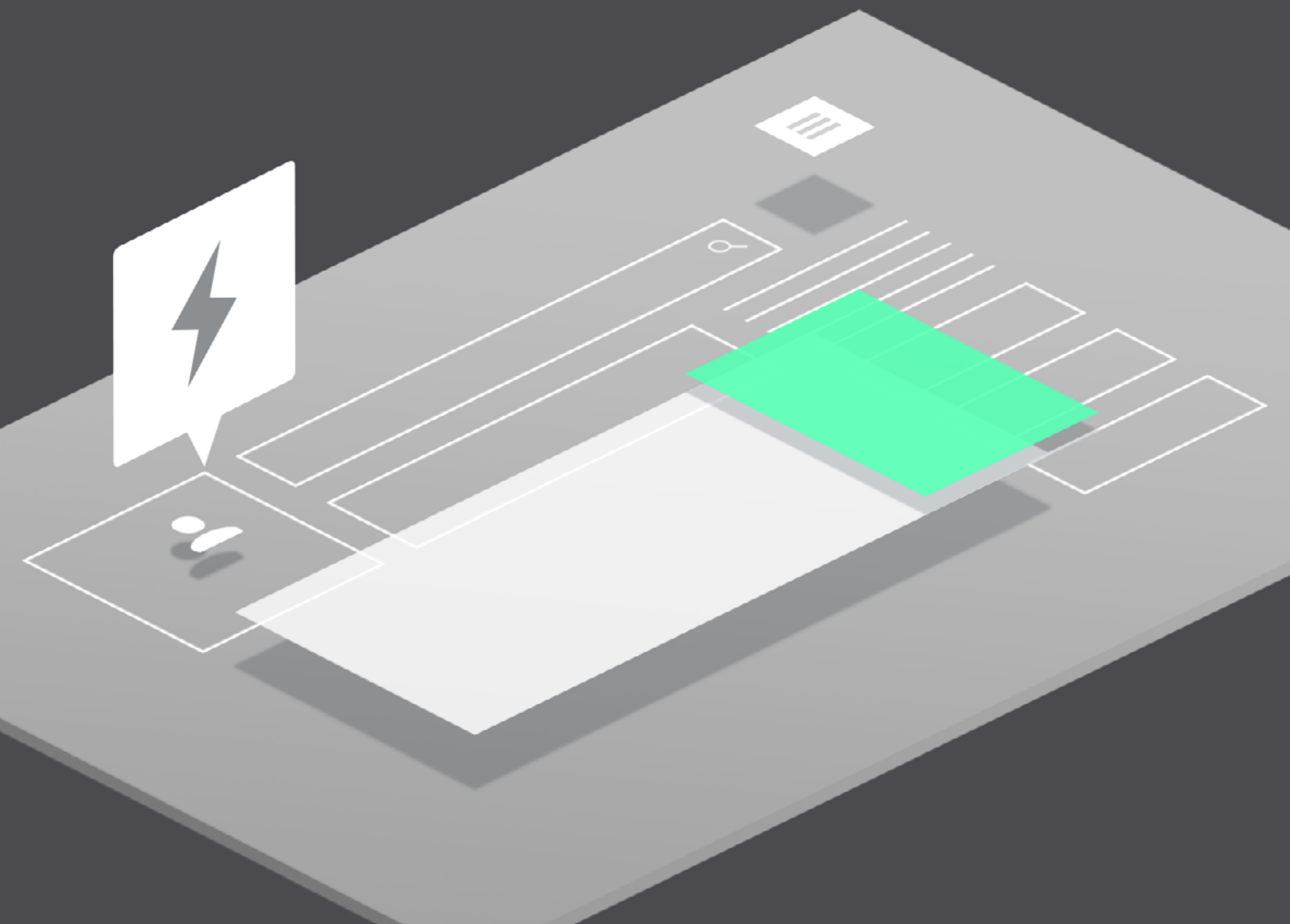


UXPin

The Ultimate Guide to **Prototyping**

The best prototyping methods, tools, and processes



UXPin

The Ultimate Guide to **Prototyping**

The best prototyping methods, tools, and processes

Copyright © 2015 by UXPin Inc.

All rights reserved. No part of this publication may be uploaded or posted online without the prior written permission of the publisher.

For permission requests, write to the publisher, addressed “Attention: Permissions Request,” to hello@uxpin.com.

Index

0. Introduction	6
1. A Practical Look at Prototypes	9
Clearing the Air: Wireframes vs. Mockups vs. Prototypes	10
Why You Must Prototype	11
How Prototypes Improve Collaboration & Communication	14
How Prototypes Add Balance to Design	17
How Prototyping Makes Usability Testing Easier	18
Takeaway	20
2. Traditional Prototyping Methods and Tools	21
Paper Prototypes	22
Wizard of Oz Prototypes	27
Takeaway	30
3. Digital Prototyping Methods and Tools	31
Presentation Software	32
Coded (HTML) Prototype	35
Prototyping Software ⁷ & Apps	38
Takeaway	41

4. 9 Prototyping Best Practices	43
1. Know Your Audience and Your Goals	43
2. Prime Your Audience Beforehand	45
3. Involve the Users (Participatory Design)	46
4. Focus on Flows and User Scenarios	46
5. Keep Clicking Simple	50
6. Don't Neglect Animations	51
7. Sketching: The Prototype for the Prototype	52
8. Don't Let Coding Hold You Back	53
9. Use Prototypes for Usability Tests	54
Takeaway	56
5. Wireframing & Prototyping:	
The Past, Present, and Future	57
Present: The Current State of Design	57
Present: The Current State of Prototyping	61
Past: A Prototyping Timeline	62
Future: The Age of Prototyping	65
Takeaway	68
6. Creating Interactive Prototypes from Photoshop Files	69
Importing from Photoshop – overview video	71
Importing from Photoshop into UXPin – step by step	72
General notes on using interactions and animations	77
Button: scrolling the page after click, changing style on hover	81
Form: triggering visibility on scroll	85
Form: interactive inputs	87

Form: interactions after signing up	88
Previewing and gathering feedback	90
7. How to Create Interactive Prototypes From Sketch Files	92
Overview Video	93
Importing from Sketch into UXPin	94
Prototyping Animations & Interactions	99
Previewing Animations & Interactions	101

Introduction

A quick note from the authors

When it comes to prototyping, there's quite a bit of debate over low fidelity vs. high fidelity, paper versus digital. As you'll discover in this book, each tool has its place depending on personal preferences and design stage.

We'll share expert advice, design theories, prototyping tips, and screenshots of prototypes created using the most popular methods.

For beginners, you'll learn the many different tactics and processes for prototyping and why prototyping is required for crafting memorable product experiences. For more advanced readers, you'll learn how to select the right fidelity and prototyping tool and where prototyping is headed in the future.

You'll find that we've also analyzed prototyping best practices from companies like **Buffer**, **Apple**, **Google Ventures**, **ZURB**, **Groupon**, **IDEO**, **Virgin America**, and dozens others. We've made this book

as practical as possible and included advice based on our own prototyping experience at UXPin.

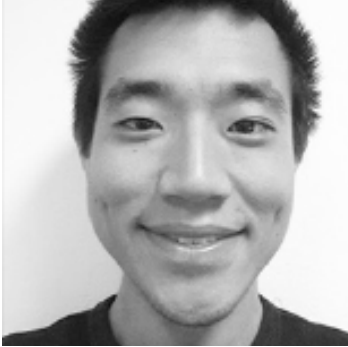
Prototyping helps unveil and explore human needs, opening the door to insightful interaction and more empathetic design solutions. If a wireframe is worth 1000 words, then a prototype is worth at least 10,000.

We'd love your thoughts on what we've written, and feel free to share with anyone who might find this helpful.

Happy prototyping,

Jerry Cao

co-written by Kamil Zieba & Matt Ellis



Jerry Cao is a content strategist at UXPin where he gets to put his overly active imagination to paper every day. In a past life, he developed content strategies for clients at Brafton and worked in traditional advertising at DDB San Francisco. In his spare time he enjoys playing electric guitar, watching foreign horror films, and expanding his knowledge of random facts.

[Follow me on Twitter.](#)



Co-founder and head of product, Kamil previously worked as a UX/UI Designer at Grupa Nokaut. He studied software engineering in university, but design and psychology have always been his greatest passions. [Follow me on Twitter @ziebak](#)



With a passion for writing and an interest in everything anything related to design or technology, Matt Ellis found freelance writing best suited his skills and allowed him to be paid for his curiosity. Having worked with various design and tech companies in the past, he feels quite at home at UXPin as the go-to writer, researcher, and editor. When he's not writing, Matt loves to travel, another byproduct of curiosity.

A Practical Look at Prototypes

How & Why Prototypes Are Mandatory for Good Design

Nothing brings you closer to the functionality of the final product than prototyping. While [wireframes](#) sketch out the blueprint and [mockups](#) show the feel and texture of the design, it is the prototype that brings to life the “experience” behind “user experience.” That beautiful call-to-action may look great on the screen, but you won’t know if it works on end users until the clickable prototype. Not only do prototypes help provide proof of concept, they more importantly expose any usability flaws behind the wireframes and mockups.



Photo credit: UXPin

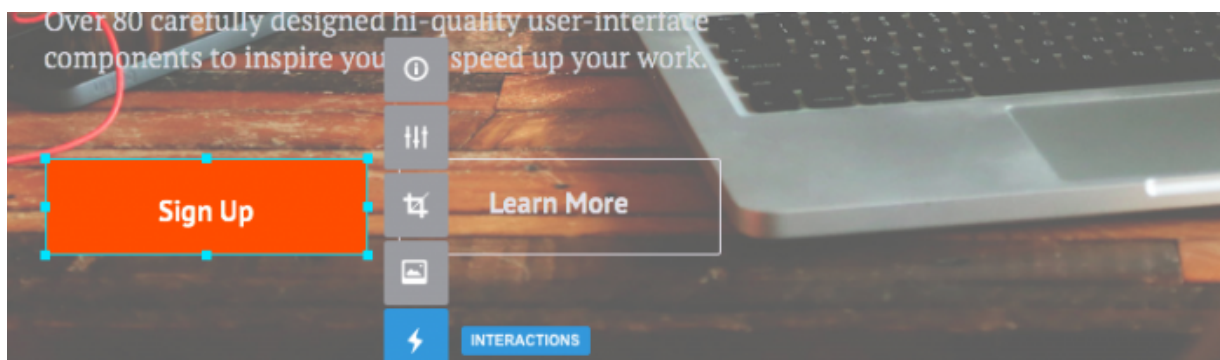
So how do we actually put into the practice this safeguard against emergency stakeholder meetings, endless revisions, and painful late nights in the development phase? While we previously touched upon proper prototyping in our [Guide to UX Design Process & Documentation](#), here we've devoted an entire e-book to explaining how prototyping can make or break a product's success.

In this chapter, we'll begin by clarifying some misused terminology, then we'll look at the most compelling reasons to prototype and how prototypes improve collaboration, design, and usability testing.

Clearing the Air: Wireframes vs. Mockups vs. Prototypes

Say it with us: a prototype is not a wireframe. And it certainly is not a mockup.

It's much easier to confuse wireframes with mockups since they're both static designs, but we've also seen prototypes getting thrown into the same realm of misuse.



Source: [UXPin](#)

Wireframes are low-fidelity designs, usually defined by rough shapes and X-ed boxes to define the content structure. Mockups, on the other hand, are a step above: they're usually mid to high fidelity, but still a static design.

Prototypes exist on a whole other realm of fidelity because they are interactive designs. You can certainly build them with low visual detail, but they will always function more than any wireframe or mockup ever will.

In the waterfall methodology, you'll build a wireframe, followed by a mockup, and finally the prototype. However, as we start adopting Lean and Agile UX processes, it's actually much more common to see designers start with a rough prototype, figure out the rough interactions, then either jump into wireframing or improve the visual detail in Photoshop/Sketch and finish up in code. Alternatively, they might also use prototyping tools that support a wide range of visual fidelity and functionality.

As long as it works, and it gets you working on interaction design with others, it's good to go.

Why You Must Prototype

While skipping prototyping might save some time during design, that surplus can be lost many times over in development. If people

try on clothes before buying them and test-drive cars before signing the check, then it only makes sense to test your designs interactively before they go into development. Interaction, after all, is how users access the design solutions to their problems.

To better understand why you should prototype, let's look to **Todd Zaki Warfel**, the designer who wrote the book on prototyping... [literally](#). We highly recommend checking out and buying his book since it contains plenty of real-world advice. Warfel summarizes his excellent book's points in [a slideshow presentation](#), where we can see all the relevant reasons to embrace prototyping.

While these vary depending on designers and their needs, some universal benefits (which he touches upon) include:

- **Better collaboration** – It's one thing to discuss requirements documentation, but it's a whole other level of imaginative collaboration when both parties can play with a prototype and explore limitations and possibilities. Documentation can be misinterpreted, but experiences are shared.
- **More realistic learnings** – Wireframes, mockups, and requirement documents live in paper, not reality. Prototyping allows teams to experiment, giving them the freedom to fail cheaply while learning real lessons about the design.
- **Experience instead of explain** – Prototypes can be great for conveying ideas that you just can't quite put into words clearly.

For example, try explaining or annotating a complex animation through wireframing...somehow, you're going to miss the mark. On the other hand, actually showing the real-life website or app proves your vision more than any amount of documentation.

- **Quick & efficient usability testing** – The earlier you have a live interactive design, the sooner you can test it with the people who matter most: your users. It's difficult to gauge authentic reactions with static designs, but prototypes actually tease out real user behavior. Users don't look at designs, they use them. Prototypes let you test how your design actually works, not just how it looks.
- **Prioritize interaction design** – On a related note, prototypes force you to think about interactions early instead of treating them as a nice-to-have. If your visual design is stunningly beautiful but the interaction design feels mechanical, your design probably isn't going to be very usable. The visuals must fulfill the experience, not the other way around.

Not every prototype needs to be elaborate and consuming – in fact, [some schools of thought believe in a rapid, low-investment form of prototyping](#). The point is that it's almost always beneficial to devote some resources to prototyping, how many depends on your specific needs.

How Prototypes Improve Collaboration & Communication

Showing is always better than telling, and experience is king. If people can interact with your ideas, then they're better able to understand them. If nothing else, you'll at least encourage someone to try your idea before they axe it (and that can make all the difference in whether a design lives or dies).



Source: "IMG_8298." [jeanbaptisteparis](#). *Creative Commons 2.0*.

Prototypes clarify internal communication in a few ways.

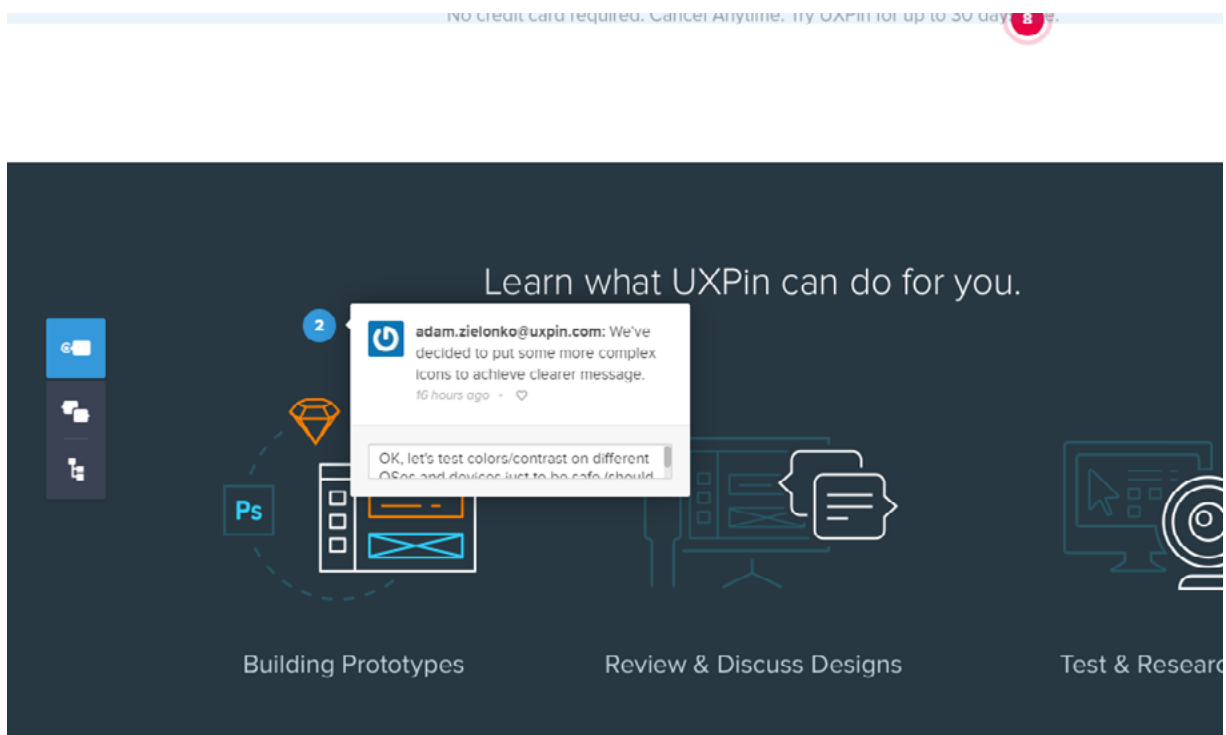
First, it creates common ground between developers and designers. Developers understand the language of interaction, because every element in their code framework acts and reacts to multiple conditions. Likewise, designers create humanized solutions with which people can interact to solve their problems.

