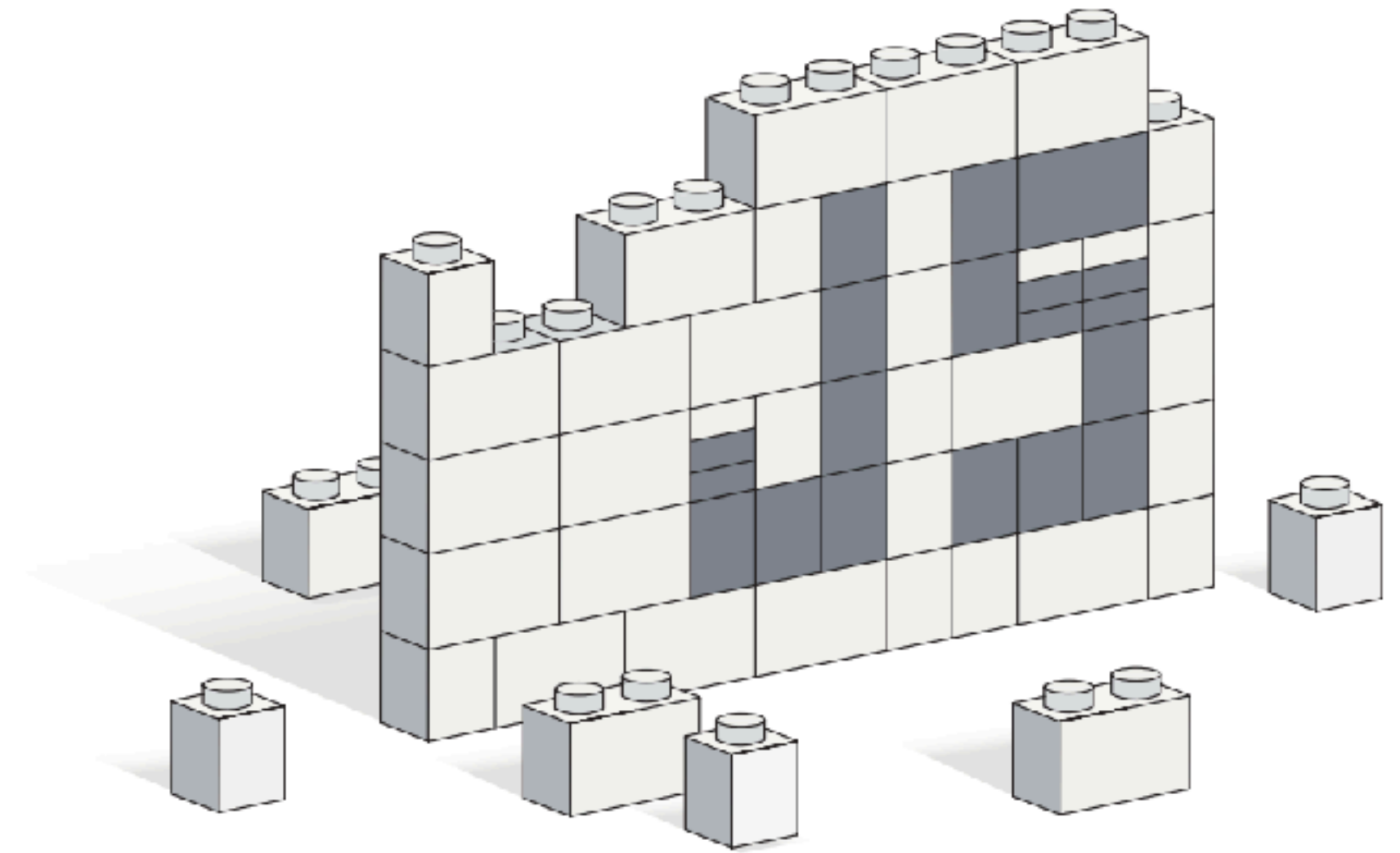


# WebAssembly

and the death of JavaScript?

