DataRobot

What's Old is New Again

An Introduction to WebAssembly

DataRobot

Guy Royse

Developer Evangelist

DataRobot

@guyroyse

github.com/guyroyse

What is WebAssembly?

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

WTH, JavaScript?

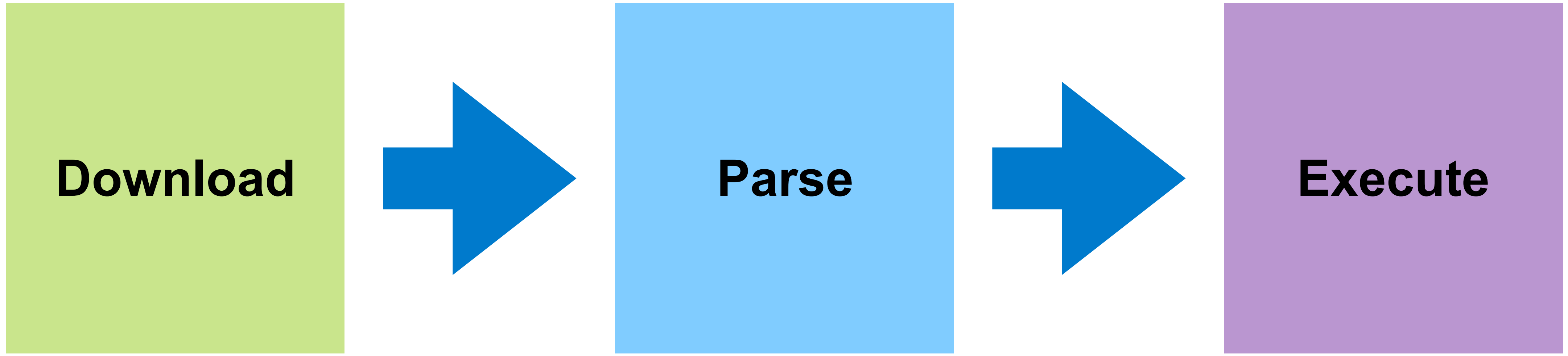
```
[] == ![]; // -> true

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

Modern JavaScript Environment

