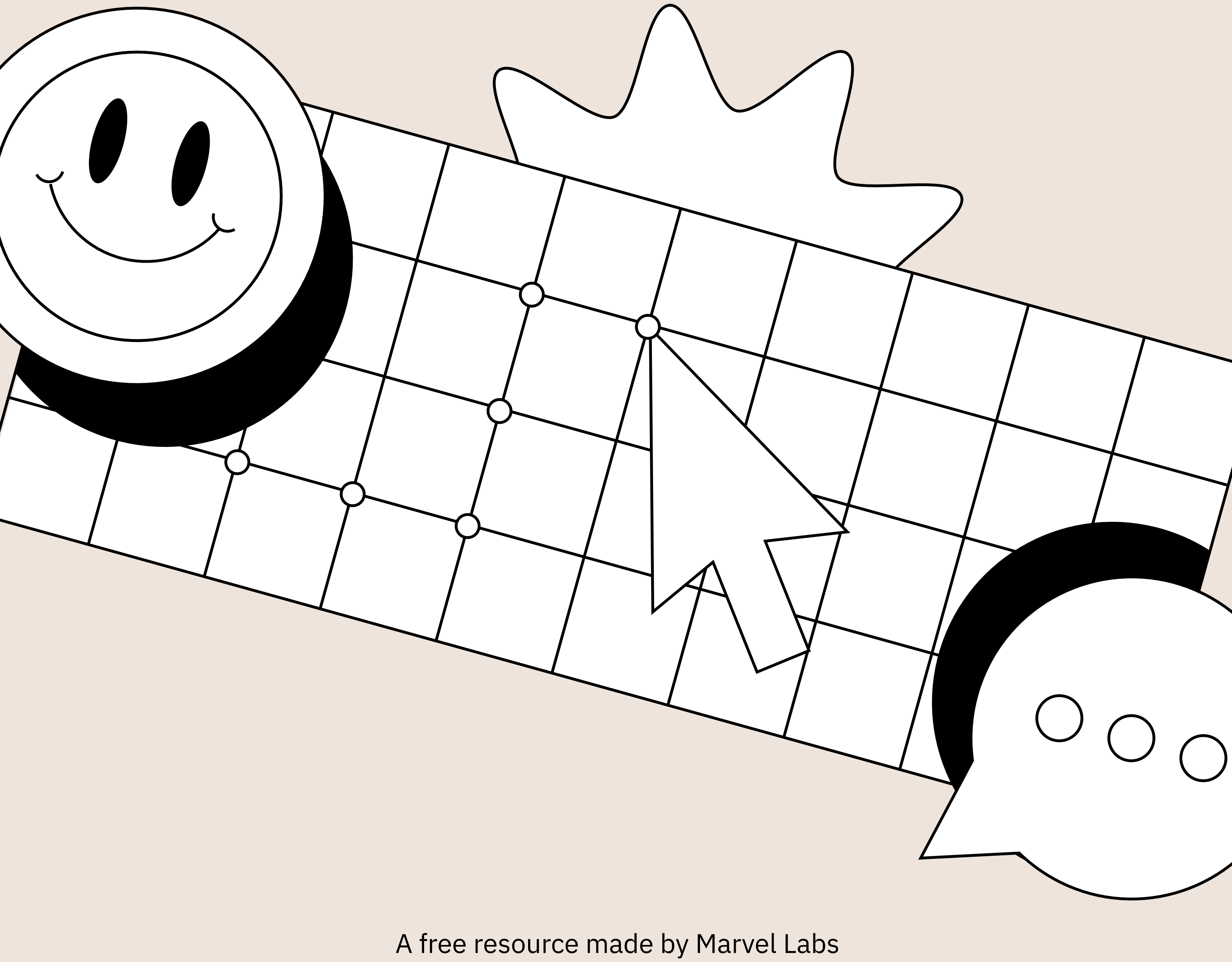
Marvel

THE BOOK OF COLLABORATION: HANDOFF EDITION



A free resource made by Marvel Labs

THE BOOK OF COLLABORATION: HANDOFF EDITION

The power of design in everyone’s hands.

