An Introduction to WebAssembly

redislabs
HOME OF REDIS

redislabs
HOME OF REDIS

Guy Royse
Developer Advocate
Redis Labs

@guyroyse
github.com/guyroyse
guy.dev

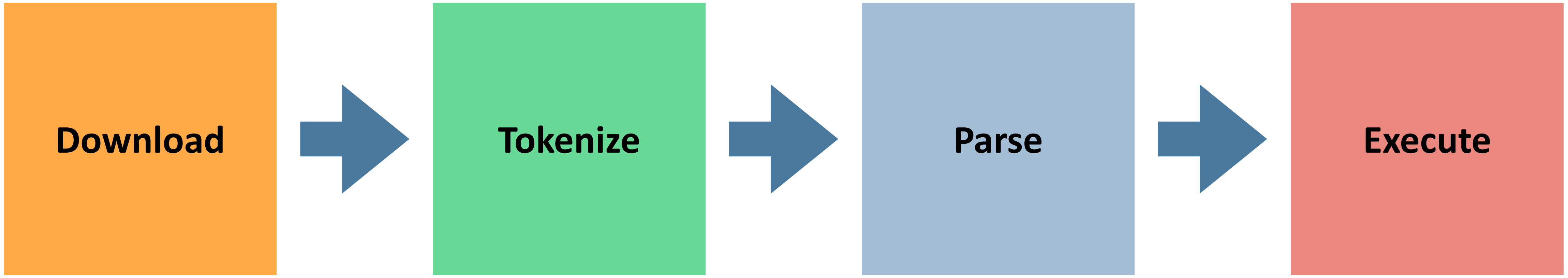
What is WebAssembly?

(Hint. It's a solution to a problem.)

WTH, JavaScript?

```
[] == ![]; // -> true

NaN === NaN; // -> false

Number.MIN_VALUE > 0; // -> true

parseInt("firetruck"); // -> NaN
parseInt("firetruck", 16); // -> 15

console.log.call.call.call.call
  .call.apply(a => a, [1, 2]);

Math.min() > Math.max(); // -> true
```

So Many Choices

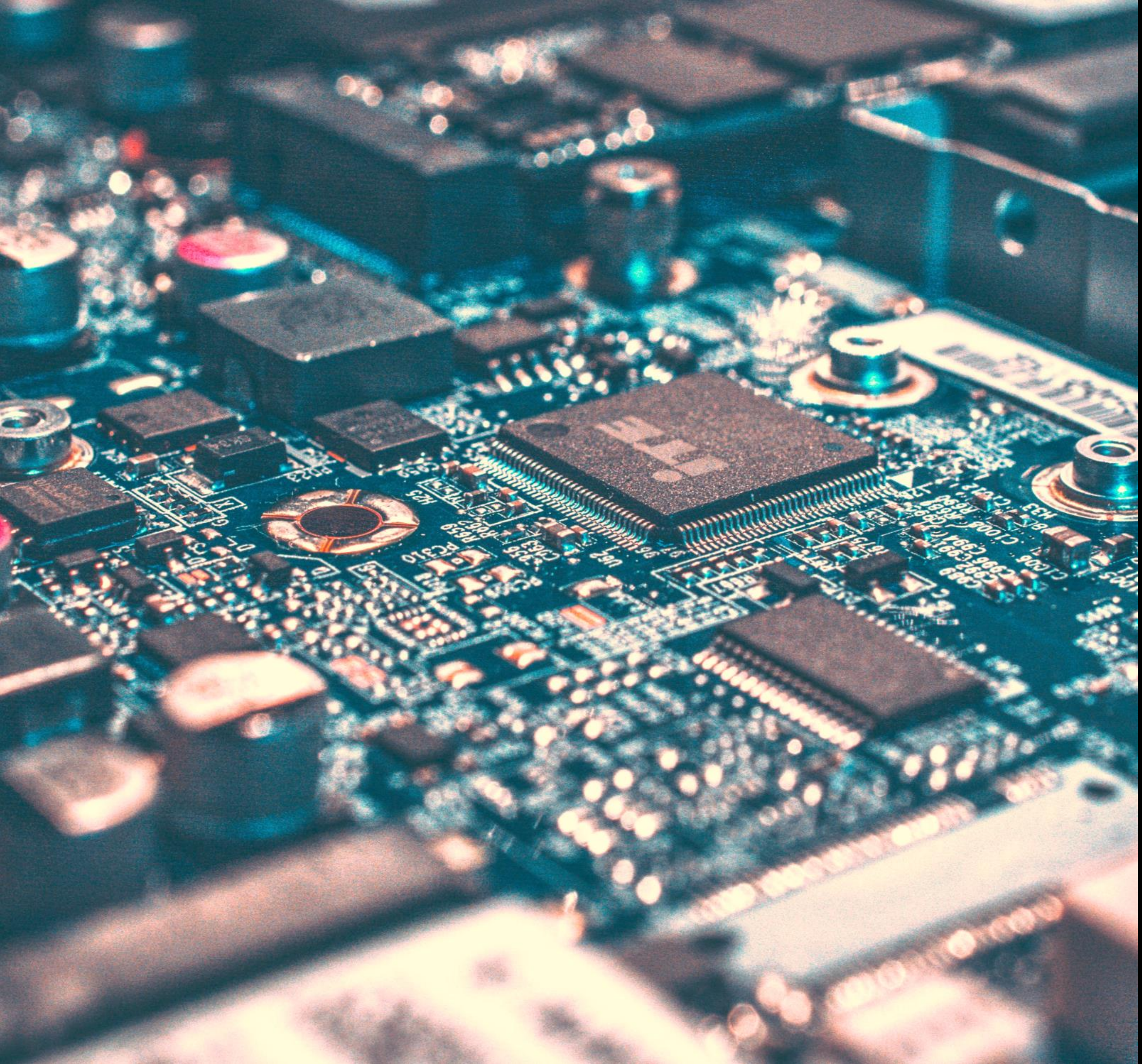

Download
Tokenize
Parse
Execute

What If You Could Compile Code for the Browser?

