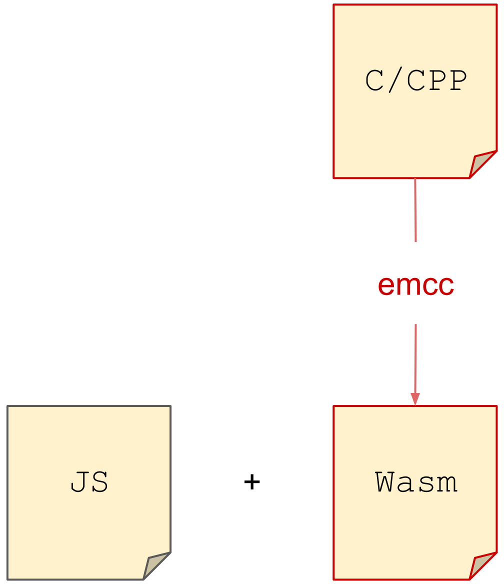
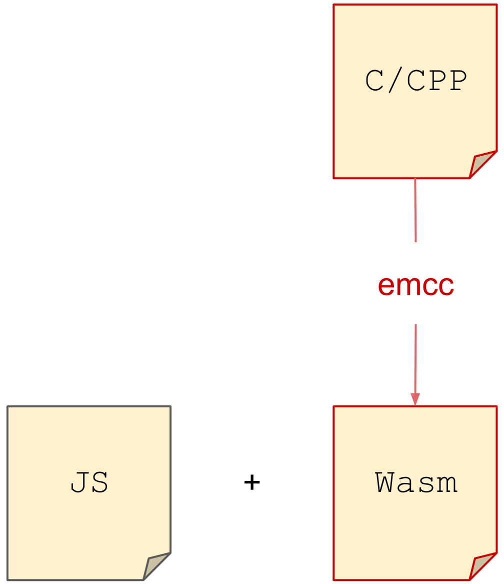
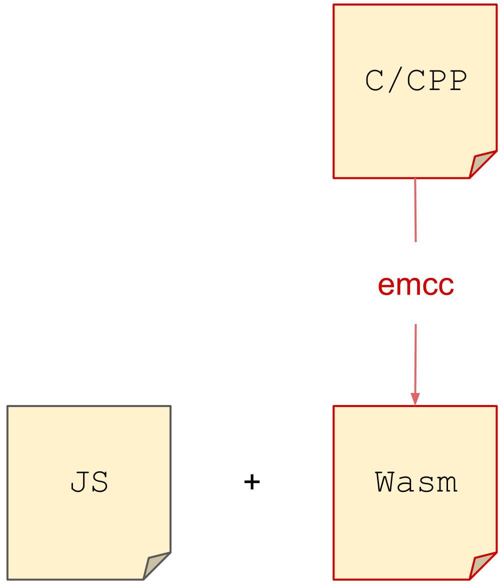

C/CPP
emcc
JS
+
Wasm

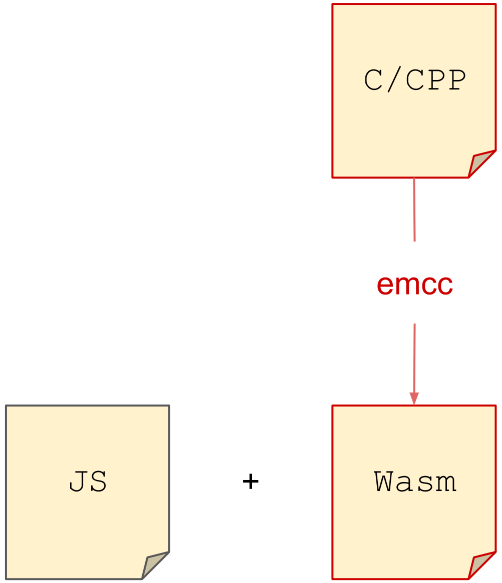


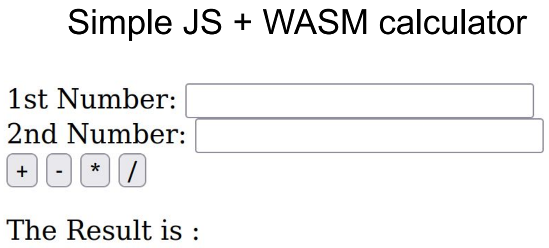
Observation:
DCE is pretty strong
(LLVM DCE + Binaryen DCE)



Observation:
DCE is pretty strong
(LLVM DCE + Binaryen DCE)

How did we come to this conclusion?