The prototype, then, is the intersection point. When a designer builds a prototype, they get a realistic feel for how somebody inter-

acts with the interface and reacts to the on-screen feedback (whether it's simple messages, modal windows, or transitional animations). When a developer plays with a prototype, they can start envisioning how the system must be built to fulfill the experience.

Different departments use different jargon, but a prototype shatters all barriers. Prototypes are the toys of design, and when something is fun, people are more eager to let down their guard and start exploring the crazy ideas that just might work.

As the Founder and CEO of Emmet Labs, David Yerba, [puts it](#), prototyping “gets the right people in the room communicating in the right ways.”



In UXPin, collaborating on prototypes is natural thanks to the live presentation and collaboration features. With iteration tracking and live commenting, wireframes and prototypes can be discussed and presented in real time without any need for email.

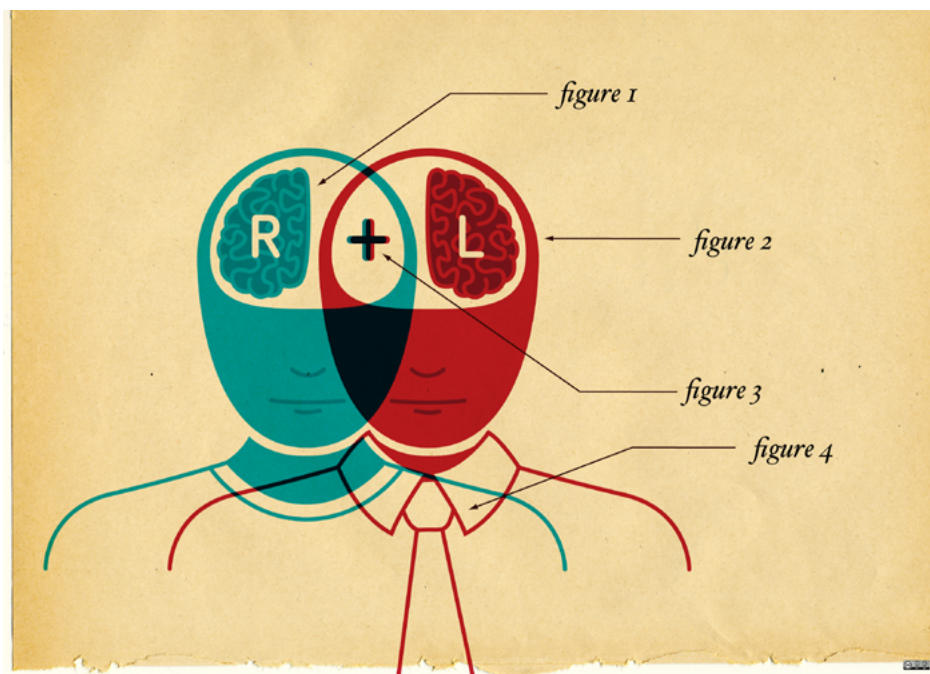
Secondly, [in the same article](#), Yerba also describes the benefits of prototyping for formal presentations. Clients and stakeholders can easily misinterpret a description, even if delivered with some convincing research or pixel-perfect mockups. An interactive prototype, on the other hand, requires little description. Prototypes help stakeholders think about the experience, instead of falling on the crutch of criticizing visual elements just because they're right in front of their eyes.

A prototype is a powerful weapon to bring to any presentation. David and Tom Kelley, Founder and Partner at IDEO, [ascribe to what they call “Boyle’s Law”](#) (named after one of IDEO’s master prototypers, Dennis Boyle). Boyle’s Law is simple but effective: never go to a meeting without a prototype. In terms of its communicative and persuasive abilities, this presentational aid often makes the difference between a “yes” or a “no.”

How Prototypes Add Balance to Design

Prototyping is the phase in which the conceptual becomes real, so it requires both creativity with practicality, rationale with intuition.

Prototyping forces you to think in terms of multidimensional UX design.



Source: *"Is the traditional business world at war with creativity?"*
[Opensource.com](https://opensource.com). Creative Commons 2.0.

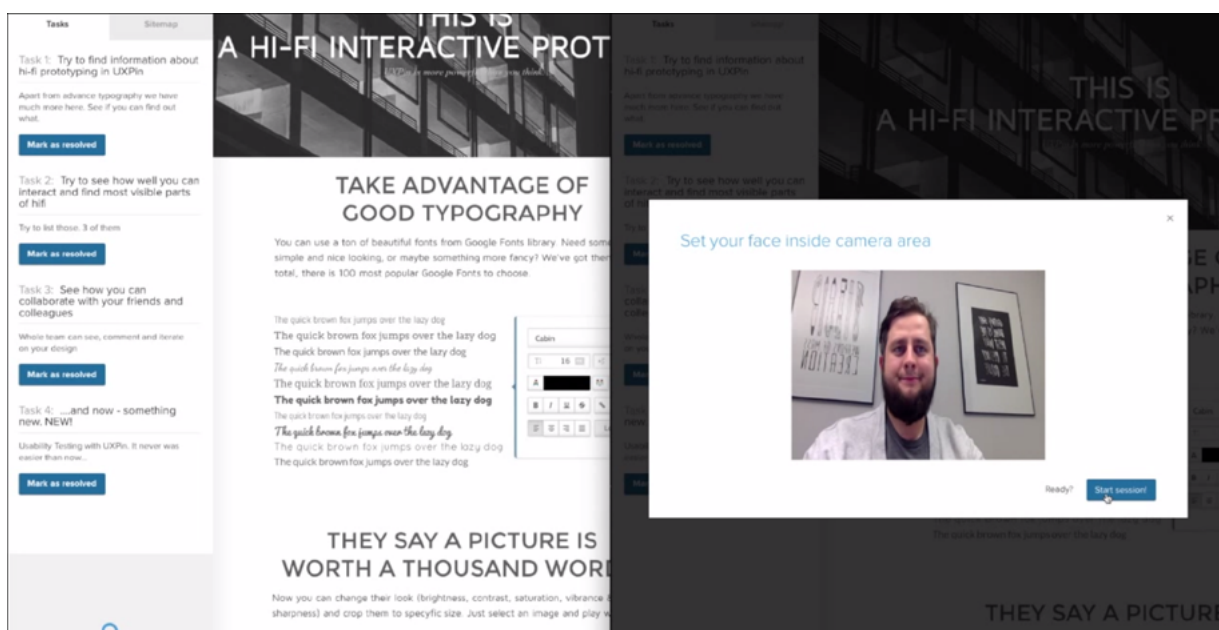
When we [worked on improving the Yelp website purely as an exercise](#), we built first a [lo-fi](#) and then a [hi-fi prototype](#). Initially the interactivity was simple, but in designing it, we were confronted with decisions and obstacles that would not surface in a static design. Through prototyping, we identified and solved UX problems such as how to transition from the homepage to the search results without suddenly overwhelming the user with too much information.

Long story short, the prototyping phase was when we were able to merge our conceptual goals with our practical reality. It was when visual design and interaction design came into balance.

How Prototyping Makes Usability Testing Easier

The reason we divide the tasks into wireframes, mockups, and prototypes is because each has a different purpose.

By focusing only on structuring in wireframing, for example, we're able to create a solid structure without distracting ourselves just yet with mockup visuals or prototype functionality. While wireframing and mockups lean more towards the creative and abstract, prototyping is about knuckling down and bringing ideas to life – and that requires usability testing to get right.



Source: *Usability Testing for Prototypes*

To make an analogy, prototypes are to the final product what sketches are to wireframes/mockups. Prototyping is a way to “sketch with interactions” to create a rough model of usability, then refine and perfect.

For this, usability testing is vital. We recommend running usability tests and iterating accordingly at each stage of the prototype so that its fidelity and functionality move in the right direction. While usability testing is a dense topic all its own, here are some broad basics to keep in mind:

- **Be realistic** – If your site or app is still in a rough stage of design, then you’ll want to make sure someone is there to moderate the testing session (since users will probably have plenty of questions). If your prototype is more refined and close to resembling the final product, you can explore remote usability testing options like those offered by [UserTesting](#).
- **Recruit the right test-takers** – Chances are your product is aimed at a target user-group, so make sure your test-takers represent that group. Qualitative tests can be run with as few as [5 people](#), quantitative tests require [at least 20 people for statistical significance](#). For a full list of user recruiting tips, check out [Jakob Nielsen’s list of 234 tips and tricks](#) to recruiting people for usability tests.
- **Set up the right test** – There are many [different types of usability tests](#) and many different ways to conduct them. Choose the

one that match your budget, timing, and project needs.

- **Analyze results** – This is when usability data turns into design insights. Review the results with your team to minimize bias, then iterate accordingly.

For more information on what types of usability tests would work for you and how to conduct them, download [our free ebook *The Guide to Usability Testing*](#). We describe over 30 methods with examples of best practices from **Apple, DirecTV, Microsoft**, and others.

Takeaway

We always prototype, and advise others to do the same. While we understand that certain restrictions may make prototyping harder for some companies, we strongly suggest making it work, even if only a limited prototype.

The benefits far outweigh the costs: better collaboration among the team, finding/fixing problems early, improving existing designs, and communicating through action rather than words. These advantages are well worth the extra time and effort.

Traditional Prototyping Methods and Tools

The Non-digital Methods of Paper Prototyping and Wizard of Oz

Whether you prefer working with your hands or are just a technophobe (though if the latter's true, you might be in the wrong industry), there are ways to build a prototype that don't require much technology or money.

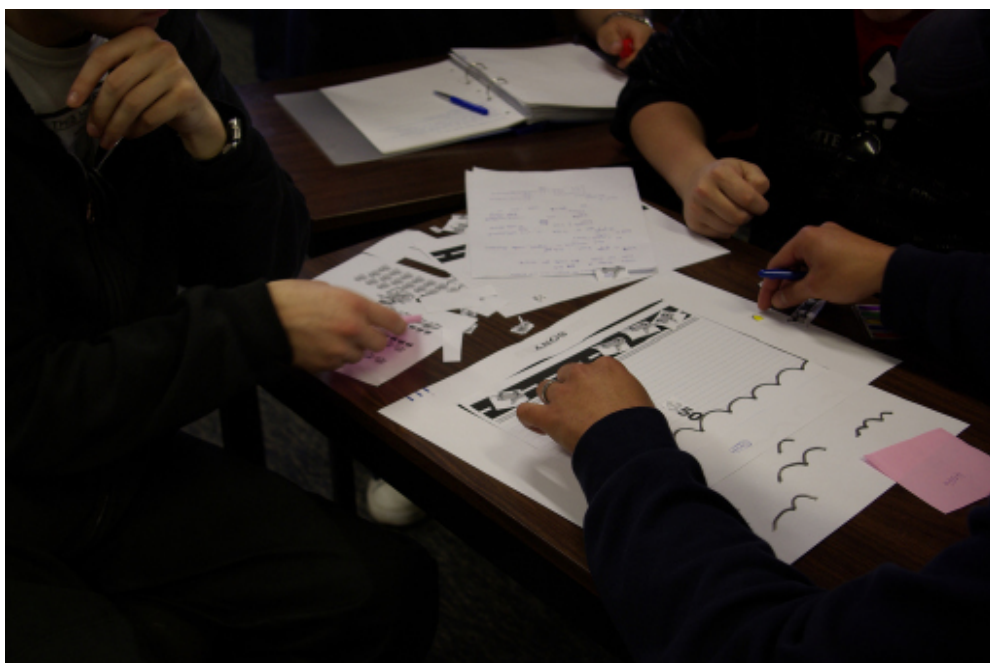


Photo Credit: Samuel Mann. Creative Commons CC BY-SA 2.0

These traditional prototyping methods – paper prototyping and Wizard of Oz – trade fidelity and functionality for speed, which can

make them quite useful in the early stages of design. We'll explain the advantages and disadvantages of both methods and how you should use them.

Paper Prototypes

In the age of modern technology, it can sometimes be refreshing getting back to the tools we used in kindergarten. Paper, scissors, glue, coloring utensils – what we once used to make unicorn collages can now be co-opted for creating useful yet quick prototypes. Let's explore the benefits, disadvantages, and methods for paper prototyping.

1. Advantages of Paper Prototyping

The benefits of paper prototyping are quite straightforward: they're cheap, fast to make, and easy to collaborate with. Here's a few reasons why:



Photo Credit: [Samuel Mann](#). Creative Commons CC BY-SA 2.0

- **Low budget** – Paper prototypes can be built with supplies already in your office. All you need is paper, pencil or pen, scissors, and some creativity. If you want to get fancier, grab some index cards or Post-Its.
- **Rapid iteration** – Which would you rather throw away, 2-hours worth of coding or a sketch that took 20 minutes? Due to the highly dynamic nature of the prototyping phase, you're always going to create some waste, so sometimes it doesn't make sense to spend a lot of time on any one design.
- **Fun collaboration** – It's hard not to bond when a group of people are given art supplies and asked to create. The casual nature of paper invites more participation and feedback, which you should encourage as much as possible during early product stages.



Source: [Converting Offline to Online Prototypes](#)

Moreover, a quick session of paper prototyping can help brainstorm solutions if you're currently stuck with your digital prototype. Prototyping with paper or software don't have to be exclusive – they can complement each other well if you play on each's strengths.

Interestingly enough, UXPin actually launched in 2011 as a [paper prototyping tool](#). Once we transitioned into a cloud app, we still complemented the paper process by letting people scan and upload paper prototypes into UXPin.

2. Disadvantages of Paper Prototyping

However, paper prototyping isn't for everyone. Jake Knapps, the designer behind **Google Hangouts**, [claims that it is a “waste of time.”](#) While we think it has its time and place, these are some of its glaring disadvantages Knapps pointed out that we feel are very valid (along with some of our own thoughts)..

- **Lack of direction** – While paper prototyping can be useful for individual processes, you need to explain the context. Otherwise, you might get feedback based on your efforts, not based on your actual product.
- **Inaccurate reactions** – Research should be based on your users' reactions, as those come naturally without thought. Because paper prototypes require the user to imagine what the final product will be like, you're getting feedback on a deliverable instead of reactions to something that resembles the product.

- **Can be slower than prototyping tools** – Given the wide selection of prototyping tools out there like UXPin, Invision, Omni-graffle, and others, you can get prototyping quite quickly (and less messier) with the added benefit of collaborative features. It all depends on your preferences.

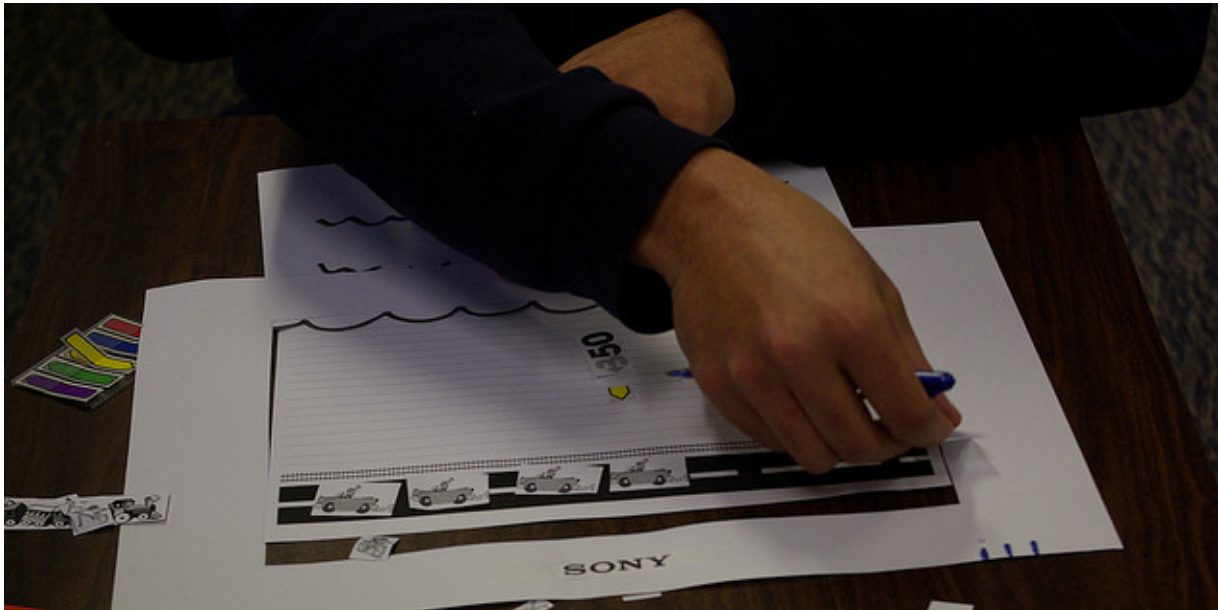


Photo Credit: Samuel Mann. Creative Commons CC BY-SA 2.0

In our experience, paper prototyping can be an excellent way of exploring ideas (as long as you know its limits). In fact, our CEO even [wrote a piece for UXMag](#), where he explained how you get the most out of paper prototyping when you treat it more as an informal concepting exercise.