WA

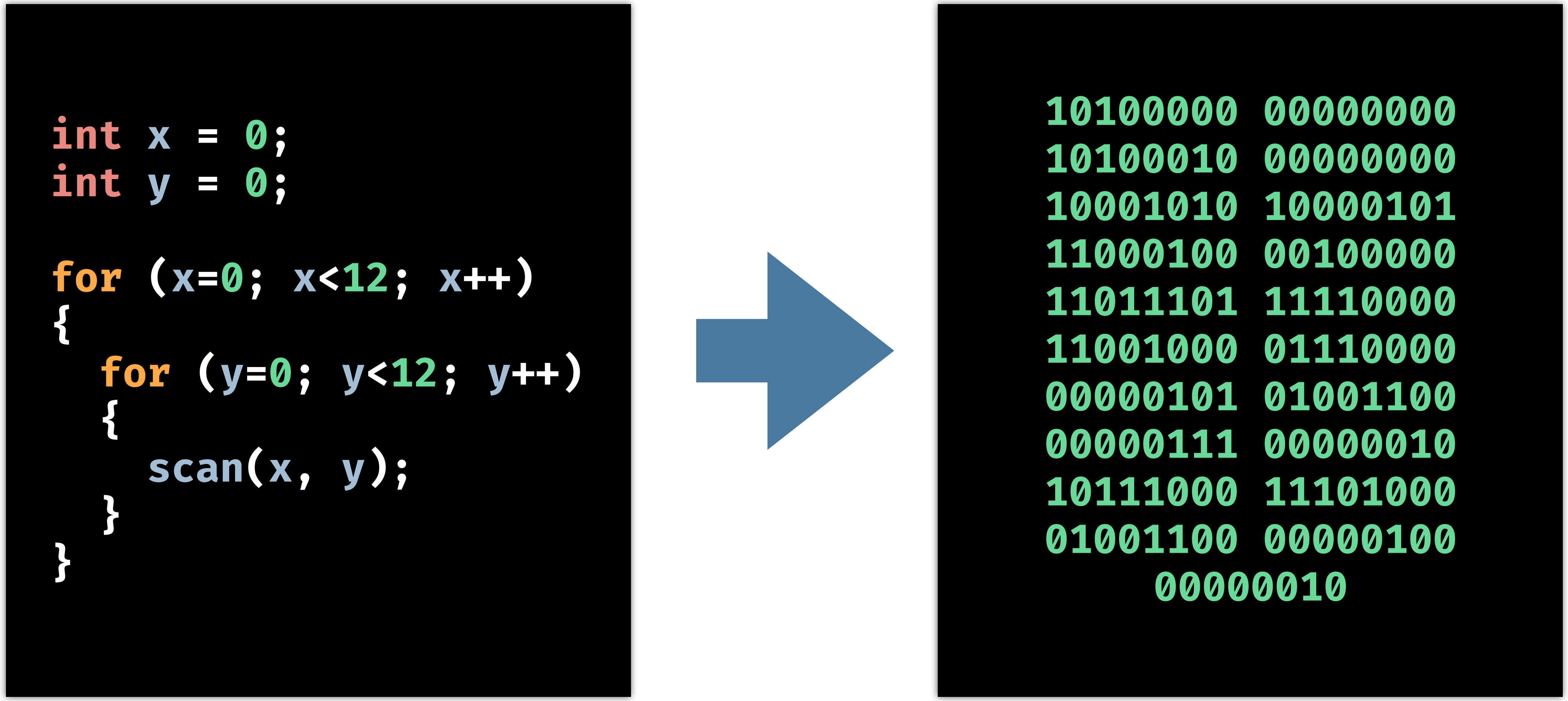
The Run Up to WebAssembly

redislabs
HOME OF REDIS



```
10100000 00000000
10100010 00000000
10001010 10000101
11000100 00100000
11011101 11110000
11001000 01110000
00000101 01001100
00000111 00000010
10111000 11101000
01001100 00000100
    00000010
```

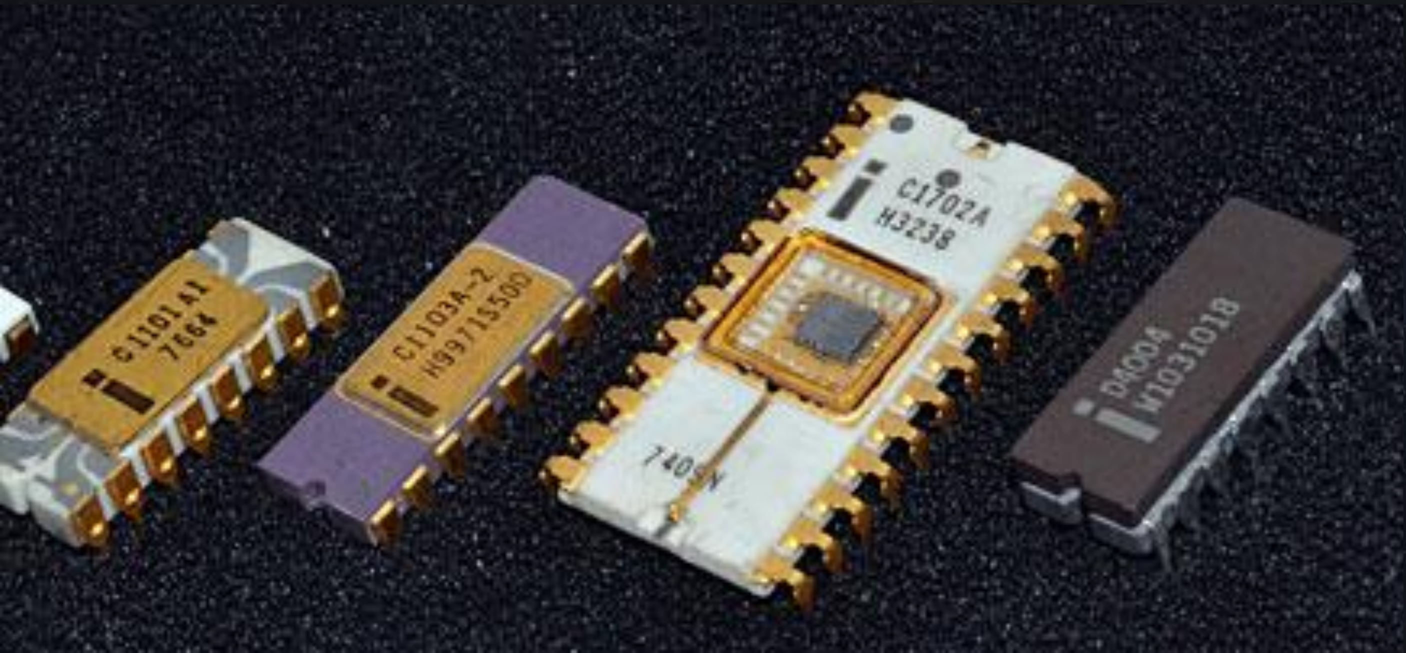
Address	Binary	Hex	Assembly	
0200	10100000 00000000	A0 00		LDY #$00
0202	10100010 00000000	A2 00		LDX #$00
0204	10001010	8A	LOOP2	TXA
0205	10000101 11000100	85 C4		STA $00C4
0207	00100000 11011101 11110000	20 22 F0	LOOP1	JSR SCAN
020A	11001000	C8		INY
020B	01110000 00000101	70 05		BVS RESET
020D	01001100 00000111 00000010	4C 07 02		JMP LOOP1
0210	10111000	B8	RESET	CLV
0211	11101000	E8		INX
0212	01001100 00000100 00000010	4C 04 02		JMP LOOP2