Marvel provides a cloud based platform that automates the design process, allowing companies and individuals to effortlessly create user-centric digital products and experiences.

marvelapp.com

We’ve also created a space for our community to share all things design and product, from opinions to processes.

blog.marvelapp.com

Design: Filippo Chiumiento
Content: Naomi Francis and Jarrod Drysdale

For more like this, visit marvelapp.com/resources

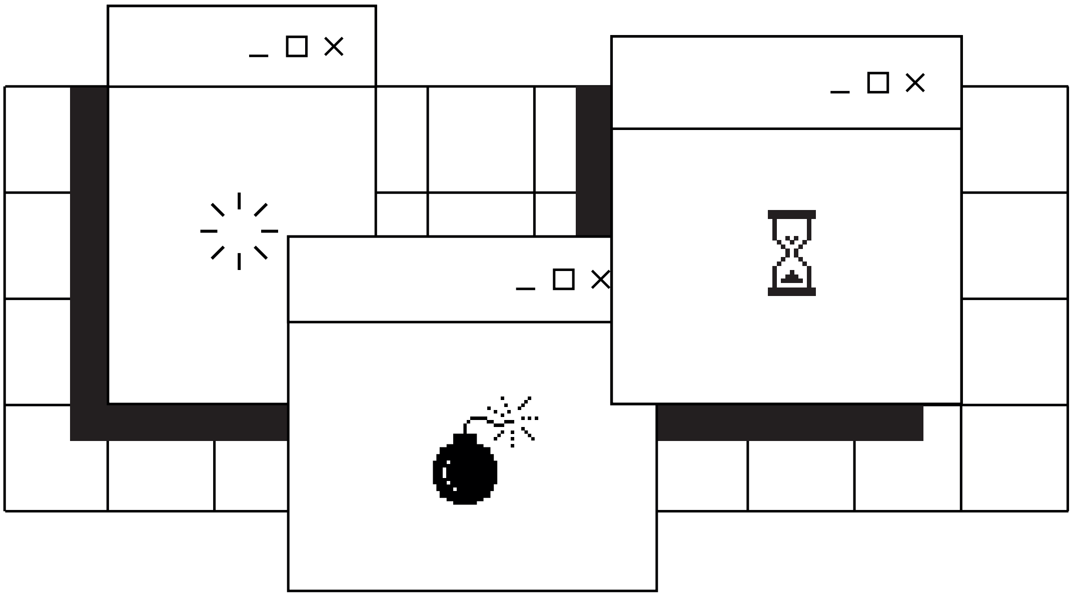
THE BOOK OF COLLABORATION

TABLE OF CONTENTS

01

01

MANAGING DESIGN HANDOVER



Many designers and developers hesitate at the idea of collaboration because it sounds like accepting compromises.

Nowhere is this dilemma more apparent than when our workflows overlap and the design is ready for developers to begin building.
When designers deliver a complete design mockup to developers for implementation, both sides tend to view it as a competition; both sides are pushing for their own concerns and trying to avoid unnecessary compromises and wasteful rework.
If the designer gets their way, the developer must make a sacrifice, and vice-versa. The designer wants to

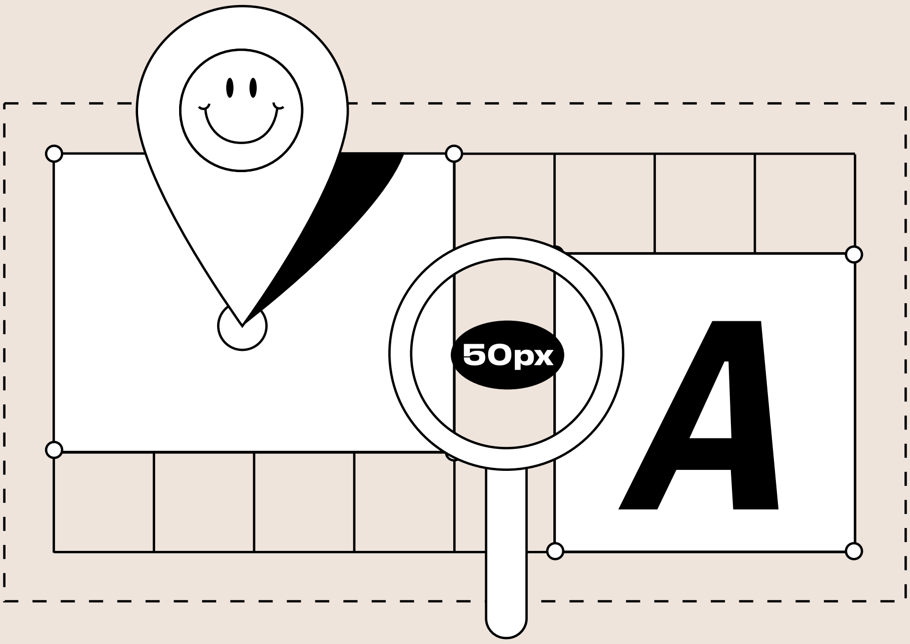
avoid major changes and is invested in seeing the design vision achieved accurately because it's better for the business and users.

