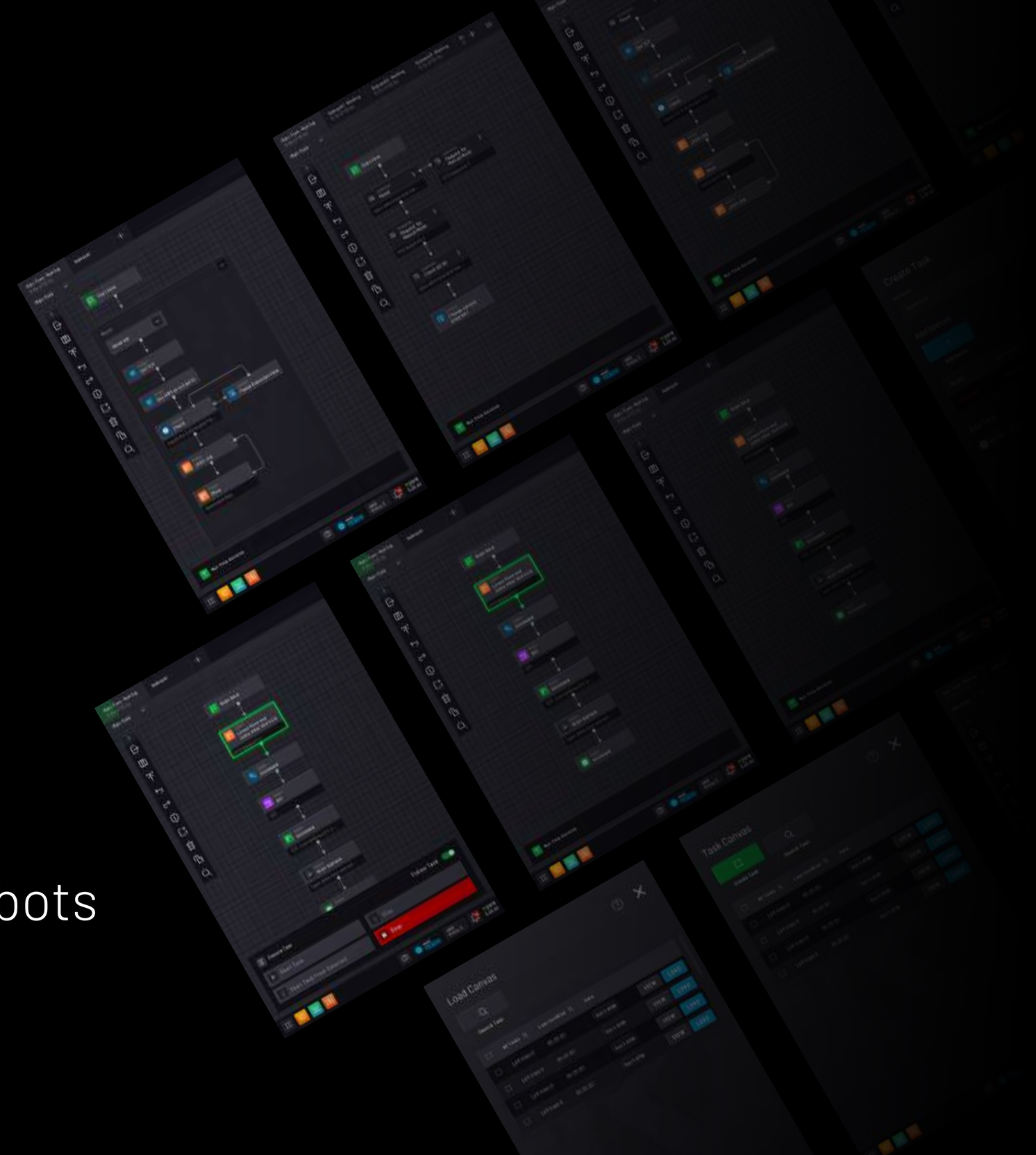