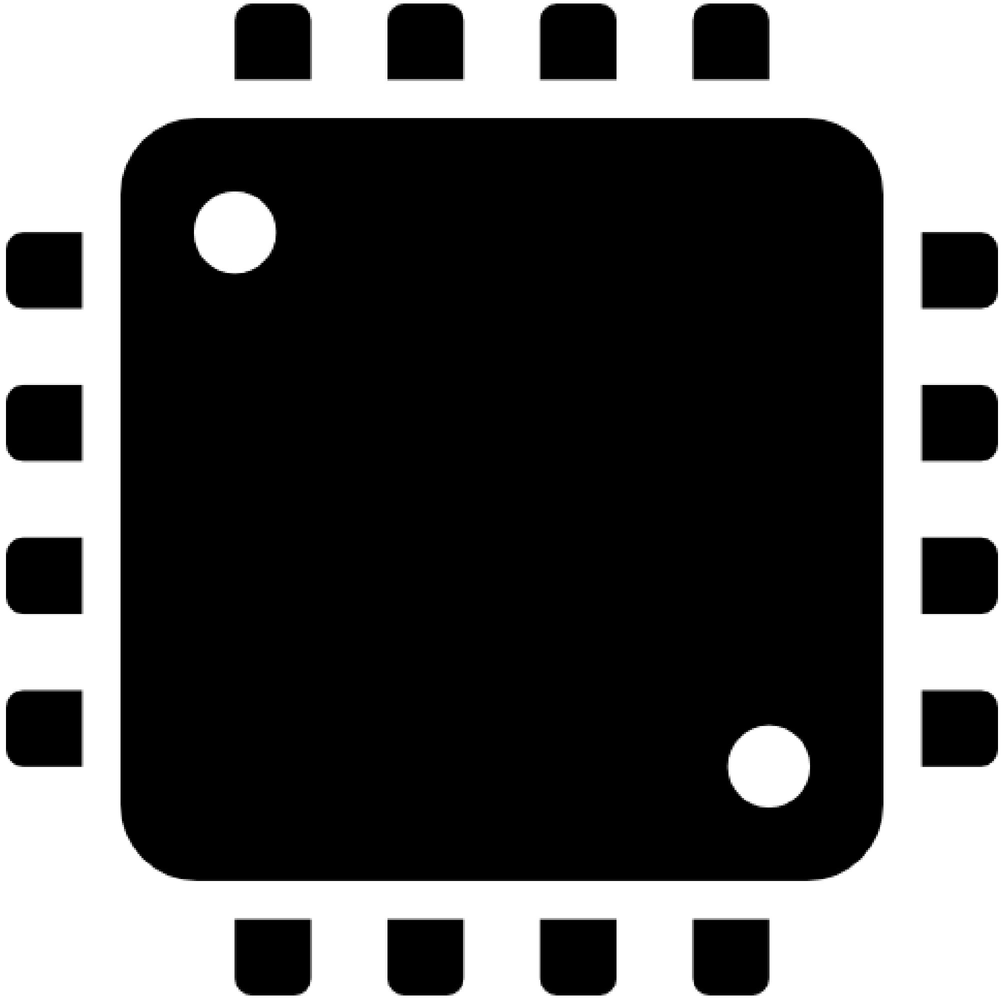
```
int x = 0;
int y = 0;

for (x=0; x<12; x++)
{
  for (y=0; y<12; y++)
  {
    scan(x, y);
  }
}
```

```
10100000 00000000
10100010 00000000
10001010 10000101
11000100 00100000
11011101 11110000
11001000 01110000
00000101 01001100
00000111 00000010
10111000 11101000
01001100 00000100
     00000010
```

redislabs
HOME OF REDIS

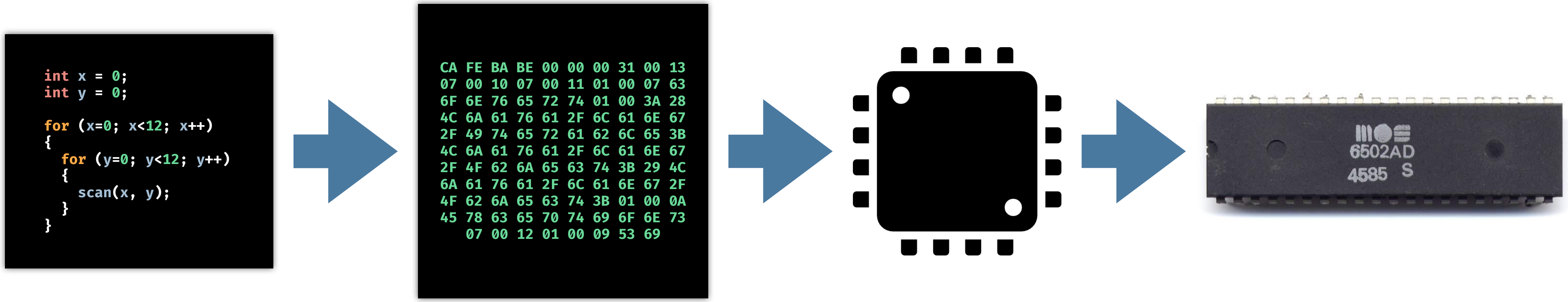
INTEL 1975
XILINX
HKb2bsite.com

C1101A1
7664
C1103A-2
H9971550
C1702A
H3238
7409N
D4004
W1031018

8231346
D5061
3001
MICROCODE ©
1985 INTERGRAPH
INTERGRAPH
C322C3Q
F307009B
8944 USA
ST15500MVB
PMC
SMC
MALAYSIA
intel
StrongARM
SA-110
ARM
21281-EB
MOXA
MOXA ART
C-BGA292-V0C
TZIK01140515
RoHS
AMD
R80286-12/S
8741HP
INTEL 1982
MOTOROLA
QLOGIC
ISP4022
TAIWAN
Raritan
SIEMENS
SAB
80268-R
BROADCOM
BCM8603KPBG
HA0716 P13
819010 P2
A5
DS89C420
MCS
0119A2
075AB
ES
MC68000L8
GN78331
SIEMENS
SAB
82258-R
Agilent
N2530/SP2.0E
ARM710 POWERED
0710KU200 JAPAN



redislabs
HOME OF REDIS

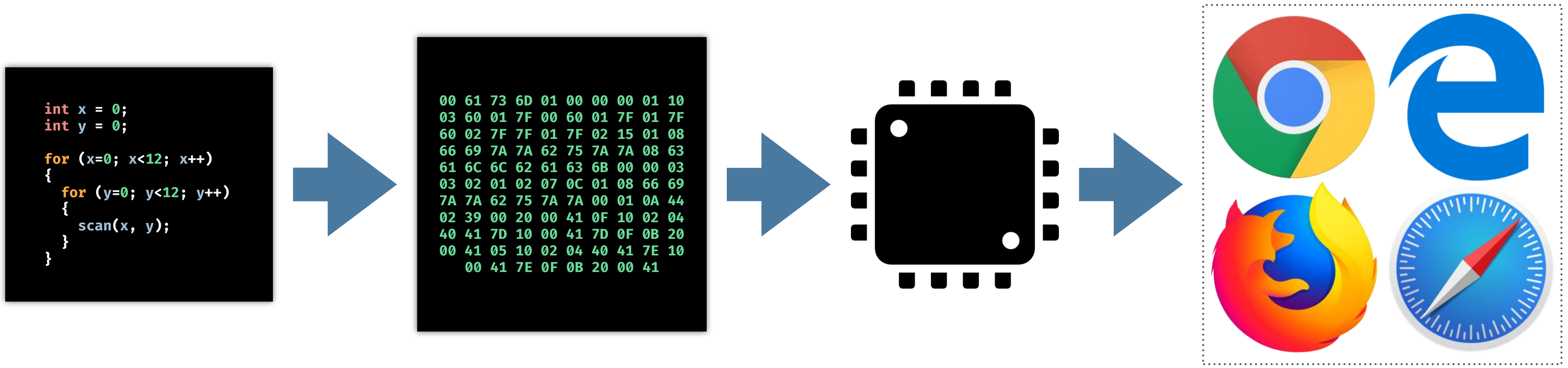
How Virtual Machines Work



```
int x = 0;
int y = 0;

for (x=0; x<12; x++)
{
  for (y=0; y<12; y++)
  {
    scan(x, y);
  }
}
```

```
CA FE BA BE 00 00 00 31 00 13
07 00 10 07 00 11 01 00 07 63
6F 6E 76 65 72 74 01 00 3A 28
4C 6A 61 76 61 2F 6C 61 6E 67
2F 49 74 65 72 61 62 6C 65 3B
4C 6A 61 76 61 2F 6C 61 6E 67
2F 4F 62 6A 65 63 74 3B 29 4C
6A 61 76 61 2F 6C 61 6E 67 2F
4F 62 6A 65 63 74 3B 01 00 0A
45 78 63 65 70 74 69 6F 6E 73
   07 00 12 01 00 09 53 69
```