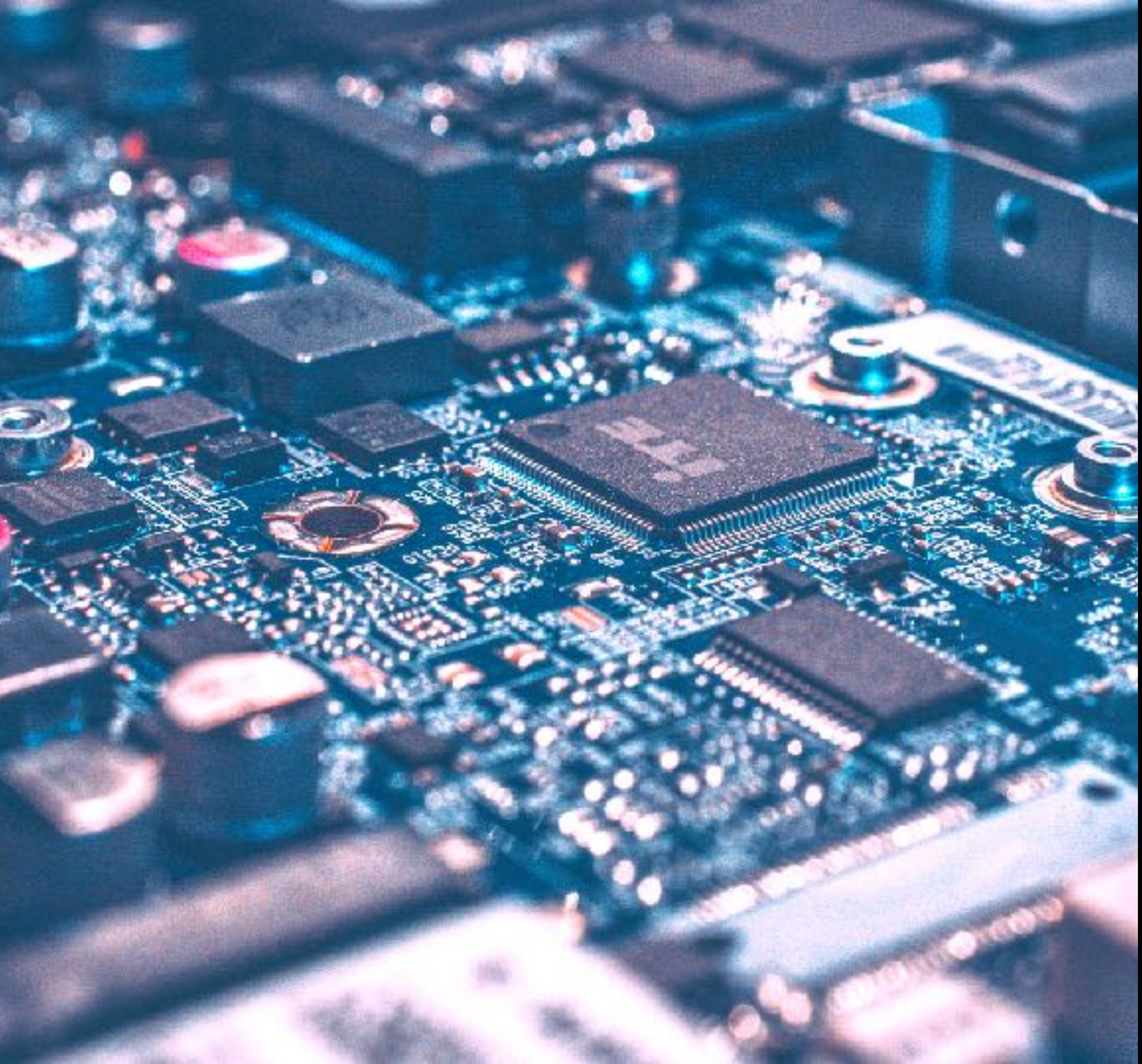
It's easy! Just use these simple tools:

- npm or yarn
- Bower
- Webpack
- Babel
- Grunt or Gulp
- live-server
- React and Redux
- Mocha, Chai, and Sinon


Download
Parse
Execute

What If You Could Compile JavaScript?

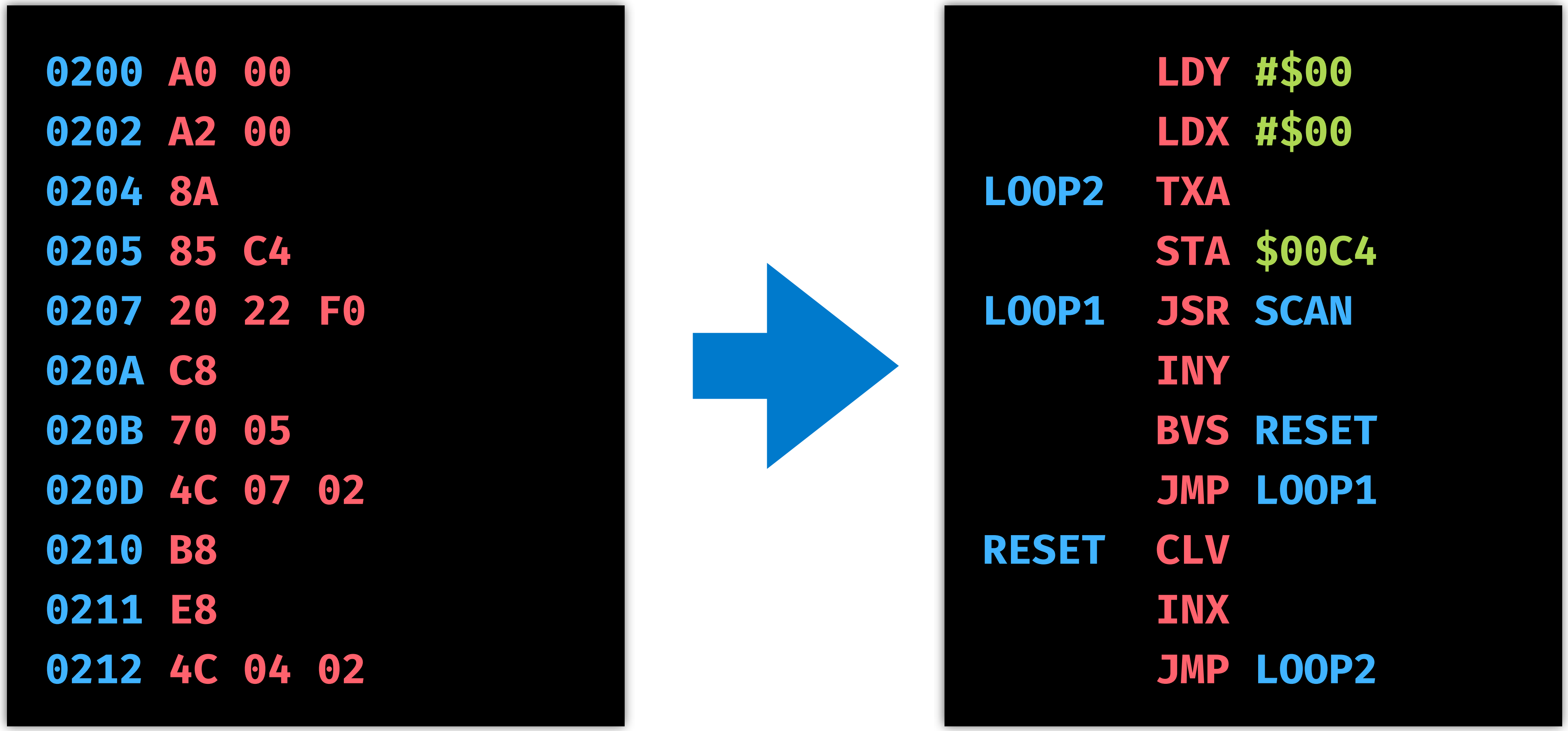
THE *Silver* BULLET®



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

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