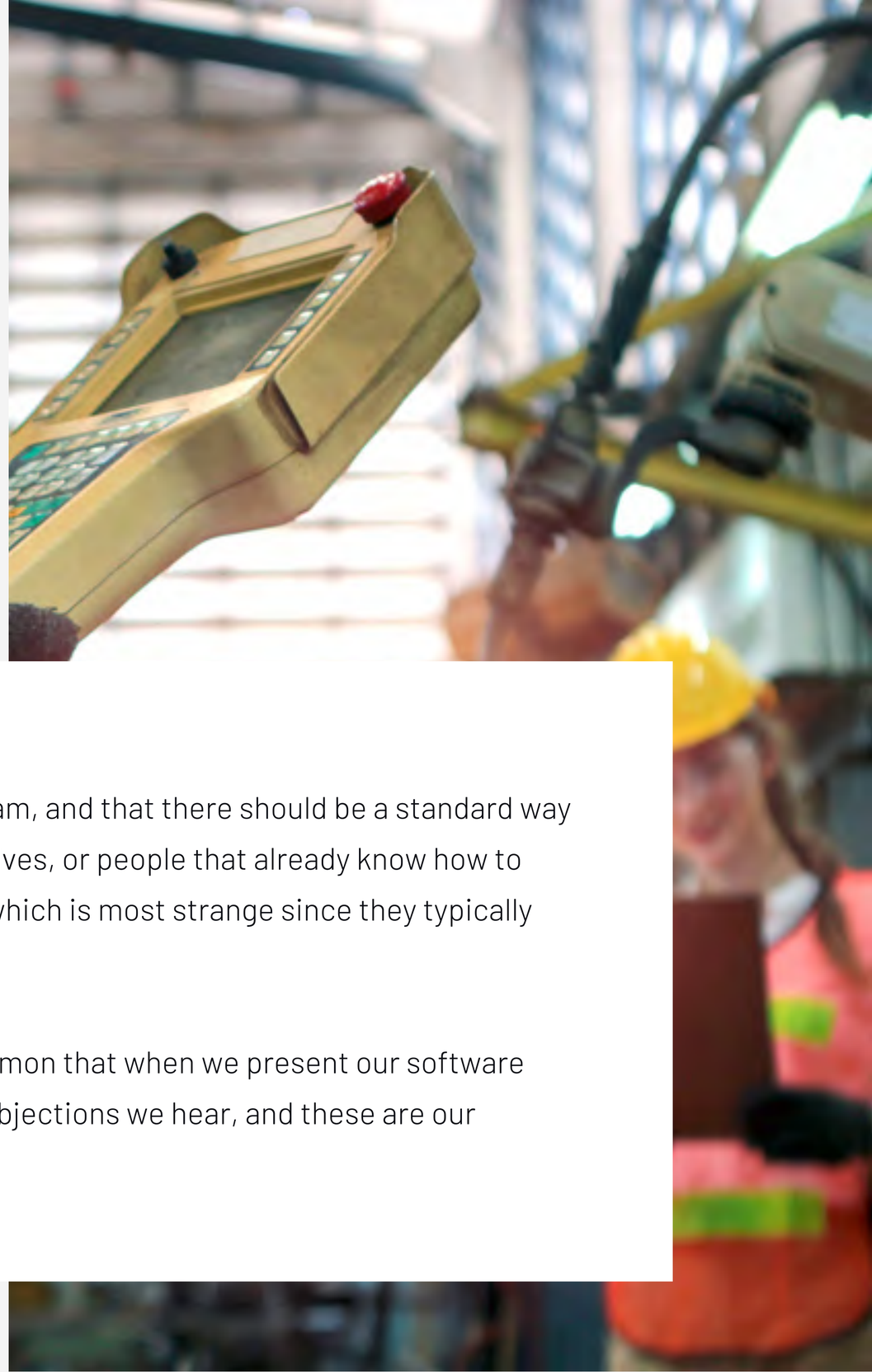
READY ROBOTICS

