API Modernization

Building Bridges As You

@codebytere

What IS Modernization?

The conversion, rewriting
or porting of a legacy system to a
modern **language**, software
library, **protocol**, or hardware
platform.

Software Modernization

What does it mean for something to be **modern**?

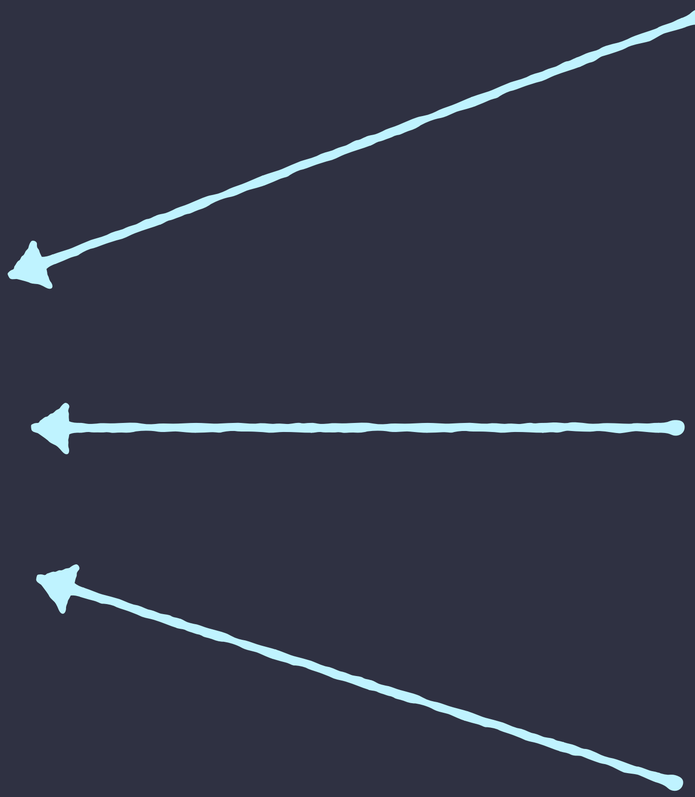
Does modernization take the **same form** across contexts?

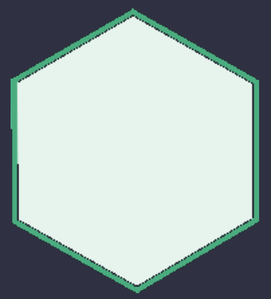
Can we all **understand it** the same way?

Modernization Goals

To **retain** and **extend** the value of a legacy investment, which is an investment that continues to provide core **services** to an **organization**.