```
0200 A0 00
0202 A2 00
0204 8A
0205 85 C4
0207 20 22 F0
020A C8
020B 70 05
020D 4C 07 02
0210 B8
0211 E8
0212 4C 04 02
```

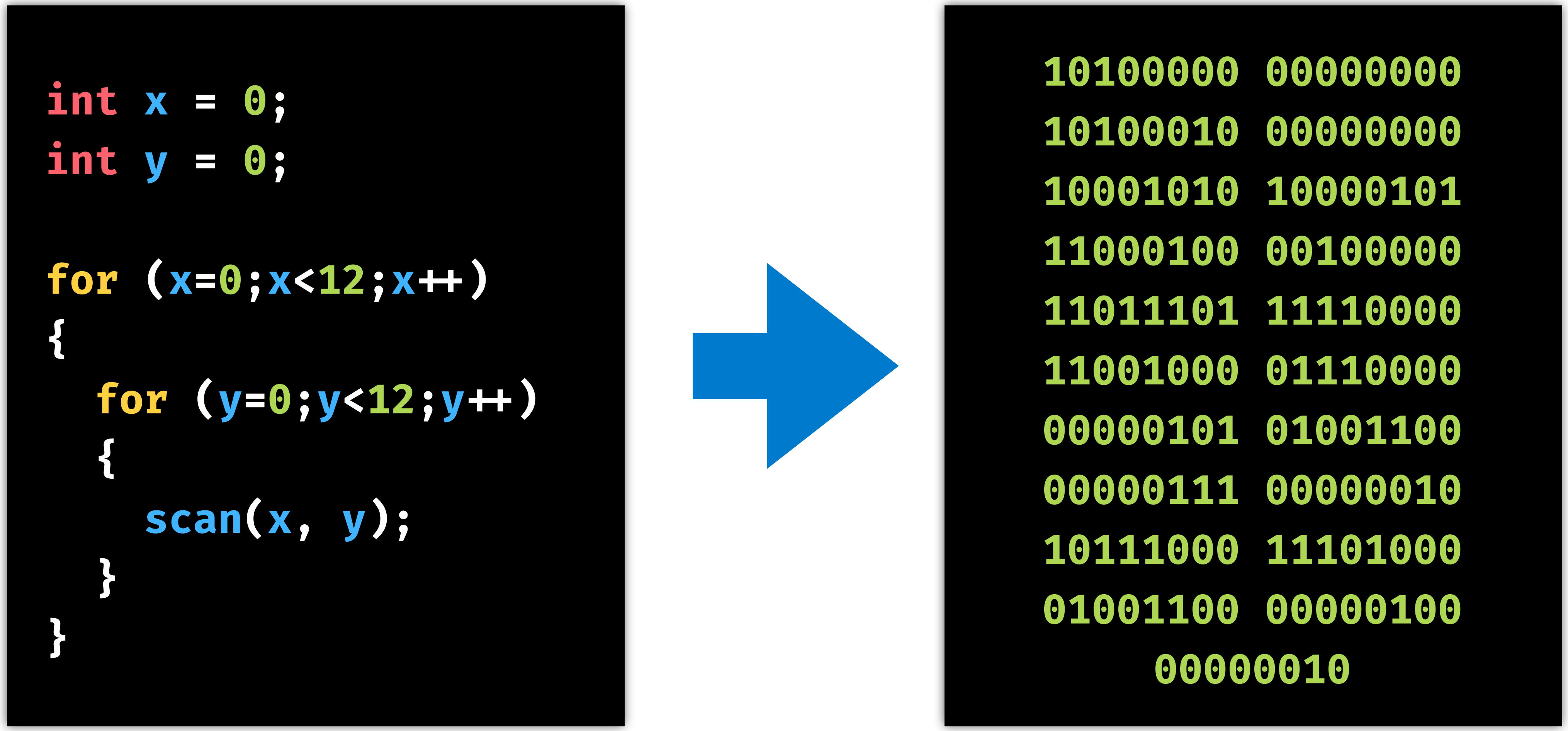


