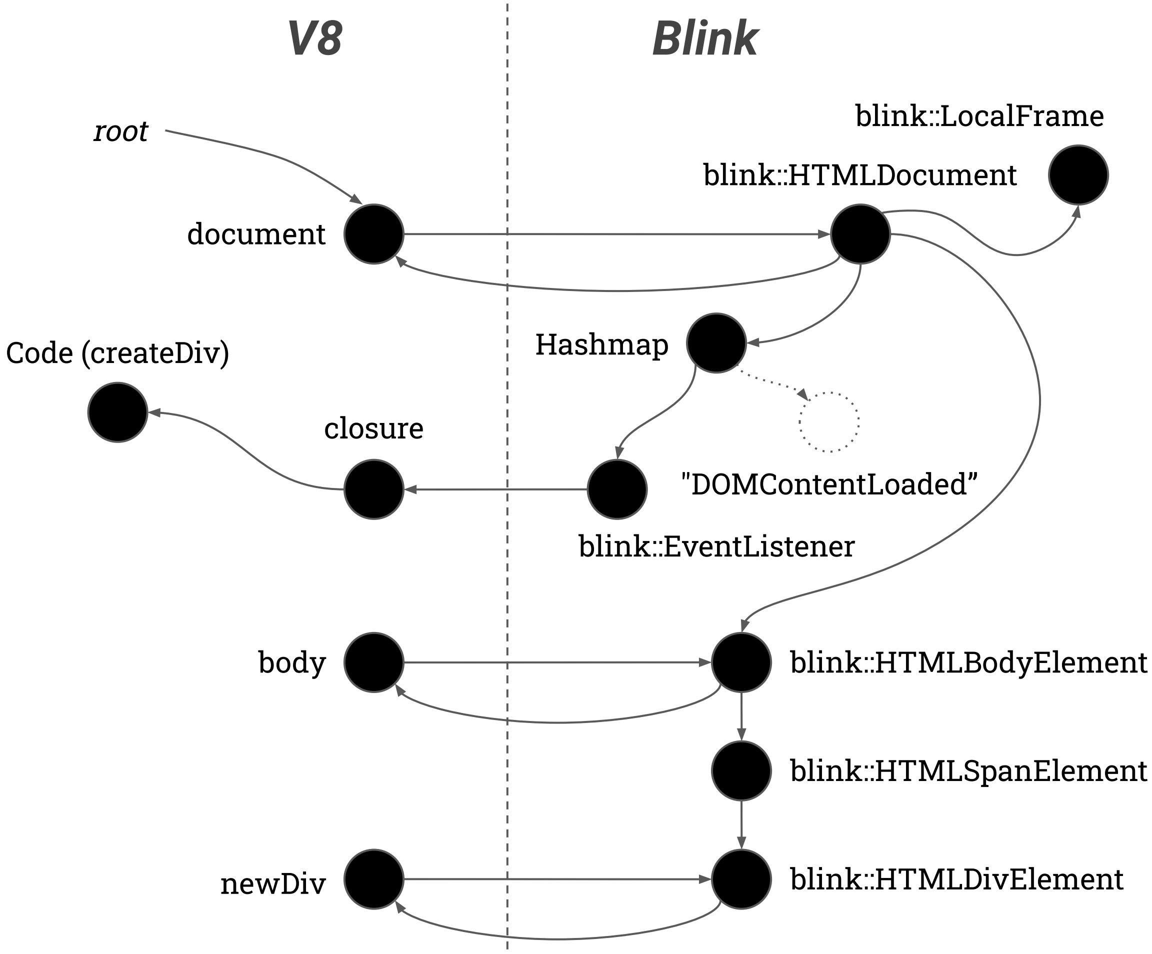

V8
Blink
root
blink::LocalFrame
blink::HTMLDocument
document
Code (createDiv)
Hashmap
closure
"DOMContentLoaded”
blink::EventListener
body
blink::HTMLBodyElement
"div”
blink::HTMLSpanElement
newDiv
blink::HTMLDivElement

# V8 & Blink

Concurrently...