Objections to Making Robots **Easier to Program**

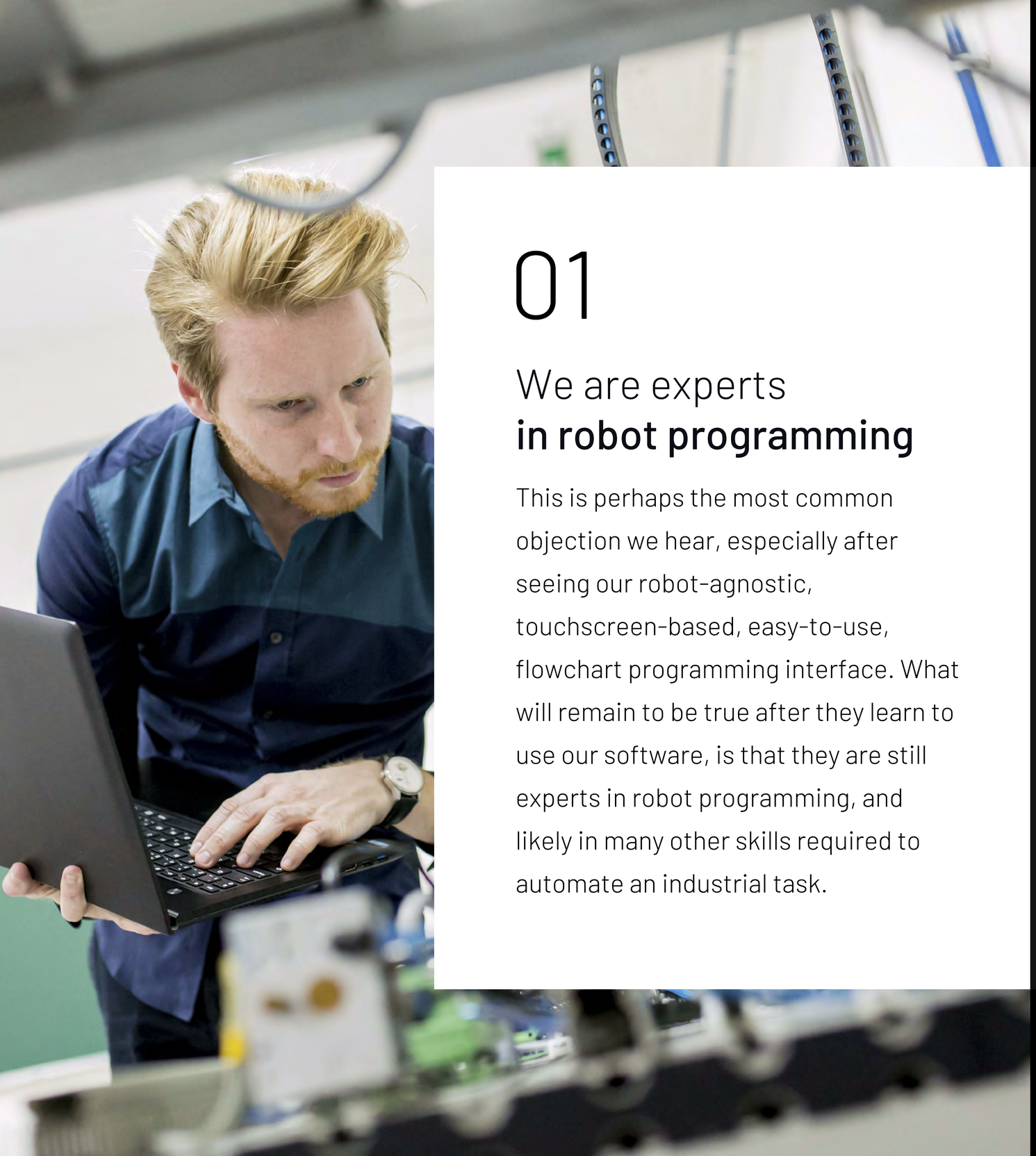


Industrial and collaborative robots are too hard to program, and even if you learn how to program one, your skills are not transferable to other robots. Our software, Forge/OS, makes robots much easier to program, and programming skills are transferable since our software is not tied to a single brand of robot.



It's pretty simple to make a case for why robots should be easier to program, and that there should be a standard way of doing it. The main counterargument comes from robot brands themselves, or people that already know how to program a robot. Even some educational institutions object to the idea, which is most strange since they typically only have one brand of robot in their labs.

For the group of people that know how to program a robot, it’s not uncommon that when we present our software they search for reasons why our approach is wrong. There are six major objections we hear, and these are our responses.