How WebAssembly Works



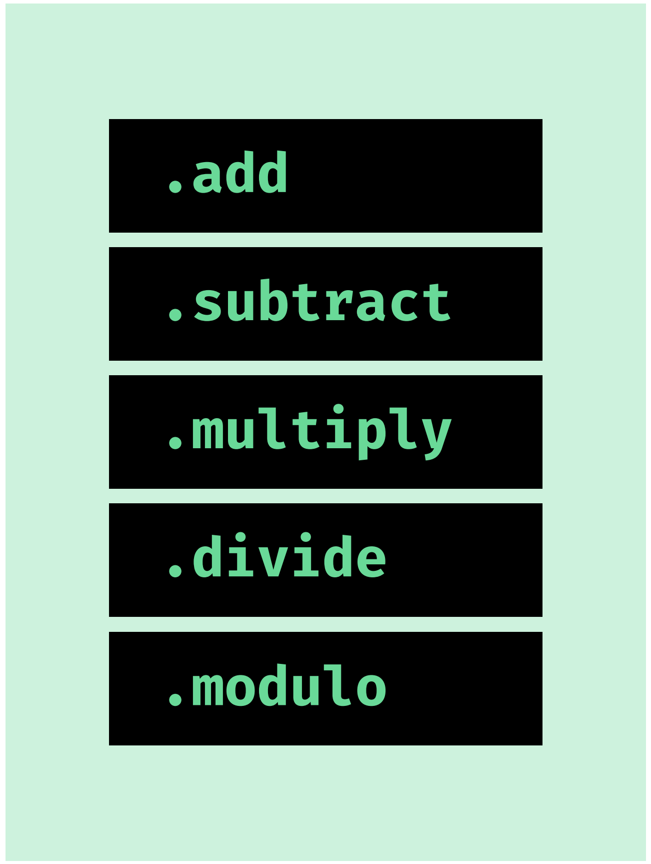
```
int x = 0;
int y = 0;

for (x=0; x<12; x++)
{
  for (y=0; y<12; y++)
  {
    scan(x, y);
  }
}
```

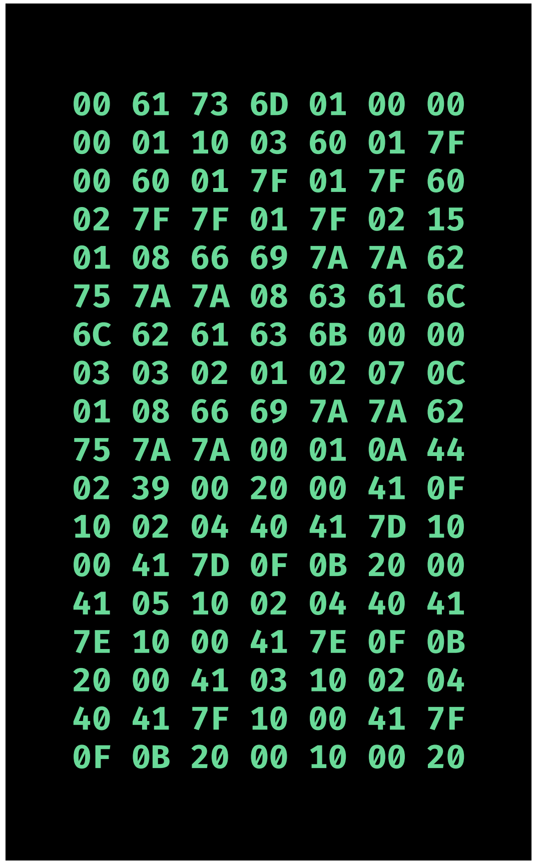
```
00 61 73 6D 01 00 00 00 01 10
03 60 01 7F 00 60 01 7F 01 7F
60 02 7F 7F 01 7F 02 15 01 08
66 69 7A 7A 62 75 7A 7A 08 63
61 6C 6C 62 61 63 6B 00 00 03
03 02 01 02 07 0C 01 08 66 69
7A 7A 62 75 7A 7A 00 01 0A 44
02 39 00 20 00 41 0F 10 02 04
40 41 7D 10 00 41 7D 0F 0B 20
00 41 05 10 02 04 40 41 7E 10
   00 41 7E 0F 0B 20 00 41
```