```
0200 A0 00
0202 A2 00
0204 8A
0205 85 C4
0207 20 22 F0
020A C8
020B 70 05
020D 4C 07 02
0210 B8
0211 E8
0212 4C 04 02
```

```
        LDY #$00
        LDX #$00
LOOP2   TXA
        STA $00C4
LOOP1   JSR SCAN
        INY
        BVS RESET
        JMP LOOP1
RESET   CLV
        INX
        JMP LOOP2
```

In the 1950's von Neumann was employed as a consultant to review proposed and ongoing advanced technology projects. One day a week, von Neumann "held court" at 590 Madison Avenue, New York. On one of these occasions in 1954 he was confronted with the FORTRAN concept; John Backus remembered von Neumann being unimpressed and that he asked "why would you want more than machine language?" Frank Beckman, who was also present, recalled that von Neumann dismissed the whole development as "but an application of the idea of Turing's `short code'." Donald Gilles, one of von Neumann's students at Princeton, and later a faculty member at the University of Illinois, recalled in the mid-1970's that the graduates students were being "used" to hand assemble programs into binary for their early machine (probably the IAS machine). He took time out to build an assembler, but when von Neumann found out about he was very angry, saying (paraphrased), "It is a waste of a valuable scientific computing instrument to use it to do clerical work." (source: John von Neuman and von Neumann Architecture for Computers (1945))