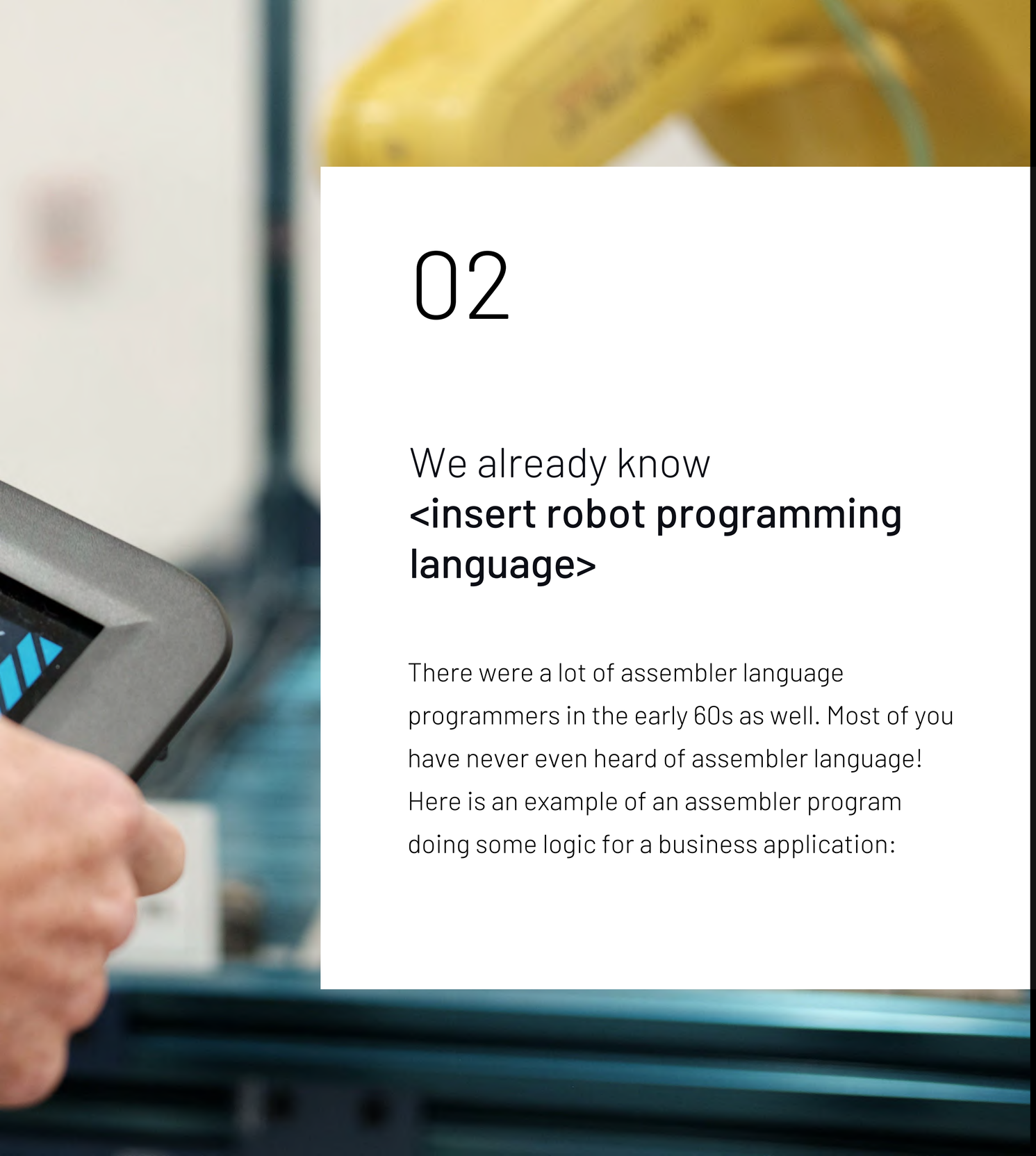
01

We are experts **in robot programming**

This is perhaps the most common objection we hear, especially after seeing our robot-agnostic, touchscreen-based, easy-to-use, flowchart programming interface. What will remain to be true after they learn to use our software, is that they are still experts in robot programming, and likely in many other skills required to automate an industrial task.

Especially with integrators, they are implementing robots that their customers request. If a new robot is requested, they'll have to invest in the training to learn how to work with that interface, or risk losing the project.

The logic also breaks down since technology is not static. There is not a single example of a technology that has not evolved in computing. If the statement is that we are experts in programming X, and we are not willing to program in Y, then they will become irrelevant if a robot vendor changes their interface.