This is why teams should change how they approach the transition from design to development. The old way is damaging and sets up unhealthy competition when everyone should be collaborating and supporting one another's goals. It's destructive to teamwork and removes important opportunities for both designers and developers to do their best work.

The other downside is that when designers hand over source files so developers can take over the project, this implies that design work is done, which couldn't be further from the truth. Even the best design mockup can't foresee every technical challenge or hurdle during implementation. A mockup is a plan, and even the best plans have to adapt.

Your team can remove common frustrations and conflicts by approaching this phase of each project differently.

> *Designers and developers competing during this phase is a shame because both perspectives are important. Design and development concerns shouldn't compete—they should support one another. Both teams are trying to create a great product, and all concerns should be welcome if they support that goal.*



IMPLEMENTATION IS LIKE TRANSLATION

For designers, this starts with changing expectations about what should happen when a design is built.

Expect a design to change during development. For many designers, that’s a hard pill to swallow because they think it’s their responsibility to protect and enforce the design vision. After all, if the designer doesn’t advocate for the design, changes and compromises could interfere with its effectiveness. Of course, no one wants to intentionally change the design so that it’s less effective, but it is the designer’s job to make sure that doesn’t happen. Because of this designers often see any change as negative.

Trying to view implementation as an opportunity instead of a conflict can help designers find worthwhile improvements they hadn’t considered.

However, if a design changes during development, that’s not always a bad thing. Some of these changes are better for users. When designers are in the default mode of defending and protecting their work, they might not realize this.

Development is not just the process of building a design—it’s a translation from concept to real product. During any translation, things get lost. Plans and reality don’t always match up perfectly: maybe a menu component can’t function the way the designer planned because of device support or a UI concept will require so many API calls that the page will load slowly. In many cases like these, enforcing the original design vision would be a bad thing for the end user, and designers should care about fixing these issues.

When designers realize there is still work to be done when mockups are complete, they will start to see collaborating with developers during implementation as a normal part of the design process. Design isn’t done when development begins, but instead, one of the most important parts of the process begins then too. Getting involved while the design is being built is a designers’ chance to help ensure the design functions and looks the best it can—even if it ends up slightly different from the original plans.

> *If the designer isn’t collaborating with developers, they’re missing the chance to make important design decisions and outcomes can be affected if they don’t contribute during implementation. So, designers, open up to collaboration and your product design will be better for it.*

ENSURE EVERYONE UNDERSTANDS THE DESIGN

The other important part of making sure development goes smoothly is how teams handle delivering the design to developers. When designers begin that transition with a literal handover—they send over source files and the developer gets to work—we're inviting problems. That old method just doesn't work.

> *"Developers and other non-designers might not always understand a design clearly. Tossing a set of mockups at developers and expecting them to understand every detail is unrealistic. It can lead to inaccuracies and other issues. Because of this, you need to make sure everyone understands the design when handing over source files."*

Some designers don't want to have to explain their work.they want everyone to be able to appreciate and understand a design without explanation. They believe the value should be obvious to everyone. Designers want others to trust them and sometimes, simply, go along with their recommendations out of respect. When they have to explain every decision, they can often feel insulted or even that people are doubting their skill.