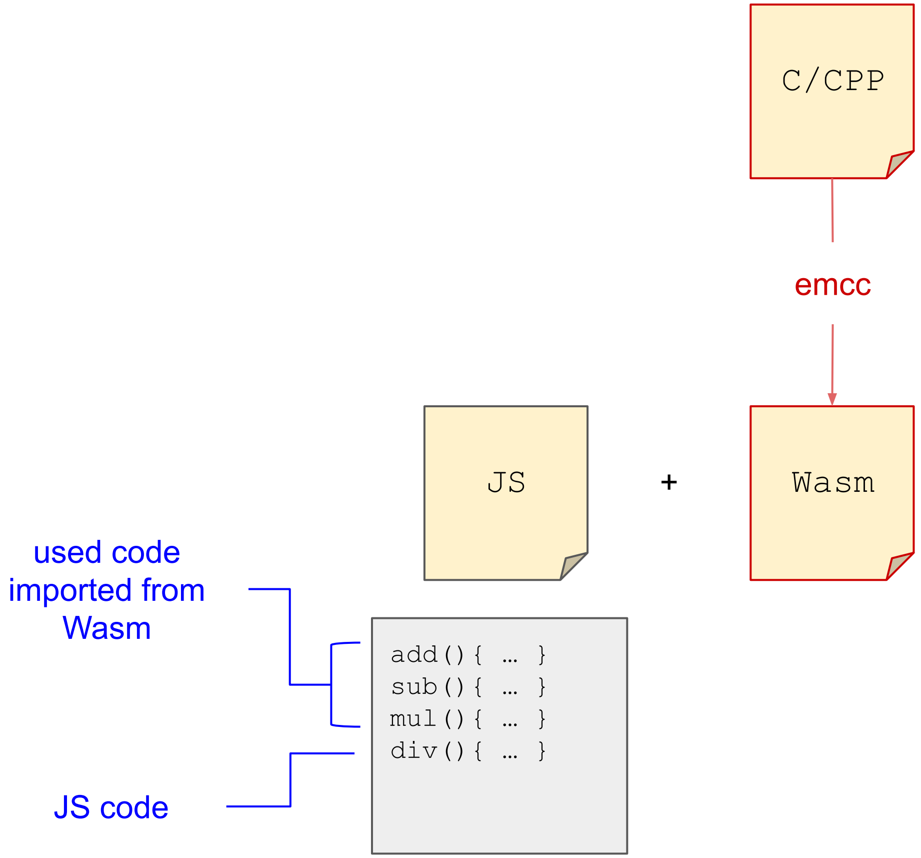


Simple JS + WASM calculator

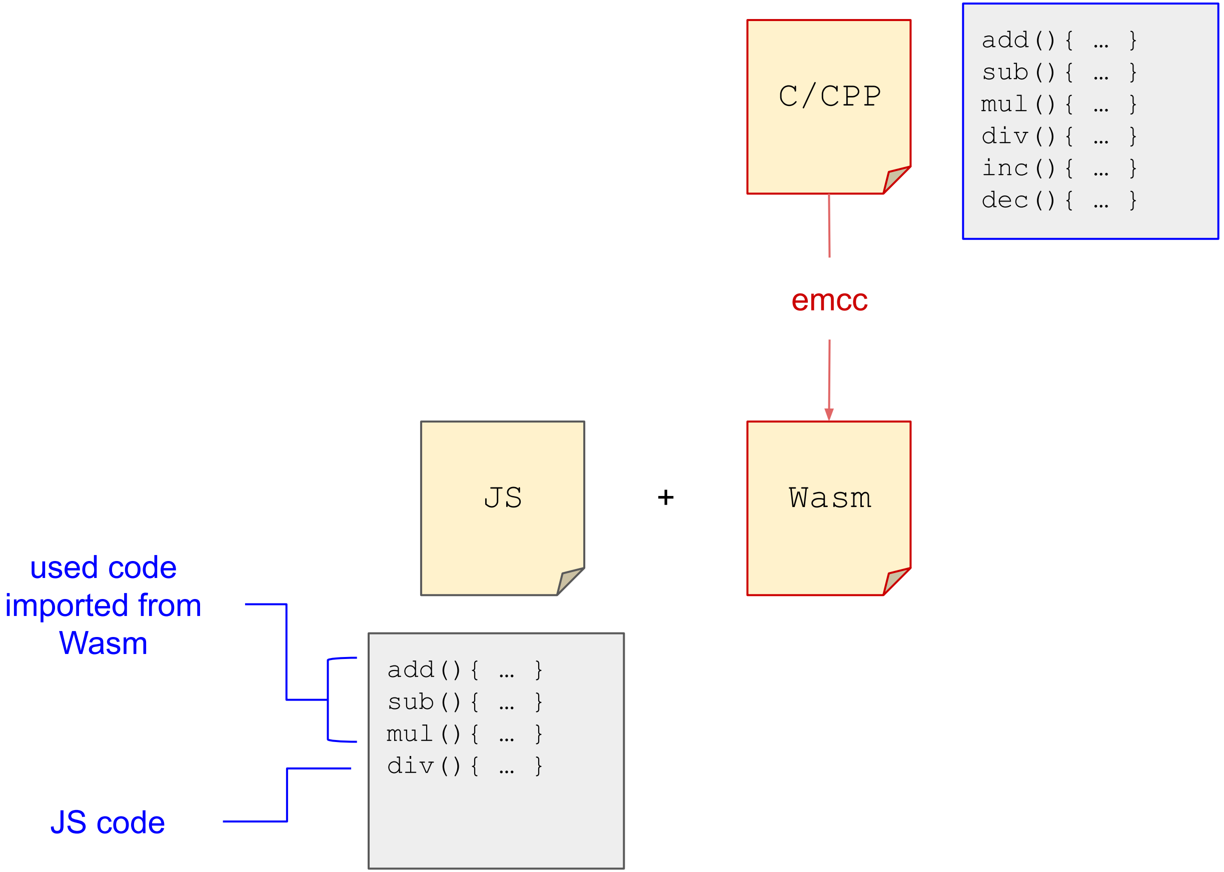
1st Number:

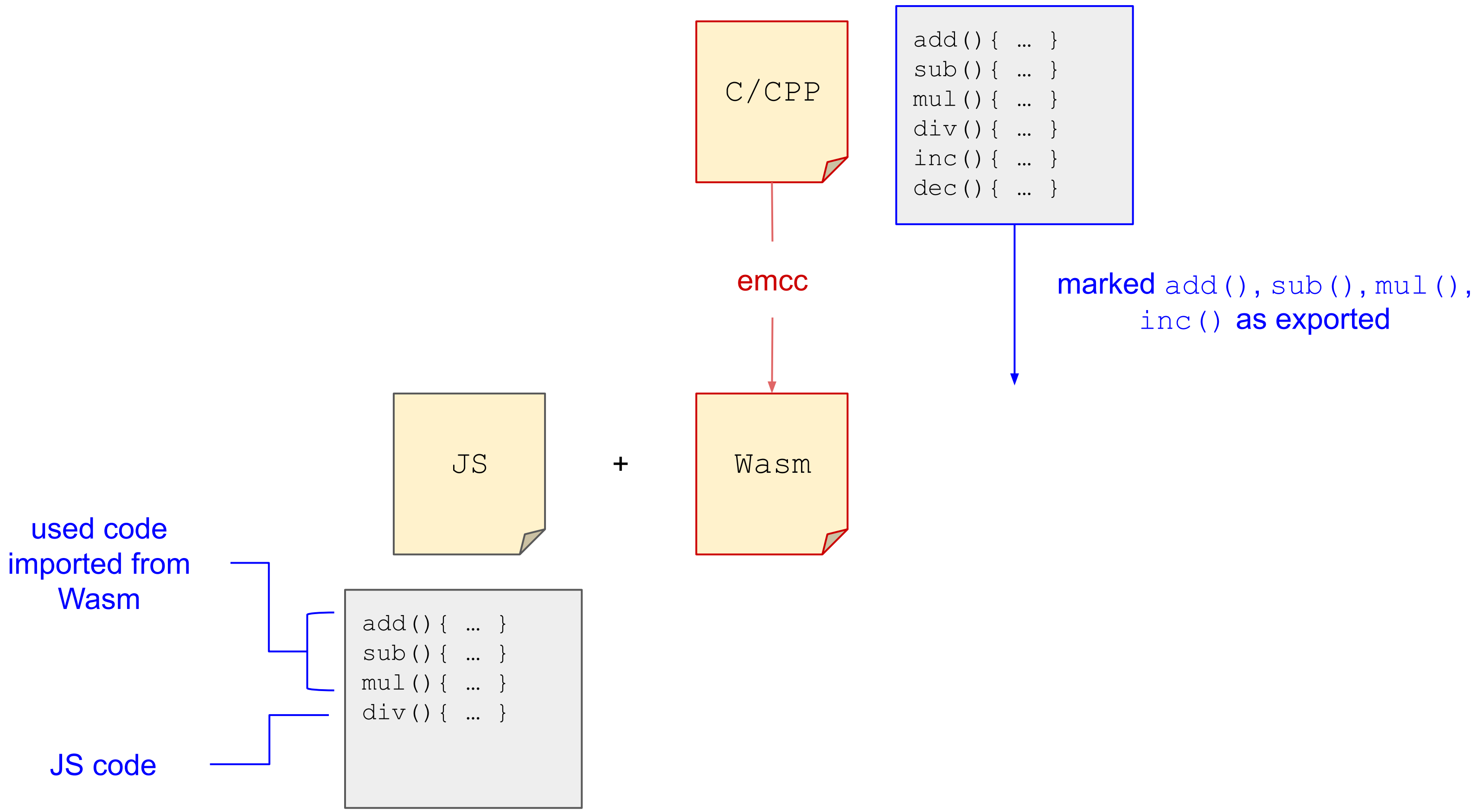
2nd Number:

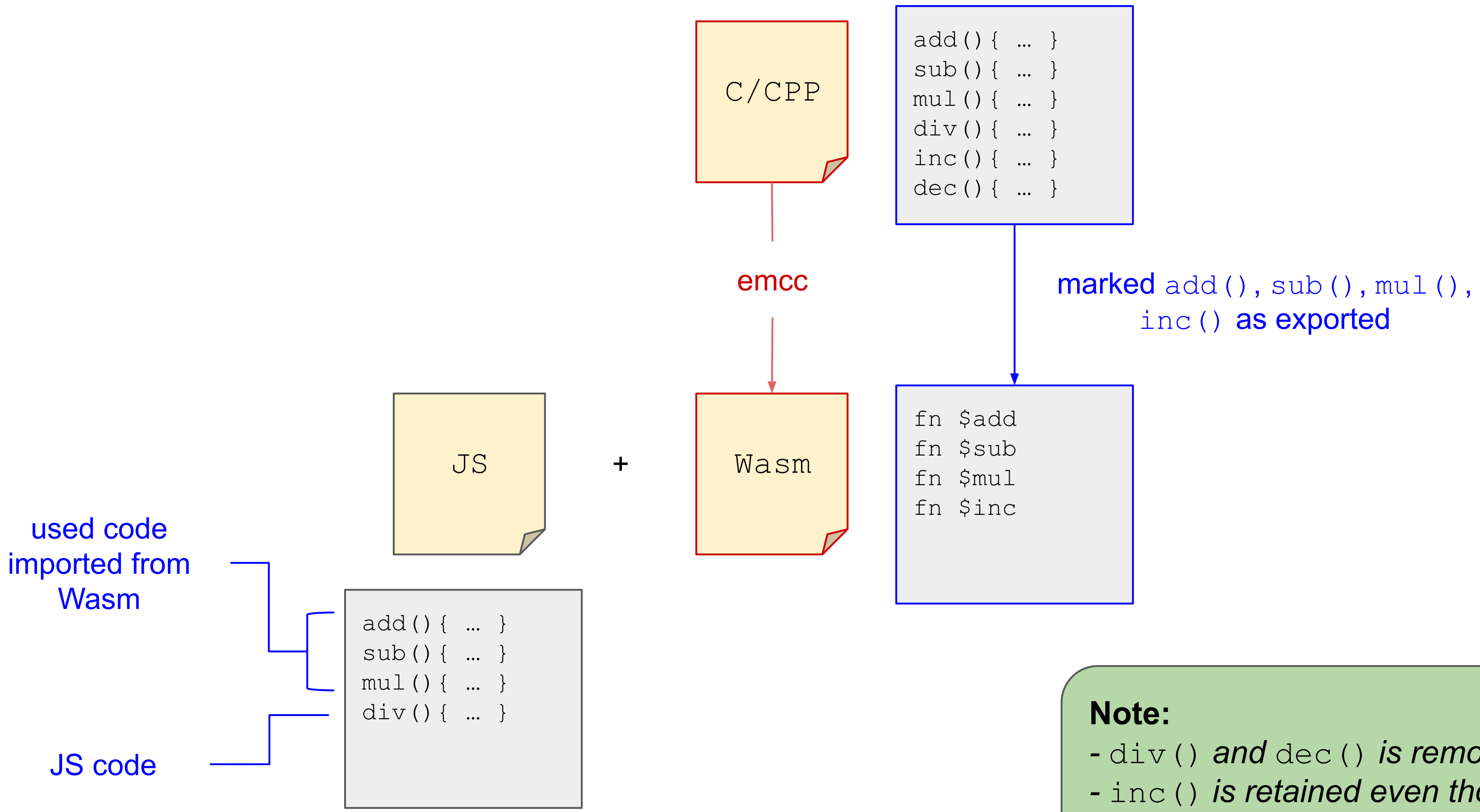
The Result is :



```
add(){ … }
sub(){ … }
mul(){ … }
div(){ … }
```


C/CPP
add(){ … }
sub(){ … }
mul(){ … }
div(){ … }
inc(){ … }
dec(){ … }
emcc
JS
+
Wasm
used code imported from Wasm
JS code
add(){ … }
sub(){ … }
mul(){ … }
div(){ … }


C/CPP
add(){ … }
sub(){ … }
mul(){ … }
div(){ … }
inc(){ … }
dec(){ … }
emcc
marked add(), sub(), mul(), inc() as exported
JS
+
Wasm
used code imported from Wasm
add(){ … }
sub(){ … }
mul(){ … }
div(){ … }
JS code

