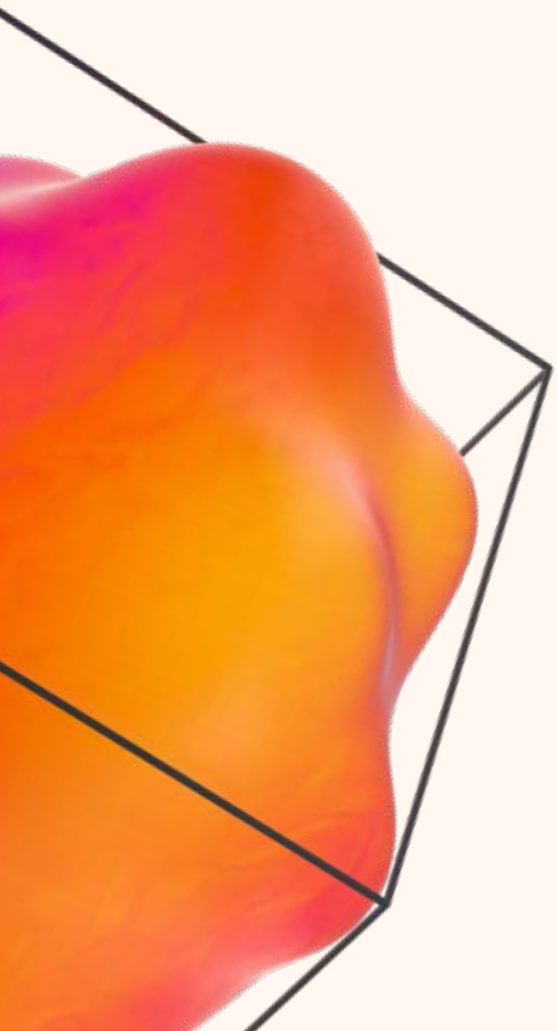
From Outsourced to In-House

Case study:
UK Department for Work and Pensions



Introduction

After almost 30 years of outsourcing technical work to cut costs, the Information and Communication Technology (ICT) department at the UK Department for Work and Pensions (DWP) decided to bring its code development back in-house. npm was a crucial tool in that transition.

Here's why: Outsourcing was a common business practice for decades, but the trend waned when cheap labor was replaced by even cheaper automation. However, making the shift from outsourced to in-house when ICT had very limited in-house staff was daunting. They needed to complete a series of major technical projects and make multiple new hires before they could automate much of anything.

The UK Department for Work and Pensions' investment outlays carry substantial consequences in both the short- and long-term. However, more was at stake than just the budget.

It was critical to find a solution that is scalable to 100K internal and 50M external users and beyond, as is typical of government projects. It had to be simple in terms of training, daily use, and troubleshooting resources. The solution also had to solve debilitating dependencies in the code while fostering collaboration within a team of 40–50 developers of varying skill levels.

Thus, it was no small matter that in 2015, the DWP chose npm to fast-track staffing, modernize languishing applications, and bring the department up to speed.

The cost of outsourcing

In the early 2000's, many organizations large and small, public and private, sought to cut costs by outsourcing. According to a Statista report, the size of the global outsourcing market in 2000 was $45.6 billion. The market peaked in 2014, at $104.6 billion, dropping to $88 billion in 2017. Processes that were not a direct function of an organization's core competencies were sent to workers in another company, or in some cases, another country.

The outsourcing model works as a concept—the benefits include lower labor costs and increased expertise when there are talent shortages. The problems come from its execution in the real world, wherein outside forces, elongated timetables, and unintended consequences can impact outcomes, as was the case with DWP.

"We used big incumbent suppliers who possessed excellent technical skills and project expertise," said Adam Moss FBCS CITP, Head of Technical Leadership at the UK Department for Work and Pensions. "We would give them requirements, they'd go away and after some time and in true waterfall fashion come back with something which theoretically met our requirements. But by then our business had changed because business doesn't stand still. Invariably what came back was no longer what was actually required."