Draw out sketches for the sake of exploring your own ideas, then run a quick [hallway usability test](#) with 3-5 people. Afterwards, you can move to a digital platform for wireframing and higher-fidelity prototyping.

3. Process

Don't let the scissors and the pretty colors fool you – paper prototyping isn't just fun and games. Whether you're creating one for your team, for usability testing, or for a stakeholder presentation, your paper props should be both professional and functional.



Photo Credit: [Prittammets](#). [Creative Commons CC BY-SA 2.0](#)

The first thing to know about paper prototyping is that someone has to play the role of the “human computer.” [In his list of prototyping tips](#), Jim Ross, Senior UX Architect for **Infragistics**, suggests that one member's main and only part should be shuffling around the right screens at the right time. This isn't easy, but it's needed to best maintain the illusion.

Though the technique has been around since the 1980s and has had different variations, there are some standard practices to follow. Justin Mifsud, **UI Designer and Owner of UsabilityGeek**, [explains these steps in a post for his site](#):

- **Sketch each screen** – Each screen’s sketch should be creative – for example, a pull down menu can be folded underneath the paper at first – but also individual and separate from the rest.
- **Create user scenarios** – As recommended in *The Guide to UX Design Process & Documentation*, create a realistic scenario to run through, either to demonstrate for a presentation, or for your test-takers to try to figure out.
- **Rehearsal** – Run through the different scenarios until the “human computer” works like a real machine.
- **Presentation/Test** – Now you’re ready for the presentation or usability test. For presentations, just run through the scenario as rehearsed. For usability tests, stay on your toes and adjust the screens accordingly.

If you’d like to learn more, check out [this video](#) showing how you might explain a paper prototype to stakeholders. Then, get started with these [printable templates](#).

Wizard of Oz Prototypes

“Pay no attention to that man behind the curtain,” you tell your stakeholders right before you blow their mind with a stellar prototype that, technically, doesn’t exist. In our industry, this is called

the Wizard of Oz, an illusion of what the final product will look like and how it will work, but without it really looking or working like that... yet.

The Wizard of Oz is essentially a “fake it until you make it” strategy. If you are unable to realize your vision before a presentation for

whatever reason – not enough time, limited resources, etc. – the next best thing is to create a fabrication of your idea. This works especially well for presentations and testing early [minimum viable products](#). Of course, it may not be extremely polished (especially if you need a complex GUI), and you’ll eventually need to build the real thing.

1. Wizard of Oz for Presenting Prototypes

Perhaps the most successful proponents of the Wizard of Oz are David and Tom Kelley, Founder and Partner at **IDEO**. [In their love letter to prototypes](#), they tell the story of how the IDEO team wanted to pitch their proposed “smart car” features to an auto-maker. Because they didn’t have the time to develop that kind of technology so soon, they showed a video with a little movie magic to illustrate how the features *would* work. The automakers were able to see and envision the features in reality, and they loved it.

The pictures in this section show a similar story. Toy inventor Adam Skaate and gaming expert Coe Leta could not rally enough

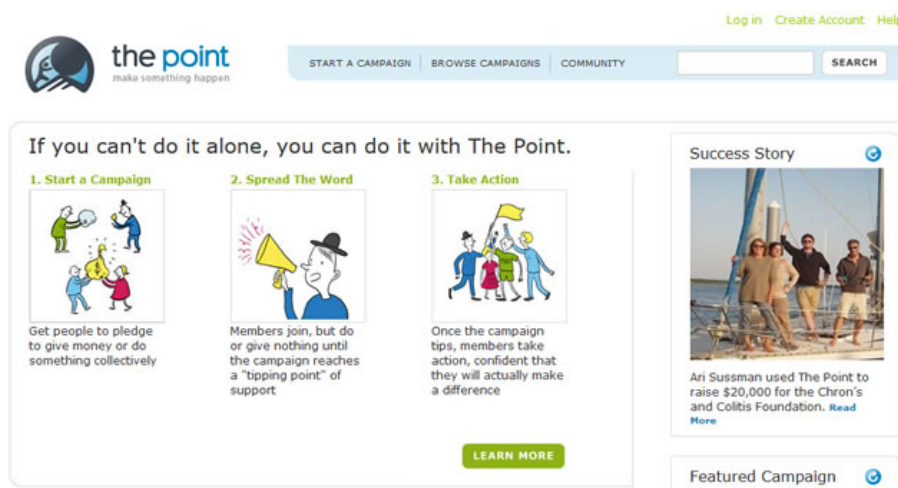
support for their *Elmo's Monster Maker app*, so they decided to show instead of tell. Only an hour before the pivotal conference call, they took a video of Skaate standing behind an oversized iPhone cutout in the role of the “monster,” with Leta moving her finger in the foreground as the user. In this way they showed how their app would function, and you can download *Elmo's Monster Maker* from the app store today.

2. Wizard of Oz for Testing Business Ideas (Minimum Viable Product)

If you wanted to build a standalone version that users can play with, you would need to follow a process similar to what [Groupon did for their minimum viable product](#): use existing technology, throw in some manpower, and get ready for manual work. In the early version of **Groupon** (which was just built on a WordPress site), founder Andrew Mason's team actually sent all the coupons to people *manually* via Apple Mail.

The Groupon prototype wasn't very pretty, but it let the team test the same functionality that is at the core of the current site. It is important to note, however, that this approach requires more human resources than other methods (like paper prototyping), and everyone needs to work in synch or you'll suffer from slow response times (which ruins the illusion).

The essence of a prototype is to give a precursory glimpse of the final product, and in this sense the Wizard of Oz can be a perfectly appropriate approach. Just avoid the temptation to overdo it



Source: *10 Massively Successful Minimum Viable Products*

and *never* promise more than you can deliver. To learn more, you can check out this [practical guide](#) by the **Stanford HCI Group** on creating and testing Wizard of Oz prototypes

Takeaway

As you prototype, remember that you don't need to limit yourself to only traditional methods. In our experience, we've found both methods to work well for quickly validating concepts, but they become less effective as your need for fidelity and functionality increases. Just remember to prototype smarter, not harder. Don't undercut the quality of the final product by avoiding an option that might have a higher learning curve, but don't push yourself into biting off more than you can prototype either.

In the next chapter, we'll look at some of the most popular methods of creating digital prototypes.

Digital Prototyping Methods and Tools

The different ways to build a ready-to-use digital prototype

Asking *What's the best way to prototype?* is like asking *What's the best way to make a website?* – there is no single “best” way. Each individual prototype, like each individual website, has its own styles, objectives, and strategies. What works well for a cloud CRM website might not work as well for an ecommerce business.

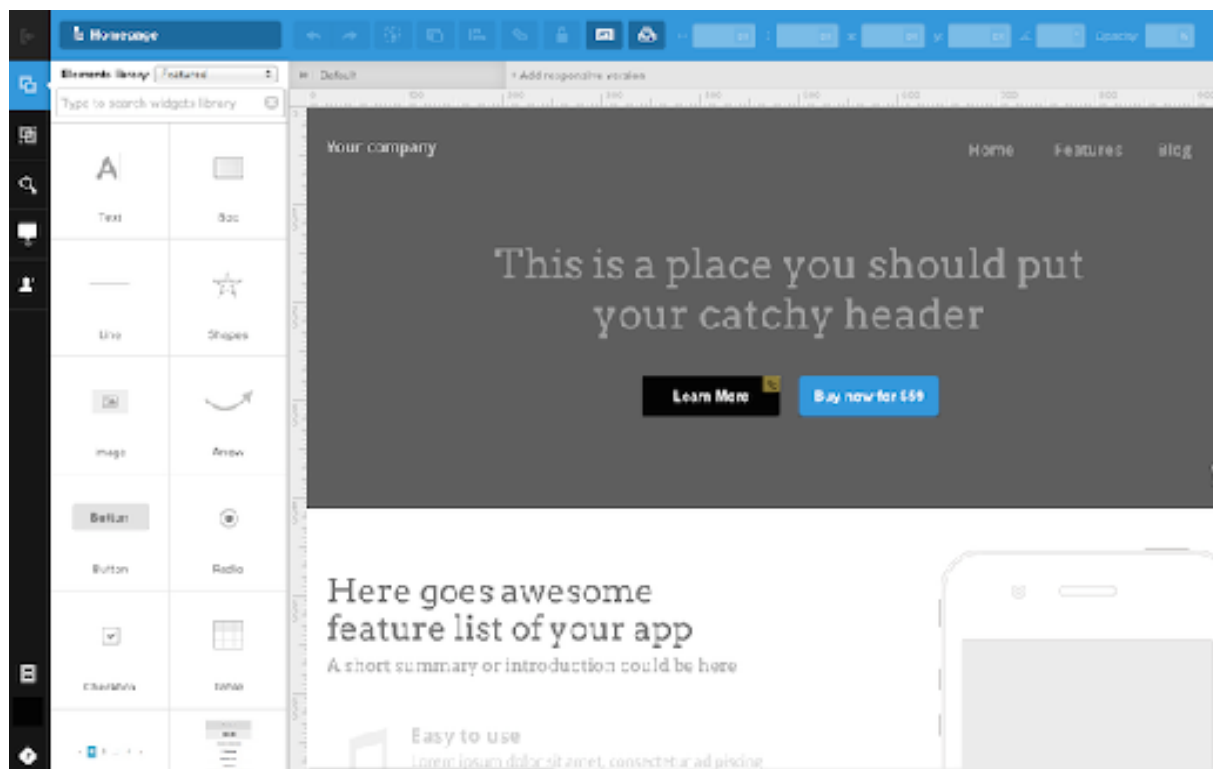


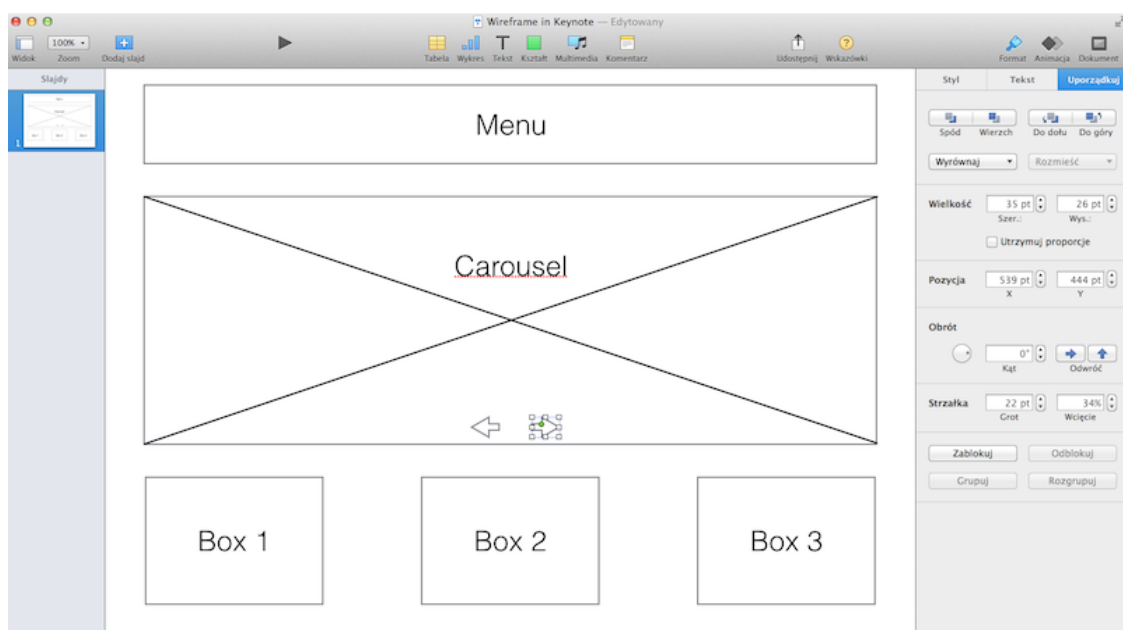
Photo Credit: www.uxpin.com

We'll explore 3 of the most common digital prototyping methods: presentation software (mostly for beginners), coded prototypes (intermediate to advanced), and specialized prototyping apps (for all levels of expertise).

Presentation Software

For starters, there's the traditional PowerPoint, a reliable business staple used for presentations for over two decades. If you're looking for a more modern alternative, [Keynote](#) is rising in popularity. Interestingly enough, **Google Ventures** mentioned [Keynote](#) as a [secret weapon](#) for their “design sprint” initiative.

Let's take a look at the pros and cons so you can make an educated decision.



Source: [4 Digital Wireframing Weapons](#)

1. Advantages of Presentation Software

Almost everyone has used presentation software before, so they're a quick way to start a simple prototype.

- Familiarity – You know the basics, and it's not that hard to learn more advanced features like animations, slide transitions, and linking slides for interactions.
- Basic element libraries – Thanks to simple wireframing libraries like [Keynotopia](#), you can quickly create low-fidelity wireframes and then link them together for a clickable prototype. You can also use master templates, and reuse slides or parts of slides as needed.
- Natural linear flow – The slideshow nature of these tools takes you through a sequential user flow, which forces you to think about the experience aside from visuals. For more advanced users, you can link slides in complex ways that go outside the linear progression. Most wireframing and prototyping apps can still be clunky for visualizing user flows, but UXPin, Flinto, and Invision do a great job.

2. Disadvantages of Presentation Software

Like we described in [The Guide to Wireframing](#), once you start playing around with advanced user flows and interactions, you've basically hit the limit of presentation software.

- Non-stock element libraries – It's not easy finding the right element libraries (if they exist at all). Unlike dedicated prototyping tools, presentation libraries aren't updated as frequently and their quality usually isn't as good.
- Limited collaboration – Most presentation software doesn't offer any collaboration (except for Google Presentation). The tradeoff though is that collaborative presentation software lacks interactivity, graphics manipulation, shapes, text, and color options that make them worthwhile for prototyping. If you want to collaborate without compromise, stick to a prototyping tool.
- Limited flow charting & user flows – As we discussed, you can communicate advanced user flows since you can link slides together for user flows that aren't purely linear. But it's not easy to do and the sitemaps aren't linked to the prototypes in a way that Axure or UXPin can do.
- Limited interactivity – Resourceful users can get pretty far if they use all the features in Keynote or Powerpoint. But once you think about how easy it is to add basic interactions with prototyping tools, and the sheer breadth of options available in the combinations of elements, content, views, and animations, it might just be easier to switch over to something specialized.

If you'd like to learn more, Keynotopia has some basic prototyping tutorials for [Powerpoint](#) and [Keynote](#).

Coded (HTML) Prototype

Furthering the discussion from the previous chapter, one of the biggest questions designers have about prototyping is whether or not to use code. This uncertainty stems from some designers' lack of comfort with coding: they either don't know how to do it, or don't like doing it. When faced with the more fun and intuitive method of using a prototyping tool or even sketching by hand, writing code can feel tedious.

1. Advantages & Disadvantages

Today there are more reasons than ever to start coding early, [as explained in a UX Booth article](#) by Andy Fitzgerald, Senior UX Architect at **Deloitte Digital**. The “I design it, you build it” waterfall mentality taken by designers in the past is becoming outdated as technology advances in large strides and collaboration becomes mandatory.

“When we address the architectural underpinnings of our content’s choreography early on, we ensure that we haven’t driven off course, and left our intent on the side of the road. What’s more, the benefits of HTML prototyping present themselves when we apply even the most basic of HTML’s elements. Creating a linear, semantic document calling out our navigation, header, teaser, aside, and paragraph elements forces us to think critically about how these elements relate to each other in context.”

There are a [few distinct advantages](#) of prototyping in code, mostly owing to the fact that you're starting the design in something that resembles the final form. Some advantages include:

- Platform agnostic – HTML prototypes work on any operating system, and nobody needs outside software to use it.
- Modularity – HTML is component-based, which can help with productivity.
- Low cost (aside from time) – There's many free HTML text editors, but you'll need to spend some time learning the language before it's helpful.
- Technical foundation for the product – Provided you're creating production-ready code (and not just throwaway for the sake of a quick prototype), you can end up saving time in development.