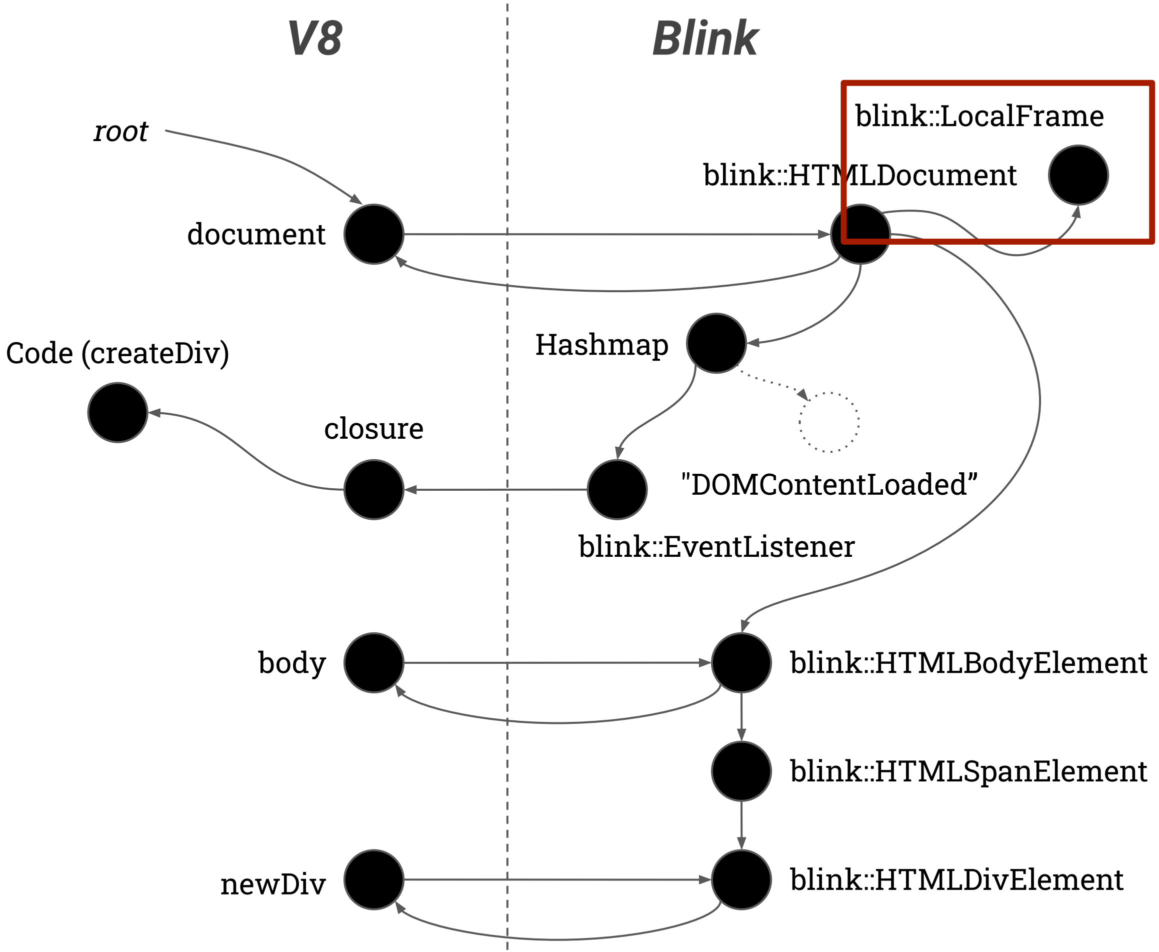


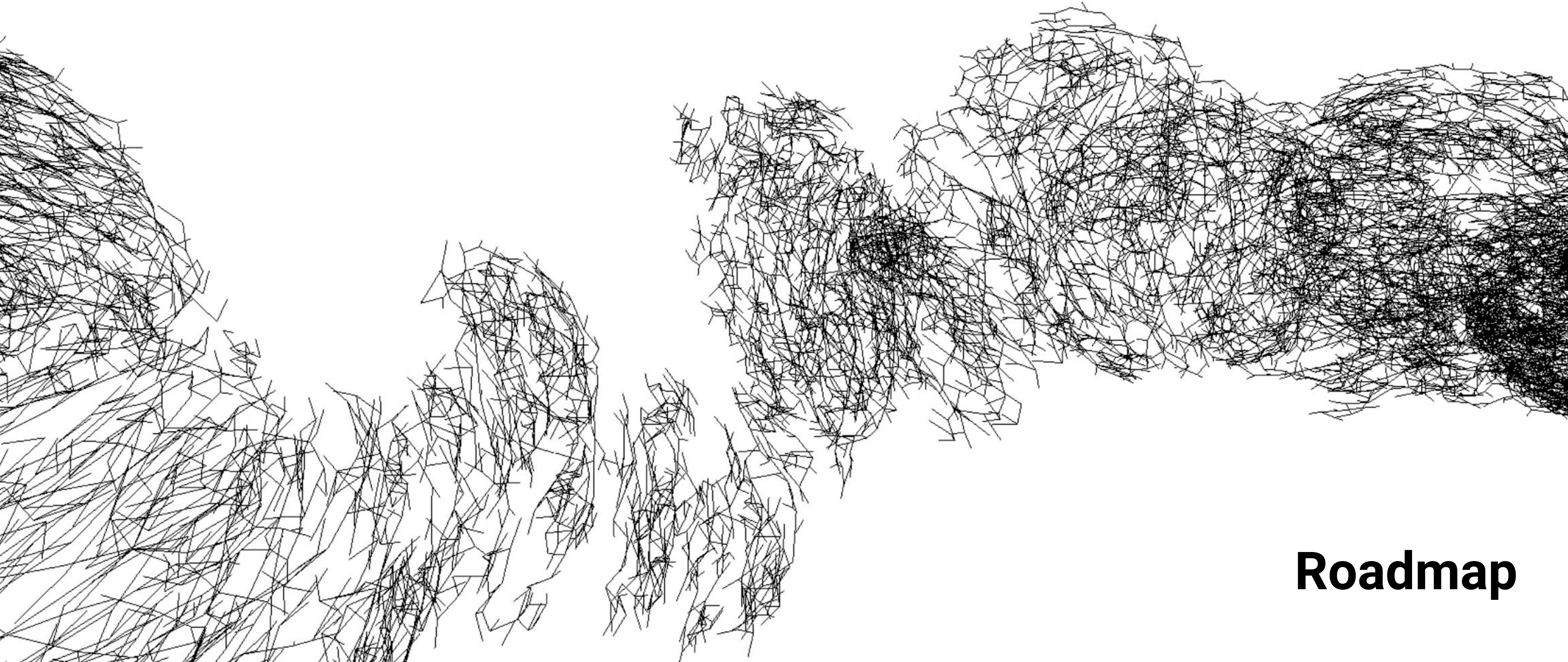
# Unified heap



# Wrapper tracing Δ

- Proper stack handling in all code paths
- Covers *all* paths on Oilpan's heap
- Allows sweeping both, V8 and Blink after marking





# Roadmap

# Unified Heap

*Requirement: Low-latency on garbage collection operations*

- **V8**
  - Concurrent marking ✓
  - Concurrent sweeping ✓
  - Parallel compaction ✓
- **Oilpan**
  - Non-incremental marking ❌
  - Incremental sweeping ±

# Roadmap

**Oilpan: Incremental Marking**

**(Oilpan: Concurrent Marking)**

**Unified Heap**

# Oilpan: Incremental Marking

- No changes in non-platform Blink code
- Implemented basic infrastructure
  - gn arg: `enable_blink_heap_incremental_marking`
- Unit test suite for
  - Write barriers
  - WTF collection integration
- Verification of mark bits
  - gn arg: `enable_blink_heap_verification`
  - Verifies after marking that there are no transitions from marked to unmarked objects
  - Landed on CI (fyi) for current Oilpan
- Currently working on performance tuning

bit.ly/oilpan-incremental-marking

# Oilpan: Concurrent Marking

- (Ideally) no changes in non-platform Blink code
- Needed immediately if incremental marking performance regresses too much
- Based on incremental marking
  - Same barriers and consistency models
- Concurrent read-only access on object payload
  - Using existing Member<T> abstractions
- Complex object types (e.g. collections) are handled on the main thread

bit.ly/oilpan-concurrent-marking

# Unified Heap

- Currently working on sanitization
- Intermediate step: Merge visitation
    - Visible in non-heap Blink code!

```
void ContainerNode::Trace(
    blink::Visitor* visitor) {
  visitor->Trace(first_child_);
  visitor->Trace(last_child_);
  Node::Trace(visitor);
}
void ContainerNode::TraceWrappers(const
    ScriptWrappableVisitor* visitor) const {
  visitor->TraceWrappers(first_child_);
  visitor->TraceWrappers(last_child_);
  Node::TraceWrappers(visitor);
}
```