@ColinEberhardt

**JavaScript**  
**Brendan Eich**  
**1995**  
**10 days**



# ActiveX

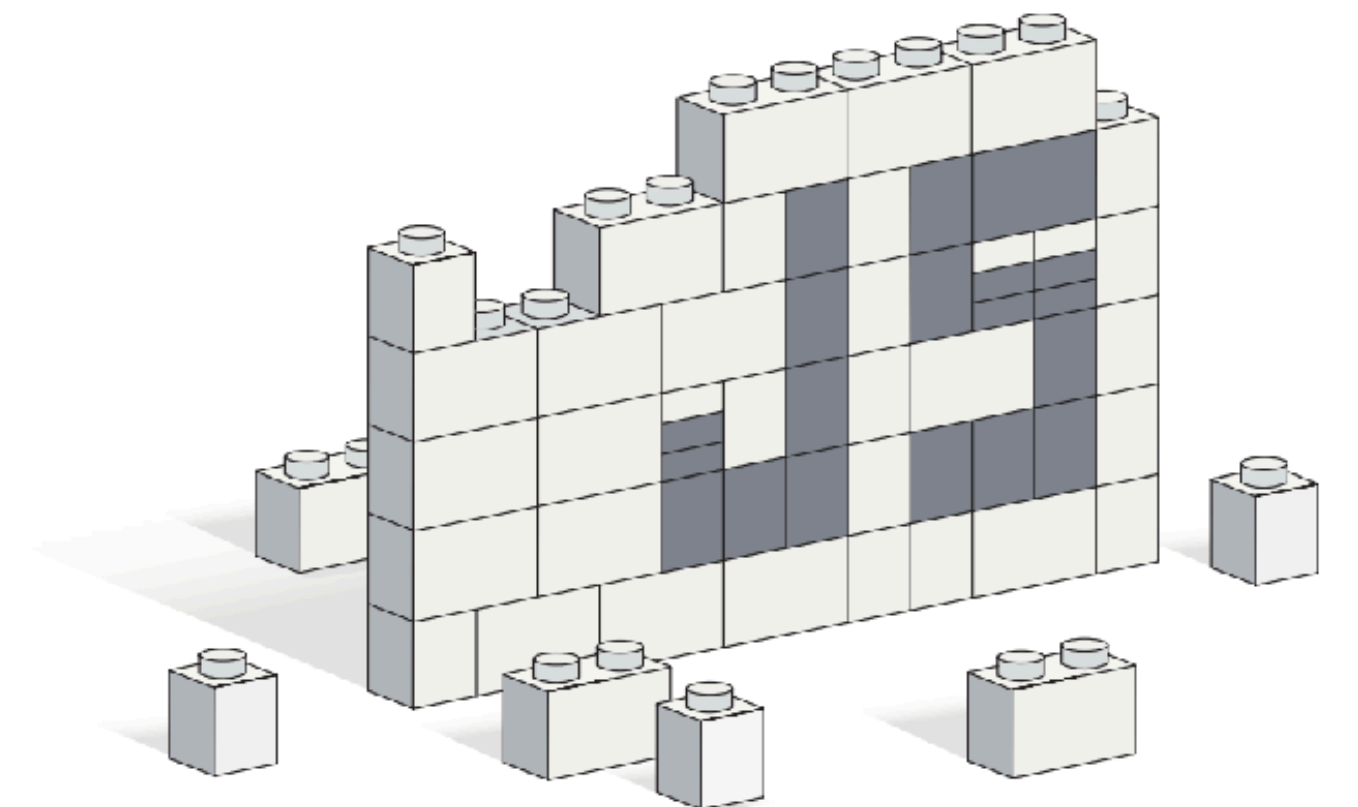
# Flash

# Silverlight

# Dart

**2018, JavaScript  
... still**

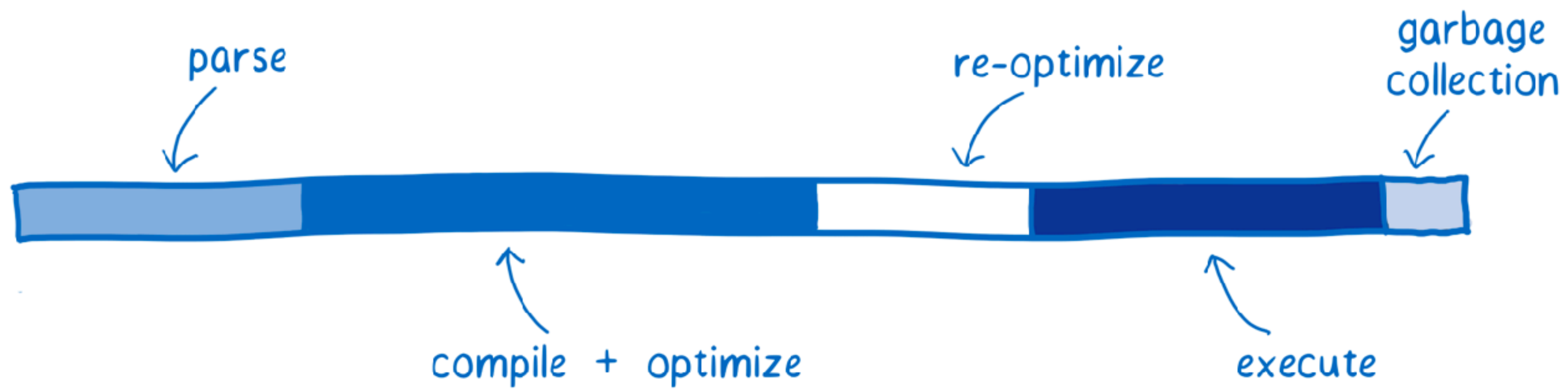
**We are writing a *lot* of  
JavaScript**





```
a(c,d,{configurable:!0,writable:!0,value:b}}},ca=function(){ca=function(){};
(r.Symbol=da)},da=function(){var b=0;return function(c){return"jscomp_symbol_"+(c||"")+b++}}(),
.iterator;b||(b=r.Symbol.iterator=r.Symbol("iterator"));typeof Array.prototype[b]!==m&&aa(Array.
0,value:function(){return ea(this)}});fa=function(){}},ea=function(b){var c=0;return ha(function
ne:!0}})},ha=function(b){fa();b={next:b};b[r.Symbol.iterator]=function(){return this;
ia=function(b){fa();var c=b[Symbol.iterator];return c?c.call(b):ea(b)};ba(function(b){function
of f?b:new f(function(c){c(b)}))}if(b)return b;c.prototype.c=function(b){null==this.b&&(this.b=[
e.g=function(){var b=this;this.f(function(){b.i()});var e=r.setTimeout;c.prototype.f=function(
b&&this.b.length;){var b=this.b;this.b=[];for(var c=0;c<b.length;++c){var d=
e b[c];try{d()}catch(l){this.h(l)}}this.b=null;c.prototype.h=function(b){this.f(function(){th
d 0;this.b=[];var c=this.f();try{b(c.resolve,c.reject)}catch(n){c.reject(n)}};f.prototype.f=fun
!0,b.call(c,e))}}var c=this,d=!1;return{resolve:b(this.B),reject:b(this.g)}};f.prototype.B=func
'A Promise cannot resolve to itself'));else if(b instanceof f)this.C(b);
ch(typeof b){case "object":var c=null!=b;break a;case m:c=!0;break a;default:c=!1}c?this.A(b):
d 0;try{c=b.then}catch(n){this.g(n);return}typeof c==m?this.D(c,b):this.i(b)};f.prototype.g=fun
e.i=function(b){this.j(1,b)};f.prototype.j=function(b,c){if(0!=this.c)throw Error("Cannot settl
state"+this.c);this.c=b;this.h=c;this.l()};f.prototype.l=function(){if(null!=this.b){for(var b
c<b.length;++c)b[c].call(),b[c]=null;this.b=null}};var g=new c;f.prototype.C=function(b){var c
e.D=function(b,c){var d=this.f();try{b.call(c,d.resolve,d.reject)}catch(l){d.reject(l)}};f.prot
typeof b==m?function(c){try{e(b(c))}catch(Za){g(Za)}}:c}var e,g,h=new f(function(b,c){e=b;g=c})
e["catch"]=function(b){return this.then(void 0,b)};f.prototype.o=function(b,
d()\[switch(c,c)\[case 1:b(c,b);break;case 2:c(c,b);break;default:throw Error("`"+c+"`");\]]var
```