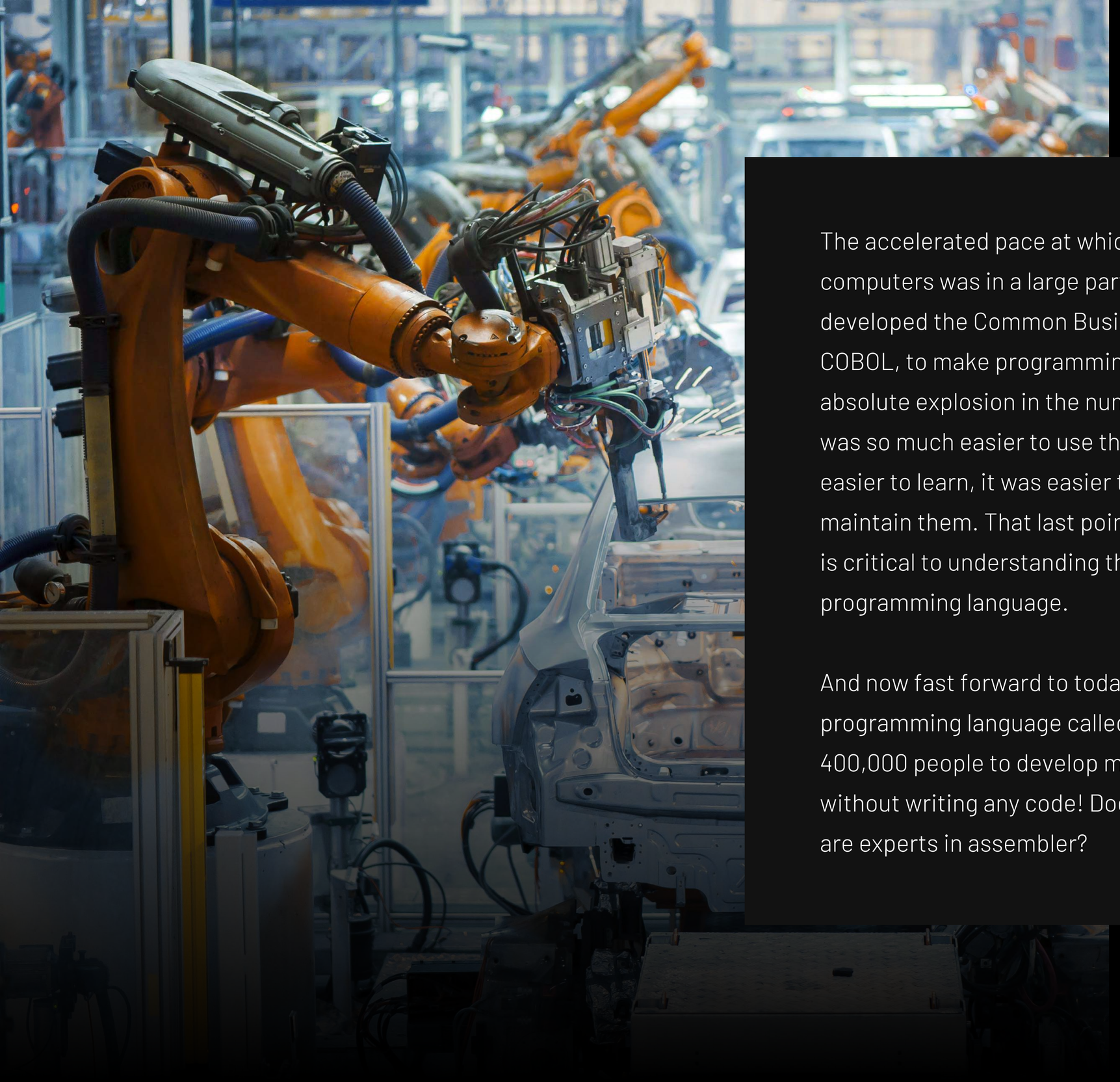
02

We already know
<insert robot programming language>

There were a lot of assembler language programmers in the early 60s as well. Most of you have never even heard of assembler language! Here is an example of an assembler program doing some logic for a business application:

```
MOVX 23,I
ADD  223,I
SD   sd,j
```

That doesn’t look easy, does it?However, there were a good number of experts in assembler language in the 60s and 70s writing programs that enabled businesses to use the first computers to automate their businesses.