Understanding WebAssembly

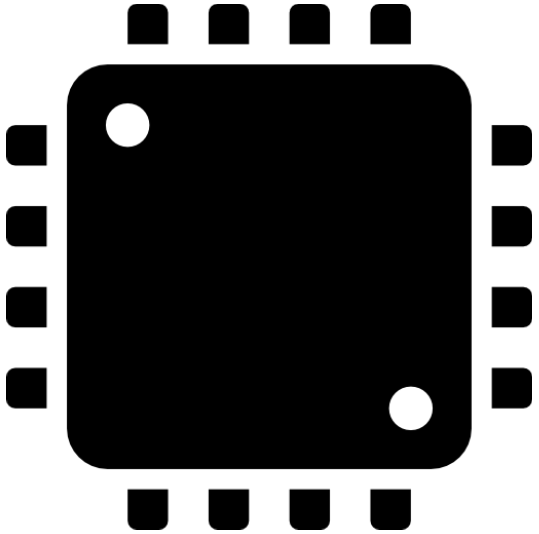
redislabs
HOME OF REDIS

WebAssembly Modules

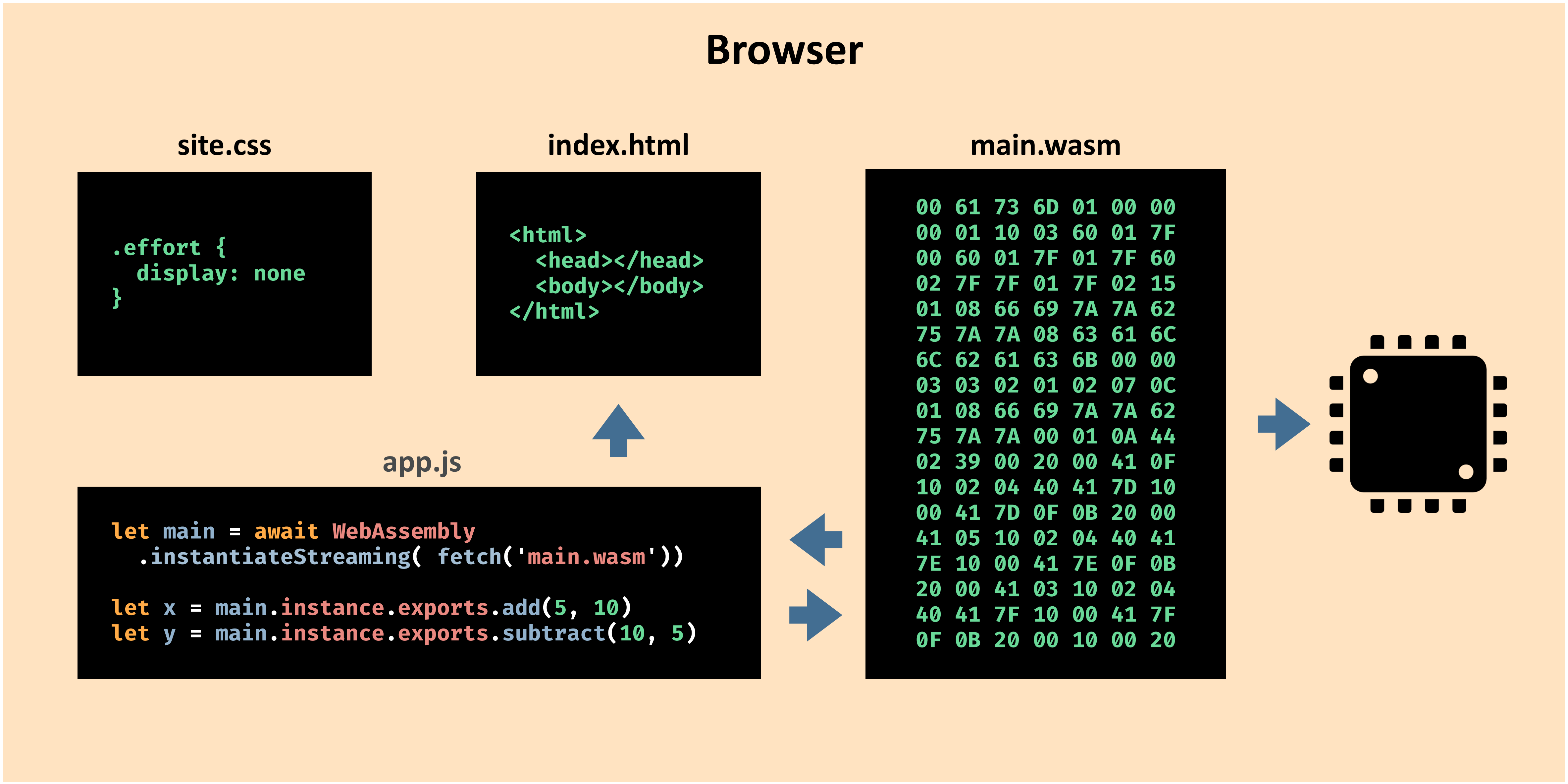




```
00 61 73 6D 01 00 00
00 01 10 03 60 01 7F
00 60 01 7F 01 7F 60
02 7F 7F 01 7F 02 15
01 08 66 69 7A 7A 62
75 7A 7A 08 63 61 6C
6C 62 61 63 6B 00 00
03 03 02 01 02 07 0C
01 08 66 69 7A 7A 62
75 7A 7A 00 01 0A 44
02 39 00 20 00 41 0F
10 02 04 40 41 7D 10
00 41 7D 0F 0B 20 00
41 05 10 02 04 40 41
7E 10 00 41 7E 0F 0B
20 00 41 03 10 02 04
40 41 7F 10 00 41 7F
0F 0B 20 00 10 00 20
```



WebAssembly in the Browser



```css
.effort {
  display: none
}
```

```html
<html>
  <head></head>
  <body></body>
</html>
```

```js
let main = await WebAssembly
  .instantiateStreaming( fetch('main.wasm'))

let x = main.instance.exports.add(5, 10)
let y = main.instance.exports.subtract(10, 5)
```

```
00 61 73 6D 01 00 00
00 01 10 03 60 01 7F
00 60 01 7F 01 7F 60
02 7F 7F 01 7F 02 15
01 08 66 69 7A 7A 62
75 7A 7A 08 63 61 6C
6C 62 61 63 6B 00 00
03 03 02 01 02 07 0C
01 08 66 69 7A 7A 62
75 7A 7A 00 01 0A 44
02 39 00 20 00 41 0F
10 02 04 40 41 7D 10
00 41 7D 0F 0B 20 00
41 05 10 02 04 40 41
7E 10 00 41 7E 0F 0B
20 00 41 03 10 02 04
40 41 7F 10 00 41 7F
0F 0B 20 00 10 00 20
```

Using WebAssembly From JavaScript

```
let imports = {
  math : {
    callback : x => console.log("result is", x)
  }
}


let module = await WebAssembly.instantiateStreaming(fetch('main.wasm'), imports)

let x = module.instance.exports.add(5, 10)
let y = module.instance.exports.subtract(10, 5)
let z = module.instance.exports.multiply(2, 5)
```

Instantiating Modules

```
let imports = {
  math : {
    callback : x => console.log("Result", x)
  }
}


let module = await WebAssembly.instantiateStreaming(fetch('main.wasm'), imports)

let x = module.instance.exports.add(5, 10)
let y = module.instance.exports.subtract(10, 5)
let z = module.instance.exports.multiply(2, 5)
```

Using Functions in Modules

```
let imports = {
  math : {
    callback : x => console.log("result is", x)
  }
}


let module = await WebAssembly.instantiateStreaming(fetch('main.wasm'), imports)

let x = module.instance.exports.add(5, 10)
let y = module.instance.exports.subtract(10, 5)
let z = module.instance.exports.multiply(2, 5)
```

Importing JavaScript Functions into WebAssembly Modules

```
let imports = {
  math : {
    callback : x => console.log("result is", x)
  }
}


let module = await WebAssembly.instantiateStreaming(fetch('main.wasm'), imports)

let x = module.instance.exports.add(5, 10)
let y = module.instance.exports.subtract(10, 5)
let z = module.instance.exports.multiply(2, 5)
```

Imports as Callbacks

```
let imports = {
  math : {
    callback : x => console.log("result is", x)
  }
}


let module = await WebAssembly.instantiateStreaming(fetch('main.wasm'), imports)

let x = module.instance.exports.add(5, 10)
let y = module.instance.exports.subtract(10, 5)
let z = module.instance.exports.multiply(2, 5)
```

WA

A Simple Module in WebAssembly Text Format

```
00 61 73 6D 01 00
00 00 01 10 03 60
01 7F 00 60 01 7F
01 7F 60 02 7F 7F
01 7F 02 15 01 08
66 69 7A 7A 62 75
7A 7A 08 63 61 6C
6C 62 61 63 6B 00
00 03 03 02 01 02
07 0C 01 08 66 69
7A 7A 62 75 7A 7A
00 01 0A 44 02 39
00 20 00 41 0F 10
02 04 40 41 7D 10
00 41 7D 0F 0B 20
00 41 05 10 02 04
40 41 7E 10 00 41
7E 0F 0B 20 00 41
03 10 02 04 40 41
7F 10 00 41 7F 0F
0B 20 00 10 00 20
```

```
(module

  (import "math" "callback" (func $callback))

  (export "add" (func $add))
  (export "subtract" (func $subtract))

  (func $add (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.add
  )

  (func $subtract (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.sub
  )
)
```

Declaring a Module

```
00 61 73 6D 01 00
00 00 01 10 03 60
01 7F 00 60 01 7F
01 7F 60 02 7F 7F
01 7F 02 15 01 08
66 69 7A 7A 62 75
7A 7A 08 63 61 6C
6C 62 61 63 6B 00
00 03 03 02 01 02
07 0C 01 08 66 69
7A 7A 62 75 7A 7A
00 01 0A 44 02 39
00 20 00 41 0F 10
02 04 40 41 7D 10
00 41 7D 0F 0B 20
00 41 05 10 02 04
40 41 7E 10 00 41
7E 0F 0B 20 00 41
03 10 02 04 40 41
7F 10 00 41 7F 0F
0B 20 00 10 00 20
```

```
(module

  (import "math" "callback" (func $callback))

  (export "add" (func $add))
  (export "subtract" (func $subtract))

  (func $add (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.add
  )

  (func $subtract (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.sub
  )
)
```