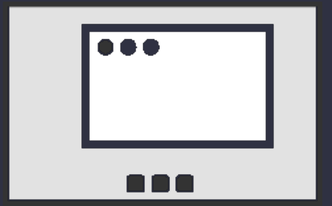
Electron





Node.js for filesystems and networks

Chromium for making web pages



Native APIs for three systems

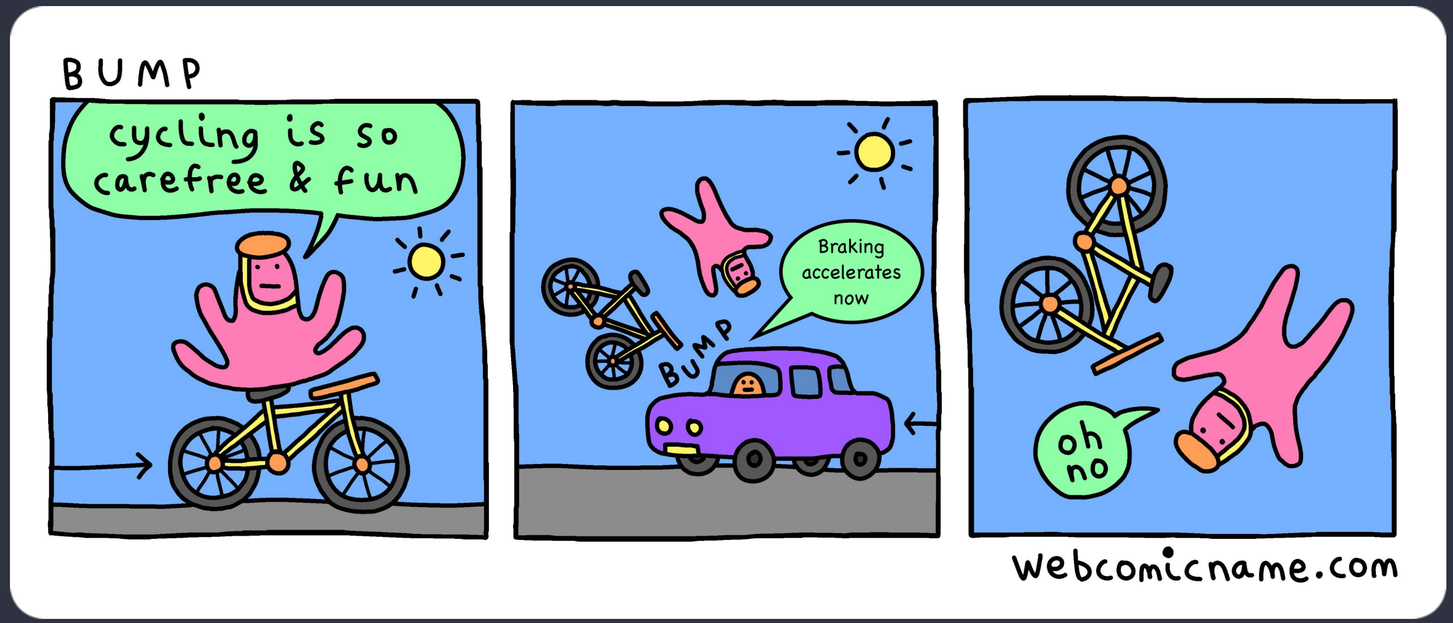
APIs & Versioning

APIs are **contracts** between maintainers and those who **consume** the API.

They rely on **expectations** you have **set out**.

How do you **set these out**?

API Contract Breakage



You don’t want this.

Your consumers don’t want this.

Semantic Versioning

- **MAJOR** bump when you make incompatible API changes
- **MINOR** bump when you add functionality in a backwards compatible manner
- **PATCH** bump when you make backwards compatible bug fixes

(Some) Types of Modernization

Platform Updates

Dependency Updates

Language Updates

ES5 → ES6

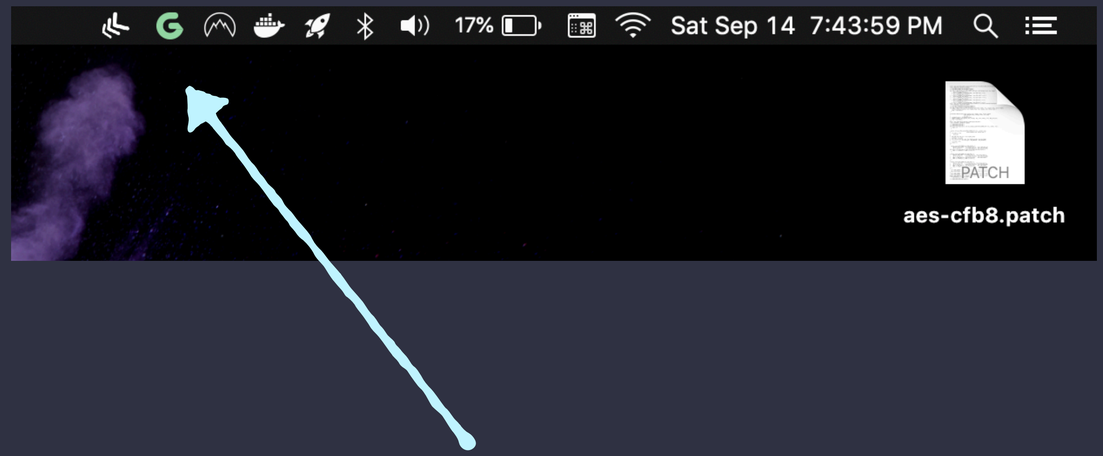
Platform Modernization

Operating system can
update APIs across
major releases.