The accelerated pace at which businesses could adopt computers was in a large part thanks to **Grace Hopper**, who developed the Common Business Oriented Language, or COBOL, to make programming easier. COBOL resulted in an absolute explosion in the number of programmers, since it was so much easier to use than assembler. Not only was it easier to learn, it was easier to develop programs and maintain them. That last point about maintaining programs is critical to understanding the value of a simpler programming language.

And now fast forward to today, there is a no-code programming language called **Bubble** that is enabling over 400,000 people to develop mobile and web applications, without writing any code! Does it matter that none of them are experts in assembler?

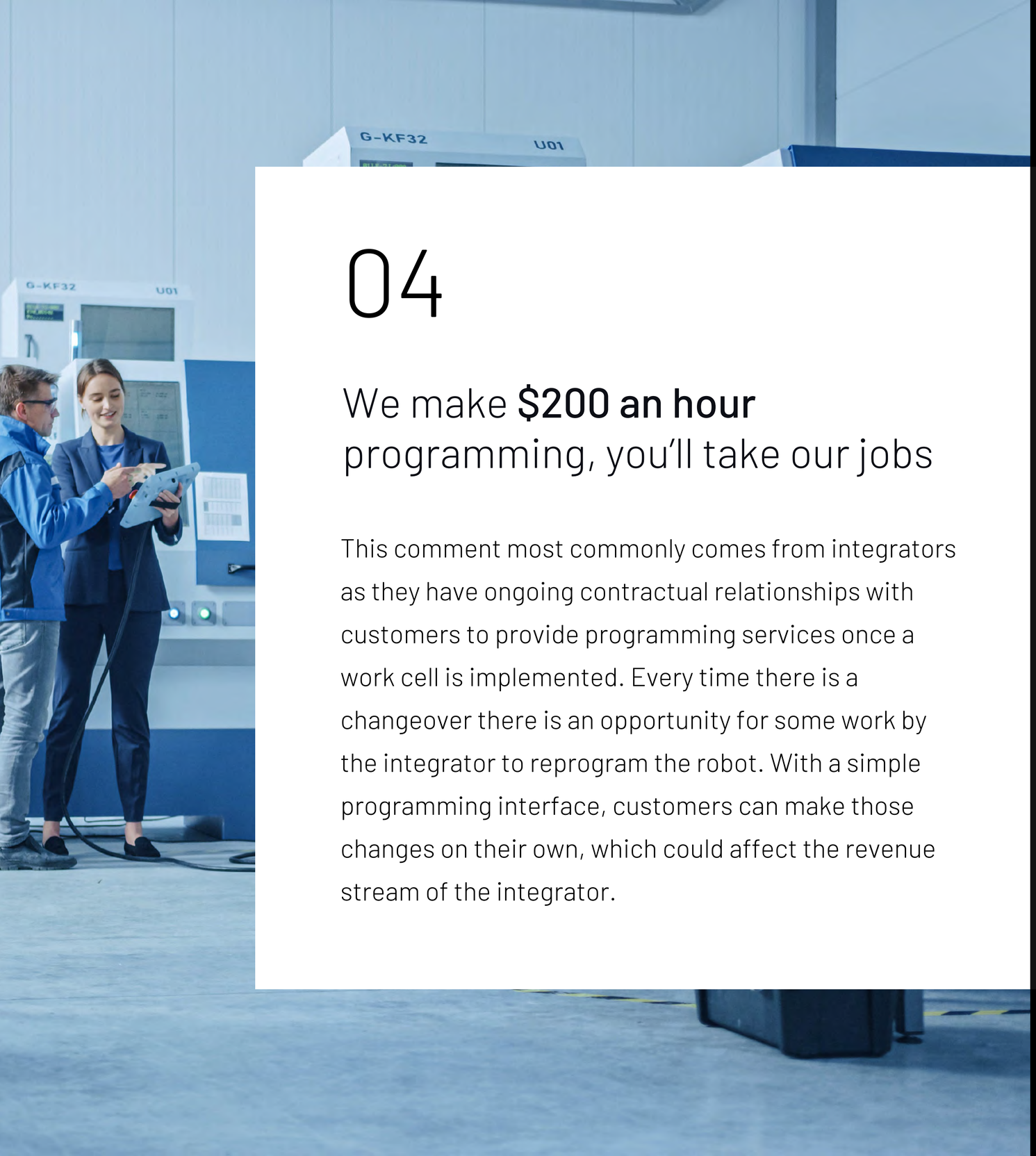
3

You do not support **<insert feature here>**

We get a lot of questions about our current release:

- **Can you do welding?**
- **Can you pick from a conveyor?**
- **Can you calculate the optimal path for picking from a bin?**

As of today, we can either do some of these tasks, integrate with the robot's built-in feature to do the task, or integrate with software that performs these functions. The important point is not whether we can do it today, but do we have the capability to add these features? And, along the way, make them easier and more functional than how they are done today. In addition, we are providing an open architecture where we are not the ones having to implement every single feature, but enabling a large ecosystem of providers who are experts in complex tasks such as bin picking and path optimization. These are areas of not only active research, but fierce competition in the space, and our goal is just to make it easier for developers to iterate and improve on their features and make them accessible to the widest set of end customers as possible.

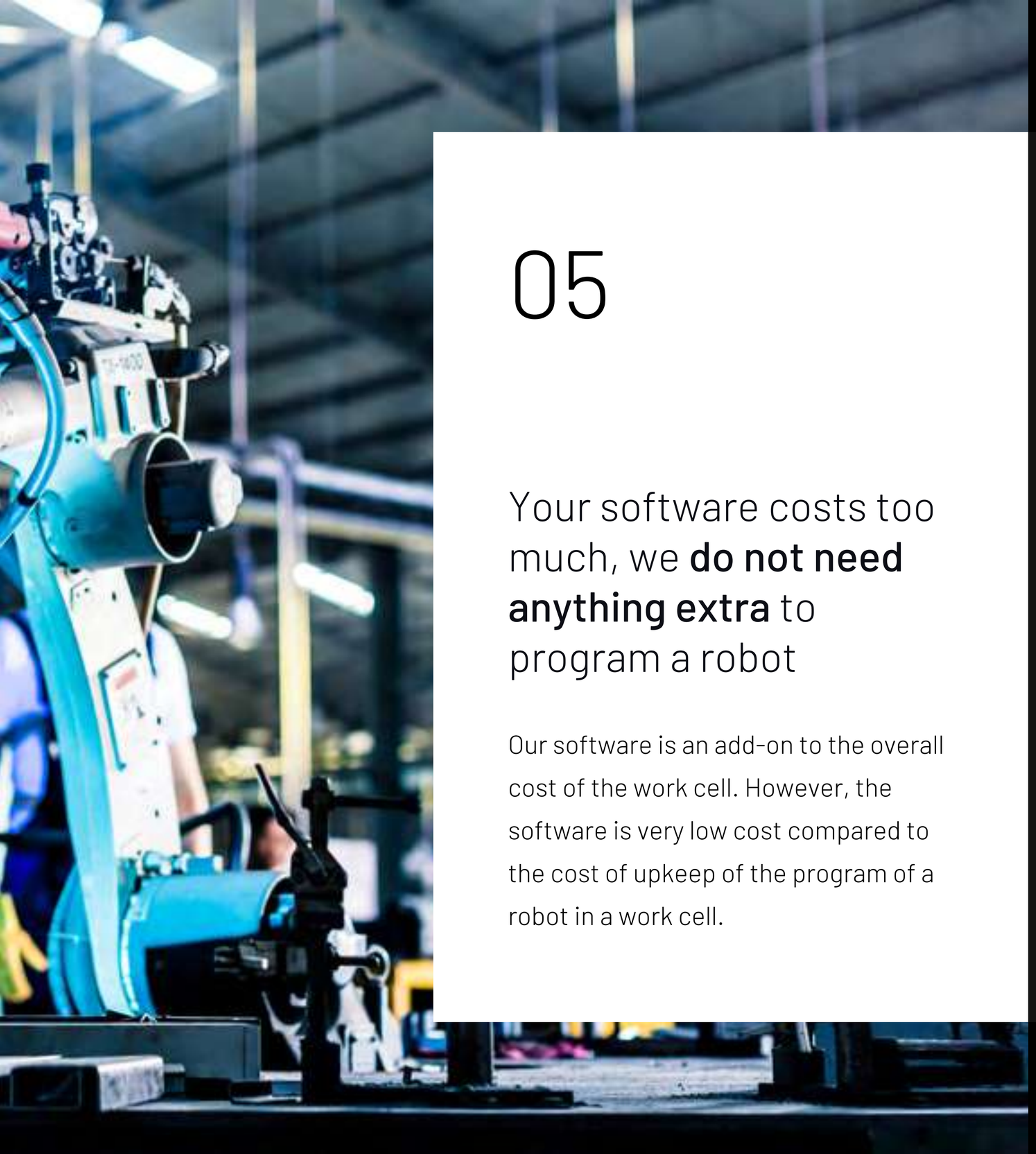


04

We make **$200 an hour** programming, you'll take our jobs

This comment most commonly comes from integrators as they have ongoing contractual relationships with customers to provide programming services once a work cell is implemented. Every time there is a changeover there is an opportunity for some work by the integrator to reprogram the robot. With a simple programming interface, customers can make those changes on their own, which could affect the revenue stream of the integrator.

However, this is not what has happened in the software programming world. Programmers are in demand more than ever because the number of applications being developed has exploded. In robots, the dramatic change in the landscape will come from cheaper robots that have better integrated hardware options such as those needed for safety. This means there will be much more demand for programmers since there will be so many more robots. Even if robots are easier to program, there will be more opportunities for integrators to provide value add services to their clients.



05

Your software costs too much, we **do not need anything extra** to program a robot

Our software is an add-on to the overall cost of the work cell. However, the software is very low cost compared to the cost of upkeep of the program of a robot in a work cell.

It's a dirty little secret in the industry that robots require constant touch ups to their programs to account for robot drift, changes in their environment caused by accidents (e.g. a forklift bumping a fixture) or an accident on a line (a part colliding with the robot). When amortized over even just a few years, easier-to-use software has a very large ROI based on its impact reducing ongoing maintenance costs.