Coded prototypes can be built in a variety of ways like HTML (or even Python), depending on your preferences. Ash Maurya, Founder and CEO of **Spark59** and design speaker, [suggests using Ruby on Rails](#) because of the ease in which he can set placeholders for each page and link them together for navigation. However, the most popular code choice for prototyping will likely still be HTML.

Of course, the real consideration in deciding whether or not to use code in your prototype is your skill level. Not all designers have the

ability to code, so don't overextend yourself unless you're technically confident. Furthermore, diving straight into code may inhibit creativity – ask yourself how many interactions and page flows you can create with 30 minutes in a prototyping tool versus a code like HTML or Javascript.

2. Real-world Example – ZURB's Verify

One of the most successful interaction design companies **ZURB** – designer for sites such as eBay, Facebook, Photobucket, and NYSE – [stresses the importance of coding in the prototyping phase](#).

In their experience, coded prototyping (specifically rapid prototyping) is most valuable for usability testing, namely in finding what's wrong early on and correcting it. If the prototype fails the usability test, another round of revisions are in order; if it passes, then it's time to proceed to the next step.

Verify is the ZURB app that allows users to gather feedback on their own websites and apps. While this is a useful app for designers and developers, it is [the hybrid design process for the app](#) that we're going to focus on here:

- Sketching – ZURB likes sketching because it's a quick and effective way of generating ideas. In the first week, they went through around 80 sketched screens.

- **Front-end Prototyping** – Then they immediately dove into creating the front-end, which they accomplished in 2 weeks. They credit the use of a global stylesheet with grids (global.css file), a preexisting visual style to work from, and no delusions that the first version would be rough.
- **Engineering** – Two more weeks after the first pass, the team had a working version of their app, though rough. Working with engineering helped curb the glaring technical problems.

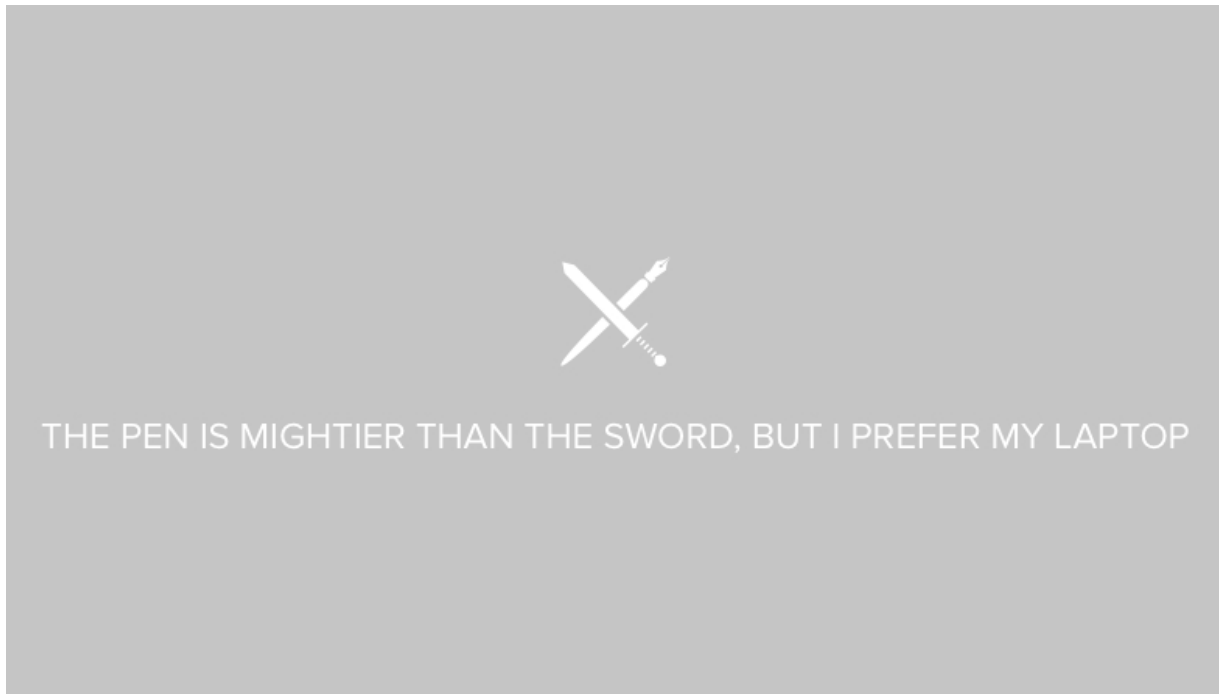
After creating the first version of the app in a little over a month, ZURB used the time they saved for final refinements.

Prototyping Software & Apps

Eager to dive straight into a computer program that's an actual representation of your idea? The beauty of prototyping software and apps is that it's specifically designed for this purpose, so they provide the perfect balance between functionality, learning curve, and ease-of-use. Both beginner and veteran designers use specialized tools like the ones below – beginners for the ease-of-use, and veterans for the controls crafted to their particular needs.

These tools vary in their capabilities, with some being better attuned to certain situations than others, so it's best to find the one best suited to your needs. To start on your search, you can check

out tools like [UXPin](#), Invision, MockFlow, JustInMind, Axure, Omnigraffle, JustProto, Flinto, or Marvel.



Source: *4 Digital Wireframing Weapons*

1. Advantages of Prototyping Tools

As described in *The Guide to Wireframing*, these tools have an advantage in that they are built specifically for wireframing and prototyping. Once you learn the basic features, you may find it even faster to prototype with these versus traditional methods like paper prototyping.

- Speed – From speaking to our own customers, we’ve found that power users can work in specialized tools even faster than paper prototyping because they can create, copy, and [produce advanced interactions](#) with just a few mouse clicks.

- Element libraries – While tools like Invision work great for quick clickable prototypes that link together multiple screens with simple interactions (like click and hover), other tools like JustInMind, JustProto, and UXPin come with built-in element libraries (and let you create your own for repeated use).
- Advanced flowcharting & user flows – Flow and functionality are the most important aspects of prototyping, and most tools come with these features built in. Most tools allow you to generate sitemaps as you create new screens for prototyping, and let you see these screens laid side-by-side so you can navigate them.
- Built-in collaboration – First, make sure the tool you select has basic commenting and resolving comments capabilities. Secondly, the tool needs to be able to allow [collaborative editing and sharing of prototypes](#) (as links). Finally, revision history and cloud storage simplifies your workflow by making it device agnostic. While UXPin and Invision are the most robust, JustInMind, JustProto, Flinto, and Marvel also have some level of collaboration.
- Streamlined presentation – This can mean exporting to PDF, a [built-in presentation mode](#), or exporting to a web or mobile app for a real prototype experience. Most tools will export to PDF, while only a few like Mockflow export to presentation software, and only a few like UXPin let you export the file as an HTML-based prototype. Some tools like UXPin and Invision also have presentation and screen-sharing capabilities

2. Disadvantages of Prototyping Tools

All prototyping tools require a little bit of time to become familiar, but they can be well worth the effort. Nonetheless, let's make sure you consider all aspects before deciding.

- Lack of familiarity – Like anything in life, if you've never used it, you'll need to learn it. But a [handful of tools like UXPin and Invision](#) are known for usability and ease of learning.
- Limited fidelity & functionality – With the exception of tools like UXPin and Axure, most prototyping tools are low fidelity and/or low functionality. For example, Invision is used mostly for quick clickable prototypes, so it's limited to two interactions (hover and click) and you can't create anything high-fidelity in the app (only import from elsewhere). Invision and Flinto also let you import files from Photoshop and Sketch, but the files are flattened so you lose your layers (UXPin [doesn't flatten anything](#)).

Takeaway

To take a stab at the question we posed at the beginning of the last chapter, *What is the best way to prototype?*, we'll go with the answer *Whichever way works best for you*. That's not a euphemism for *Whichever is easiest* or *Whichever requires the least work*. The best way to prototype is the way that, for you, will produce the best results.

You must consider the specifics of both the product and the design team. Which platforms and methods are the designers and developers best at or most comfortable with? What are the specific goals, time frames, and limitations involved with the product you're working with. Knowing where you're going is the most important part – we're just explaining the different routes to get there.

While we previously touched on prototyping's role in usability testing, in the next chapter we'll delve deeper, explaining how to build and test for usability with your prototype.

9 Prototyping Best Practices

Practical tips for any kind of prototype

Even though there's multiple fidelities and forms of prototypes, they all share a lot of common ground. In this chapter, we've distilled years of prototyping experience from some of the best advice we've read, plus added some spice from our own trial and error.

What we came up with are 9 guidelines that can help you prototype, no matter what method you choose.

1. Don't Neglect Your Content

Lorem ipsum content is fine for your first low-fidelity prototype since it gives you a rough structure to play with, but ditch the placeholder text once you move on to mid and high fidelities.

As dictated by the [content-first design](#) philosophy, your design must prioritize content because that is what users care about most. Users don't want to look at your site, they want to use it to accomplish

their goals. And the only way they can do that is by interacting with your content.



Source: [UXPin](#)

For example, take a look at the above layout. It might look fine right now, but what if you drop in your final copy afterwards and you find that your entire layout must shift? That may then affect your visual hierarchy, which of course affects how someone would interpret and interact with your prototype.

When possible, try to at least get a head start on the content before the design is in a near-finished state. Even playing around with rough text is better than defaulting to Lorem Ipsum content and assuming that the real content will flow into your prototype as fluidly as water.

2. Involve the Users (Participatory Design)

Participatory design builds user input into the actual design process. This can be done any number of ways, including usability tests, brainstorming sessions, paper prototyping exercises, etc.

The idea is that the users offer insight into how to improve the UX that the designers wouldn't have thought up on their own. One of the greatest misconceptions about participatory design is that the designer must hand over the reins to inexperienced amateurs, a common concern that is dispelled by **Beaudouin-Lafon & Mackay** in *Prototyping Tools and Techniques*.



Photo credit: [Samuel Mann. Paper Based Prototyping. Creative Commons.](#)

Instead, participatory design is more of an observant collaboration – by involving users in the creation process, designers will naturally see what features are important and where the gaps lie. This is different than usability testing because you are involving the user right in the beginning stages of ideation and concepting, not purely for the sake of validation. The end result is not something that's the product of one side or the other, but both.

If you'd like to learn more about participatory design – including different types, guidelines, and examples – check out our [The Guide to Usability Testing](#) and Frog Design's [excellent tutorial](#).

3. Focus on Flows and User Scenarios

Prototypes don't need to be pretty, but they need to work. No matter the fidelity, prototypes must have a degree of functionality and interactivity. Morgan Brown's [piece for Smashing Magazine](#) explains the benefits of designing a website for the user flow, not for individual pages.

That's not always easy, especially when you work hour after hour hammering out details on one specific page. Luckily, creating personas and user stories will help you focus on the journey and not just the steps. Remember whom you're designing for and recall their behavior in any number of scenarios.

For more information, see Chapter 4 [The Guide to UX Design Process and Documentation](#).

4. Keep Clicking Simple

Related to the previous tip about flow, less clicking means less friction, which means better flow.

The 3-Click Rule may have been around for so long that it's taken as truth, but it has been called into question, specifically by David Hamill in this [UX Booth piece](#). It was once believed that a web page should not be more than 3 clicks away from any other page on that site.

Over a decade ago, before the rule was even popularized, Joshua Porter [called it “well-intentioned but misguided.”](#) The consensus is that the rules for user flow don't have to be as rigid as “3 clicks or else,” but each click must feel as effortless as possible.

As you can see in the below example from [Web UI Patterns 2014](#), the airline booking process can't really be completed in 3 clicks or less. But when you can't shorten the line, it's not a bad idea to make the wait more pleasant. **Virgin America's** stepped form design and simple aesthetics make the process feel much easier than it is.

The screenshot shows the Virgin America flight booking interface. At the top, there is a navigation bar with a back arrow, the Virgin America logo, and flight details: SFO → LAX, LAX → SFO, 1 Adult, and a price of \$0.00. Below this, the main heading is "Who's flying?". There are three input fields: "Adults" with the number 1, "Children (Age 2-14)" with 0, and "Lap Infants (Age 0-2)" with 0. Each field has a plus and minus button. Below the "Adults" field is a link for "Have special requests?". At the bottom, there is a purple button labeled "CONTINUE WITH 1 ADULT".

Source: [Virgin America](#)

Any kind of fanatical adherence to strict guidelines like this is inadvisable – plenty of successful sites require more than 3 clicks to fully appreciate.

For example, let's look at these series of clicks during when signing into Chase to redeem my rewards for cash:

1. I visit the homepage. Navigation options include *Products & Services*, *News & Stories*, and *Log In or Enroll*. I see the call-to-action to log in and type in my info.

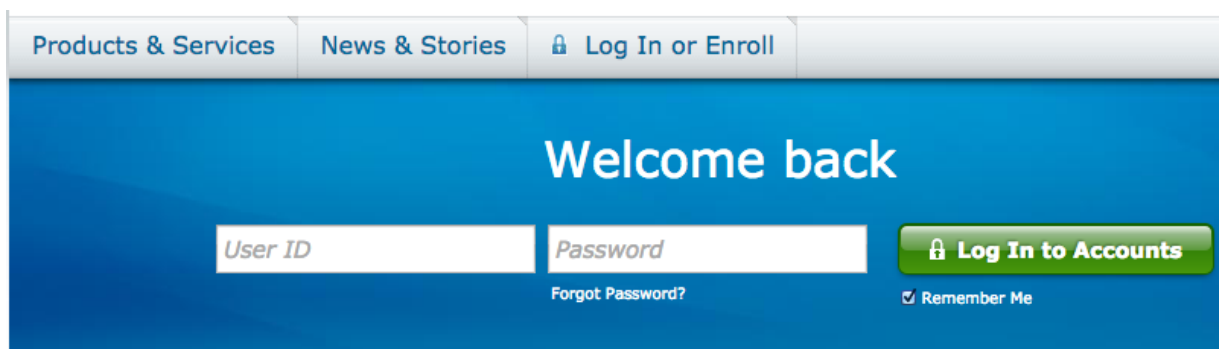


Photo credit: Chase

2. My account page loads. I see a call to action for *Ultimate Rewards* and I click it.

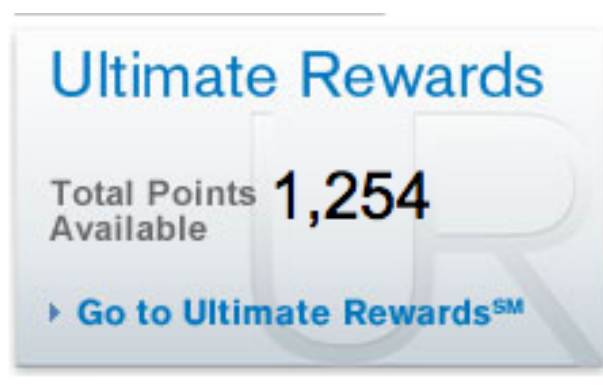


Photo credit: Chase

3. The rewards page loads. I see options to either Use Points or Earn Points. I click on Use Points.

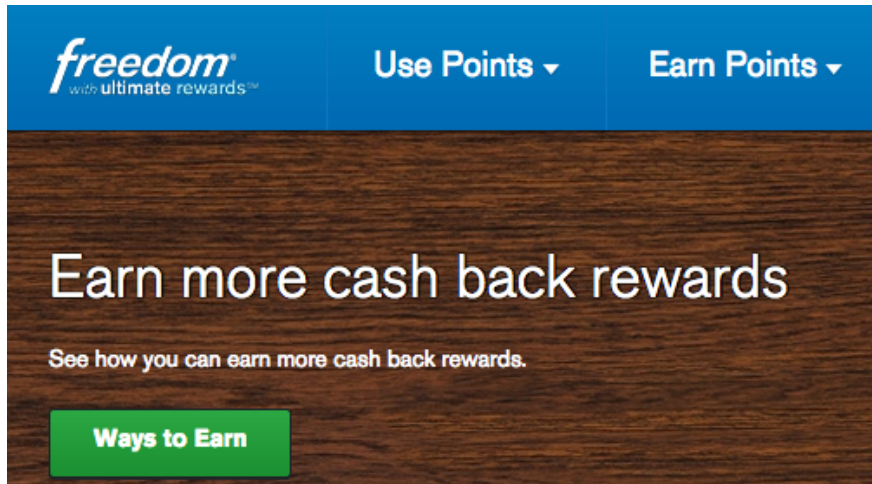


Photo credit: Chase

4. Once the point redemption page loads, I can see how many points are available and how many I can redeem. In this case, I know that I still need to rack up more points.

You need a minimum of 2,000
points to redeem.