Creating Functions

```
00 61 73 6D 01 00
00 00 01 10 03 60
01 7F 00 60 01 7F
01 7F 60 02 7F 7F
01 7F 02 15 01 08
66 69 7A 7A 62 75
7A 7A 08 63 61 6C
6C 62 61 63 6B 00
00 03 03 02 01 02
07 0C 01 08 66 69
7A 7A 62 75 7A 7A
00 01 0A 44 02 39
00 20 00 41 0F 10
02 04 40 41 7D 10
00 41 7D 0F 0B 20
00 41 05 10 02 04
40 41 7E 10 00 41
7E 0F 0B 20 00 41
03 10 02 04 40 41
7F 10 00 41 7F 0F
0B 20 00 10 00 20
```

```
(module

  (import "math" "callback" (func $callback))

  (export "add" (func $add))
  (export "subtract" (func $subtract))

  (func $add (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.add
  )

  (func $subtract (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.sub
  )
)
```

Exporting Functions

```
00 61 73 6D 01 00
00 00 01 10 03 60
01 7F 00 60 01 7F
01 7F 60 02 7F 7F
01 7F 02 15 01 08
66 69 7A 7A 62 75
7A 7A 08 63 61 6C
6C 62 61 63 6B 00
00 03 03 02 01 02
07 0C 01 08 66 69
7A 7A 62 75 7A 7A
00 01 0A 44 02 39
00 20 00 41 0F 10
02 04 40 41 7D 10
00 41 7D 0F 0B 20
00 41 05 10 02 04
40 41 7E 10 00 41
7E 0F 0B 20 00 41
03 10 02 04 40 41
7F 10 00 41 7F 0F
0B 20 00 10 00 20
```

```
(module

  (import "math" "callback" (func $callback))

  (export "add" (func $add))
  (export "subtract" (func $subtract))

  (func $add (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.add
  )

  (func $subtract (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.sub
  )
)
```

Importing Functions

```
00 61 73 6D 01 00
00 00 01 10 03 60
01 7F 00 60 01 7F
01 7F 60 02 7F 7F
01 7F 02 15 01 08
66 69 7A 7A 62 75
7A 7A 08 63 61 6C
6C 62 61 63 6B 00
00 03 03 02 01 02
07 0C 01 08 66 69
7A 7A 62 75 7A 7A
00 01 0A 44 02 39
00 20 00 41 0F 10
02 04 40 41 7D 10
00 41 7D 0F 0B 20
00 41 05 10 02 04
40 41 7E 10 00 41
7E 0F 0B 20 00 41
03 10 02 04 40 41
7F 10 00 41 7F 0F
0B 20 00 10 00 20
```

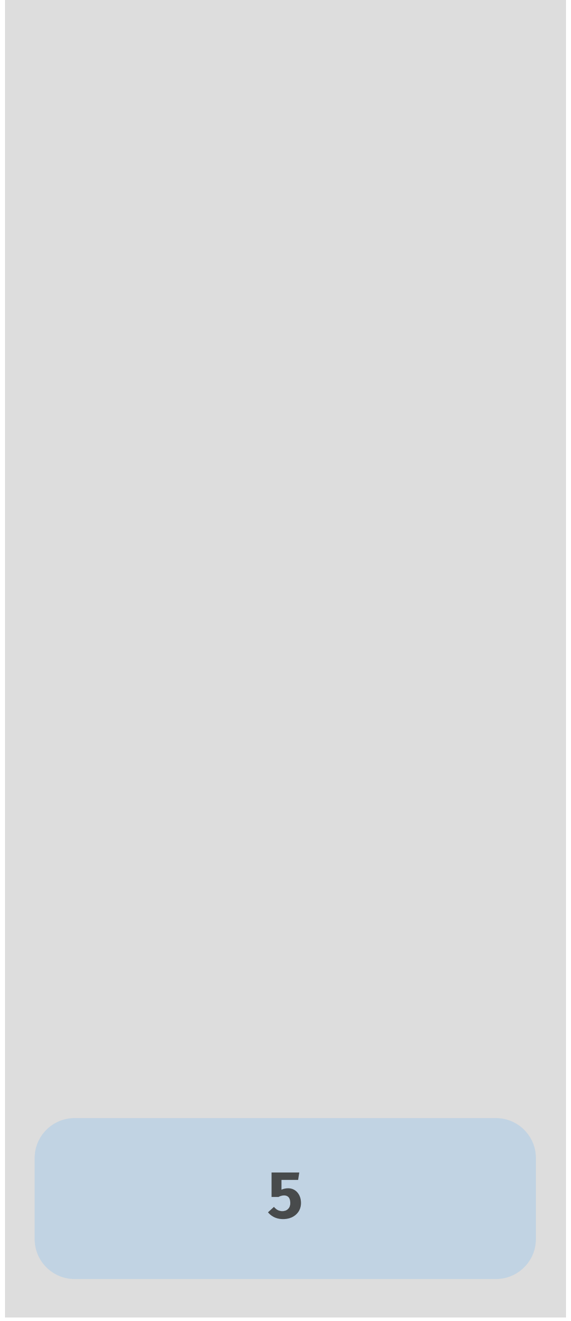
```
(module

  (import "math" "callback" (func $callback))

  (export "add" (func $add))
  (export "subtract" (func $subtract))

  (func $add (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.add
  )

  (func $subtract (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.sub
  )
)
```

WebAssembly is Stack-based

```
(module

  (import "math" "callback" (func $callback))

  (export "add" (func $add))
  (export "subtract" (func $subtract))

  (func $add (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.add
  )

  (func $subtract (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.sub
  )
)
```