But it doesn't just stop with maintenance. Imagine the possibilities if you can swap in a different robot, one that is newer and cheaper, and be able to make only minor updates to the program.

6

We've already standardized on **<insert robot OEM brand here>**

The argument goes: we have already chosen a particular robot brand, and we've got too much invested to use something else. In addition, we are told that they get incentives to continue to stay with that brand. This is contrary to nearly any service or product sold today. Isn't it more logical that a competing brand would give a discount to win your business? Doesn't blind loyalty lead to higher prices and less competition over time? It's fair to say that industrial robots have seen very little innovation in the past 30 years. The biggest advance has been with collaborative robotics (also known as "cobots"). Initially led by Rethink Robotics, Universal Robots picked up where they left off after bankruptcy, to become the undisputed market leader for cobots. However, Universal Robots is facing intense competition with dozens of startups and established robot OEMs now manufacturing collaborative robot arms.

These new entrants are only exacerbating the fragmentation facing the industry, with every new cobot OEM bringing their own proprietary programming interface. This makes the need for a standardized, easy-to-use programming interface all the more critical. Standardization drives the next wave of innovation. Google's Android platform did so for the nascent smartphone industry, triggering a massive wave of innovation in mobile apps. Microsoft's Windows operating system did so for the budding PC industry, triggering a massive wave of innovation in computer software. Robotics is overdue for a common operating platform that reduces fragmentation, reduces the programming barrier, and unlocks the innovation that manufacturing needs.

Conclusion

At READY Robotics, our software, **Forge/OS**, makes industrial and collaborative robots vastly easier to program.

In addition, Forge/OS provides a standard interface to robots, with an open platform architecture, where partners can build plugins that work with any robot.

Contact

Ben Gibbs, CEO and Co-Founder
ben@ready-robotics.com

Author

Ben Gibbs, CEO and Co-Founder
ben@ready-robotics.com

READY ROBOTICS

1080 STEELWOOD RD. COLUMBUS, OH 43212
(833) 732-3967

Thought Leadership Team

Kel Guerin
CIO and Co-Founder

Josh Davis
PhD in Robotics, VP of Robotics

Jake Huckaby
PhD in Robotics, VP of Strategic Partnerships

Production Team

Erik Bjornard
VP of Marketing

Kirk Higgins
Kirk Higgins

For more resources and thought leading content and case studies on automation:

ready-robotics.com/resources

For press inquiries, contact
erik.bjornard@ready-robotics.com

Linkedn • Facebook • Twitter
Instagram • YouTube