Keep earning points so you can redeem for cash back!

Photo credit: Chase

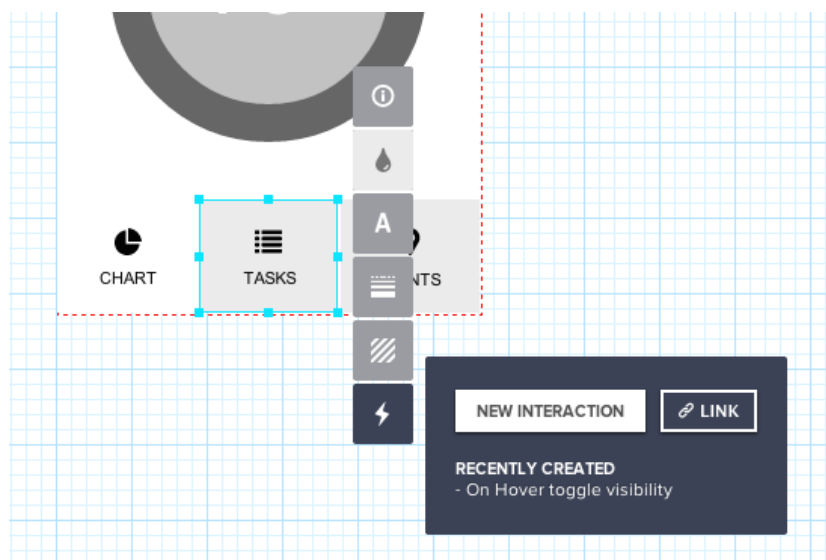
More than 3 clicks are required, but each of the clicks requires very little effort. Each click also moves the user forward a step on their path to the goal. Now, if you were adamant about the 3-click rule, you might make one of the top-level navigation labels “Use Your

Rewards”. The clicks are certainly reduced, but this wide-and-flat strategy will eventually present too many items to sift through at once. Usability is therefore sacrificed for the sake of a shorter click path.

We want to emphasize the spirit behind the 3-click rule: clicking should be as simple and organic as possible. Make sure that the time users spend on site isn't just minimal, but worthwhile.

5. Don't Neglect Animations

Elements like animations can be put off to the end of the design process – which is acceptable, so long as you don't forget about them. As we mention in *Web UI Best Practices*, these may seem extraneous, but certainly are not; they serve the function of revealing content in a charming and delightful manner.



Source: *Advanced Interactions*

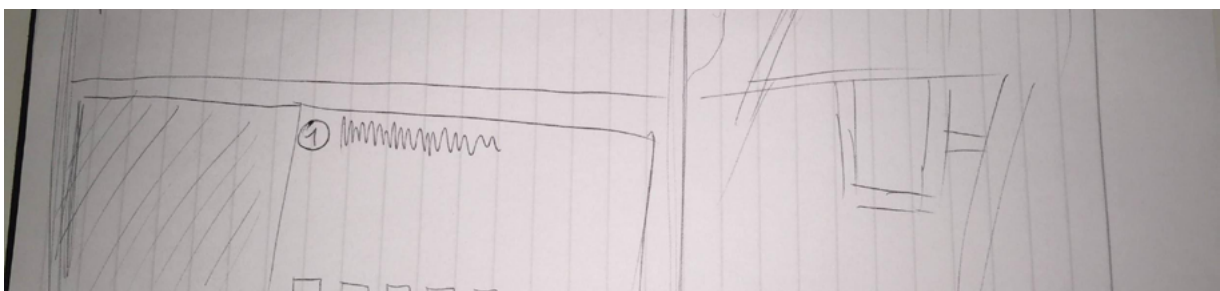
While animations might be simplified in the prototype, or completely absent in a lo-fi or paper prototype, the important thing is that they're taken into account for later. Knowing where and how your animations will be implemented – whenever you get to them – will contribute to your understanding of the user experience as a whole. These will have to be developed seriously at some point, sooner if you're doing a hi-fi prototype, so at least keep them in the back of your mind.

6. Sketching: The Prototype for the Prototype

Just like the prototype can be a simplified outline of a website, a sketch can be a simplified outline of the prototype. The traditional design process goes something like:

Sketching ▶ Wireframing ▶ Mockups ▶ Prototypes ▶ Development

... but often varies based on the project's needs, resources, and limitations. No matter what method you use, though, starting with a rough sketch is a quick and inexpensive way to help organize your thoughts and turn abstract ideas into something concrete. Mike Rohde talks in-depth about it [in this article for *A List Apart*](#).



Source: *User Testing & Design*

This was actually our first step [when we redesigned Yelp as an exercise](#). If you have concerns about your artistic ability – don't! As you can see by the photo of our first sketch above, it's not about looking pretty. The goal is organizing your thoughts, exploring concepts, and creating a basic structure.

Alternatively, you could start in a wireframe and build your basic structure that way, beginning digitally right away. If you're using a design software like [UXPin](#), you can even add higher fidelity, interactive elements, and animations to that wireframe and build your mockup and prototype from the same document.

7. Don't Let Coding Hold You Back

If you're a designer first and foremost, code might not be your strong suit, but don't let that stop you from prototyping. While coding prototypes does have its benefits (as we discussed earlier), it's not a requirement. There are other plenty of other options: paper prototyping, the Wizard of Oz method, Keynote, and specialized prototyping tools.

Obviously, a lo-fi prototype can let you get away with minimal or no coding. A paper prototype requires no coding whatsoever, and could be an easy way to clearly express your ideas to a developer who knows code better than you do.

You can still build a hi-fi prototype without code, too. Tools like [UXPin](#), Invision, and Axure are designed to facilitate the design process without bogging you down with additional concerns like coding. The point is, don't use an aversion to code as a reason for not prototyping.

8. Use Prototypes for Usability Tests

As we discussed in the last chapter, testing prototypes lets you find and gauge problems early and collects valuable feedback for iteration before it's too late.

38 Responses + 6 Abandoned

Task 1: You're going out for dinner in San Francisco and want to find out about some good nightclubs for afterwards.

Clicks: 38
Skips: 0
Average time taken: 13 seconds

Find Near

Home About Me Write a Review Find Friends Messages Talk Sign Up Log In

Browsing San Francisco, CA Businesses Showing 1-10 of 9013

Active Life
Arts & Entertainment
Automotive
More categories

Beauty & Spas
Education
Event Planning & Servi...

Financial Services
Food
Health & Medical

Home Services
Hotels & Travel
Local Services

Nightlife
Pets
Professional Services

Real Estate
Restaurants
Shopping

Hide Filters

Sort By
Best Match
Highest Rated
Most Reviewed

Neighborhoods
 Financial District
 SoMa
 Union Square
 Mission
More Neighborhoods

Distance
Bird's-eye View
Driving (5 mi.)
Biking (2 mi.)
Walking (1 mi.)
Within 4 blocks

Features
 Offering a Deal
 Open Now 12:33 PM
 Sells Gift Certificates
 Free Wi-Fi
More Features

Photo credit: [Optimal Workshop](#)

The rapid prototyping strategy handles testing usability well. Basic prototypes are built quickly, tested, and then scrapped as the feedback is iterated into the next version.

As our CEO **Marcin Treder** points out in [an article for UX Magazine](#), rapid prototyping greatly reduces after-launch fixes. He adds that, by altering the scope to only the major issues, you can still finish the project in a quick and timely manner without sacrificing quality.

9. Focus on the Right Fidelity, Not the Best Fidelity

Don't forget that prototyping is a means to an end, not the end itself. It can be tempting to get caught up in making the perfect prototype, but that only postpones development on the real product.

While you don't want to rush the prototyping phase – iterating and refining should come naturally – you also don't want to hold on too long. Some of the minor hang-ups can be perfected during development, so only prototype the serious concerns.

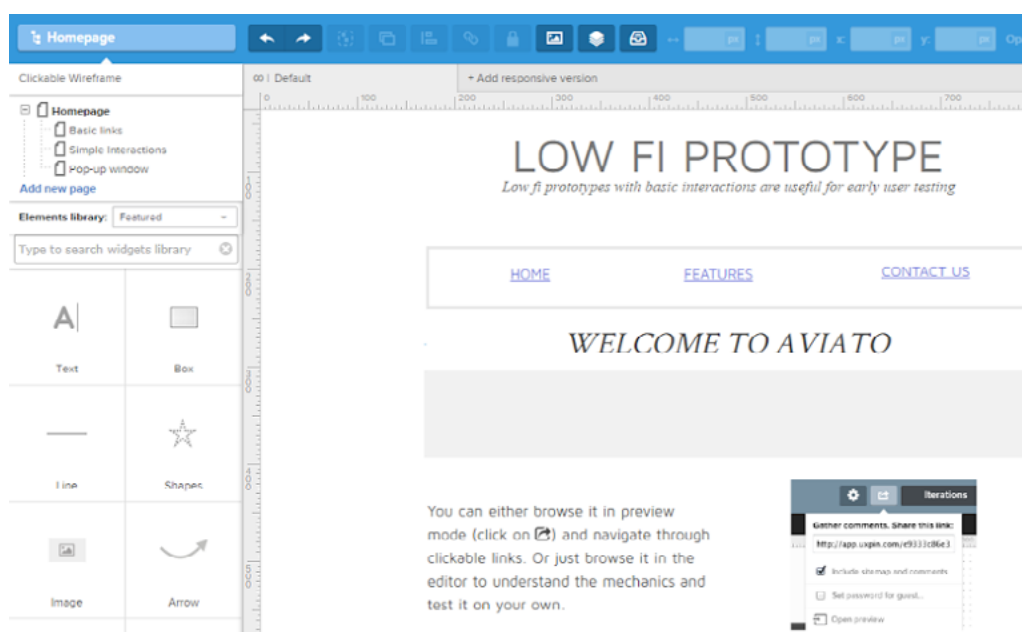


Photo credit: [UXPin](#)

As interaction designer Ivana McConnell [explained in our blog](#), overbuilding a prototype is just asking for the wrong type of feedback. In her case, she jumped headfirst into a high-fidelity prototype when a low-fidelity prototype was more appropriate. She was still able to get the right feedback from the client, but it took much more guiding and time than was necessary.

The piece in *Smashing Magazine* by **Lyndon Cerejo** cites the famous [80-20 rule](#), stating that 80% of the effects come from 20% of the causes. In the case of UX design, the rule states that 80% of your users' interaction usually involves only 20% of the functionality available – your main links, menus, etc. Therefore, if the goal is efficiency, focus on just this top fifth to address most of the product's usability.

On the other hand, if you anticipate more than a few animations and interactions on your site, then you'll definitely want to create at least mid-level functionality in your prototype.

Nevertheless, build the appropriate fidelity, then make sure you guide stakeholders by explaining precisely which core features are included (and why) and on which aspects of the design you seek feedback.

Takeaway

Maybe these tips are new to you. Maybe you've heard them over and over by this point. In either case, the important thing is that they're fresh in your head during the prototyping process, because there's a reason that this is the advice that designers find most helpful.

In the next two chapters, we'll explore how you can start prototyping with your existing Photoshop and Sketch files.

Wireframing & Prototyping: The Past, Present, and Future

Where These Design Methods Came From, and Where They're Going

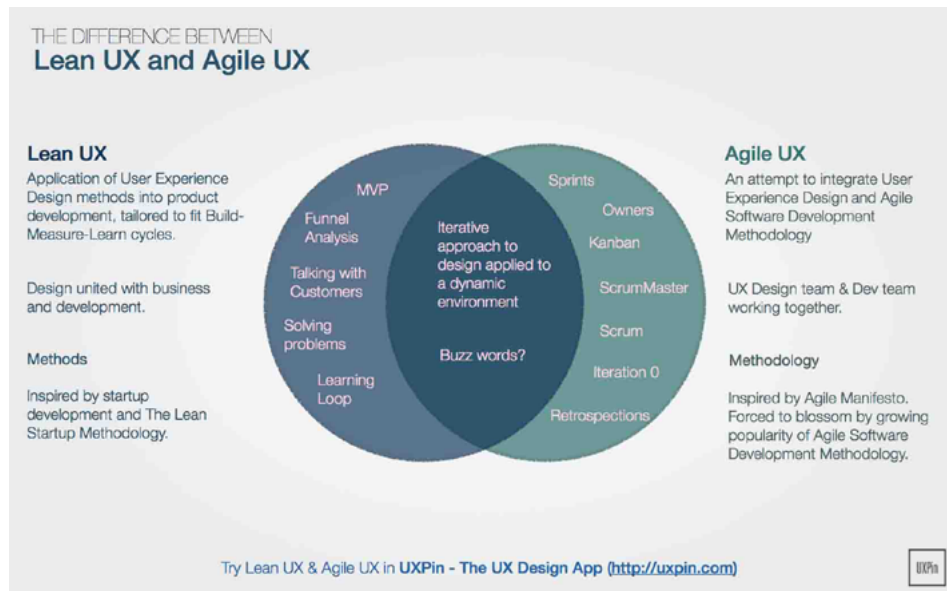
Knowing the origins of wireframing and prototyping will help you put everything into a historical context, allowing you to comprehend the practice and predict how it will evolve in the future. This chapter will be somewhat of a modern history lesson, covering digital UX design in its past, present, and future.

Present: The Current State of Design

At the moment, two of the most popular design methodologies are **Lean UX** and **Agile UX**. Both may sound similar, but their approaches to the design process differ in terms of scope. Lean UX is more of a business approach, while Agile UX is more of a project approach.

The old way of doing things – the Waterfall method where designers hand off a fully fleshed-out comp to a developer and say “good luck” – is coming to an end. [In a post heralding the end of the “PSD-era,”](#) Brad Frost, Web Designer, Blogger, and Speaker, writes that

the Waterfall method makes less and less sense given the variety of devices on the market.



Source: *Lean UX vs. Agile UX – Is There a Difference?*

In the vacuum left by the Waterfall method, two alternative schools of thought are filling its place. While both agree that the process should be more collaborative, there are subtle differences between the two. In an article for *Web Designer Depot*, UXPin CEO Marcin Treder takes a brief look at each. As you'll see below, the two processes are more complementary than they are conflicting.

1. Agile UX

A methodology following the *Agile Manifesto*, Agile UX is essentially a movement to unify designers and developers into the Agile development process. Its supporters follow these guidelines:

- People, personal interactions, and teamwork are valued over processes and tools.

- Producing working software is a better goal than comprehensive documentation.
- Working with the customer is more important than contract negotiation.
- Adapt to change instead of following a rigid plan.

At the moment, this line of thinking is “considered to be the gold standard for the development of digital products.”

While the Agile methodology doesn’t directly address UX design, it does mandate a paradigm shift in the way we collaborate with people on design projects. As discussed in our free ebook [Web UI Best Practices](#), Agile UX gives us methods like the [Design Studio](#) and [Cross-Functional Pairing](#) that help designers replace documentation with fast and meaningful interaction. Agile UX replaces the treacherously long waterfall phases with collaborative sprints involving designers, developers, and product managers.

2. Lean UX

Originating from the [Lean Startup](#) methodology, the Lean UX school of thought believes that a company must release a product that fills a previously researched niche, and it must do it as quickly as possible (with minimized waste) in order to survive. While Agile is more focused on getting the product to market, Lean UX shows us that shipping the product is only the start.

Some of Lean UX's **core principles** include:

- Validating hypotheses with customers (“getting out of the building”)
- Releasing **minimum viable products** that solve user problems
- Rapid prototyping (“learning loops”) done collaboratively
- Nimble prototypes over heavy wireframes and spec sheets

These strategies have been developed as a type of lifecraft to keep the product afloat in a time of flooded markets.

3. How They Fit Together

In a nutshell, Agile UX is more concerned with the “how” of product design while Lean UX focuses on the “why.” While Agile helps UX designers revamp outdated methods of designing and collaborating, Lean UX gives us new ways to research the product and measure quality.

Agile UX is the ‘how’ of product design while Lean UX focuses on the ‘why’.



Rather than conducting research before you build the prototype, Lean UX advises that you continuously research metrics gathered through methods like A/B testing, customer interviews, and usability testing.

Because Lean UX is an approach to the overall business strategy, you can still create your product using Agile processes. Brainstorm with your team, sketch out concepts and requirements, build quick prototypes, and start testing them. This is exactly what Spotify does, as we pointed out in *The Guide to Minimum Viable Products*.

There's a lot of common ground to be shared between the two: both champion collaboration over documentation, and both emphasize short sprints over ambitious development timelines.

In fact, Jeff Gothelf, the godfather of Lean UX, even says that Lean UX is “inspired by Lean and Agile development theories.” The bottom line is that it's not really that important whether you choose Agile or Lean UX, but that the “work smarter, not *longer*” approach behind both methodologies are driving today's rapid prototyping movement.