Stack

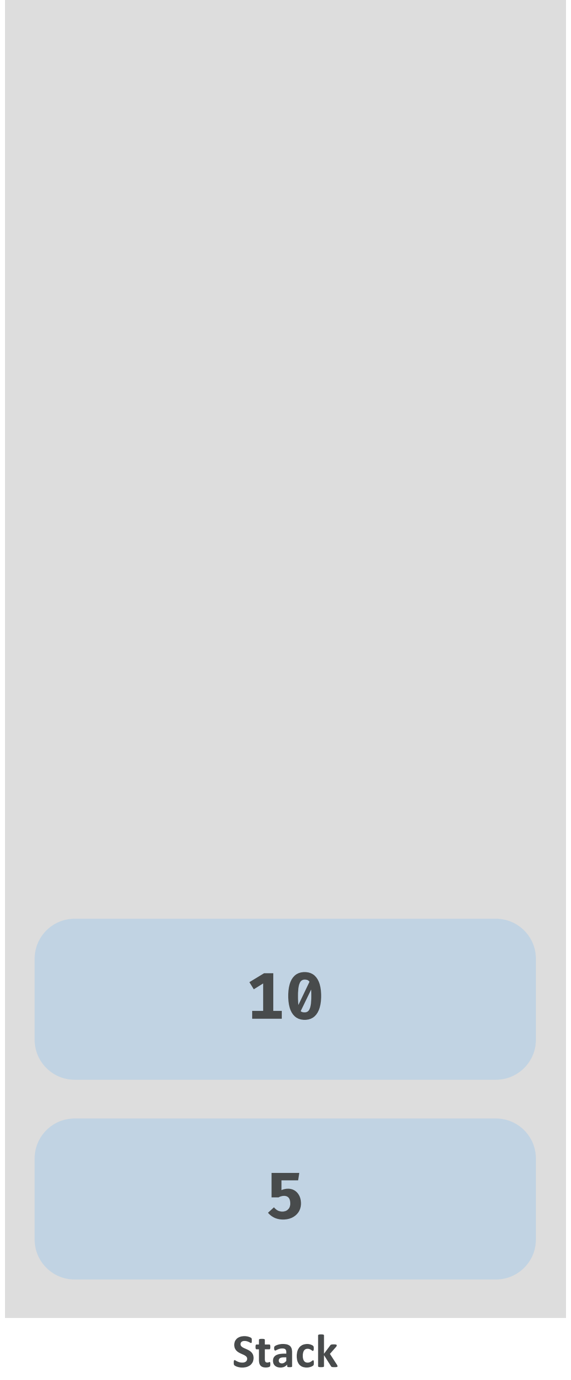
Push an Argument



Stack

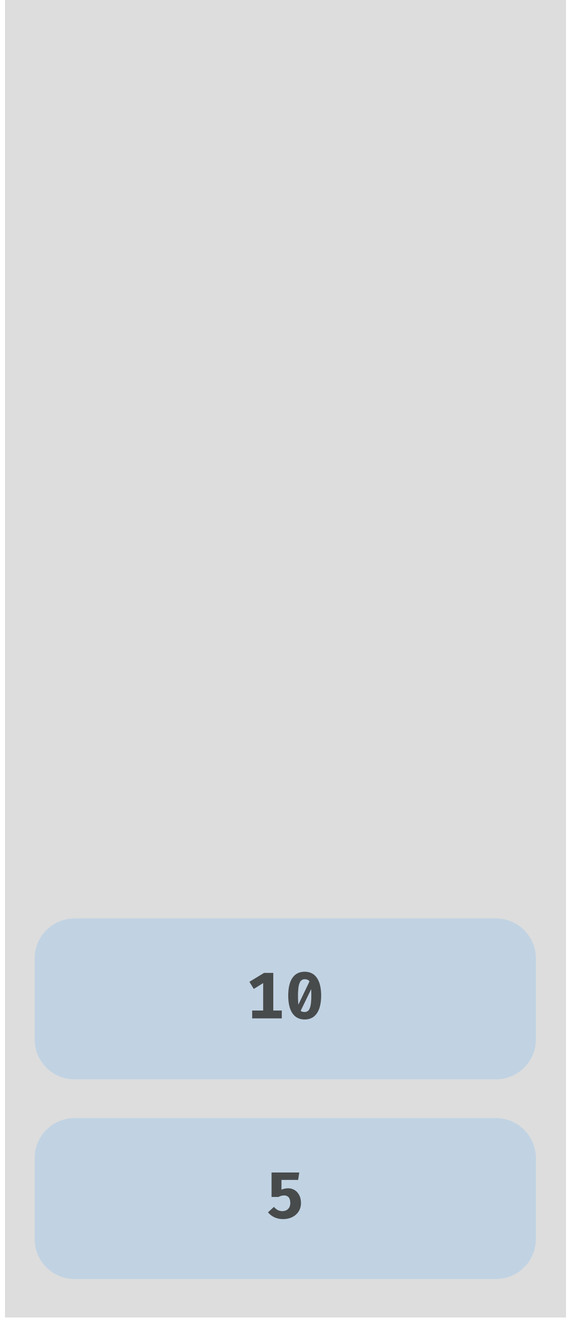
```
(module

  (import "math" "callback" (func $callback))

  (export "add" (func $add))
  (export "subtract" (func $subtract))

  (func $add (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.add
  )

  (func $subtract (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.sub
  )
)
```

Push Another Argument



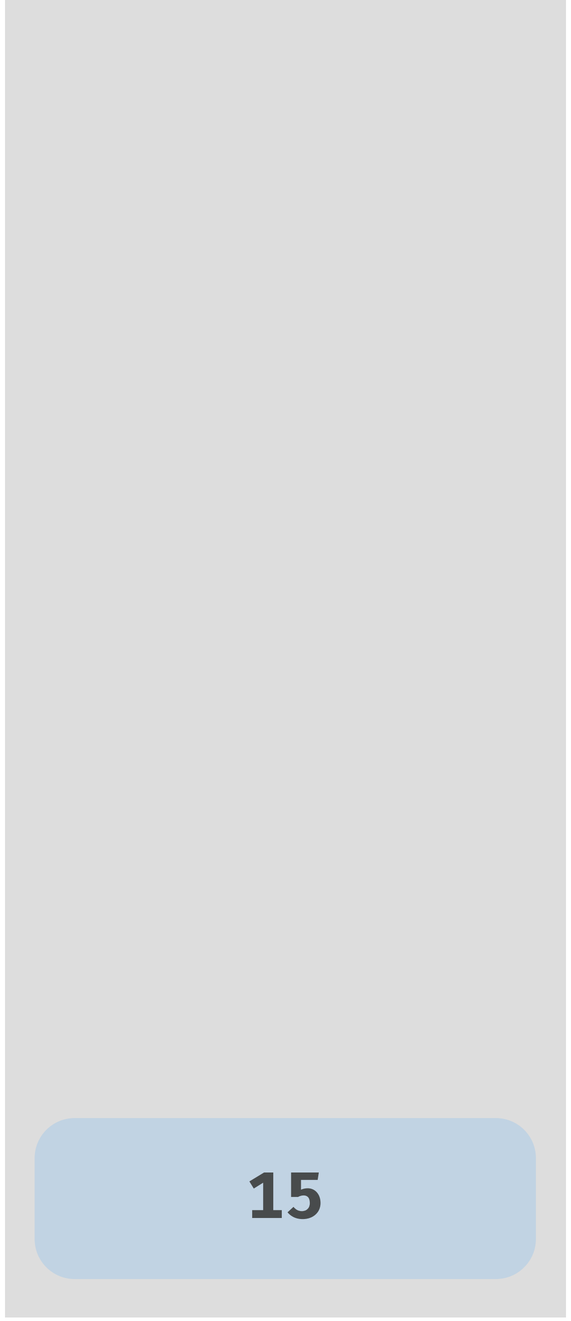
```
(module

  (import "math" "callback" (func $callback))

  (export "add" (func $add))
  (export "subtract" (func $subtract))

  (func $add (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.add
  )

  (func $subtract (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.sub
  )
)
```

Pop the Arguments



Stack

```
(module

  (import "math" "callback" (func $callback))

  (export "add" (func $add))
  (export "subtract" (func $subtract))

  (func $add (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.add
  )

  (func $subtract (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.sub
  )
)
```



Stack

Add & Push

```
(module

  (import "math" "callback" (func $callback))

  (export "add" (func $add))
  (export "subtract" (func $subtract))

  (func $add (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.add
  )

  (func $subtract (param $a i32) (param $b i32) (result i32)
    local.get $a
    local.get $b
    i32.sub
  )
)
```