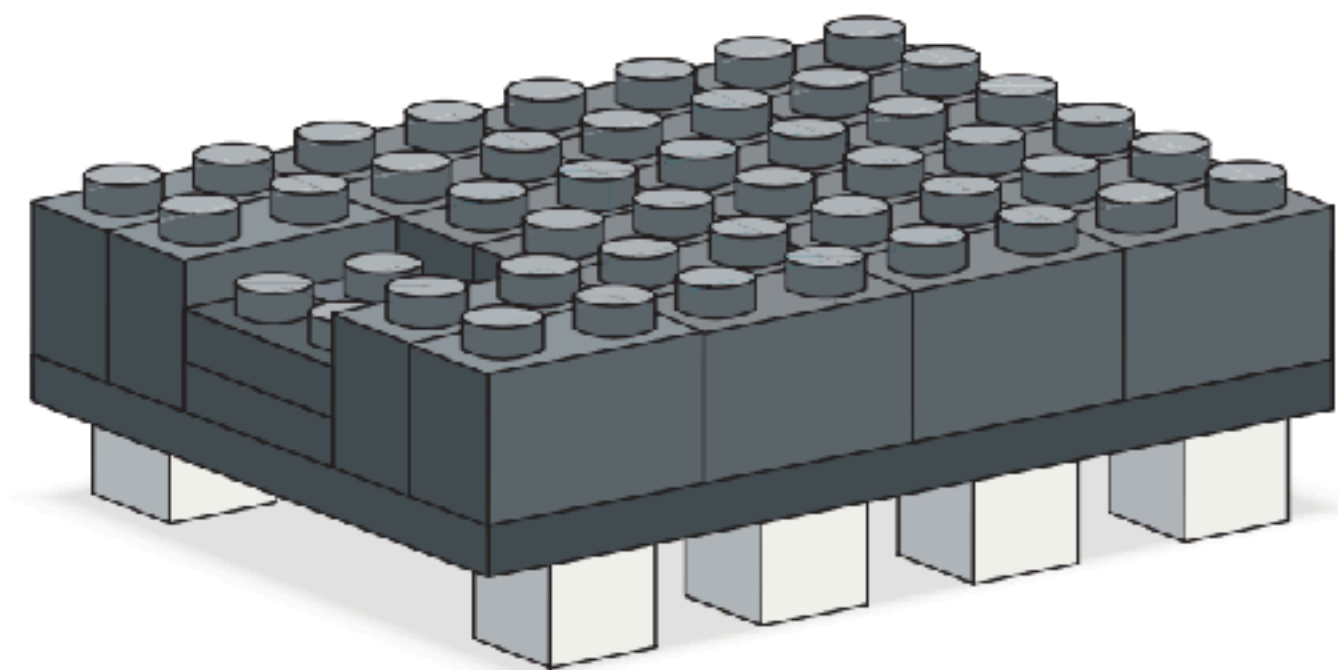


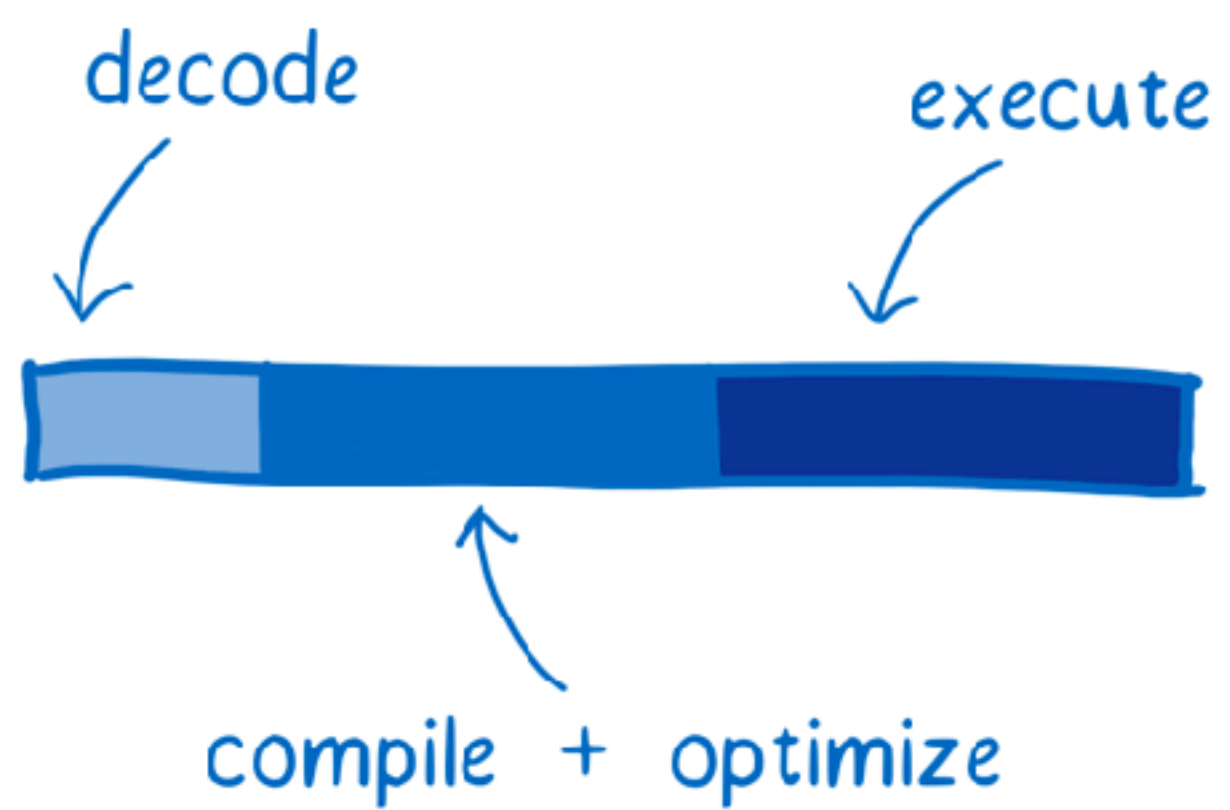
*// the Web has become the most ubiquitous  
application platform ever, and yet by historical  
accident the only natively supported programming  
language for that platform is JavaScript! //*