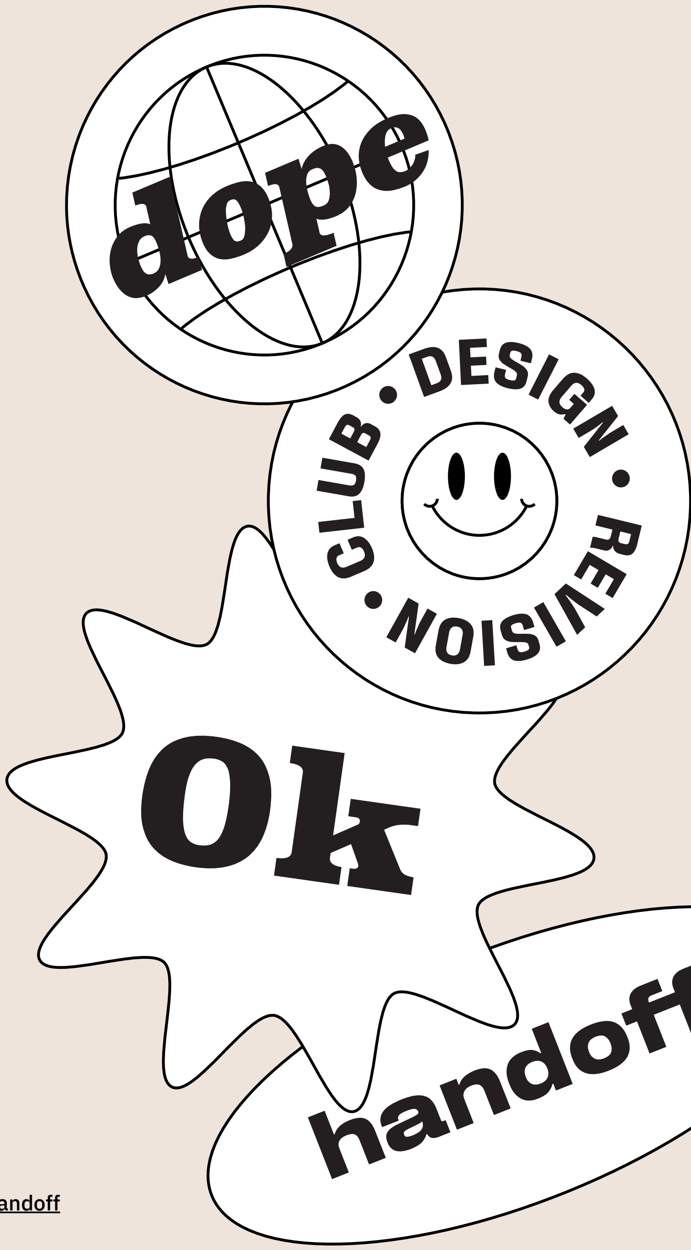
However, designers can parse and understand a piece of work and its rationale at a glance, which is a skill they've developed through lots of practice and effort. Other people haven't necessarily put in the time to gain that ability, and the design might not be totally clear to them.

“Never hand over source files without a conversation. It might seem unnecessary, but assuming everyone understands without explanation isn’t worth the risk when explaining a design is such a simple thing to do”

Asking questions and trying to understand the design is their way of respecting it, so designers need to be open to explaining.

So, before implementation begins, designers need to explain the design. Describe the details, be available to answer questions, and verify the developers understand the concept. It's their job to ensure the design functions the best it can, and for that to happen, the people building it need to understand what they're building.

Take time to walk people through the design, and let them ask questions. If a designer assumes everyone else understands, it's likely other assumptions are being made somewhere. Inconsistency and low quality implementation happen not because of unskilled designers and developers but because of designers and developers who don't talk to one another.



02

02

ENSURING **CONSISTENCY** IN DESIGN

A comprehensive implementation phase is essential as you already learned, but consistency is a concern during all phases of the design process.

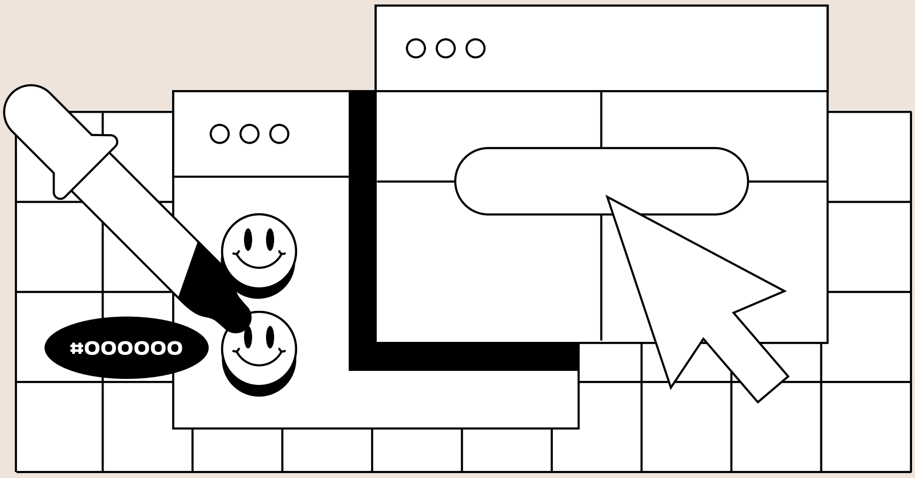
Consistency is a core design concern. Every designer wants to ensure consistency because design is fundamentally about establishing reusable methods for handling the same visual problems. If you're solving the same visual problem in a different way each time it arises, what you're doing isn't design—not really.

But achieving consistency is no small task. Anyone who's been in charge of keeping a design system or style guide updated can tell you that it sounds easier on paper than it really is. (We'll cover style guides and other methods later.)