```
        LDY #$00       0200   AØ ØØ
        LDX #$00       0202   A2 ØØ
LOOP2   TXA            0204   8A
        STA $00C4      0205   85 C4
LOOP1   JSR SCAN       0207   2Ø 22 FØ
        INY            020A   C8
        BVS RESET      020B   7Ø Ø5
        JMP LOOP1      020D   4C Ø7 Ø2
RESET   CLV            0210   B8
        INX            0211   E8
        JMP LOOP2      0212   4C Ø4 Ø2
```

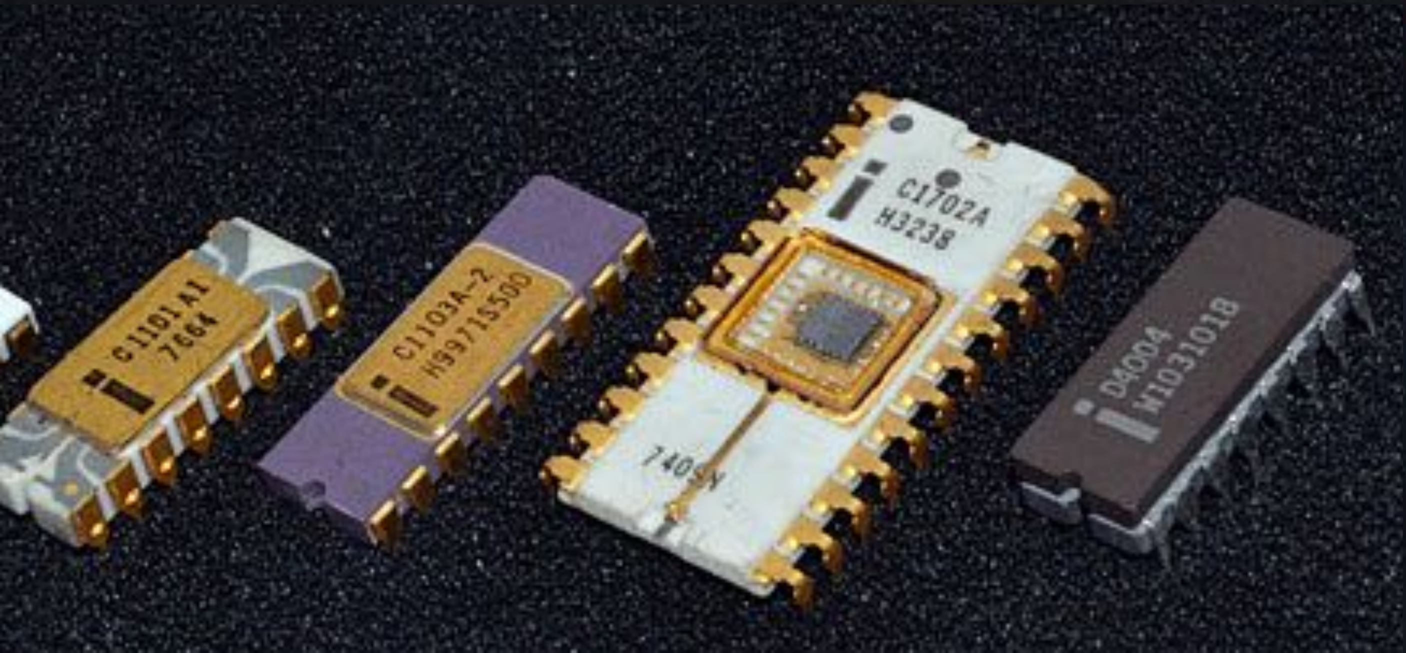


```
int x = 0;
int y = 0;