**WEBASSEMBLY**

*“WebAssembly or wasm is a new portable, size- and load-time-efficient format suitable for compilation to the web.”*

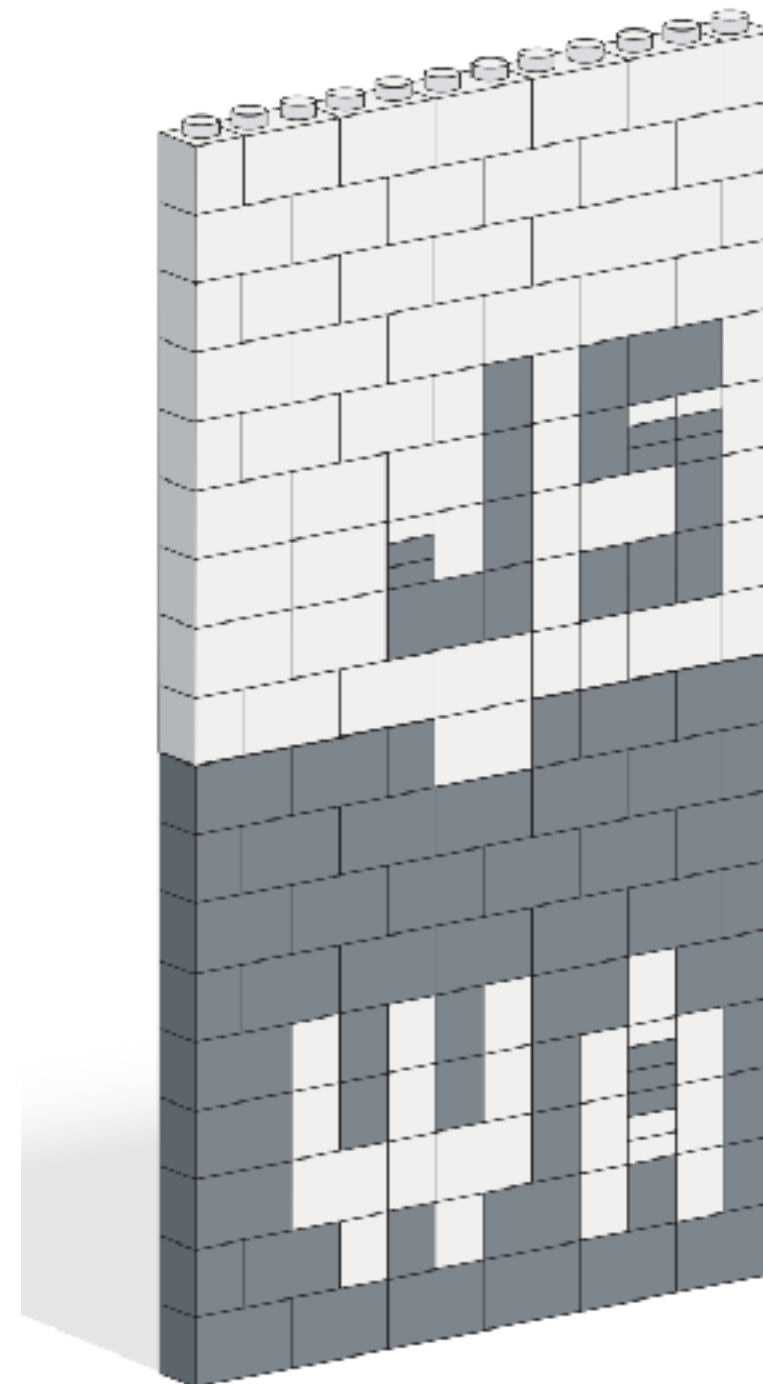




```

00000000: 0061 736d 0100 0000 0107 0160 027d 7f01  .asm.....`.}..
00000010: 7d03 0201 0004 0401 7000 0005 0301 0001  }.....p.....
00000020: 0712 0206 6d65 6d6f 7279 0200 0570 6f77  ....memory...pow
00000030: 6572 0000 0a33 0131 0101 7d02 4020 0141  er...3.1..}.@ .A
00000040: 0248 0d00 2001 417f 6a21 0120 0021 0203  .H.. .A.j!. .!..
00000050: 4020 0220 0094 2102 2001 417f 6a22 010d  @ . ...!. .A.j"..
00000060: 000b 2002 0f0b 2000 0b                .. ... ..

```



```

// read the binary into a buffer
const fs = require("fs");
const buf = fs.readFileSync("./add.wasm");

// create a wasm module
const wasmModule = new WebAssembly.Module(new Uint8Array(buf));

// construct an instance of the module
const wasmInstance = new WebAssembly.Instance(wasmModule);

// invoke the exported function
const result = wasmInstance.exports.power(2, 3)
console.log(result);

```

1 / 3



## PSPDFKit for Web Demo



Hello!