Independent of
Electron Major
Versions.

Platforms don't
always adhere to a
clear **API contract.**

Platform Modernization



Instance Property

highlightMode

A Boolean that indicates whether the status item is highlighted when it is clicked.

Deprecated

Use the `button` property.

NSStatusItem

An individual element displayed in the system menu bar.

✅ Deprecated in macOS 10.10 - 10.14

❌ Removed in macOS 10.15

@codebytere

Platform Modernization

```
const { app, Tray } = require('electron')

let tray
app.on('ready', () => {
  tray = new Tray('/path/to/my/icon')

  // always highlight the tray icon
  // Electron manually manages highlighting
  tray.setHighlightMode('always')
})
```

Note issues in v6, v5 & v4 on macOS Catalina

```
const { app, Tray } = require('electron')

let tray
app.on('ready', () => {
  tray = new Tray('/path/to/my/icon')

  // Apple now manages highlighting logic
  // for status items, so we can't change it
})
```

Remove in Electron v7 without replacement

Platform Modernization

✓ Successes

- Informing users
- Minimizing end-user breaking changes

✗ Learning Opportunities

- Less reactionary platform deprecation research
- Refactoring earlier

Language Modernization

Allowing developers to access the **latest** and **greatest** language **features** as they arrive.

Async/Await

Promises

Nullish Coalescing

Optional catch binding

Numeric separators

Array.prototype.flatMap()

BigInt

Top-level await

Optional Chaining

WeakRefs

Promise.prototype.finally

@codebytere

Language Modernization

```
// open a new file or folder on your desktop
dialog.showOpenDialog(mainWindow, {
  properties: ['openFile', 'openDirectory']
}, (filePaths => {
  console.log(filePaths)
}))
```

Electron's **dialog module**

Callbacks

Promises

```
// open a new file or folder on your desktop
dialog.showOpenDialog(mainWindow, {
  properties: ['openFile', 'openDirectory']
}).then(result => {
  console.log(result.filePaths)
}).catch(err => {
  console.log(err)
})
```

Language Modernization

How did Electron carry out this **promisification** effort?

Natively!

(wait, what?!)

Language Modernization

Aren't **Promises** a **JavaScript** thing?

Language Modernization

Google's open source high-performance **JavaScript** and WebAssembly **engine**, written in **C++**

Implements **ECMAScript** and **WebAssembly**

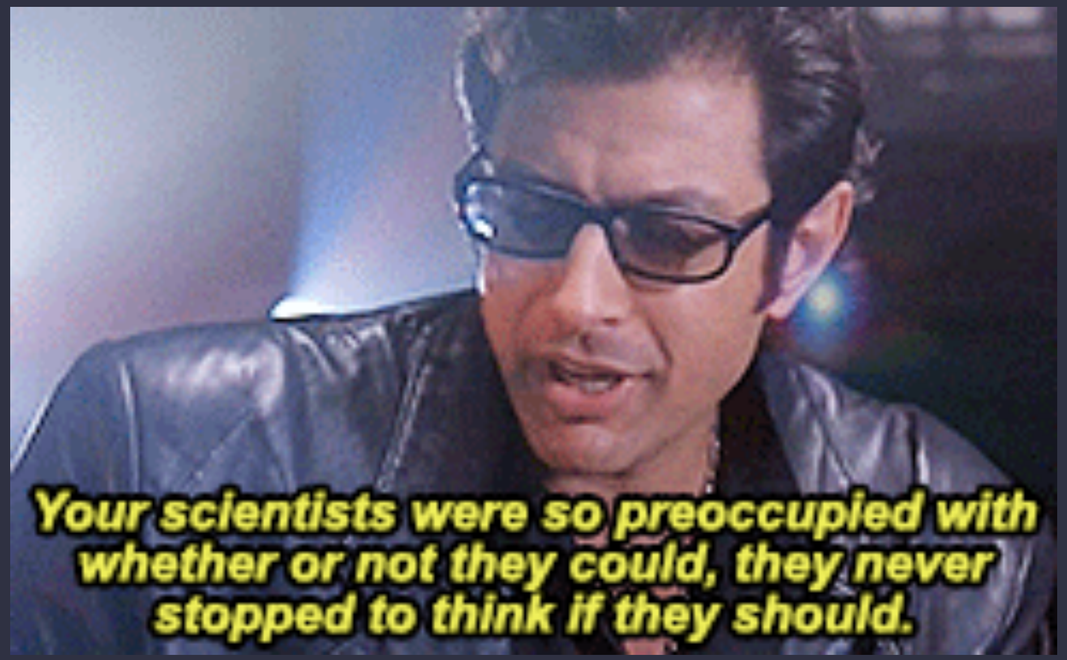
Language Modernization

```
template <typename RT>
static void RejectPromise(Promise<RT> promise, std::string errmsg) {
  if (!content::BrowserThread::CurrentlyOn(content::BrowserThread::UI)) {
    base::PostTaskWithTraits(FROM_HERE, {content::BrowserThread::UI},
                             base::BindOnce(
                                 [](Promise<RT> promise, std::string errmsg) {
                                   promise.RejectWithErrorMessage(errmsg);
                                 },
                                 std::move(promise), std::move(errmsg)));
  } else {
    promise.RejectWithErrorMessage(errmsg);
  }
}
```