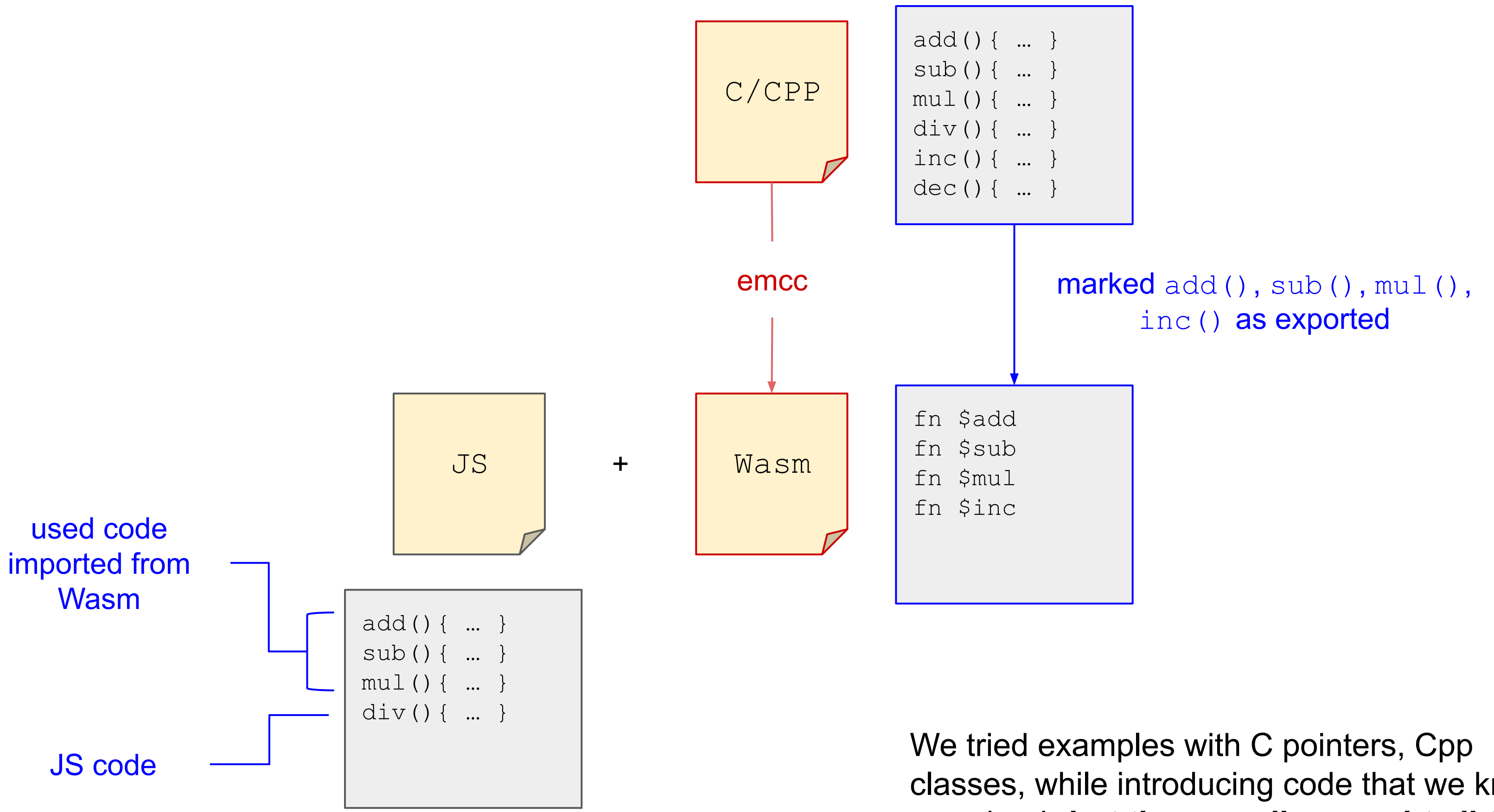

C/CPP
emcc
JS
+
Wasm
marked add(), sub(), mul(), inc() as exported
used code imported from Wasm
JS code

```
add(){ … }
sub(){ … }
mul(){ … }
div(){ … }
inc(){ … }
dec(){ … }
```

```
fn $add
fn $sub
fn $mul
fn $inc
```

```
add(){ … }
sub(){ … }
mul(){ … }
div(){ … }
```

Note:
- `div()` *and* `dec()` *is removed*
- `inc()` *is retained even though it is not used in JS*

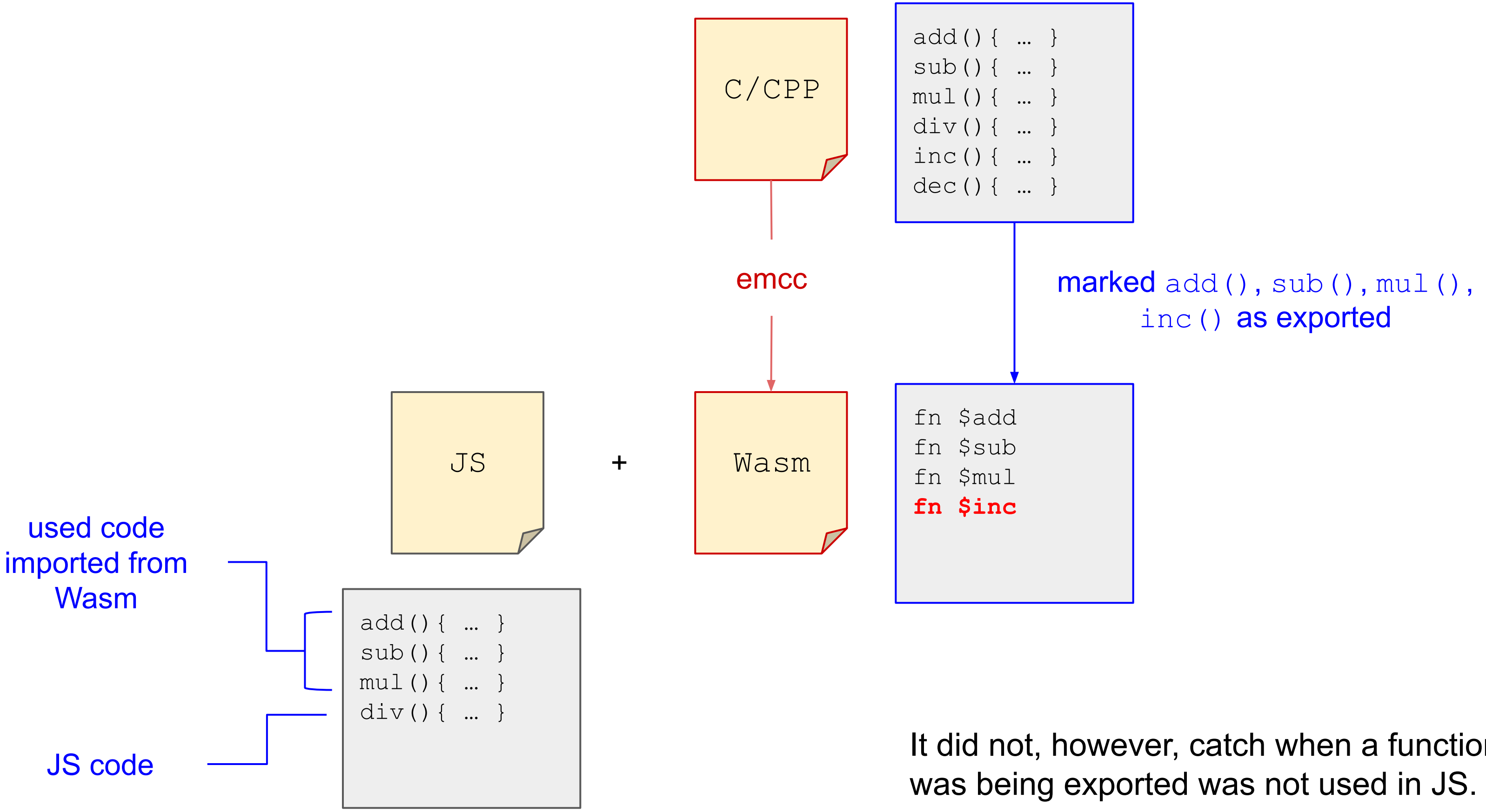

C/CPP
emcc
Wasm
JS
+
marked add(), sub(), mul(), inc() as exported
used code imported from Wasm
JS code

```
add(){ … }
sub(){ … }
mul(){ … }
div(){ … }
inc(){ … }
dec(){ … }
```