To really understand how to ensure design consistency, you need to be aware of where it comes from. There are many points at which a product can start to deviate from the design plan. Catching those points when they happen is just as important as making the design in the first place.

SOURCES OF INCONSISTENCY

While every project and team is different, here are the most common times when design inconsistencies can creep into a product.



Product Design & other long-term design work

Despite best intentions, every design evolves over time. Each new release includes new design work. When creating something new, be a it a new feature, page, campaign, or anything else, there are often new visual challenges the current design system doesn't cover, so designers need to apply the styles in new ways.

Over the long term, old views, pages, and assets can get left behind as new releases push the design system forward.

Multiple designers working on the same product or brand

Everyone will have their own interpretation of the design system. Varying techniques and perspectives are why having a diverse team is so valuable. However, these different methods can cause the design to diverge if the designers are unaware of one another's work.

Implementation without designer involvement

As you already learned, implementation is an important phase of the design process. If designers aren't involved, all kinds of unintentional mismatches and errors can become part of the live product.

Bug fixes and other changes that happen without designer involvment

Just as designers need to be involved during implementation, they need to be involved in other changes that affect the design, such as emergency bug fixes, new marketing strategies, and so on. Often out of urgency changes are implemented and launched without a designer involved. This is a common source of inconsistencies that needs to be addressed.

Design based on incorrect or insufficient information

Input from developers, marketing, business experts, and user research is critical to making the best design decisions possible.

Poorly informed designs always end up having to change, and those changes can bring inconsistency. Making it work the first time is always better than redesigns and pivots because when designers are making sweeping changes, it's possible to miss minor issues.

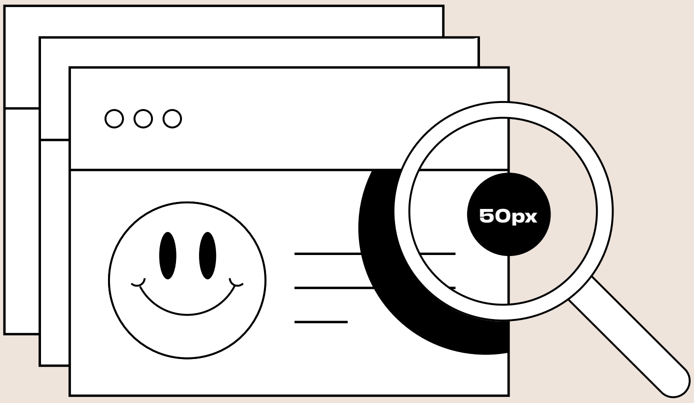
Multiple design system use cases

When a design system or style guide is used in a variety of ways or implementations, such as for a website, mobile app, and print ads all for the same brand, it's more prone to breaking. The broader a design system needs to become, the more likely it is to have inconsistencies.

Edge cases

Some features or content might differ from the rest of the product so much that the usual design approaches don't really seem to apply. Often low-usage features and pages introduce design consistency, such as: settings menus, unique UI components only used in 1 place, privacy policy pages, and so on. Designers create new styles to cover these needs, intending those styles to never be reused, but they slowly seep into the rest of the product when others reuse those
styles thinking they are part of the accepted design system.

HOW TO ENSURE CONSISTENCY



Seeing so many potential sources of inconsistency, you should be realizing that covering every possible scenario is a huge job. But you and your team can accomplish consistency by choosing the right tools and then using those tools correctly.

Design Systems you'll actually use

The first solution many teams try is a design system of some kind. However, Design Systems, living style guides, and brand guidelines are imperfect solutions because getting people to follow them can be surprisingly difficult.

The Design System is a tool for everyone because design happens in lots of phases with lots of people involved. Many believe that a Design System is a tool for designers alone, and people outside the design team don't think about using it. Worse, the way some teams use Design Systems and similar tools assumes that inconsistency only happens when a design mockup is being made, but as you know that's not the only source.