This is a sample PDF document that showcases the functionality and rendering of PSPDFKit for Web, our JavaScript-based PDF SDK. If you have any additional questions regarding PSPDFKit for Web or our other SDK products, please [get in touch!](#)

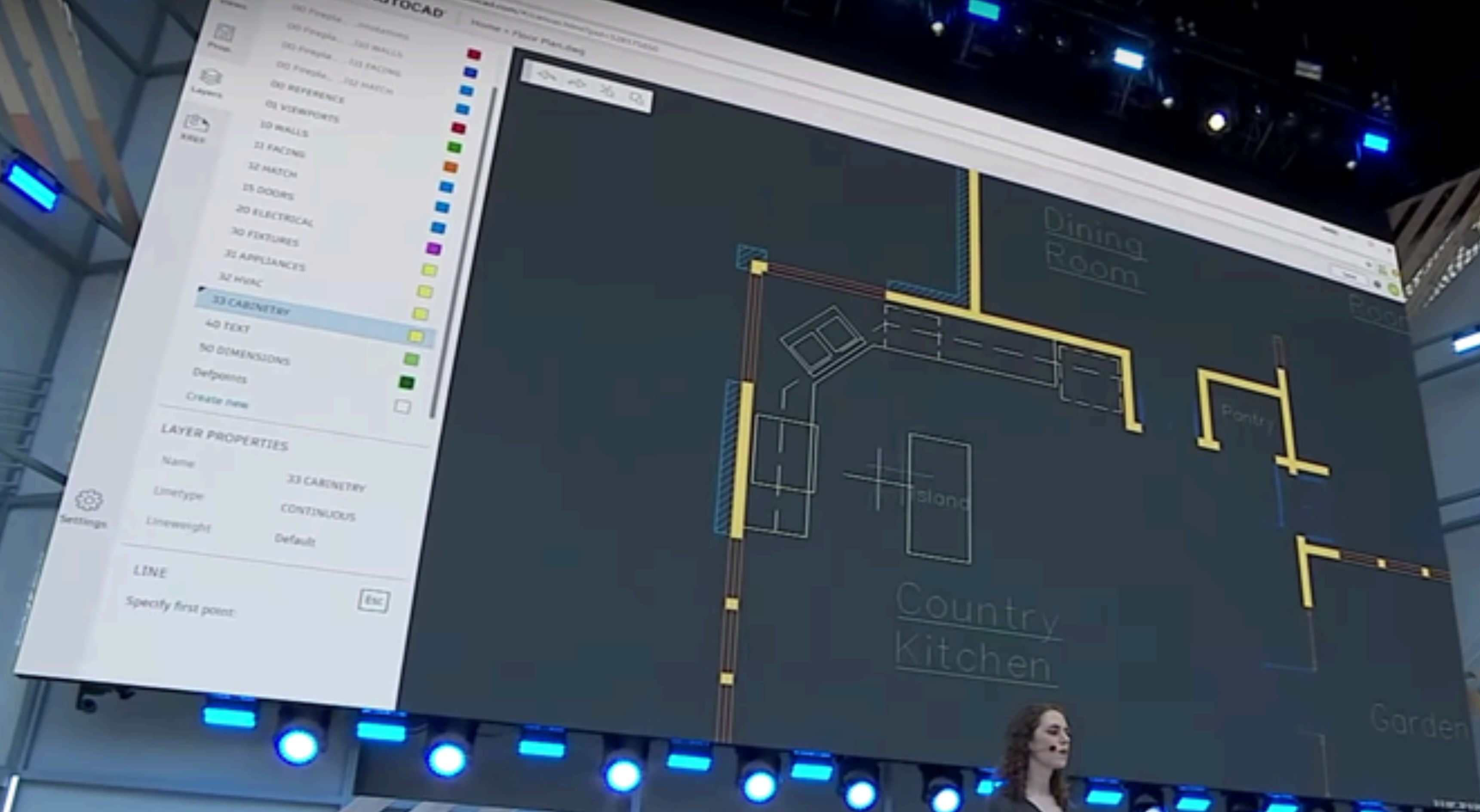
— the PSPDFKit Team



Scroll down  
for more!