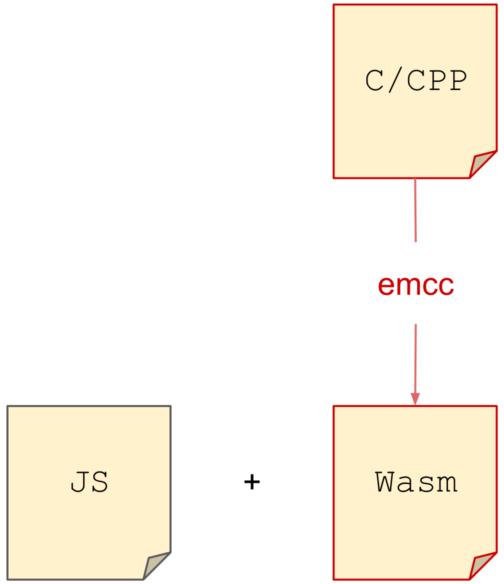
```
fn $add
fn $sub
fn $mul
fn $inc
```

```
add(){ … }
sub(){ … }
mul(){ … }
div(){ … }
```

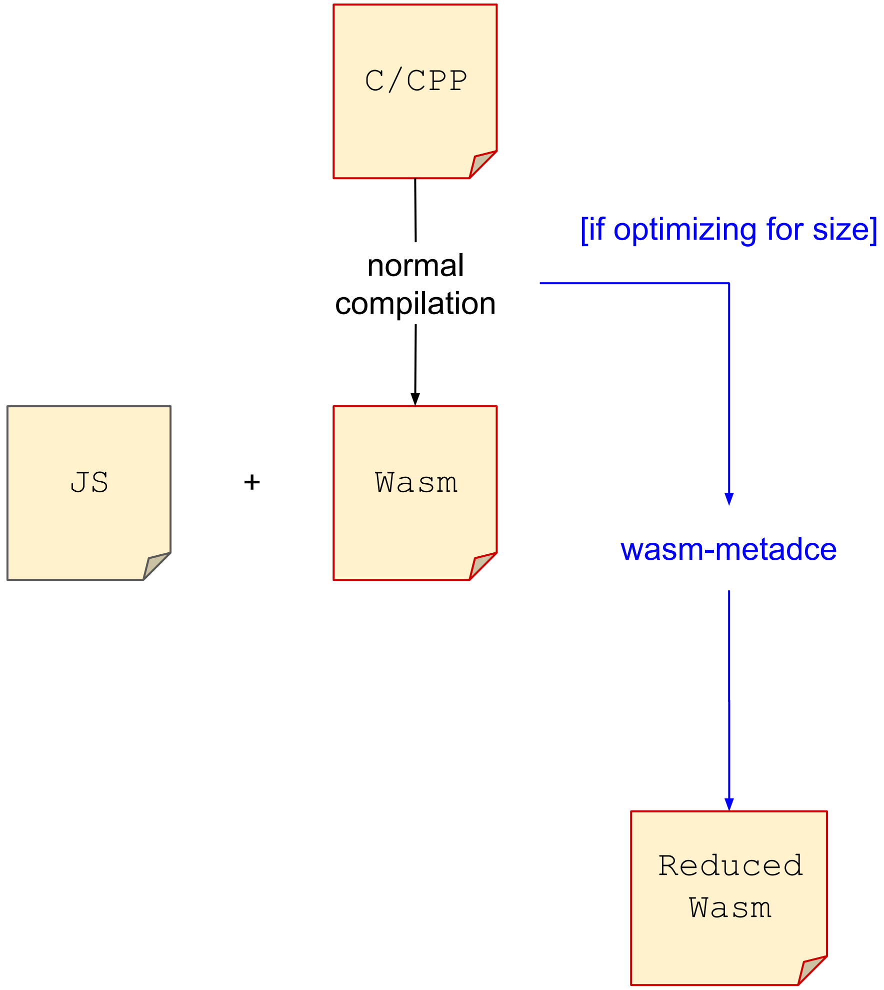
We tried examples with C pointers, Cpp classes, while introducing code that we knew was dead, ***but the compiler caught all the dead code present within Cpp itself.***