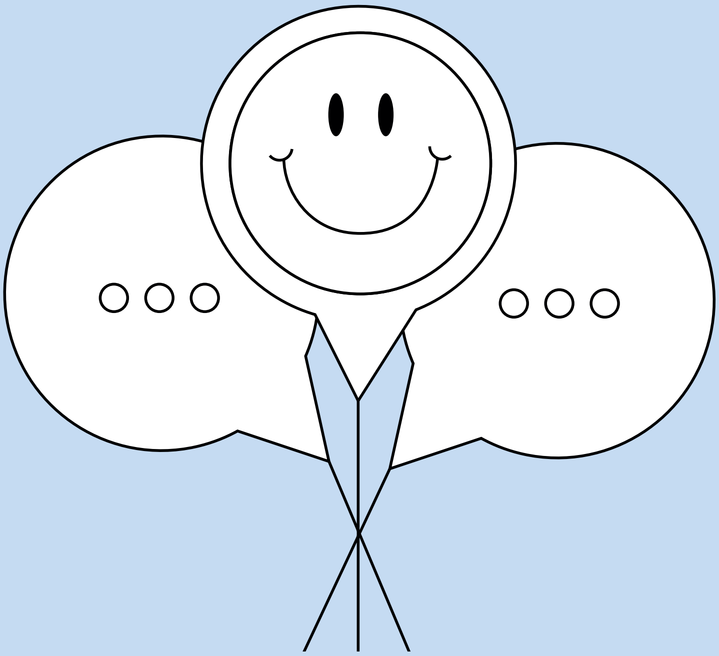
However, if design teams do try to get others to use and follow the Design System, again they view that task as one of enforcement; designers are pushing everyone to do what they ask.

No one on your design team wants to become the style police. Being the person who reviews every asset is not only tedious, it's counterproductive and frustrating for both sides. Giving someone the task of enforcing the Design System is basically assuming people are going to argue. No one wants to be micromanaged, and when people resent having a direction imposed on them, inconsistencies are more likely. Plus, it's just not an enjoyable way to work.

> *"Rather than enforcing the design approach, designers can earn consistency through agreement and mutual support. Design Systems are great, but they're not laws to enforce. Instead, use design documentation as a goal and ambition everyone can take part in."*

We're not saying design is everyone's job. Designers should be making final calls about the shape of a design. However, inviting others to do their part in bringing it to life can be powerful.

Inviting others to do their part in bringing the design to life can be powerful.



“Make design something everyone can believe in and even have some ownership over.”

Look for buy-in

Designers should teach everyone about why the design system is important; explain how to use, maintain, and watch out for it.

Here's an example of what that looks like:

> *"Imagine that a designer and developer are working on the implementation of a new software feature. In many teams, the developer would send the designer a staging link for review, and the designer would then be in the position of asking for changes to match the design system. The designer feels like a jerk for asking for minor changes, and the developer feels like they're getting bossed around."*

Now instead, imagine that everyone in the company had access to a style guide. The design team has presented this style guide to the development team, explained the rationale behind key decisions, and invited the developers to review it and use it. Then, when that same designer-developer team is implementing the new feature, instead of sending a staging link for review, the developer does something incredible.

They ask the designer this question:

"The button hover styles weren't shown in the mockups, so I used our style guide for them. Do you think this is the best style to use?"

The second dynamic is completely different. The developer has shown they care about the design perspective, and now, the designer is no longer in the position of being the enforcer and the developer is no longer merely following instructions. They're collaborating on the best way to implement and realize the design. It's teamwork instead of conflict. That, right there in such a simple question, is how great design happens.

An open design process

Getting everyone to care about the design like that is simpler than it sounds. If designers are closed off and territorial over design concerns, people will see these concerns as adversarial. But if designers are open and forthcoming, others will react in kind.

An open design process might sound like changing the way designers work by inviting compromise and interruption. We're not suggesting designers change how they design, only that designers change how they talk about design and open up specific points of the creative process that won't cause any harm.

An open design approach starts with teaching.

Designers should be sharing articles, books, and findings with the rest of the team. Truly advocating for design requires teaching how it works and why being consistent is so important.

For example, changing a font size from 14px to 16px seems like a minor issue, but when people understand that minor issues like this affect an entire visual hierarchy, they will realize it's not a petty request. As people's understanding of design grows, they'll value it more and might even be excited to support design concerns.

Further, if designers want support, they need to reciprocate. Learning about the value and perspective of other professions goes a long way toward building mutual trust and support.

> *"Non-designers might not be used to this kind of work, so the designer should make it clear that they might not necessarily use the ideas people share. Designers should clarify that the goal is to evaluate solutions, see what's doable, and choose a direction, and that they're open to other perspectives."*

Designers should learn a little bit of code, marketing strategy, business, and other areas that overlap with their work. Not only does that get others to support design goals, it makes designers better at their own profession because they understand the context surrounding their work. Not to mention, when designers support other aspects of the business, the results can be exciting. Every designer will hesitate to share early, unpolished concepts, and there's no need to do that.

Opening up the design process isn't about inviting the rest of the team to micromanage or look over shoulders, but instead choosing specific points in the process to double check information, test out concepts, and encourage involvement. Opening up these limited interactions within your team will set a positive tone for the rest of the project, which will lead to better implementation, consistency, and support for design efforts.