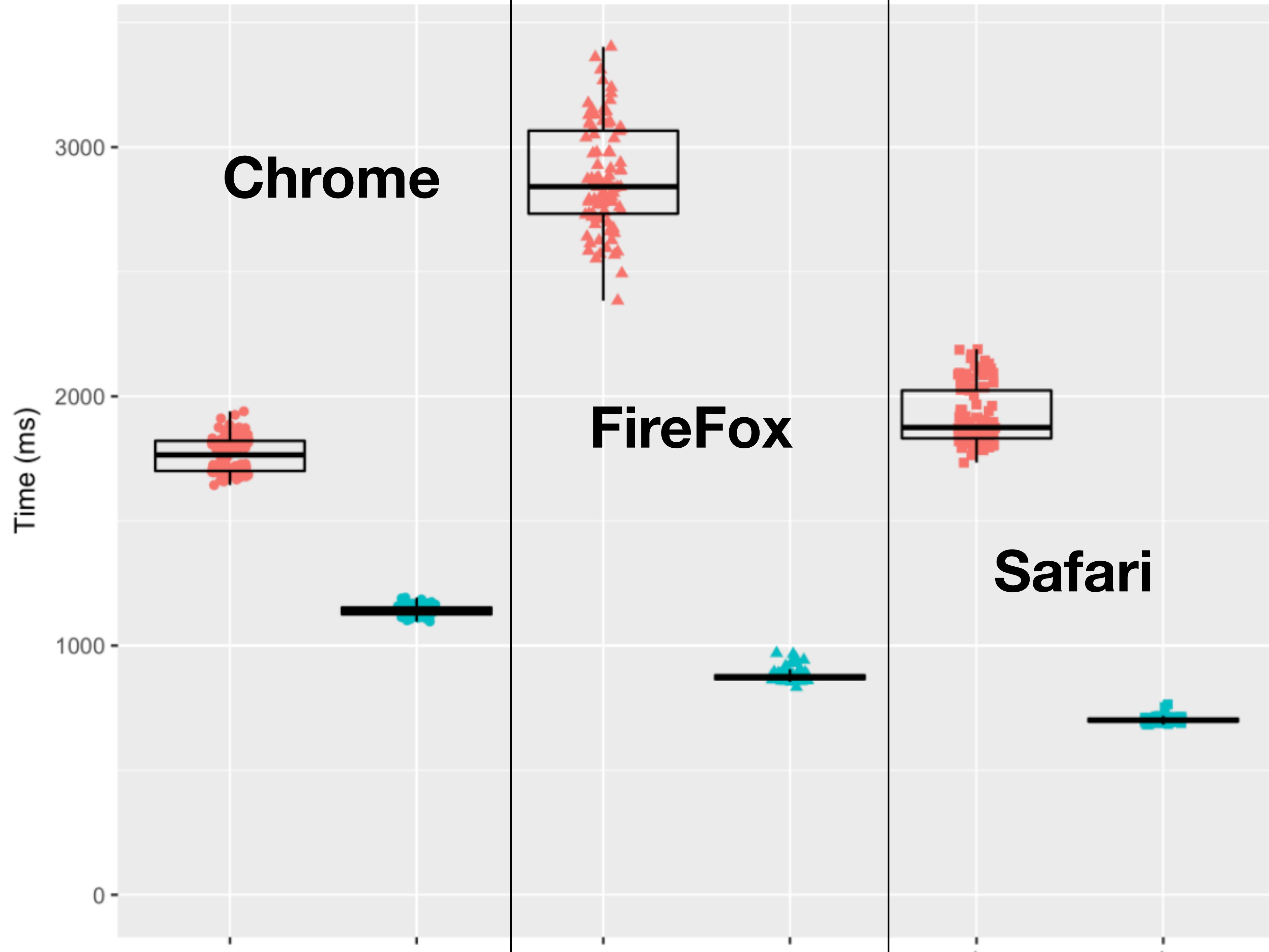




## Option Chain

An option chain demonstrating connectivity, real-time subscriptions to data and traversal of relationships. [Open the control panel](#) for stream settings and statistics.

Call						Put						
OI	Volume	Change	Last	Bid	Ask	Strike	Bid	Ask	Last	Change	Volume	OI
						Dec 1, 2017						
59				23.80 ↓	24.05 ↑	150.00		0.02 ↓	0.02	0.01	200	798
17				21.20 ↓	21.60 ↓	152.50		0.02 ↓				964
87	4	-0.10 ↓	18.90 ↓	18.80 ↑	19.10 ↑	155.00		0.03 ↓				2,098
332				16.35 ↑	16.55 ↑	157.50		0.01 ↓	0.01	0.00	2	1,605
857	1	-0.22	14.15	13.85 ↑	14.10 ↑	160.00	0.01	0.02 ↓	0.02 ↑	0.00 ↑	84	3,933
665	2	0.25	11.60	11.40 ↑	11.55 ↑	162.50	0.03 ↑	0.04 ↑	0.03 ↑	-0.01 ↑	201	4,356
2,847	3,524	-0.30 ↑	8.95 ↑	8.90 ↑	9.10 ↑	165.00	0.05 ↑	0.06 ↑	0.06 ↑	0.00 ↑	296	9,143
898	21	-0.33 ↓	6.40 ↓	6.45 ↑	6.60 ↓	167.50	0.09 ↓	0.11 ↑	0.10 ↑	0.00 ↑	648	5,920
5,897	196	-0.25 ↑	4.15 ↑	4.10 ↑	4.20 ↑	170.00	0.20 ↓	0.21 ↓	0.21 ↓	-0.01 ↓	1,952	7,856
12,536	3,258	-0.17 ↑	2.05 ↑	2.01 ↑	2.04 ↑	172.50	0.58 ↑	0.59 ↑	0.58 ↓	0.02 ↓	3,531	9,265
25,636	8,835	-0.10 ↑	0.65 ↑	0.63 ↓	0.65 ↑	175.00	1.70 ↑	1.71 ↓	1.70 ↑	0.08 ↑	4,116	10,876
19,956	1,859	-0.04 ↓	0.15 ↓	0.15 ↑	0.16 ↑	177.50	3.65 ↓	3.80 ↓	3.80 ↓	0.25 ↓	151	8,119
11,849	863	-0.02 ↓	0.04 ↓	0.04 ↓	0.05 ↓	180.00	6.05 ↓	6.20 ↓	5.95 ↓	-0.05 ↓	252	338
4,083	600	-0.01 ↓	0.02 ↓	0.02 ↑	0.03 ↑	182.50	8.55 ↑	8.65 ↑	8.35	-0.45	5	63
14,206	10	0.00	0.02	0.01	0.02 ↓	185.00	11.00 ↑	11.25 ↑	11.30	-3.45	8	47
3,390	61	-0.01	0.01		0.01 ↓	187.50	13.35 ↓	13.75 ↓				0
2,928					0.02 ↓	190.00	15.85 ↑	16.35 ↓				2
1,066					0.02 ↓	192.50	18.35 ↓	18.80 ↑				14
240					0.01 ↓	195.00	20.85 ↑	21.25 ↑				26