A Fancier Example (with Comments)

```
00 61 73 6D 01 00
00 00 01 10 03 60
01 7F 00 60 01 7F
01 7F 60 02 7F 7F
01 7F 02 15 01 08
66 69 7A 7A 62 75
7A 7A 08 63 61 6C
6C 62 61 63 6B 00
00 03 03 02 01 02
07 0C 01 08 66 69
7A 7A 62 75 7A 7A
00 01 0A 44 02 39
00 20 00 41 0F 10
02 04 40 41 7D 10
00 41 7D 0F 0B 20
00 41 05 10 02 04
40 41 7E 10 00 41
7E 0F 0B 20 00 41
03 10 02 04 40 41
7F 10 00 41 7F 0F
0B 20 00 10 00 20
```

```
(module

  (import "math" "callback" (func $callback))
  (export "add" (func $add))

  (func $add (param $a i32) (param $b i32) (result i32)
    (local $sum i32)   ;; define a local variable

    local.get $a
    local.get $b
    i32.add
    local.set $sum     ;; set the result to $sum

    local.get $sum     ;; put $sum on the stack
    call $callback     ;; call the callback with $sum

    local.get $sum     ;; put $sum on the stack again
  )
)
```

Defining and Setting a Local Variable

```
00 61 73 6D 01 00
00 00 01 10 03 60
01 7F 00 60 01 7F
01 7F 60 02 7F 7F
01 7F 02 15 01 08
66 69 7A 7A 62 75
7A 7A 08 63 61 6C
6C 62 61 63 6B 00
00 03 03 02 01 02
07 0C 01 08 66 69
7A 7A 62 75 7A 7A
00 01 0A 44 02 39
00 20 00 41 0F 10
02 04 40 41 7D 10
00 41 7D 0F 0B 20
00 41 05 10 02 04
40 41 7E 10 00 41
7E 0F 0B 20 00 41
03 10 02 04 40 41
7F 10 00 41 7F 0F
0B 20 00 10 00 20
```

```
(module

  (import "math" "callback" (func $callback))
  (export "add" (func $add))

  (func $add (param $a i32) (param $b i32) (result i32)
    (local $sum i32)    ;; define a local variable

    local.get $a
    local.get $b
    i32.add
    local.set $sum      ;; set the result to $sum

    local.get $sum      ;; put $sum on the stack
    call $callback      ;; call the callback with $sum

    local.get $sum      ;; put $sum on the stack again
  )
)
```

Calling the Callback

```
00 61 73 6D 01 00
00 00 01 10 03 60
01 7F 00 60 01 7F
01 7F 60 02 7F 7F
01 7F 02 15 01 08
66 69 7A 7A 62 75
7A 7A 08 63 61 6C
6C 62 61 63 6B 00
00 03 03 02 01 02
07 0C 01 08 66 69
7A 7A 62 75 7A 7A
00 01 0A 44 02 39
00 20 00 41 0F 10
02 04 40 41 7D 10
00 41 7D 0F 0B 20
00 41 05 10 02 04
40 41 7E 10 00 41
7E 0F 0B 20 00 41
03 10 02 04 40 41
7F 10 00 41 7F 0F
0B 20 00 10 00 20
```

```
(module

  (import "math" "callback" (func $callback))
  (export "add" (func $add))

  (func $add (param $a i32) (param $b i32) (result i32)
    (local $sum i32)    ;; define a local variable

    local.get $a
    local.get $b
    i32.add
    local.set $sum      ;; set the result to $sum

    local.get $sum      ;; put $sum on the stack
    call $callback      ;; call the callback with $sum

    local.get $sum      ;; put $sum on the stack again
  )
)
```

Mind the Stack

```
00 61 73 6D 01 00
00 00 01 10 03 60
01 7F 00 60 01 7F
01 7F 60 02 7F 7F
01 7F 02 15 01 08
66 69 7A 7A 62 75
7A 7A 08 63 61 6C
6C 62 61 63 6B 00
00 03 03 02 01 02
07 0C 01 08 66 69
7A 7A 62 75 7A 7A
00 01 0A 44 02 39
00 20 00 41 0F 10
02 04 40 41 7D 10
00 41 7D 0F 0B 20
00 41 05 10 02 04
40 41 7E 10 00 41
7E 0F 0B 20 00 41
03 10 02 04 40 41
7F 10 00 41 7F 0F
0B 20 00 10 00 20
```

```
(module

  (import "math" "callback" (func $callback))
  (export "add" (func $add))

  (func $add (param $a i32) (param $b i32) (result i32)
    (local $sum i32)    ;; define a local variable

    local.get $a
    local.get $b
    i32.add
    local.set $sum      ;; set the result to $sum

    local.get $sum      ;; put $sum on the stack
    call $callback      ;; call the callback with $sum

    local.get $sum      ;; put $sum on the stack again
  )
)
```

S-Expressions

```
00 61 73 6D 01 00
00 00 01 10 03 60
01 7F 00 60 01 7F
01 7F 60 02 7F 7F
01 7F 02 15 01 08
66 69 7A 7A 62 75
7A 7A 08 63 61 6C
6C 62 61 63 6B 00
00 03 03 02 01 02
07 0C 01 08 66 69
7A 7A 62 75 7A 7A
00 01 0A 44 02 39
00 20 00 41 0F 10
02 04 40 41 7D 10
00 41 7D 0F 0B 20
00 41 05 10 02 04
40 41 7E 10 00 41
7E 0F 0B 20 00 41
03 10 02 04 40 41
7F 10 00 41 7F 0F
0B 20 00 10 00 20
```

```
(module

  (import "math" "callback" (func $callback))
  (export "add" (func $add))

  (func $add (param $a i32) (param $b i32) (result i32)
    (local $sum i32)

    (local.set $sum
      (i32.add (local.get $a) (local.get $a)))

    (call $callback (local.get $sum))

    (return (local.get $sum))
  )
)
```

Other Stuff

Shared Memory

```
(memory 1)

(data
  (i32.const 0)
  "Hello World\n"
)

(func $foo
  (i32.store8
    (i32.const 12)
    (i32.const 0)
  )
)
```

Globals

```
(global $n
  (import "foo" "bar")
  (mut i32)
)

(func $baz (result i32)

 (global.set $n
   (i32.const 42))

 (return
   (global.get $n ))
)
```

Tables

```
(table 3 anyfunc)

(elem (i32.const 0)
  $foo)

(elem (i32.const 1)
  $bar $baz)

(func $foo)
(func $bar)
(func $baz)
```

Other Languages

Awesome Wasm Languages

https://github.com/appcypher/awesome-wasm-langs

Coming Down the Pike

Threads

Interface Types

WASI

Demo

Resources

- WebAssembly.org - official site for WebAssembly
- GitHub.com/WebAssembly - all the code and specs
- Mozilla Developer Network Documentation - JavaScript side of things
- WebAssembly Binary Toolkit - command-line assembler and disassembler
- Wasmtime - command-line WebAssembly runtime
- WASI - WebAssembly System Interface, brings I/O to WebAssembly
- Rust - has built-in support for WebAssembly
- Awesome Wasm Languages - .NET, Elixir, Go, Java, Python, et al.

Code and Slides

https://github.com/guyroyse/intro-to-webassembly

I Work For Redis Labs

Redis Discord Server

https://discord.gg/gmCACHU

Redis Community Forums

https://forum.redislabs.com/

Redis University

https://university.redislabs.com/

redislabs
HOME OF REDIS

redislabs
HOME OF REDIS

Guy Royse
Developer Advocate
Redis Labs

@guyroyse
github.com/guyroyse
guy.dev

Thanks!

redislabs
HOME OF REDIS