Promises, now with threads!

@codebytere

Language Modernization



Was this a **totally smooth** process??

...No.

Unfortunately, software.

> So it turns out we've successfully introduced a way to write non-typesafe C++.

Language Modernization

✓ Successes

- Enabling access to Promises
- Smooth transition paths with backwards compatibility

✗ Learning Opportunities

- Ensure consumers understand changes as they map to versions
- Roadmapping farther into the future

Dependency Modernization

Enabling **access** to more modern capabilities **through dependencies** - not through code we write ourselves.

@codebytere

Dependency Modernization

Node.js

- Access to **performance** improvements
- Access to new **filesystem** & **networking** capabilities

Node.js Retweeted

sMyle @MylesBorins · Aug 23

Recursive rmdir just landed in @nodejs core!!!

thanks you @cjihrig @iansu @BenjaminCoe @b0neskull
and all the others who helped make this possible!

Node.js Retweeted

Colin Ihrig @cjihrig · Aug 20

node 12.9.0 has been released. this release updates V8 and libuv, adds `fs.writev()`, makes the @nodejs HTTP `OutgoingMessage` more stream-like, and much more.

Dependency Modernization

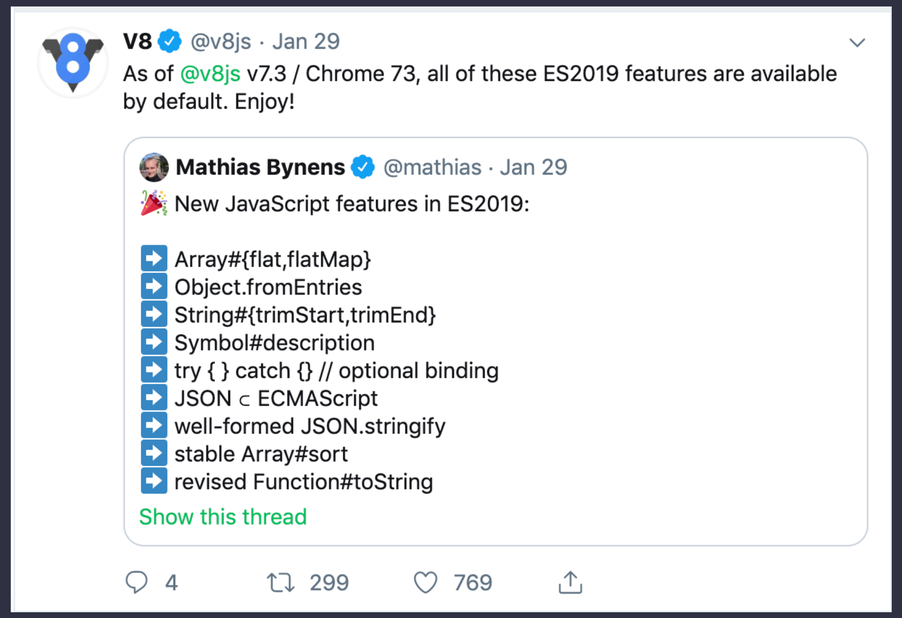
Chromium

- Access to **performance** upgrades
- Ability to reduce Electron's **code footprint**
- Access to new **capabilities**
  - Project Fugu 🐡