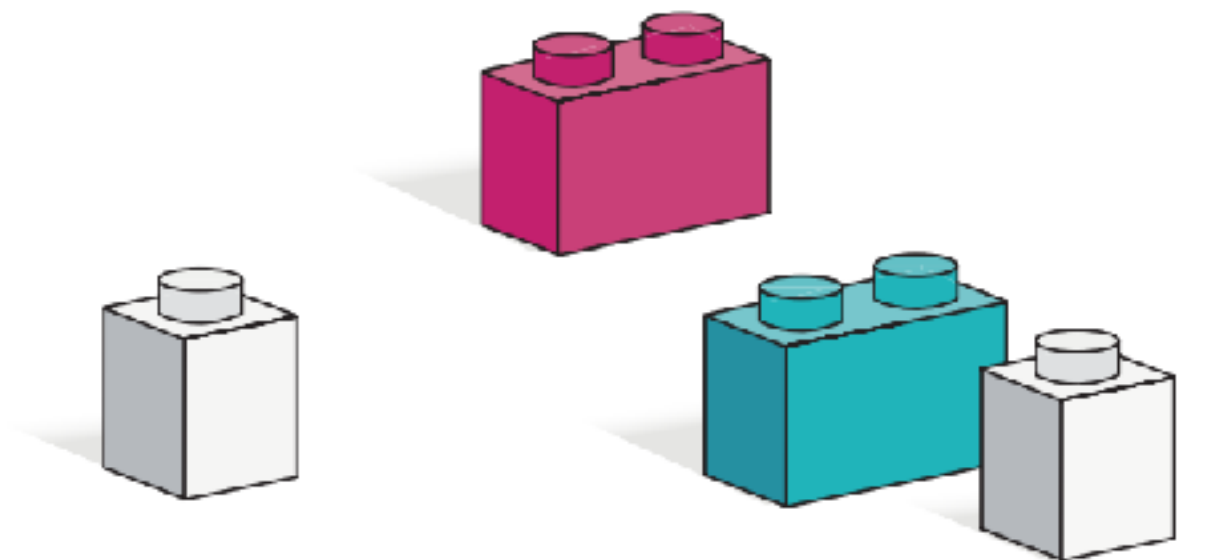


**Chrome**

**FireFox**

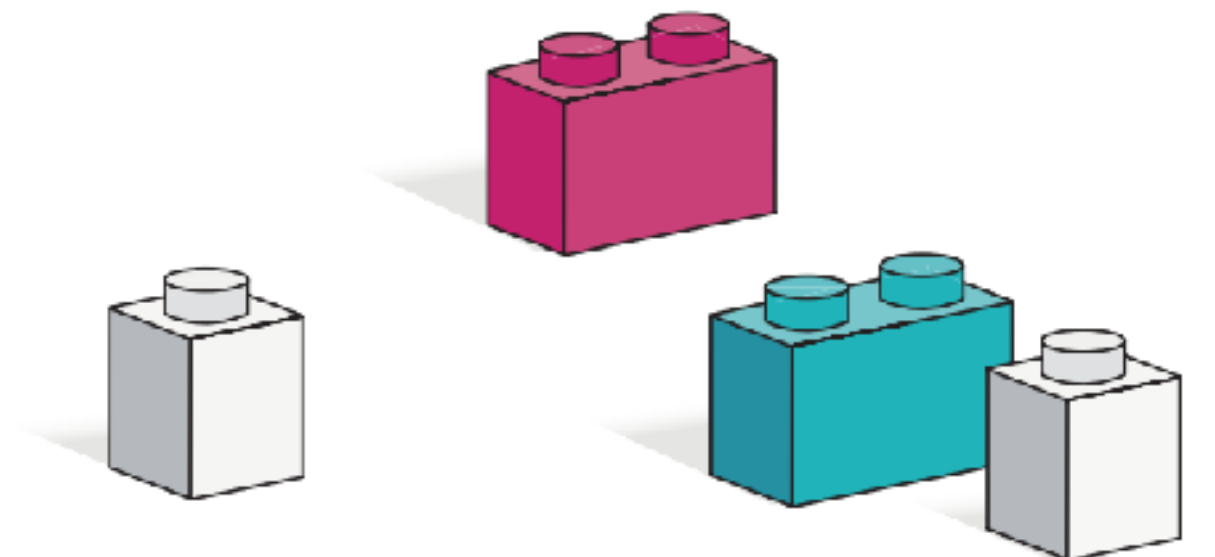
**Safari**

# WebAssembly Predictions



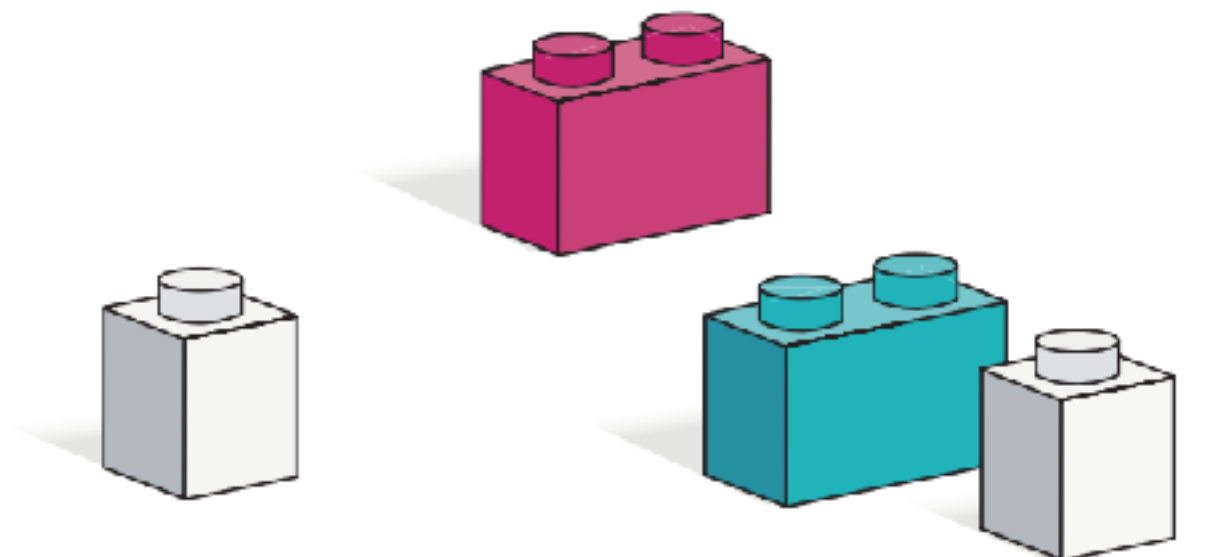
# 2018

- Rust, C, C++ used in production for performance critical, algorithmic tasks
- Java, C#, Typescript lots of creative experiments / POCs
- Threading lands in WebAssembly
- React for Rust