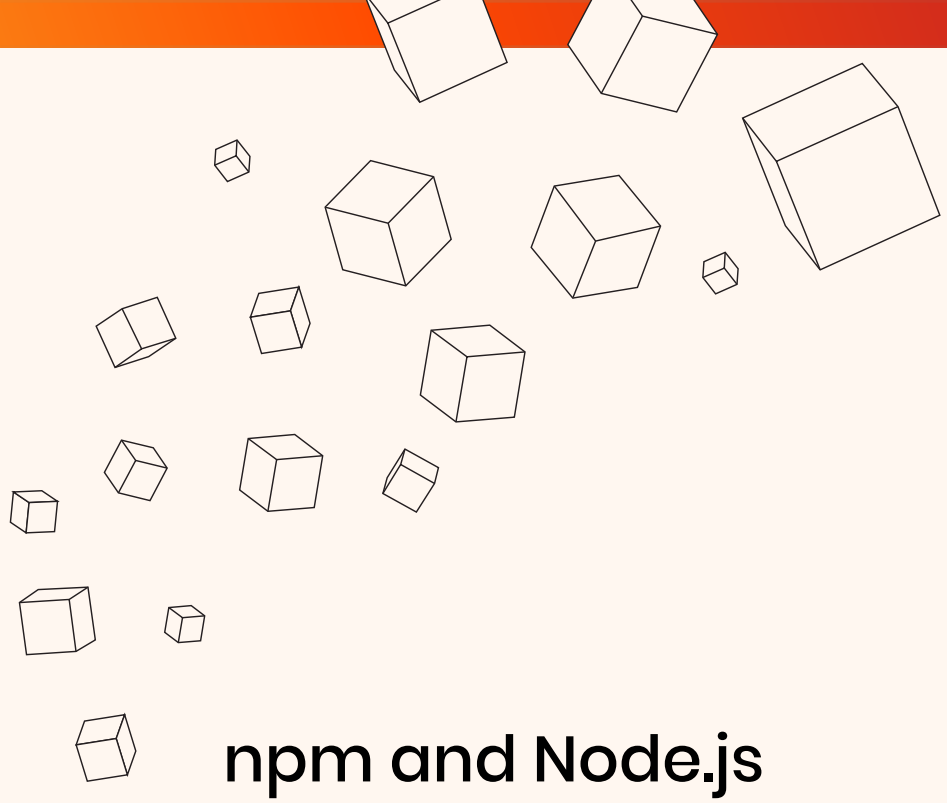
This situation was untenable given the speed with which innovation changes everything. In short, after outsourcing, ICT found itself behind where it was three decades ago and yet hurrying to get to new ground.

Rebuilding the ICT department

“There was a realization that things had to be done differently across the entire government sector within the UK,” said Moss. “We needed to bring the projects back in-house, get the right people on staff, and execute this job properly. We had to adopt more agile ways of working than waterfall methods and be able to start delivering value immediately. Which, with an outsourced supplier, is considerably more challenging because contracts get in the way.”

Moss and his team began the move to in-house by assessing the technology market. He championed the adoption of JavaScript early on in this process. One of the first projects he led was a trial for Node.js to become a formally adopted technology within the organization. While that project was successful, legacy systems still held sway.

“In terms of overall stats, I think we’ve now got more JavaScript code than we do Java code, but nowhere near as much as we do COBOL code,” Moss lamented. “I know that sounds bizarre, but from a government perspective we are cautious about what does and doesn’t get used.”



npm and Node.js to the rescue

Moss' reasons for adopting JavaScript and an npm-centric workflow were:

1. It has a low barrier to entry.
2. It facilitates collaboration.
3. It offers massive scalability.

The low barrier to entry was vital, considering much of the department's talent pool had left long ago. While many had contracted back with the DWP, veteran developers were not necessarily available for rehire to staff positions and challenges existed in re-training an internal team of Microsoft VBA developers.

"You can't just send them off to college for three months or six months and still operate the department. So, barrier to entry was one of the main reasons for adoption because JavaScript is easy, and Node is relatively simple. Then you can improve and iterate from there."

At first, npm was used to solve specific problems.

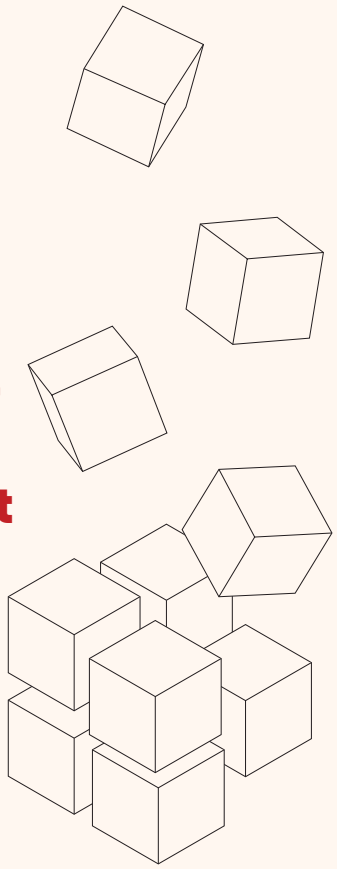
"It started with something as simple as, 'Which task runner should we use? Should it be Gulp? Should it be Grunt? Should it be this? Should it be that?'" Moss explained. "And the answer was always, 'No, just use npm scripts, it's far easier and you've got no dependencies to worry about.'"