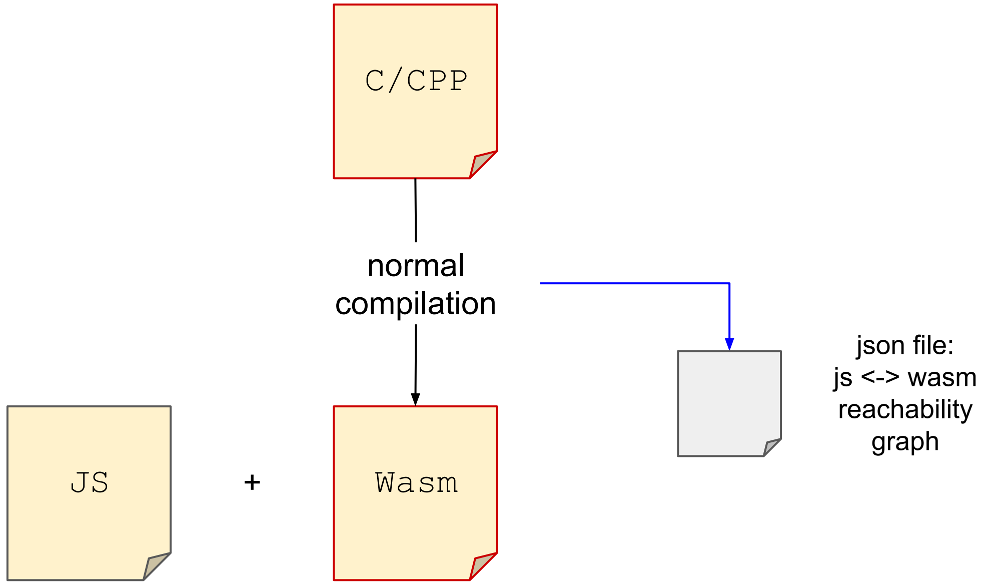


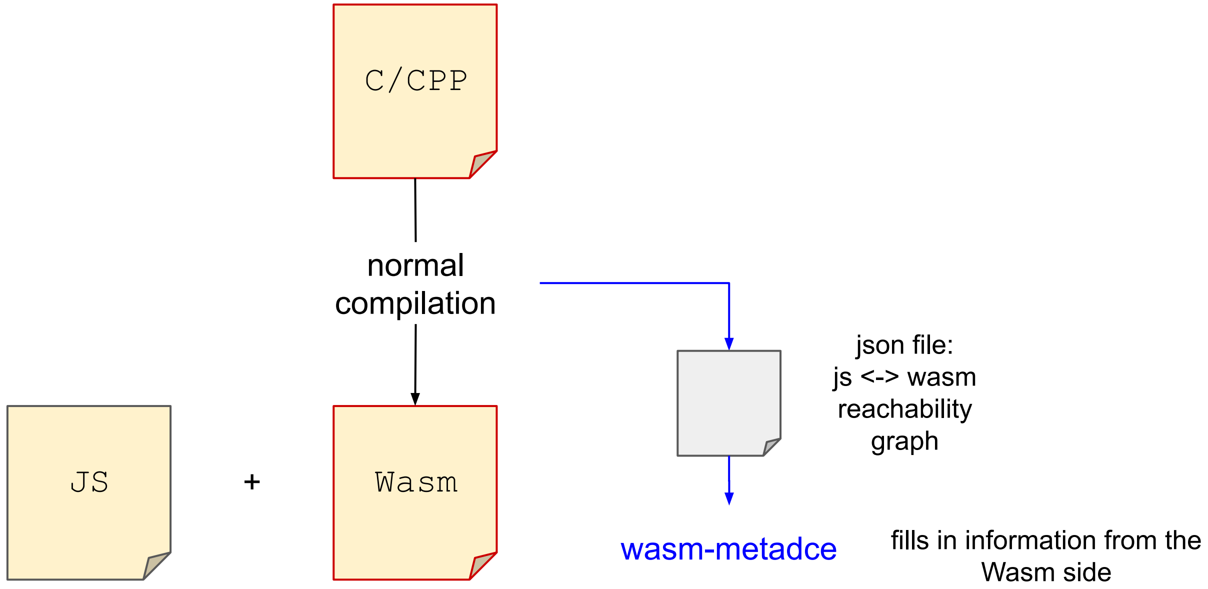
It did not, however, catch when a function that was being exported was not used in JS.