03

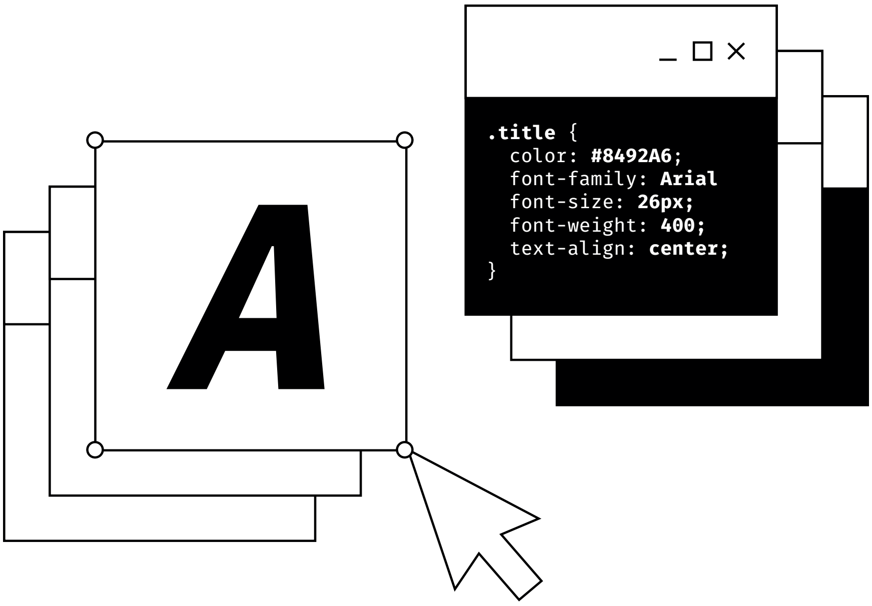
03

03

03

KEY POINTS IN **THE DESIGN PROCESS** TO CHECK IN WITH DEVELOPERS

There are times when a designer will need to disappear and focus on the creative process without interruption. However, there are a few simple times where inviting collaboration can be beneficial for everyone.



```
.title {
  color: #8492A6;
  font-family: Arial
  font-size: 26px;
  font-weight: 400;
  text-align: center;
}
```

01
Project Discovery

There are times when a designer will need to disappear and focus on the creative process without interruption. However, there are a few simple times where inviting collaboration can be beneficial for everyone.

Choosing what should be designed is a critical part of a designer's job. Any designer worth their salt will tell you that they need to be involved in choosing which features and ideas to design. They can add a lot of value when they're involved in business strategy conversations and product roadmap planning.

Project discovery—creating project plans, specifications, research, or whatever your team calls it—is the first opportunity to invite collaboration on the design. Discovery, while important, isn't always the most exciting work. Writing specs, listing features, and making job stories can be tedious. But the upside to project discovery is that everyone has to be involved. Designers need to estimate how long it would take to design a feature and developers how long to build it. Everyone will have opinions on the value and feasibility of features. All of those perspectives are required for choosing the best features and making a realistic project plan, so having everyone involved is obviously valuable.

Every project begins with discovery, which means every project begins with collaboration. And after the project is planned, rather than everyone going their separate ways, you can continue that collaboration. Discovery can set the expectation for openness during the rest of the project.

“Every project begins with discovery, which means every project begins with collaboration.”

02
Design examples or mood boards

When a project begins, designers often spend time making rough sketches, looking for inspiration, and imagining solutions. They're evaluating techniques and concepts to decide on a direction to pursue. Imagining the possibilities is one of the most enjoyable parts of design, and inviting everyone to share in that excitement is a great way to start the project out in a positive way.

Further, inviting other perspectives into early concepting can bring all kinds of new ideas to fuel the designer's creativity. Developers, clients, marketing, managers, and even execs can provide interesting ideas if designers are open to hearing them. After all, the creative process is about finding the best idea. It doesn't matter where the idea comes from.

So, make the early design concepting phase open to everyone. Invite developers and designers to sketch ideas together. Send out examples of designs that solve similar problems and ask for feedback from everyone.

Reviewing visual samples together is totally different from a list of specs or features; having a real object to discuss can bring out opinions, requirements, and other concerns that didn't surface during discovery. It's a chance to uncover issues no one has brought up yet, which designers don't want to interrupt the creative process later. It's better to know early, before unforeseen concerns cause major changes or revisions.