Collaboration wins

The team of 40–50 developers faced constant challenges with Node.js, trying to figure out how to do various functions or resolve issues without reinventing the wheel every time. Fortunately, npm's open source Registry has 775,000 pre-written solutions to answer any "how-do-I" question that exists.

To supplement existing staff that was undergoing fast track cross-training with JavaScript, Moss recruited an entire JavaScript development team from a local university.

"Obviously that was a good recruitment move because then the former university developers could pair up with our internal people and share their experiences," said Moss.



"We now had a team that was well-versed in operating in a Scrum method. They were able to support and guide the existing staff who were very unfamiliar with those processes."

Access to npm packages quickly enabled developers to process workloads on the fly, converting them into a highly functioning team.

Scalability pays off

JavaScript is not unique in how it handles asynchronous work, but using npm and Node.js made async the default position at the DWP, which was very helpful with scalability.

“When you’re rolling out a project that can be for up to 98,000 internal users or 40 to 50 million citizens, scalability is a very important consideration,” says Moss.

The DWP offers several public services that any citizen can use anytime from any device as many times as they wish from the age of 16 on.

One of the public services, Check your State Pension, allows citizens to forecast what their state pension would be if they continued working to retirement age. They can also discover steps to take if they want to increase their pension in situations such as absence from the workforce due to injury, illness, or parenthood.

“After citizens answer a series of questions, the system accesses their national insurance contributions, which is what dictates an individual’s pension payments, and calculate their benefits based on the responses given, and the legislation that’s in force,” says Moss.

Performing on-demand calculations for every citizen on-demand from the age of 16 onward requires unimaginable scale. An npm-powered JavaScript stack has delivered that scalability flawlessly.

Fun facts: npm packages Moss' team uses

As a government employee, Moss can't officiallyrecommend products for commercial use. However, he is big on sharing what he and his team have learned and on learning from the experiences of others.

The specific **npm packages** that Moss' team uses regularly are:

- AngularJS
- VueJS
- ReactJS
- Express
- Helmet
- Seneca
- Moment
- aXe
- commitlint
- eslint
- webpack
- nyc (and istanbul)
- karma
- jasmine
- mocha
- chai
- sinon
- sonarjs
- selenium-webdriver
- nsp
- husky
- lint-staged
- nunjucks
- faker

Project contributions

The use of npm packages doesn't end there. Moss and his team have contributed to projects including Renovate, commitlint, and Danger JS, and are currently experimenting with others including oclif and jaeger.

Moss says the DWP ICT team is looking to contribute more to open source in general, and npm in particular, to help projects and packages address more issues specific to government agencies.

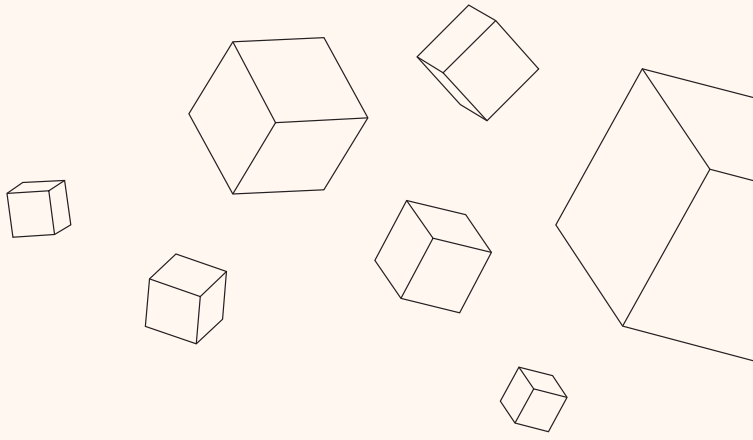
"I would really like to see us start open sourcing more of the npm packages we've been developing for various use cases. I think that's something we can more actively push and encourage with all of our project teams," said Moss.

Open sourcing is part of ICT's official job requirements as one of the criteria of the UK's Digital Service Standard. Point 8 of the Standard states, "make all new source code open," and the vast majority of government services are held accountable to it. Open sourcing code promotes reuse to eliminate wasting time and resources on starting every project from scratch.

As important as those goals are, they are truly the least of Moss' motivations behind the push to open source more code.

In particular, he hopes open sourcing DWP's code will address the lack of understanding of government environments and needs, opening the door for independent feedback.

"Open sourcing your work is also an indirect recruiting thing," says Moss. "It's a 'come see the shiny work we're doing' hook to lure talent to us. Public portfolios of work are a powerful trend in attracting talented engineers."



npm, Inc.
www.npmjs.com
1999 Harrison Street, Suite 1150
Oakland, California 94612 USA