Present: The Current State of Prototyping

While most people remain grounded supporters of prototyping, lately more and more designers are doubting the value of static wireframing. What we're seeing now, as a result, is more people merging wireframes and prototypes as a way to bypass the wireframing stage and get started on interaction design earlier.

As we mentioned above, this is in large part due to the recent rise of design tools that let you go from wireframe to prototype with just a few clicks. [In this Quora discussion thread about wireframing](#), a handful of experts all mention the benefits of “interactive wireframes” (lo-fi prototypes) over static wireframes. Their reasons vary, but all seem to echo this “two birds with one stone” approach to combining wireframes and prototypes.

With that in mind, the current state of prototyping favors practicality over pixel perfection. In the near future, we’ll likely see lo-fi prototypes replacing wireframes as a design milestone, and hi-fi prototypes continuing to be used for usability testing and presentations.

The current state of prototyping favors practicality over pixel perfection.



Past: A Prototyping Timeline

You’ve got to love a comprehensive timeline that begins in 1970. The history of software development may not be long, but it is dense, with the future being written every day. Here we’ve outlined the crucial events that shaped the birth of the information age – and where prototyping fits into it all.

1970 ▶ The Waterfall method dominates software development.

1975 ▶ The importance of information architecture is recognized

and developed.

1980 ▶ The first, basic digital prototypes – resembling flow charts – arise thanks to visual programming advances.

1985 ▶ Paper prototyping is integrated for usability testing and concept sharing.

1985 ▶ The Waterfall method is modified to incorporate **Iterative and Incremental Development (IID)**.

1986 ▶ The first visualization and design software is developed.

1986 ▶ Adobe Illustrator

1987 ▶ MS PowerPoint

1990 ▶ Adobe Photoshop

1992 ▶ MS Visio (originally Shapeware; acquired by MS in 2000)

1988 ▶ **The Spiral Model** of software development is popularized.

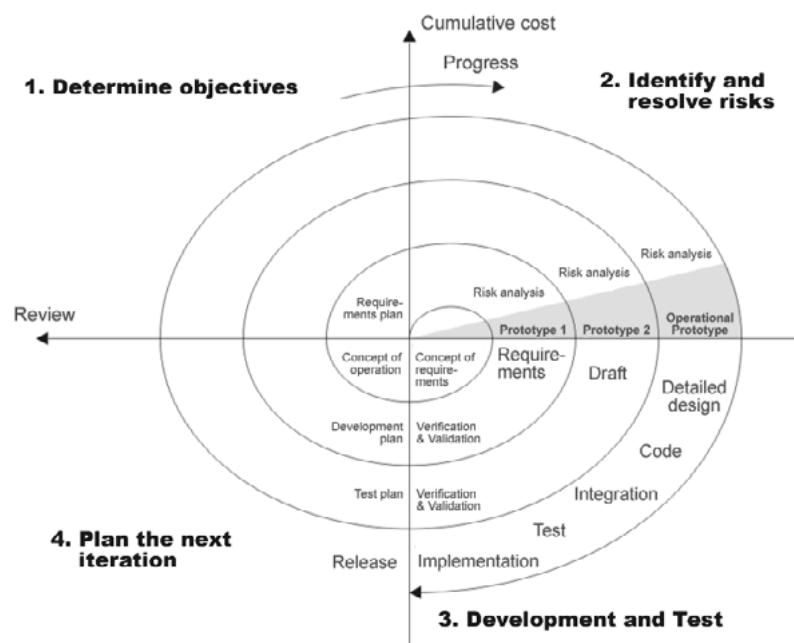


Photo Credit: "Spiral model". Wikimedia. Creative Commons.

1991 ▶ IBM introduces the **Rapid Application Development (RAD)** method of software development.

1995 ▶ Layouts become more comprehensive to showcase page or UI designs.

2000 ▶ Prototyping software emerges to meet the growing need.

2000 ▶ Omnigraffle

2003 ▶ Axure

2003 ▶ iRise

2001 ▶ The *Agile Manifesto* is released, birthing the later [Agile UX](#) movement.

2005 ▶ Web-based (SaaS) prototyping becomes more common, opening the door for low-fidelity wireframing apps, which later integrate collaboration and product management.

2005 ▶ MockupScreens

2006 ▶ Gliffy

2007 ▶ Jumpchart

2008 ▶ Balsamiq

2008 ▶ Protoshare

2008 ▶ Justinmind

2006 ▶ Cowboy coding, the code-and-fix method of software development, is popularized by Google's "20% time" policy, which allowed programmers to work on whatever else they wanted for a portion of their time. Since then, hackathons have exploded within and outside of organizations, mirroring the same software development methodologies to get a quick prototype out the door.

2008 ▶ Competition among startups leads to the Lean UX movement.

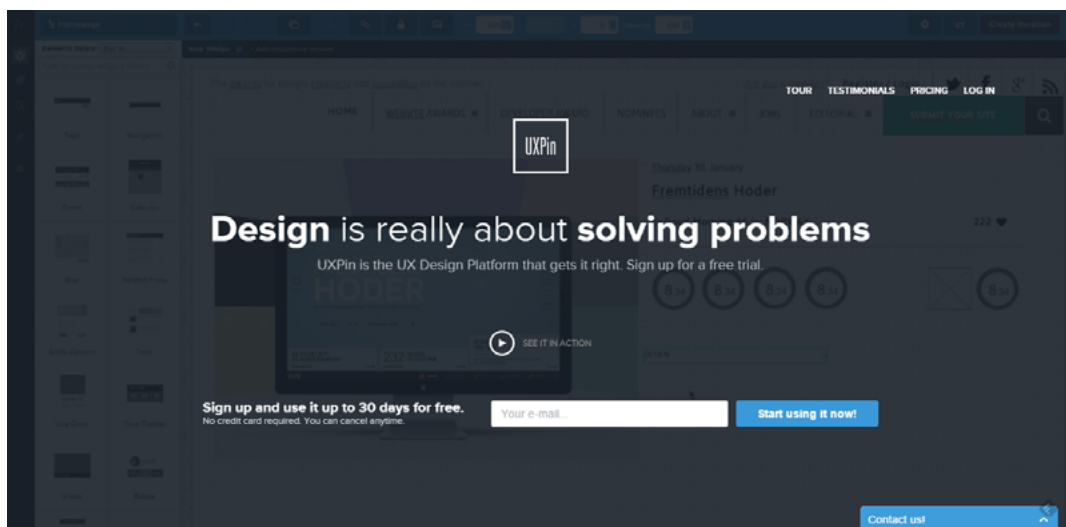
2010 ▶ Technological advances enable high-fidelity SaaS prototyping without coding.

2011 ▶ UXPin (paper, mobile, web, responsive)

- 2011 ▶ InVision (mobile, web)
- 2012 ▶ Flinto (mobile)
- 2012 ▶ POP - Prototyping on Paper (mobile)
- 2013 ▶ Marvel (mobile, web)

Future: The Age of Prototyping

Our speculation of the future begins where our discussion of the present ended: wireframes and prototypes merging together in a way where lo-fi prototypes take over the outlining and organizational purposes of wireframes.



Source: 2014 – *The Year of Interactive Design Tools*

We've read that 2014 has been referred to by some as [the Year of Interaction Design Tools](#). In the post, Emelyn Baker, Designer at **Bloc** explains the “blossoming” community of prototyping tools and lists the ones available today, including our own [UXPin](#). The surprising number of tools on the market – most of which blur the line be-

tween wireframes, mockups, and prototypes – is just a glimpse of the interactive future.

The old ways are only getting older, and static designs and the Waterfall method are slowly becoming things of the past (in fact, even POP is digitizing paper prototyping). The new wave of prototyping tools brings with it a new era of democratized design. Specialized prototyping apps have popularized two critical updates to the design process:

- **Rapid Prototyping** – The future of prototyping will see more of the *prototype, refine, repeat* method for improved functionality earlier in the design process. Considering that some prototyping tools support the full spectrum of design (from static to interactive), there is now little excuse not to practice rapid prototyping.
- **Collaboration without email** – The communication features and presentation modes on many new prototyping tools narrows the gaps between designers, developers, and stakeholders. As more people recognize the flaws in the Waterfall method, collaboration and even [participatory design](#) will only become more widespread thanks to technology.

When discussing the future of prototyping, another hot topic is [“microinteractions.”](#) To summarize, a microinteraction is a use case with a single goal – for example, unlocking a smartphone – and the trigger, rules, and feedback involved in that one task. Microinteractions shift the focus from the UX of the whole product to the UX

of individual actions and moments, a level of detail made possible in part thanks to specialized prototyping tools.

The logic behind microinteractions is that the UI details themselves (rather than the sum of all details) make the difference between a product you tolerate and one that you love. In fact, John Pavlus, Design Writer at **FastCo Design**, calls microinteractions the “[future of UX](#).” Microinteractions are a magnifying glass for interaction design, bringing into focus the delightful moments that create unforgettable user experiences. As more focus is placed on the micro-moments of how products look and feel, prototyping will become mandatory for perfecting these small, delightful details.

Microinteractions are a magnifying glass for interaction design.



tweet this

Takeaway

We must evolve or become extinct, so survival depends on reading the signals and adapting early. Wireframing still has its place, but its function now as a blueprint for prototyping is different than its function 5 years ago as a formal design deliverable, and that function was different than 10 years ago when wireframes were mostly for specifying products.

The practice of prototyping, too, has changed from being a small building block of production-ready code to a fast way of crafting and testing experiences. Through iteration, we have overcome the fear of throwaway design and code. So, in that spirit, embrace prototyping for what it is and what it's becoming. Embrace collaboration, early interactivity, and flexible iterations.

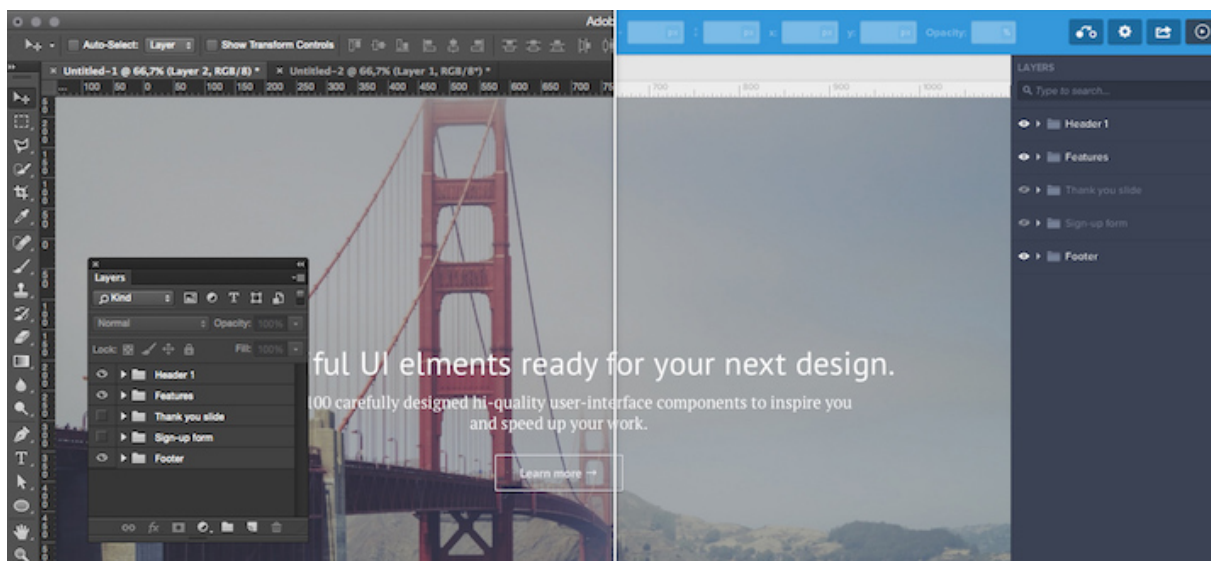
Now, let's start prototyping.

Creating Interactive Prototypes from Photoshop Files

Closing the gap between static and interactive design

So you're looking for a way to turn static a Photoshop mockup into an interactive prototype. But you want to do it quickly, without code, while preserving all of your layers. Beyond that, you need to gather feedback from your stakeholders.

This tutorial shows you how to do all that with UXPin. Once you've imported your Photoshop file into [UXPin](#), it's easy to add interactions & animations and then use our [Live Presentation tool](#) to host



Source: [UXPin](#)

a screenshare meeting to unveil your new design. There are currently 11 triggers and 20 element actions, allowing for many custom advanced interactions. Take a look at the [overview video](#) and then check out our tutorial below or [on our blog](#).

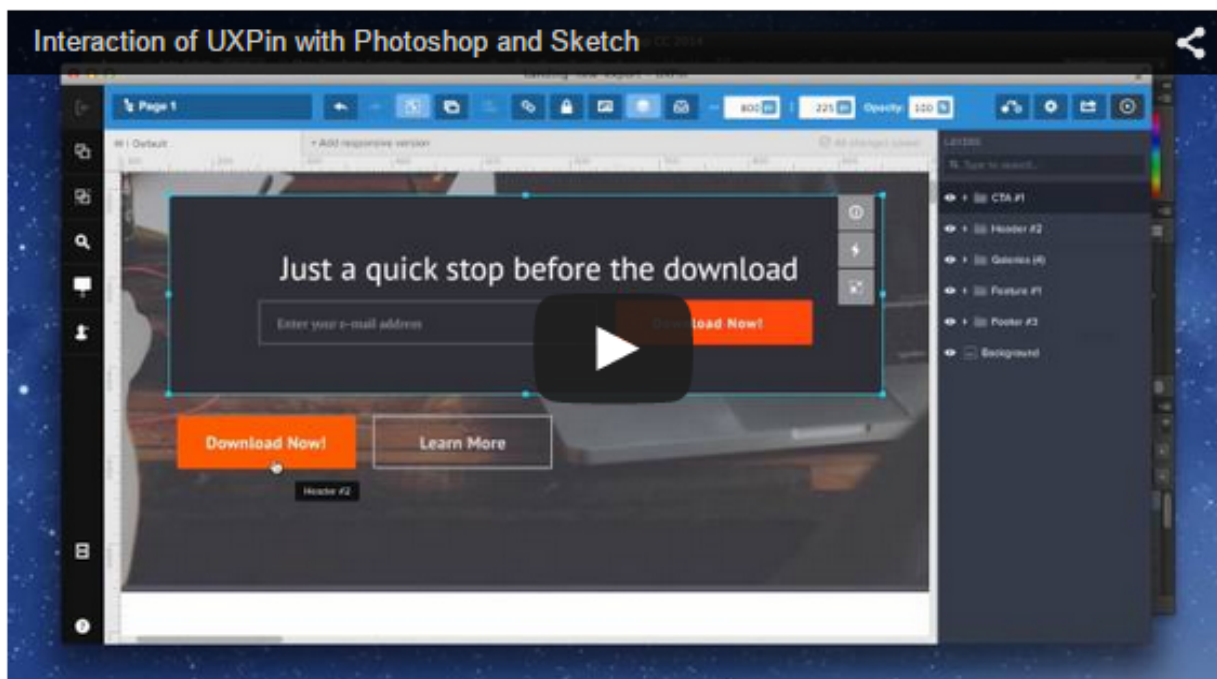
Before we start off with the tutorial, I'd like to show you the result we are going after – check out [the preview](#).

As you can see, this prototype includes several interactive points with animations. If you want to learn about specific parts, just jump to sections listed below.