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

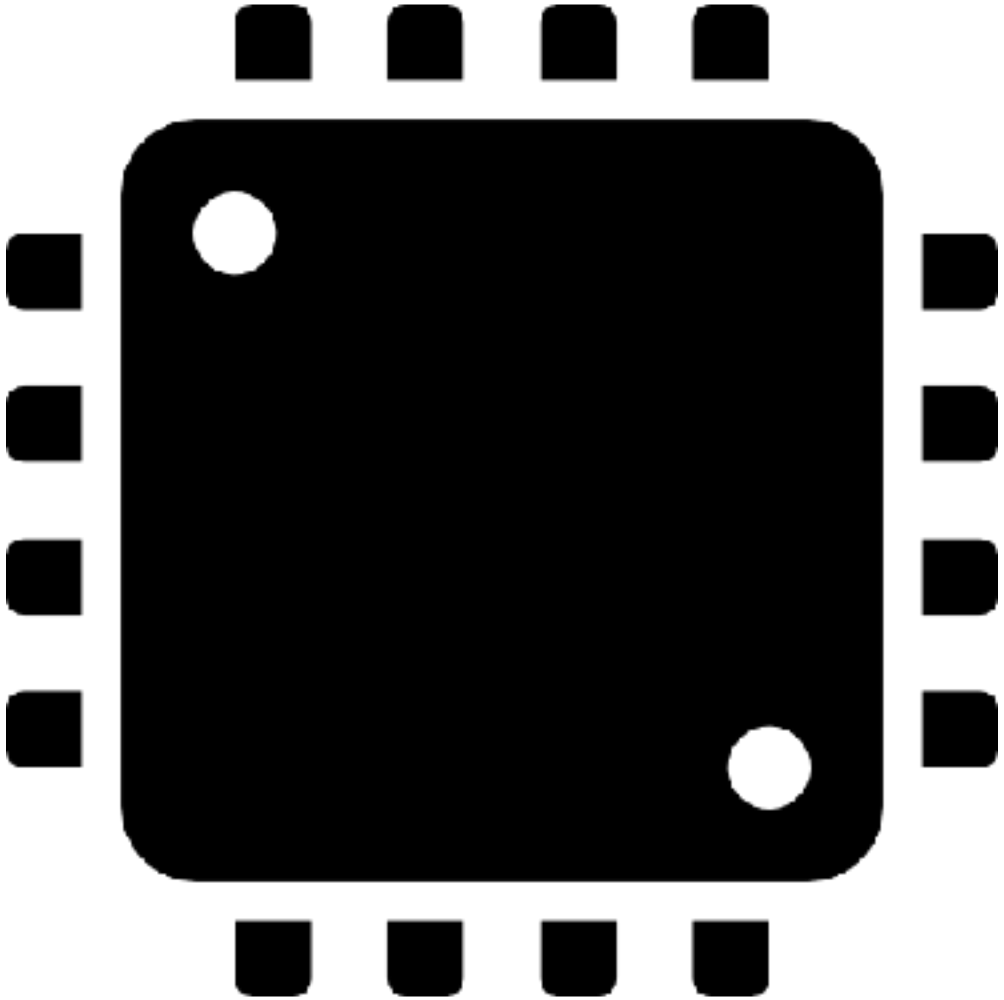
```
10100000 00000000
10100010 00000000
10001010 10000101
11000100 00100000
11011101 11110000
11001000 01110000
00000101 01001100
00000111 00000010
10111000 11101000
01001100 00000100
     00000010
```