# Unified Heap

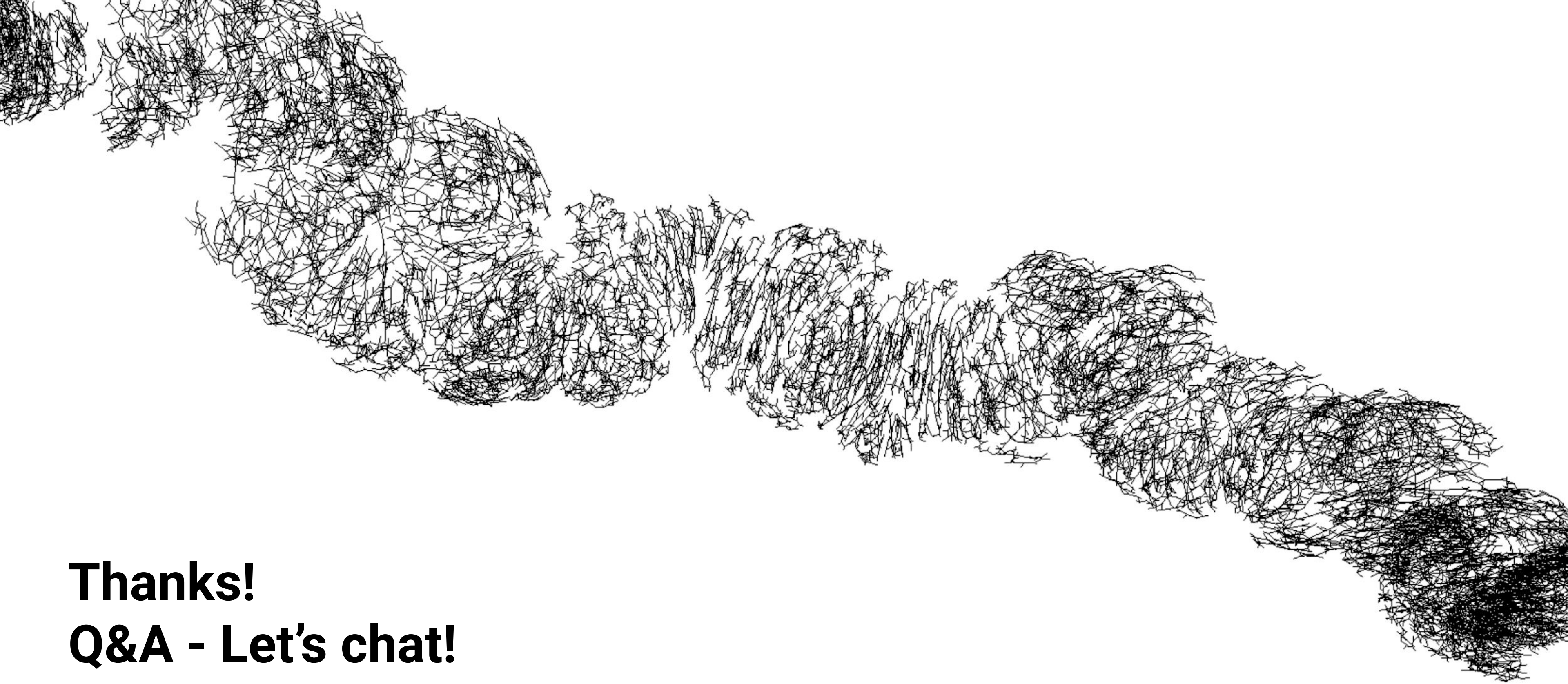
- Currently working on sanitization
- Intermediate step: Merge visitation
    - Visible in non-heap Blink code!

```
void ContainerNode::Trace(
    blink::Visitor* visitor) {
  visitor->Trace(first_child_);
  visitor->Trace(last_child_);
  Node::Trace(visitor);
}
```

~~void ContainerNode::TraceWrappers(const ScriptWrappableVisitor* visitor) const {~~
~~visitor->TraceWrappers(first_child_);~~
~~visitor->TraceWrappers(last_child_);~~
~~Node::TraceWrappers(visitor);~~
~~}~~

# Takeaways

- Memory management across V8 and Blink is tricky
- Unified heap solves this problem in a principled way
- Full garbage collections cover V8 and Blink’s heap
  - Both garbage collectors can start sweeping
- Steps on the way
  - Oilpan Incremental Marking
  - Oilpan Concurrent Marking



# Thanks!
# Q&A - Let’s chat!

mlippautz@
BlinkOn 9, Sunnyvale, Apr 2018