Index

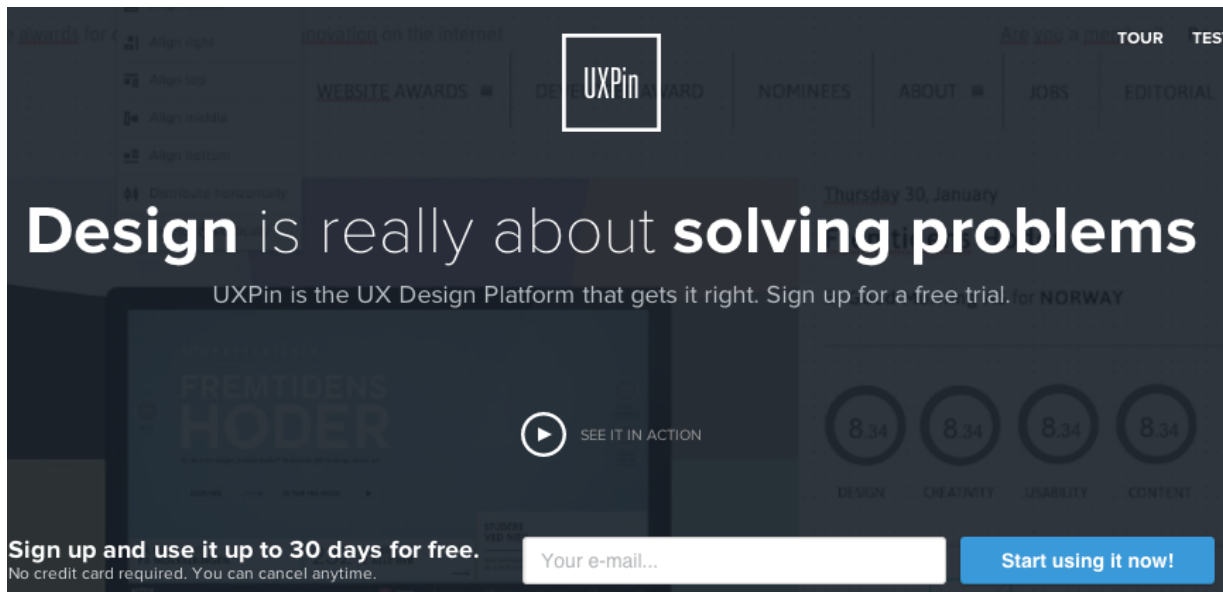
- ▶ Importing from Photoshop – overview video
- ▶ Importing from Sketch into UXPin – step by step
- ▶ General notes on using interactions and animations
- ▶ Button: scrolling the page after click, changing style on hover
- ▶ Form: triggering visibility on scroll
- ▶ Form: interactive inputs
- ▶ Form: interactions after signing up
- ▶ Previewing and gathering feedback

Overview Video (click to play)

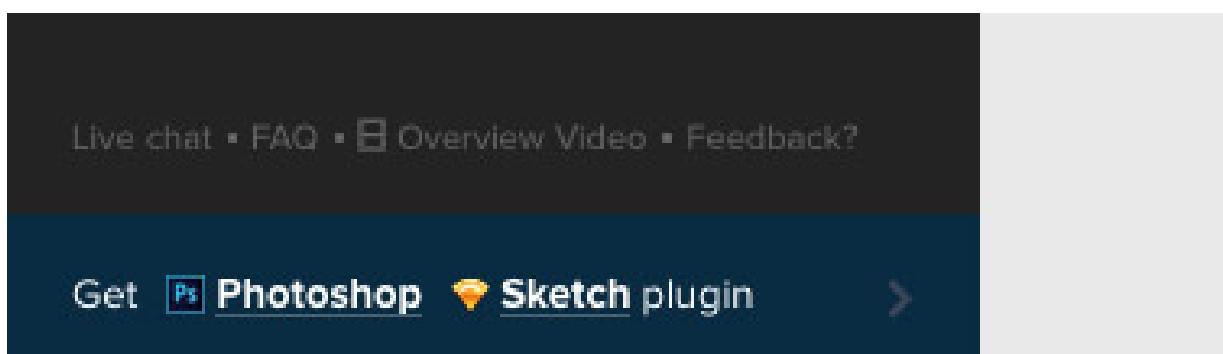


Importing from Photoshop into UXPin

For this example, we will import a Photoshop file from our free [Web UI Kit](#) into UXPin.

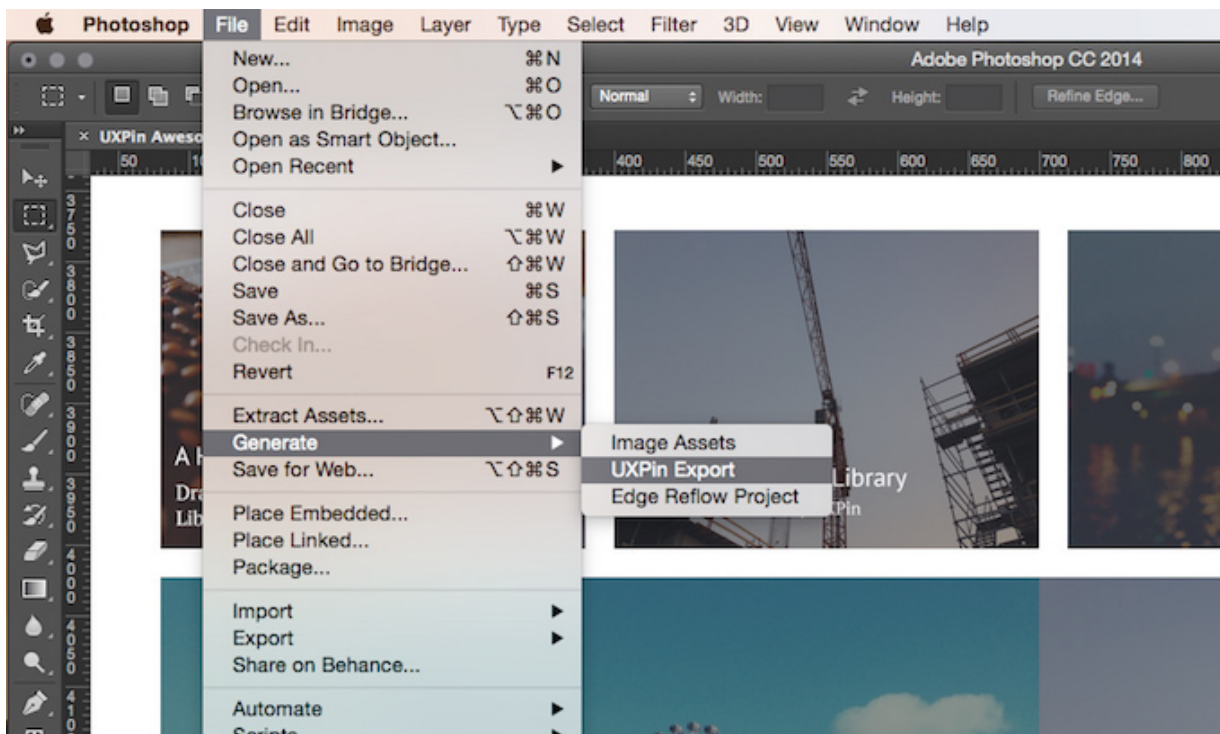


1. Sign in to your existing UXPin account (or [sign up for a free trial](#))

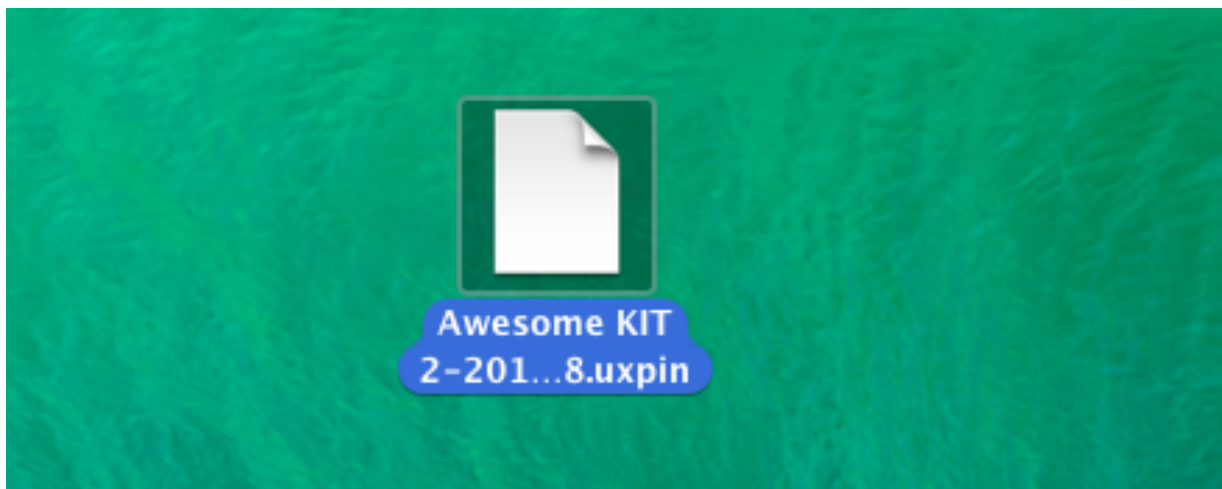


2. Click the plugin icon in UXPin (lower left hand corner)

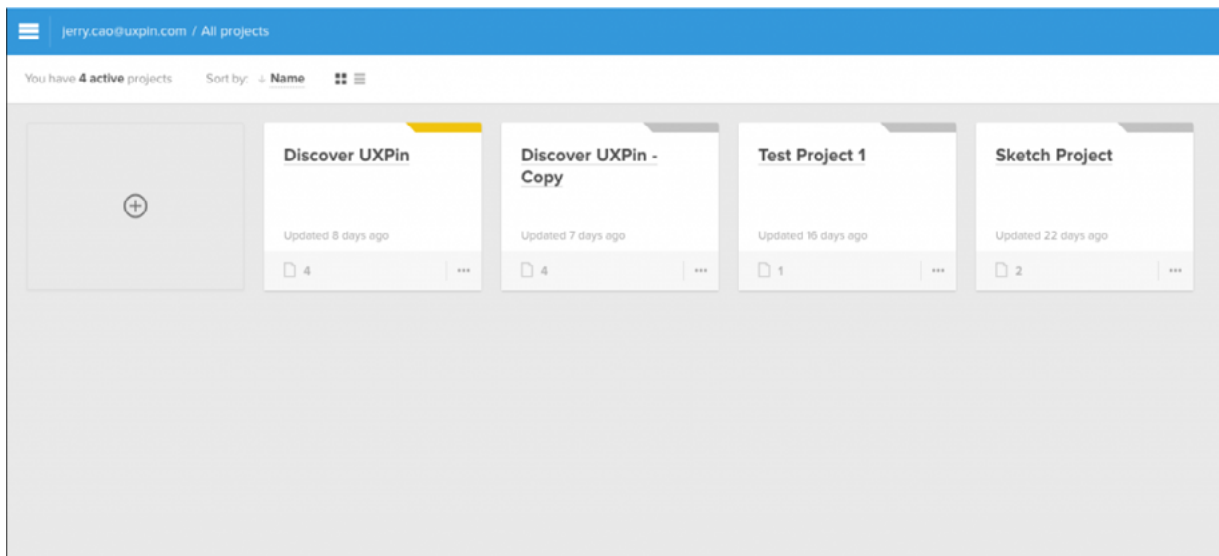
3. Download and install the Photoshop plugin



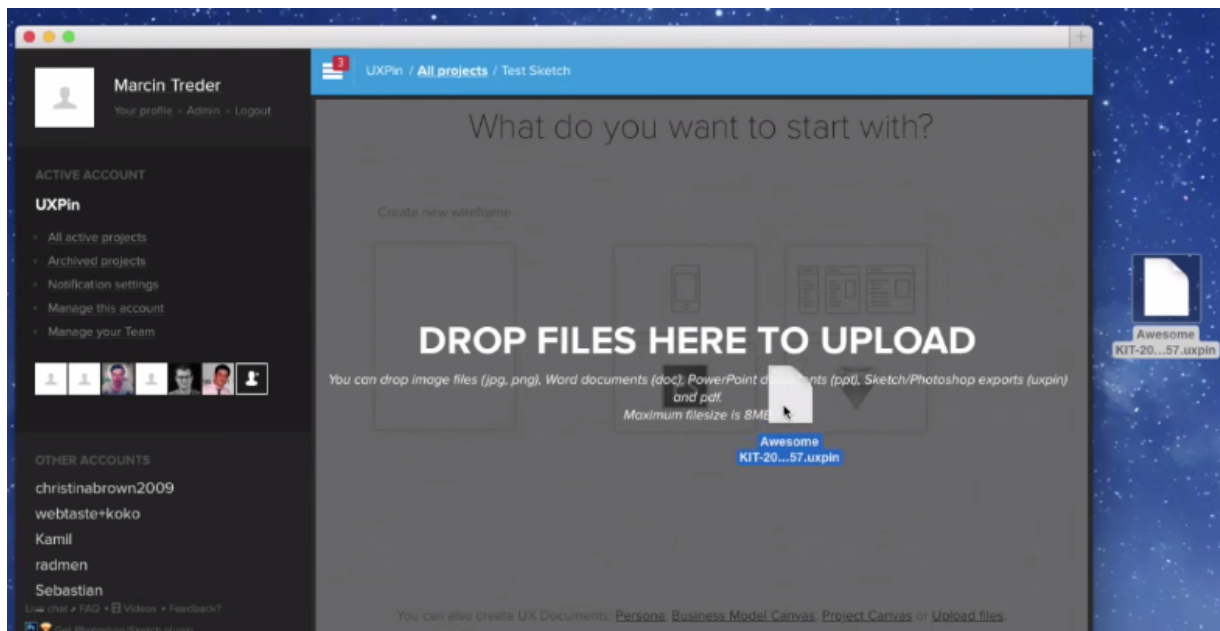
4. In Photoshop, select your elements, click File, Generate and then UXPin Export. Our file is from the [Web UI Kit](#).



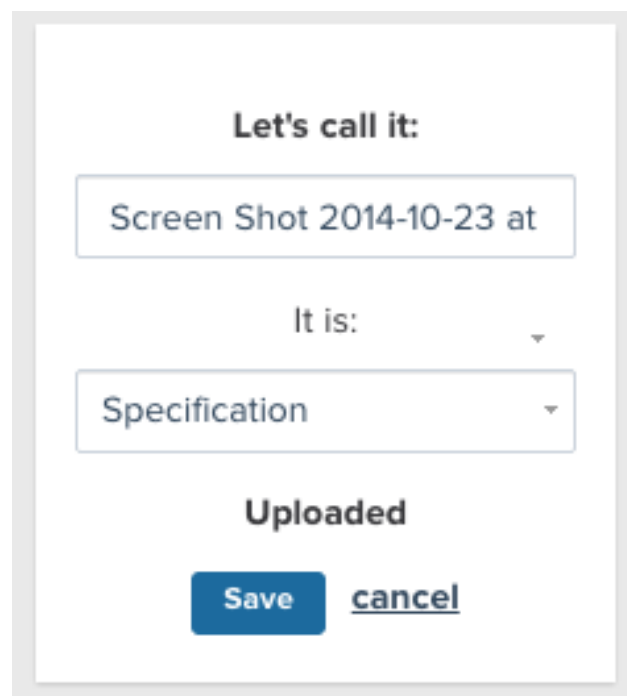
5. The file will likely export to the desktop. Your new file has a .uxpin extension.



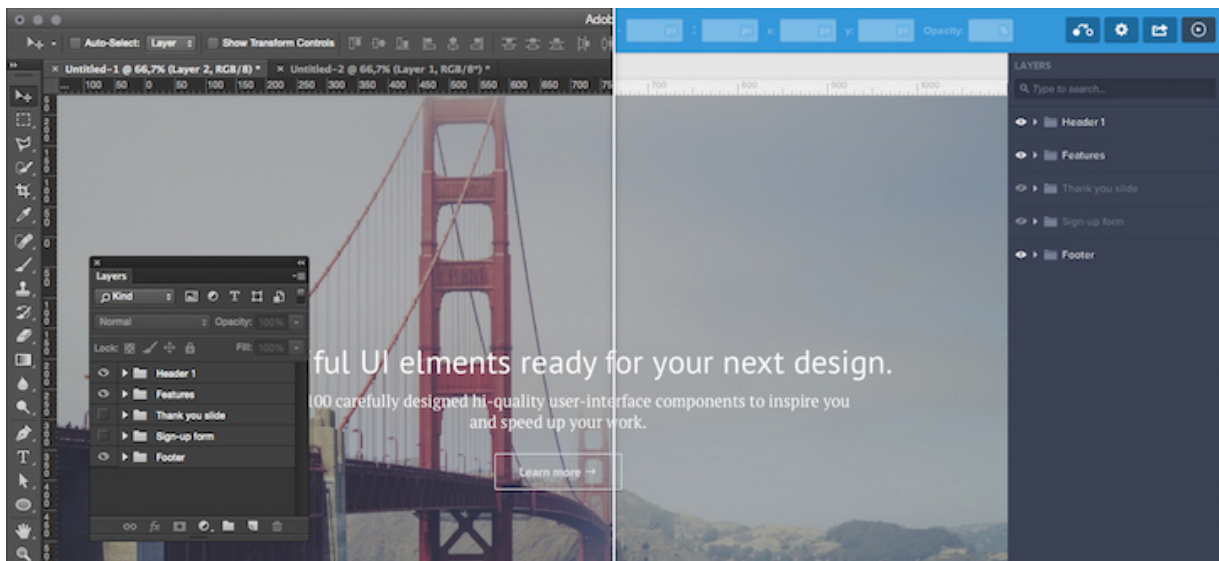
6. Return to [UXPin](#) and either create a new project or click into your existing project.