6502AD
4585 S

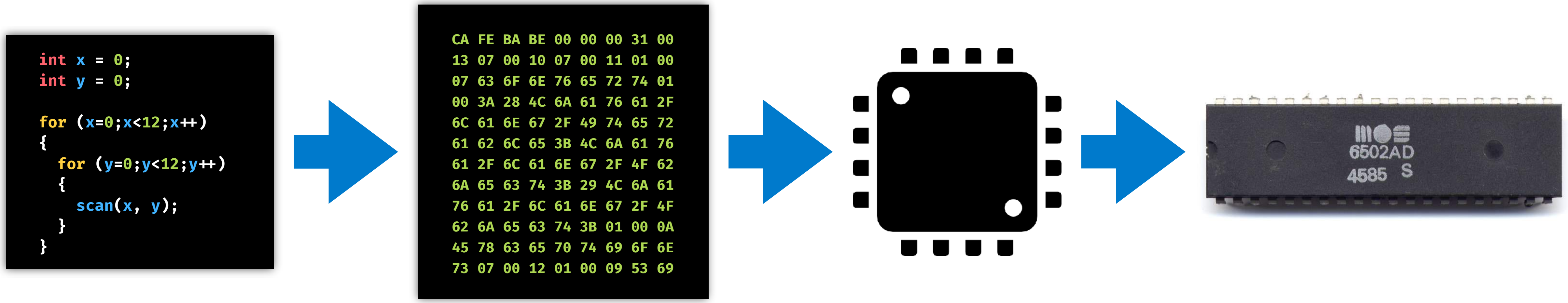
XILINX
HKb2bsite.com


C1101A1
7664
C1103A-2
H9971S500
C1702A
H3238
740SN
D4004
W1031018

8231346
MOXA
Raritan
intel StrongARM
PMC
QLOGIC
BROADCOM
Agilent
MOTOROLA
A5
DS89C420
MCS
RoHS
SIEMENS



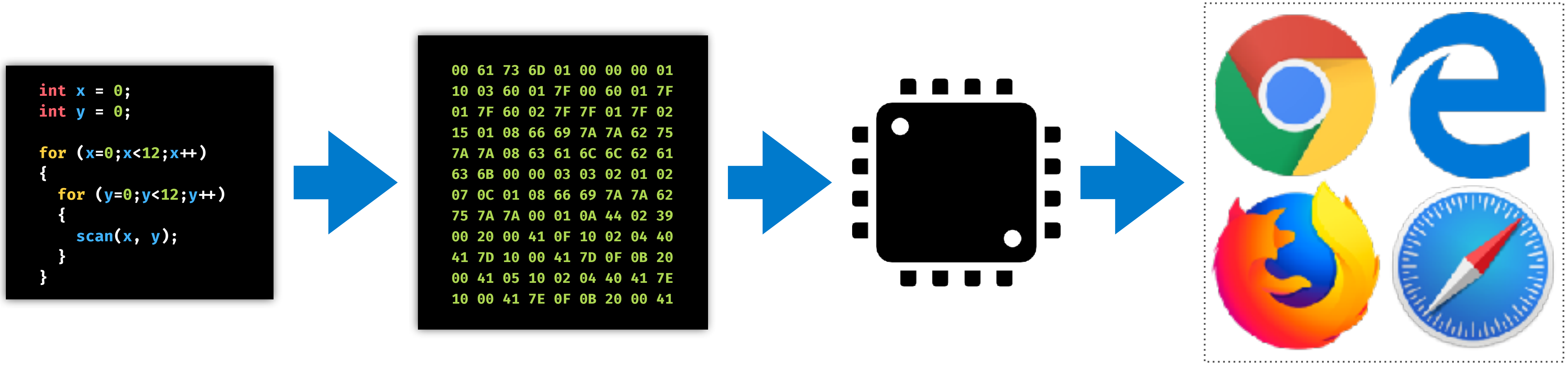
How Virtual Machines Work



```
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
```

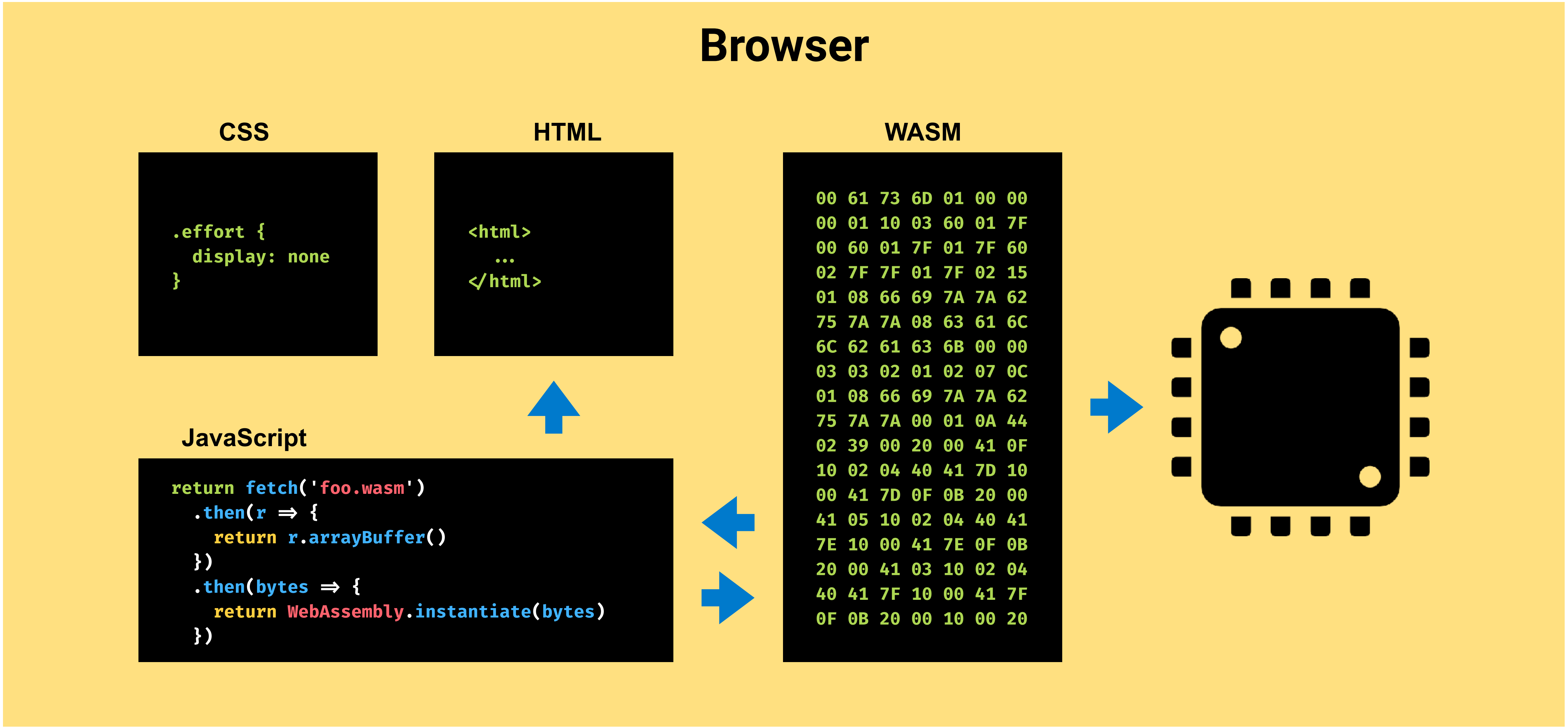
How WebAssembly Works



```
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
```