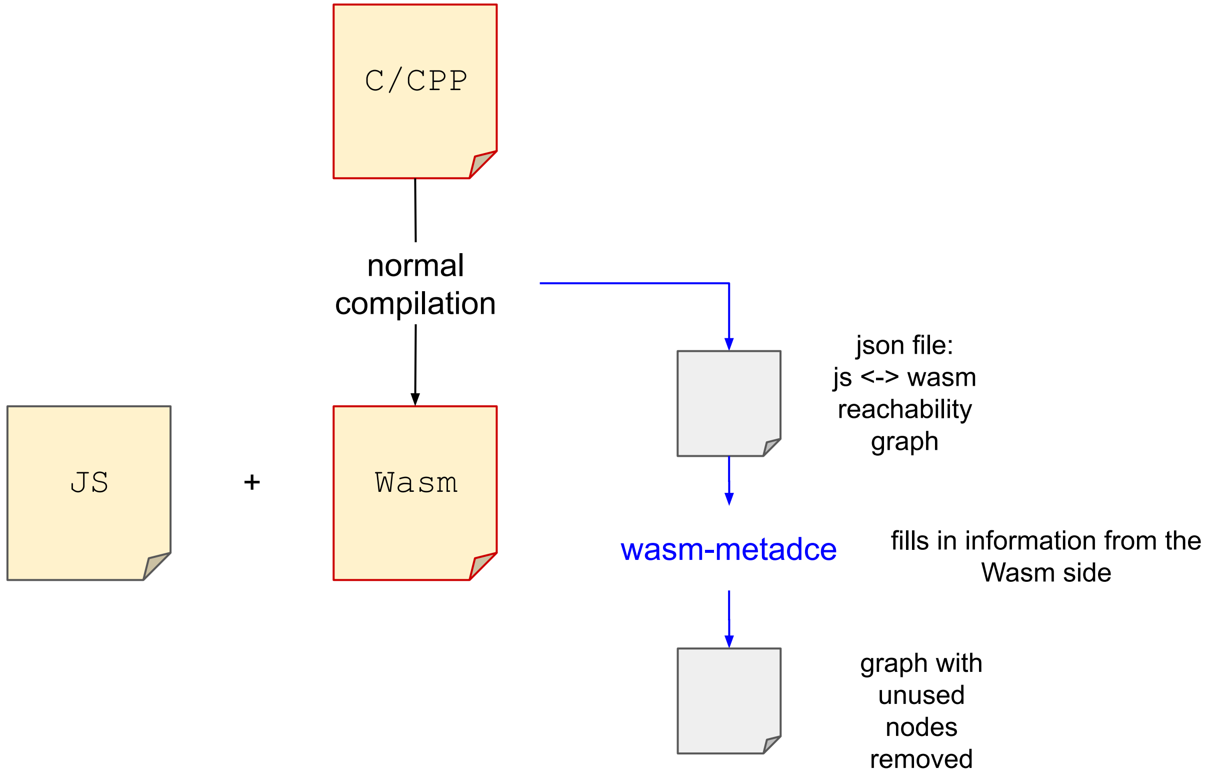


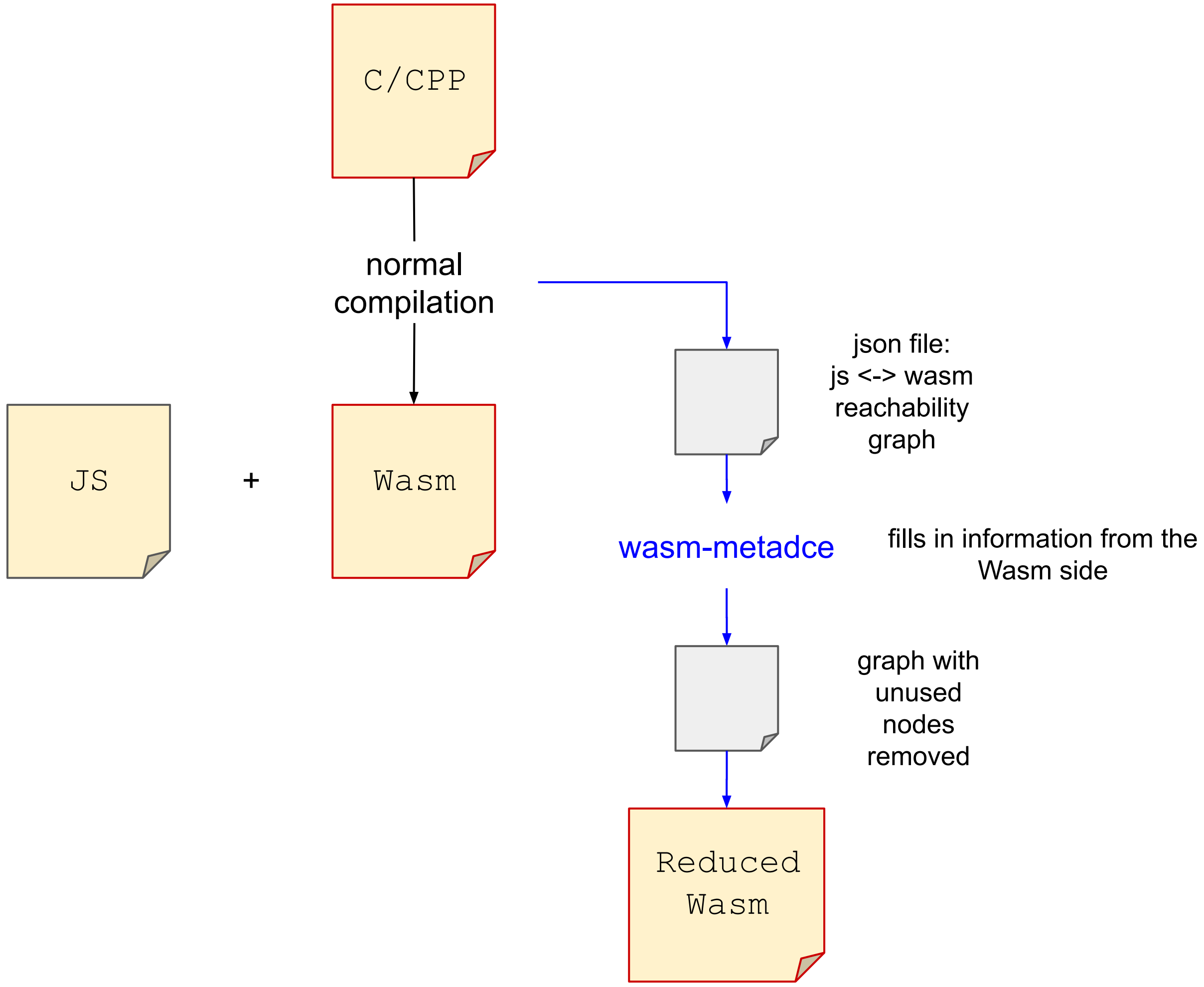
What's going on in emcc?

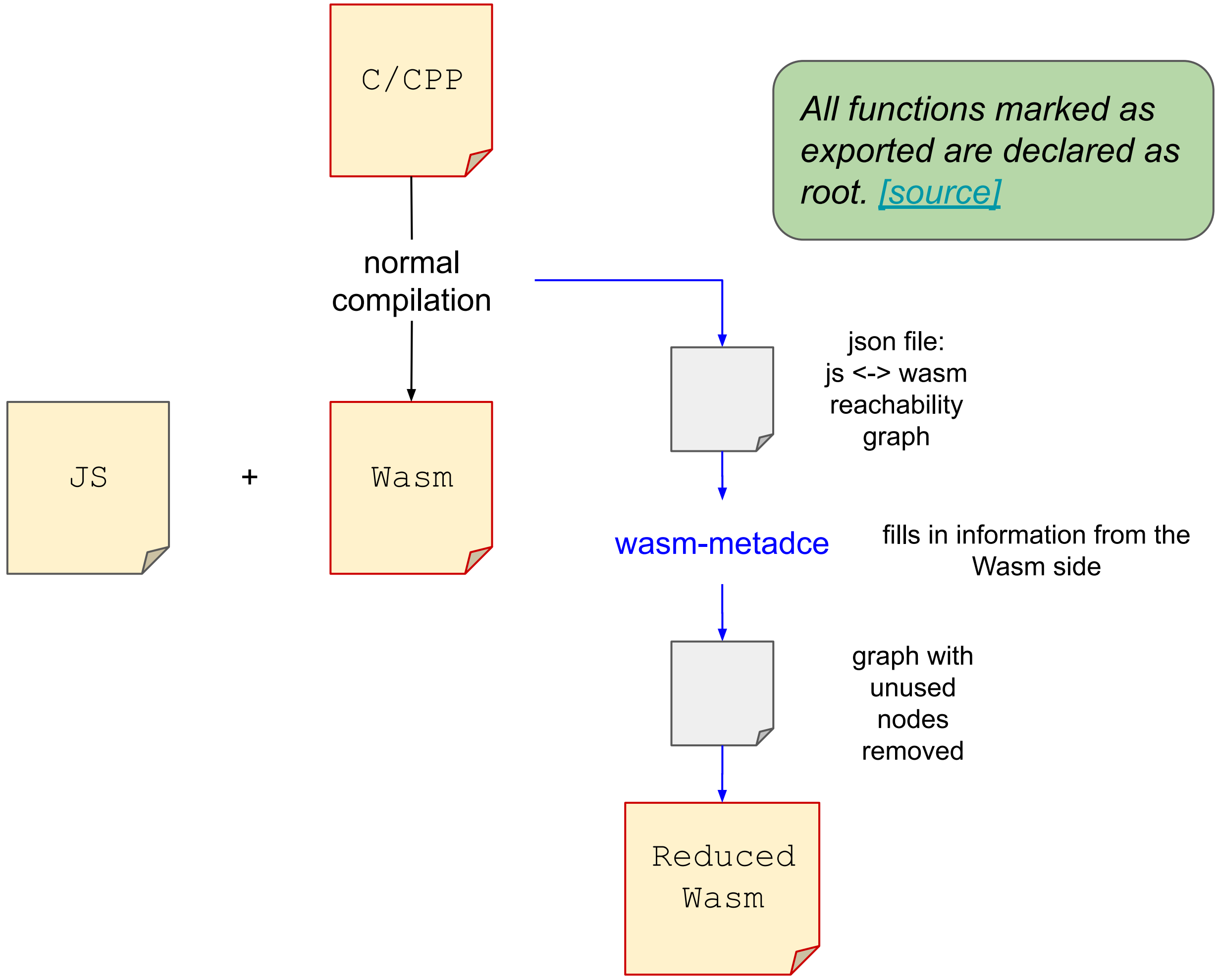

C/CPP
normal compilation
[if optimizing for size]
JS
+
Wasm
wasm-metadce
Reduced Wasm