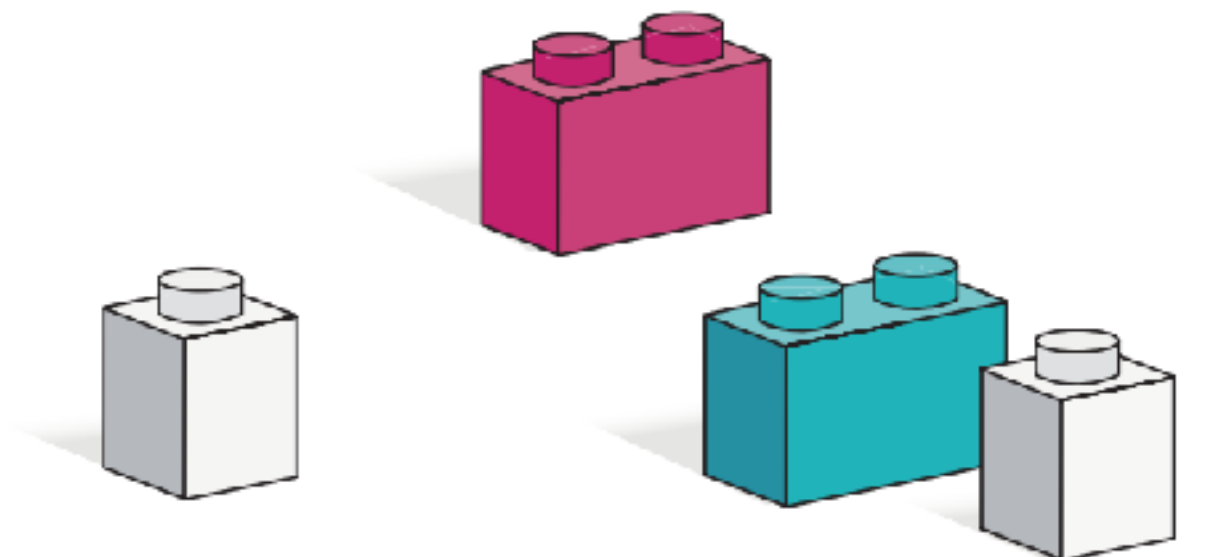
# 2019 - 2020

- Java & C# - considered production ready
- Heavyweight productivity apps move to the web
- Another wave of mobile, desktop and server-side UI frameworks will re-target the web - *write one, run everywhere*
- JavaScript's popularity starts to fall
- Native Android apps die out in favour of PWA & wasm

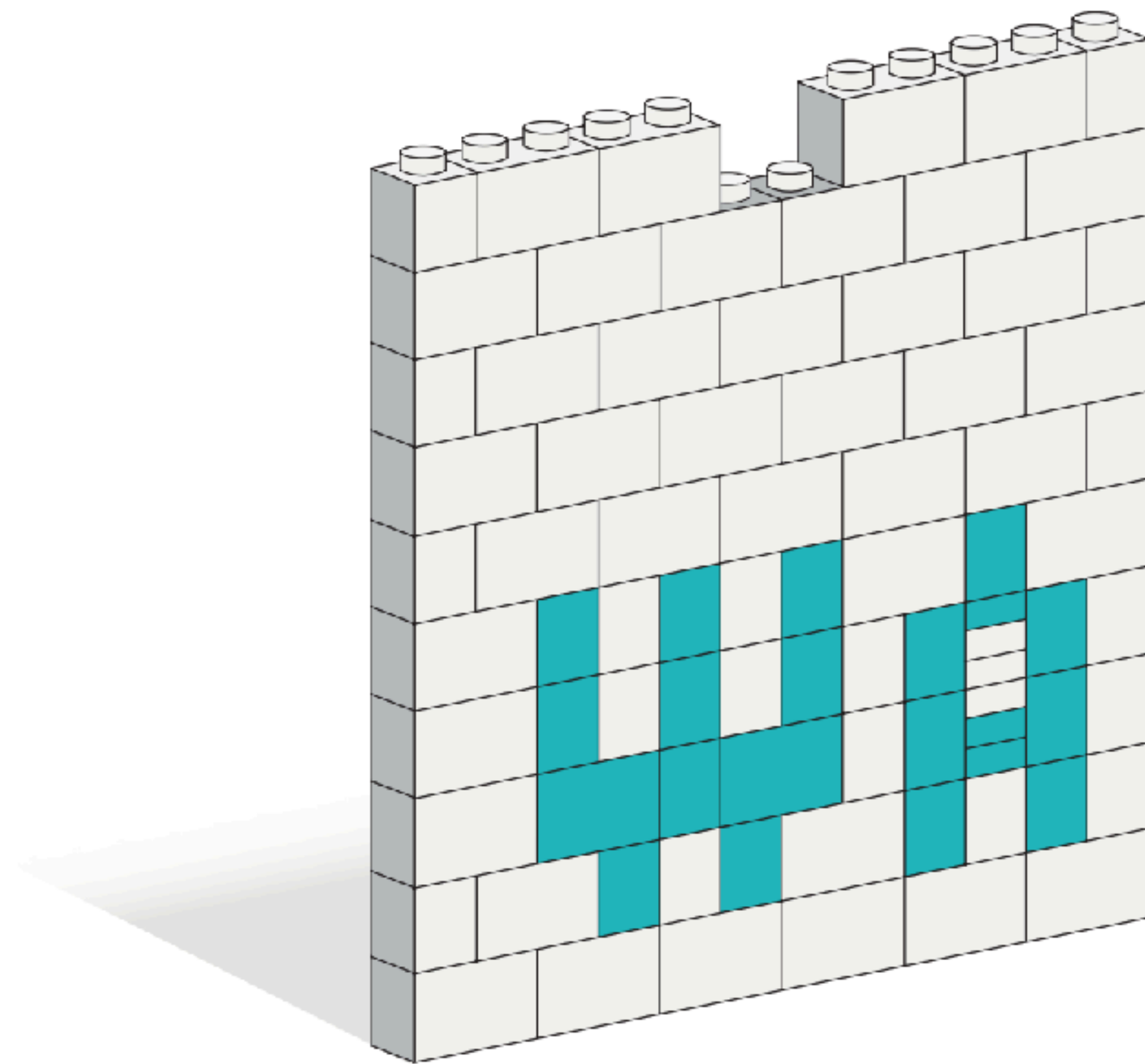


# 2021 - and beyond

- Windows Store drops support for non-web technologies
- MacOS drops support for non-web technology apps, resulting in a single unified runtime across desktop, web and mobile
- As WebAssembly has replaced JavaScript, a replacement for the DOM emerges







# WebAssembly

and the death of JavaScript?

@ColinEberhardt