03 Wireframes

In the same way, asking for feedback on wireframes is a low-risk way to uncover other requirements and concerns that would be more damaging if they surfaced later in the design process.

Often times, people won't have an opinion until they see something visual. Designers have the ability to picture how an idea might look or how two samples might combine into a unique design, but that can be difficult for non-designers.

Many layout concepts, UI component ideas, and navigation structures are simple enough to evaluate in wireframe form.

Getting early developer perspectives on these design aspects is a great way to verify your concepts are realistic, but it also makes implementation much less painful because developers will already support the design direction.

> **"Showing a wireframe early can give others a clearer idea of the designer's plans. It gives everyone more to talk about and helps avoid design waste later.**

04
Design Conversations

Even if your team chooses a more collaborative and positive route during development, change requests are still the most risky at this stage because they cause the most extra work for designers. Even minor changes to a large design system can have huge implications. Designers' instinct is still to resist any requests at this stage, even if they do try to explain the design and stick around during implementation.

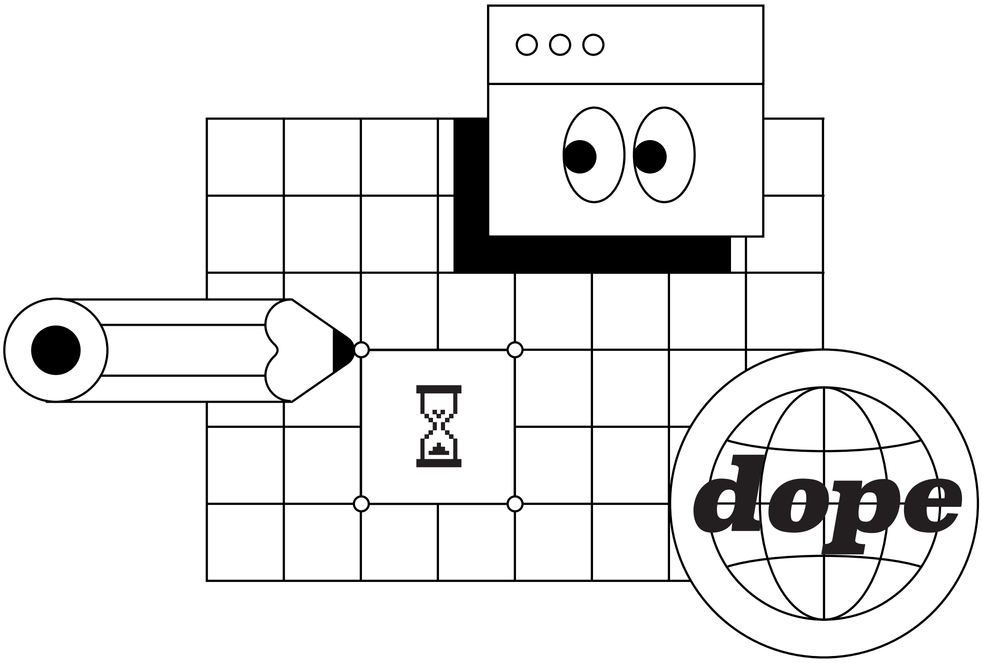
> *"Many designers follow an etiquette when critiquing design work. They focus on the reasoning behind each decision and evaluate how successful each detail is in meeting the overall purpose rather than recommending specific visual changes."*

But non-designers don't know that etiquette, so it's likely they'll ask for specific changes. Their feedback might seem blunt or even inappropriate to designers—like they're telling the designer how to design.

To avoid that, designers should model how design conversations should happen and train others in the etiquette. Explain which kinds of feedback are helpful and welcome, and which aren't. A common approach is asking for prescriptive feedback over descriptive feedback. Use this phrase: "tell me what's wrong, not how to fix it." For example, "I'm concerned users won't notice the button" instead of "Please make the button bigger". Setting clear expectations about how to provide useful feedback helps avoid conflicts and defensiveness.

CONCLUSION

CONCLUSION
CONCLUSION



The time has come to stop viewing design and programming as opposing forces. Effective design and high quality code are not mutually exclusive—they in fact support one another. A design that encourages fast and reliable code is a quality design. Code that delivers an exceptional user experience is quality code.

The process of weighing design and development concerns shouldn't be about competition; every team's goal should be finding the best possible combination of design and code. Yes, sometimes a designer or developer will need to alter their original plans, but doing so isn't a compromise.

> “Great products come from harmony between the design vision and technical details. Collaboration is the method for creating that harmony.

Marvel

Explore our handoff solution and start generating specs at the click of a button: **marvelapp.com/features/handoff**