More How WebAssembly Works



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

```html
<html>
   ...
</html>
```

```
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
```

```js
return fetch('foo.wasm')
  .then(r => {
    return r.arrayBuffer()
  })
  .then(bytes => {
    return WebAssembly.instantiate(bytes)
  })
```

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
```

- Hosted with other web content
- Content-Type of application/wasm
- Instantiated from JavaScript

```
return fetch(filename)
  .then(response ⇒ response.arrayBuffer())
  .then(bytes ⇒ WebAssembly.instantiate(bytes))
  .then(module ⇒ {
    // use webassembly module
  })
```

WebAssembly - 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
```

- Modules export functions
- Just call them

```
fetch('foo.wasm')
  .then(response => response.arrayBuffer())
  .then(bytes => WebAssembly.instantiate(bytes))
  .then(module => {
    let x = module.instance.exports.fizzbuzz(15)
    let y = module.instance.exports.volume(1, 3, 5)
    console.log(x, y)
  })
```

WebAssembly - 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
```

- Instantiate with imports
- Provides functions module can call

```
let imports = {
  imports: {
    callback: x => console.log(x)
  }
}

fetch('foo.wasm')
  .then(response => response.arrayBuffer())
  .then(bytes => WebAssembly.instantiate(bytes, imports))
  .then(module => {
    module.instance.exports.fizzbuzzCallback(15)
 })
```

WebAssembly - Shared Memory

```
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
```

- Functions can only take numbers
- Use shared memory to pass more complex data

```
let imports = {
  imports: {
    memory: new WebAssembly.Memory({ initial: 10 })
  }
}

let shared = new Uint8Array(imports.imports.memory.buffer)

shared[0] = 1; shared[1] = 2; shared[2] = 3;
```

Demo

Resources

- MDN Articles
- WebAssembly.org
- WebAssembly Binary Toolkit - assembler and disassembler
- WebAssembly Toolkit for VSCode - assembly code highlighter and assembler
- Rust - has built-in support for WebAssembly
- Emscripten - LLVM bitcode to WebAssembly
- Other Languages - .NET, Elixir, Go, Java, Python, et al.
- Awesome Wasm - curated list of awesome WebAssembly things
- WasmExplorer - compile C++ code in a browser to WASM
- WebAssembly Studio - web-based WebAssembly IDE for C, Rust, and WAT

github.com/guyroyse/intro-to-webassembly

Questions?

Guy Royse

Developer Evangelist
DataRobot

@guyroyse
github.com/guyroyse

DataRobot