Dependency Modernization

V8

- Access to **performance improvements**
- Access to **new features** of the EcmaScript **Specification**



Dependency Modernization

✓ Successes

- Catch up to Chromium ToT
- Catch up to Node.js
- Shorten feedback loop for dependencies

✗ Learning Opportunities

- Make version bundling choices well in advance
  - Beta & Stable
- Identify version expectations further in advance

Communication & Your End-Users

Ensure **Redundancy**

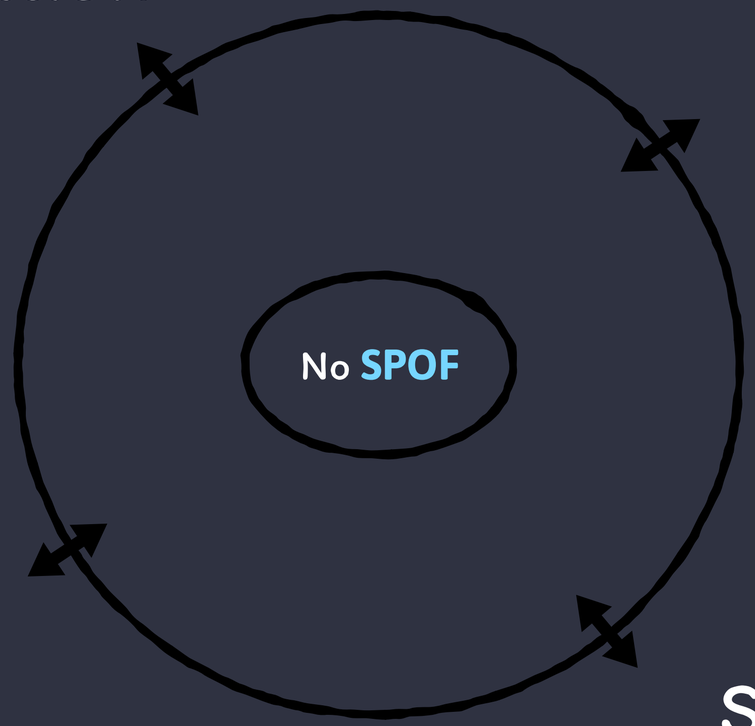
Communicate **Early**

Provide **Context**

Ensure Redundancy

API
Documentation

Deprecation
Warnings



Breaking Changes
Document

Social Media

Communicate Early

When does this change **take effect**?

On what **versions**?

Consideration for **team resources**?

Provide Context

Why are you making these changes?

How does this **benefit** them?

How might this **hurt** them?

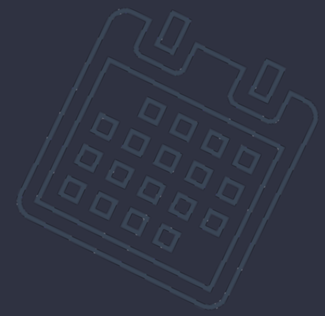
How can they **prepare**?

Minimizing Churn

You **do not** want to throw your users on an endless hamster wheel of complex updates.

Everyone suffers.

When to NOT modernize?



Sometimes, the **latest**
is not the same as the
greatest.

There is (almost) always
a **trade-off** when you
choose to alter or
modernize your software.

Wrapping Up

YOU have primary responsibility

Heuristics ≠ Rules

Modernization requires **WORK**

Come Find Me

I want to hear from you!

Future of Electron

C++/Obj-C vs JS

Contributing to OSS

Full-time OSS

Web vs. Desktop

@codebytere

THANK YOU!

@codebytere