C/CPP
normal compilation
JS
+
Wasm
json file:
js <-> wasm
reachability
graph

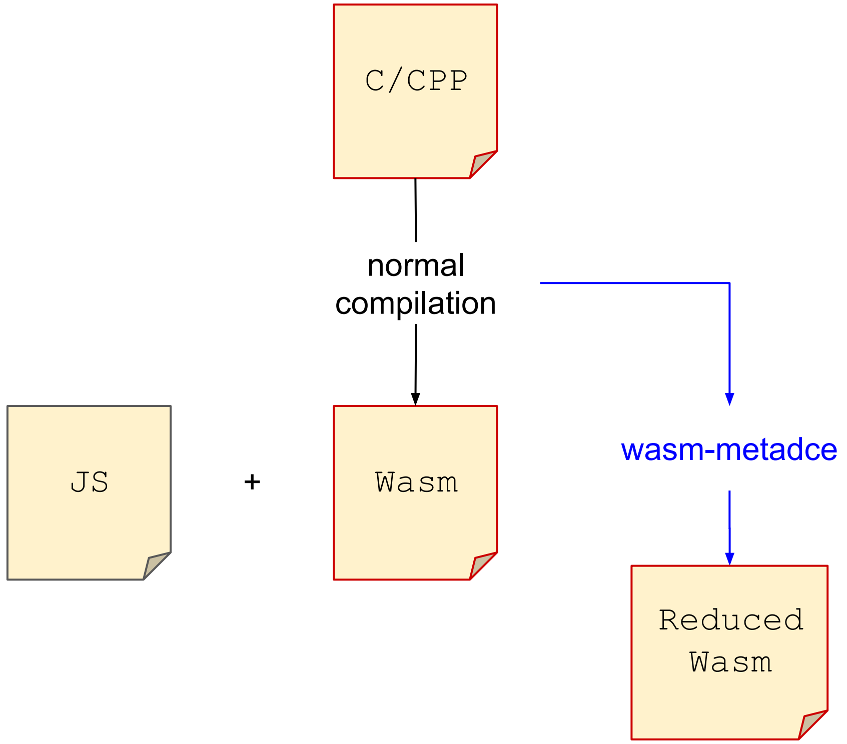

C/CPP
normal compilation
JS
+
Wasm
json file: js <-> wasm reachability graph
wasm-metadce
fills in information from the Wasm side


C/CPP
normal compilation
JS
+
Wasm
json file: js <-> wasm reachability graph
wasm-metadce
fills in information from the Wasm side
graph with unused nodes removed

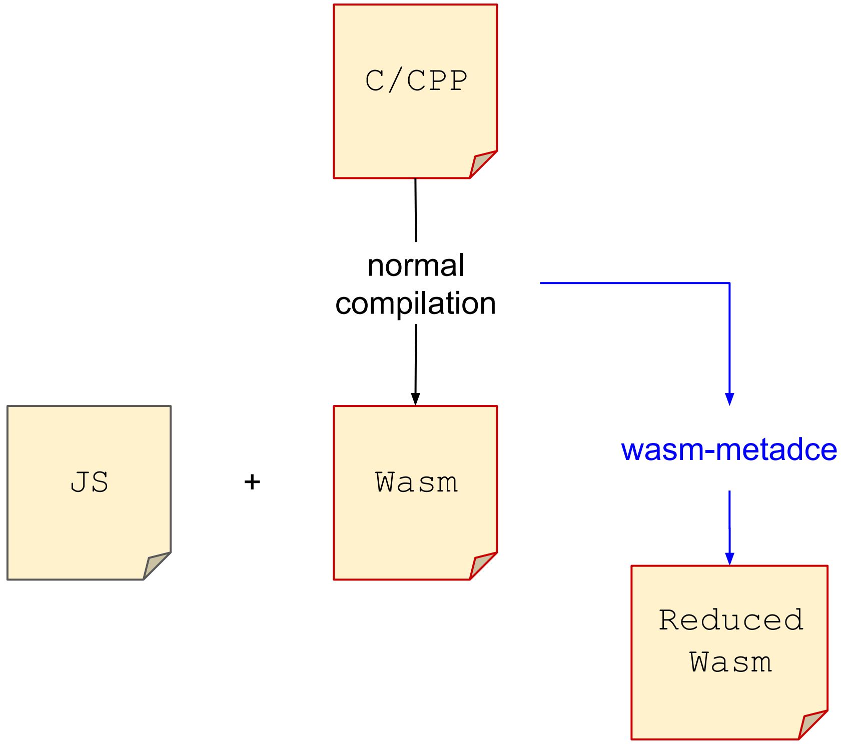

C/CPP
normal compilation
JS
+
Wasm
json file: js <-> wasm reachability graph
wasm-metadce
fills in information from the Wasm side
graph with unused nodes removed
Reduced Wasm


C/CPP
normal compilation
JS
+
Wasm
json file: js <-> wasm reachability graph
wasm-metadce
fills in information from the Wasm side
graph with unused nodes removed
Reduced Wasm

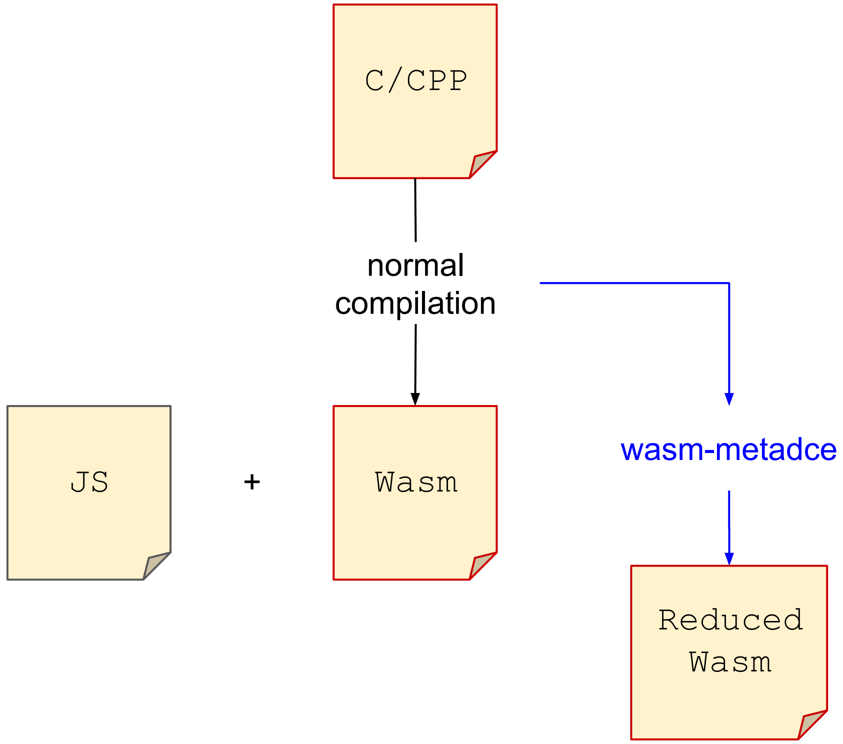
All functions marked as exported are declared as root. [source]



What are possible research questions?



***RQ1**: improve reachability analysis inside a Wasm binary for indirect calls/element section*



***RQ2**: obtaining a more precise JS reachability graph (our own tool / improve input to wasm-metadce)*