7. Once you're in the project, drag and drop your .uxpin file into UXPin.



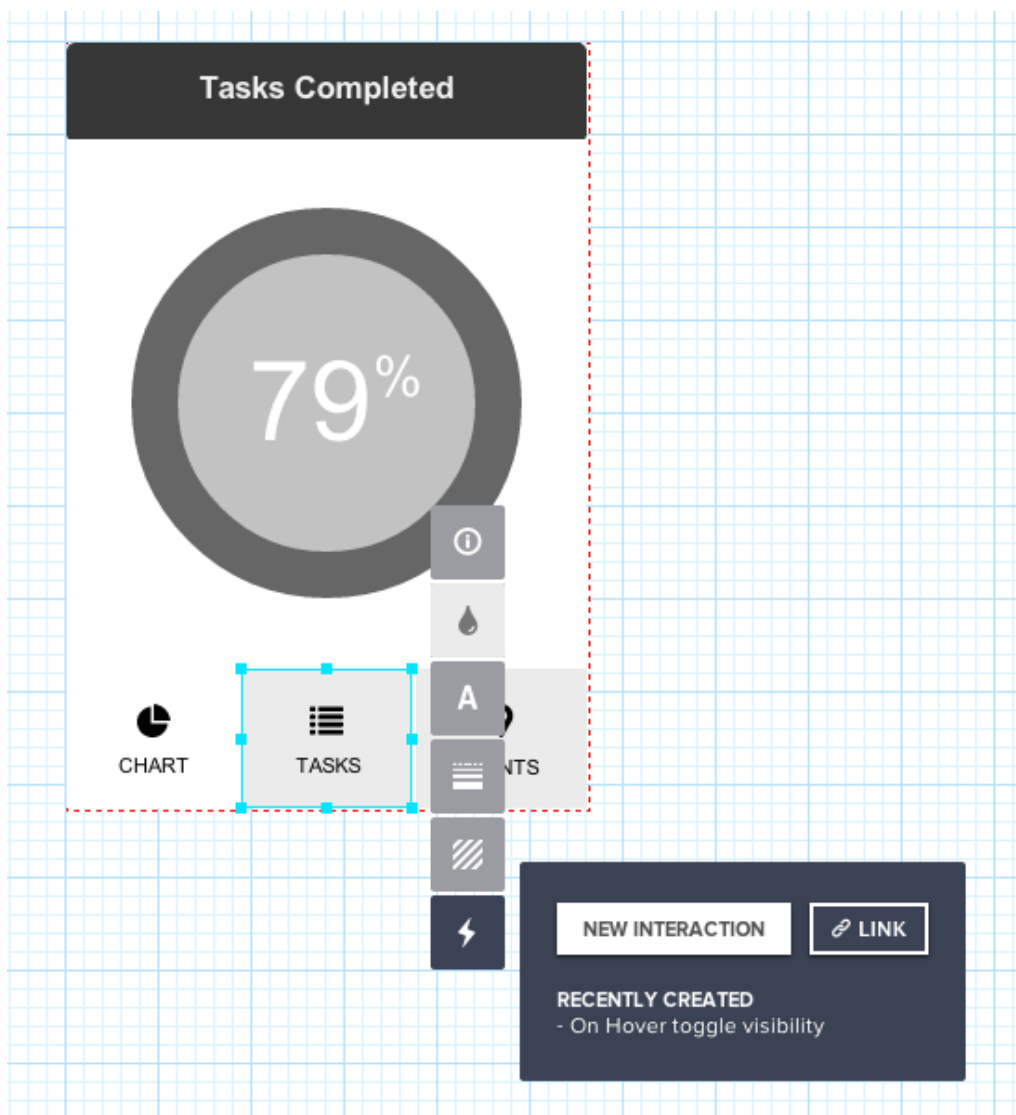
8. Once the file is loaded, feel free to rename and then click Save.



9. Done! All the elements of your Photoshop file are preserved. Feel free to click around and add interactions.

General notes on using interactions and animations

Let's go over some of the functions of Advanced Interactions. To start using it, simply pick any UI element and then can go to interactions tab in the properties manager:

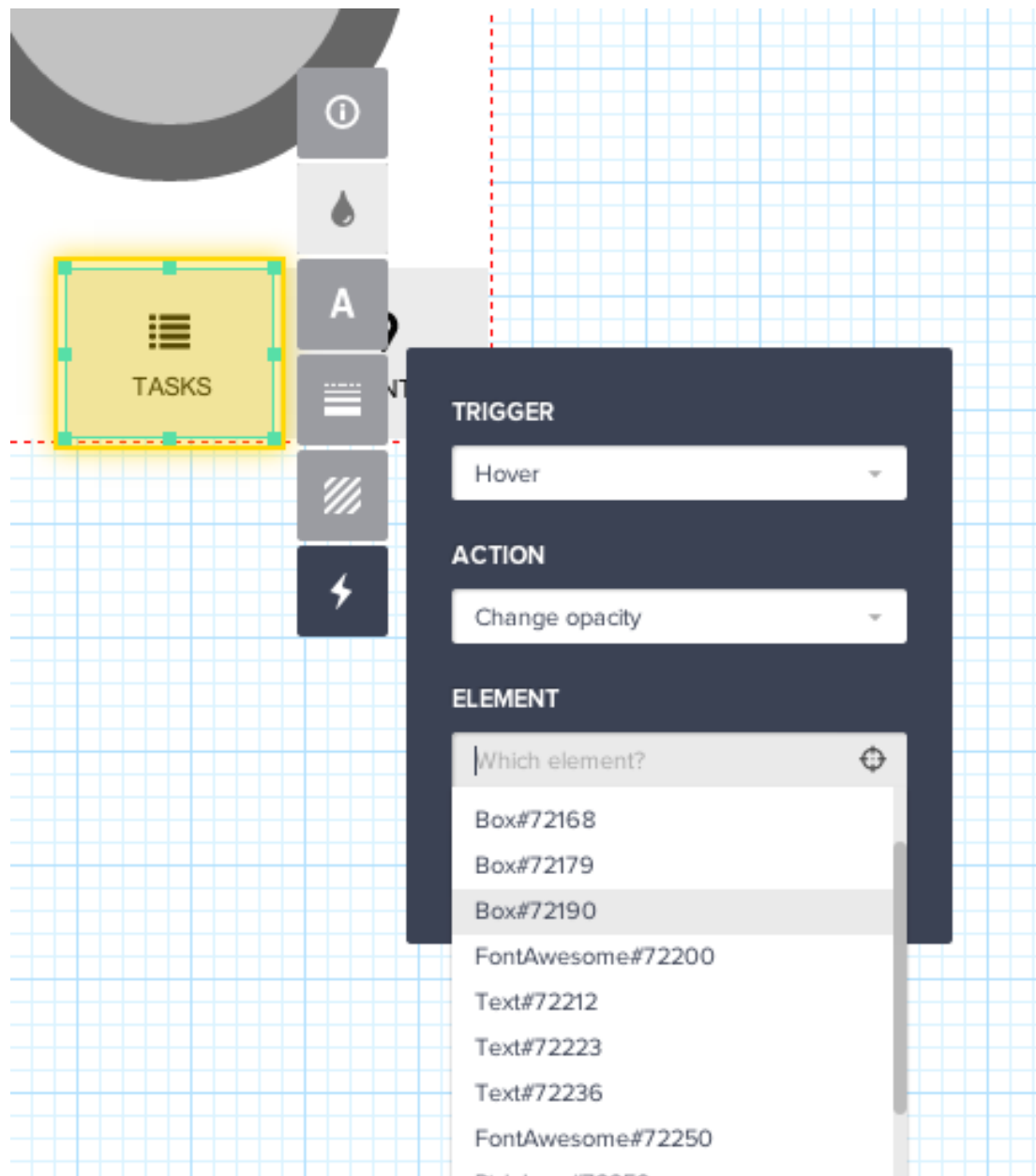


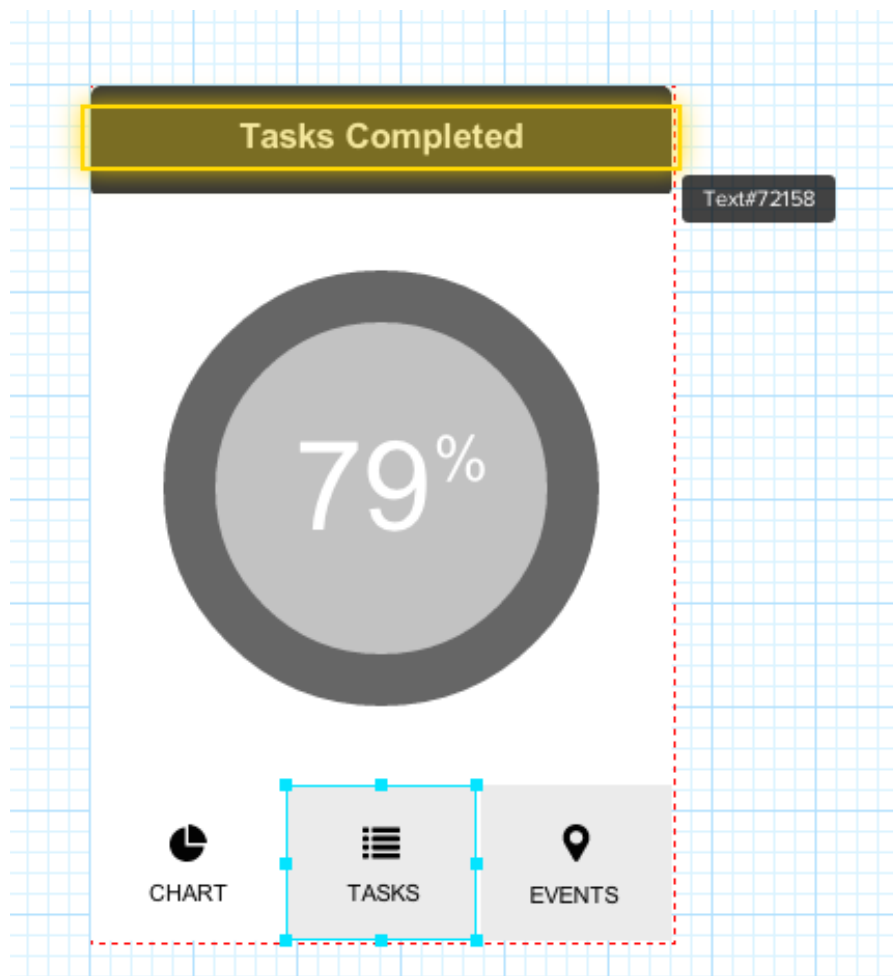
To set your first interaction, click “Lightning bolt” icon. The “Link” button is a shortcut to linking UI elements to pages. Below that, “Recently Created” is a list that shows you any recently created interactions in your design.

Setting up interactions for a given UI element is a 3-step process: first choose what will trigger the action, then what type of action should trigger initiate, and what element should be affected by it. You can see that in the images below:



There are two ways of picking the element that should be affected by the interactions – you can either choose it from the list of all elements, or use your cursor to draw a hotspot on the design.





There is a wide range of triggers, interactions and animations you can choose from. Here's some of them:

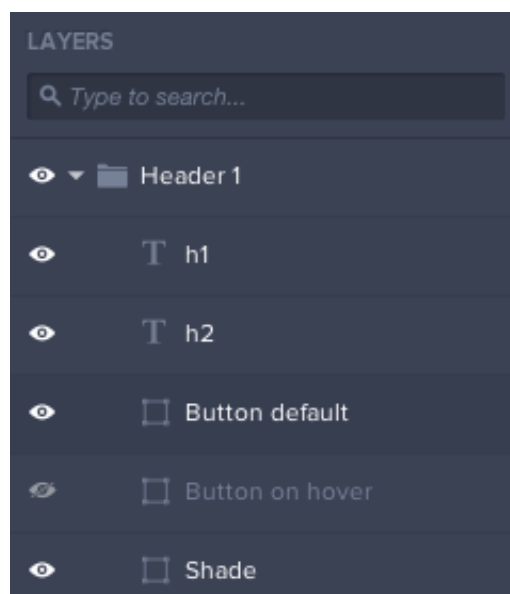
- **Triggers:** click, double, click, right-click, hover, mouse in, focus, blur, key press, window is scrolled to, page is loaded
- **Actions:** show element, hide element, toggle visibility, go to page, go back, scroll element, state: enable, state: disable, state: select/check, move by, resize element, rotate element, change opacity, change style
- **Animations:** linear, ease in, ease out, ease in out, fade, slide.

Button: changing style on hover, scrolling the page after click

Here we'll cover how to make a button that changes style when your mouse hovers over it. We'll also show how to make that same button trigger a scroll to the next section when you click the button. For this example, we'll look at the "Learn more" call-to-action.

Changing Style On Hover

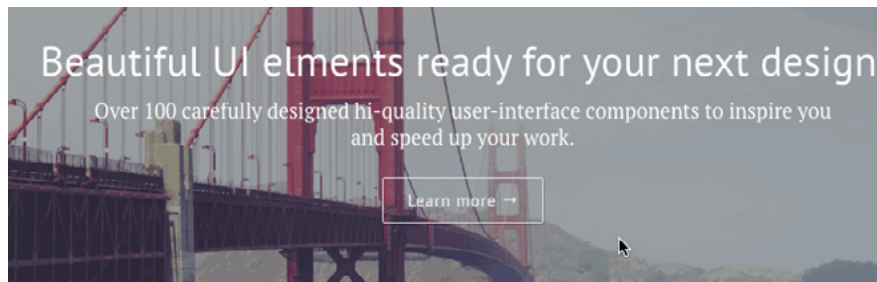
I start by making sure that I have two different buttons that will appear after specific user actions: one for the inactive state and one appearing on mouse out.



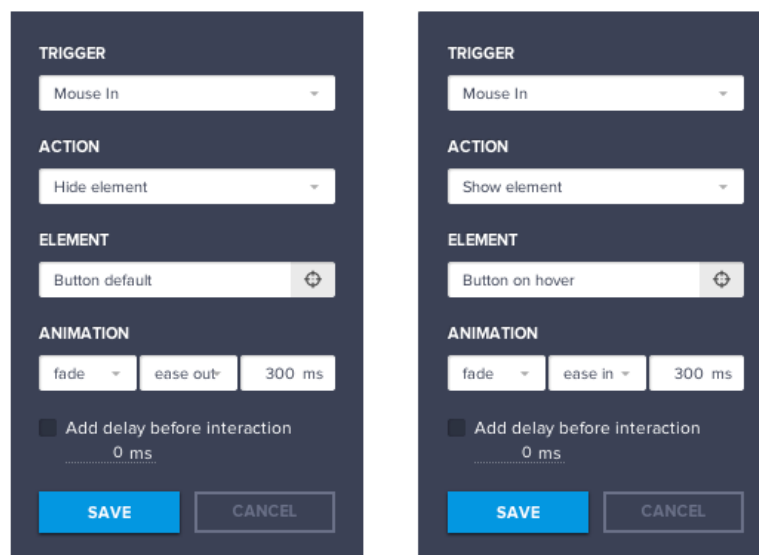
As you can see below, the buttons will switch with interactions, however they will remain in the same position, so that's how we have to put them on the prototype. Using the layers editor, you can stack them in the right order – keep the one that shows by default on top, and the one showing on hover below that.

Since they can't be shown simultaneously, hide the one that is to be activated on hover. We'll make it appear using interactions later on.

Click the image below to open the GIF in a new window.



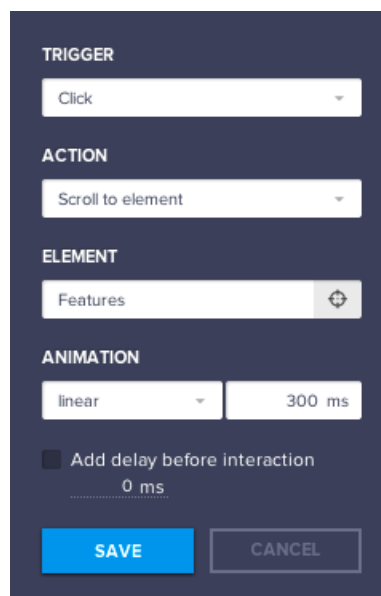
Now on to setting up interactions. For the first button we'll need a total of two interactions: showing the orange button upon hover (left panel below) and hiding the orange button on mouse out (right panel below). Here's how the setup will look:



Now we just need to make the page scroll to the next section upon clicking the orange “Learn More” button. And let's not forget that the orange button needs to be hidden on mouse out so that the default transparent button can reappear (in case you scroll back up).

That's easy to do with separate interactions. Select the on hover button and set up interactions like the instructions below.

To scroll to the next section when you click the orange button, make sure your interactions menu looks like this:



The screenshot shows the interactions menu for a 'Click' trigger. The 'ACTION' is set to 'Scroll to element', and the 'ELEMENT' is 'Features'. The 'ANIMATION' is set to 'linear' with a duration of '300 ms'. There is a checkbox for 'Add delay before interaction' which is unchecked, with a value of '0 ms'. At the bottom, there are 'SAVE' and 'CANCEL' buttons.

TRIGGER
Click

ACTION
Scroll to element

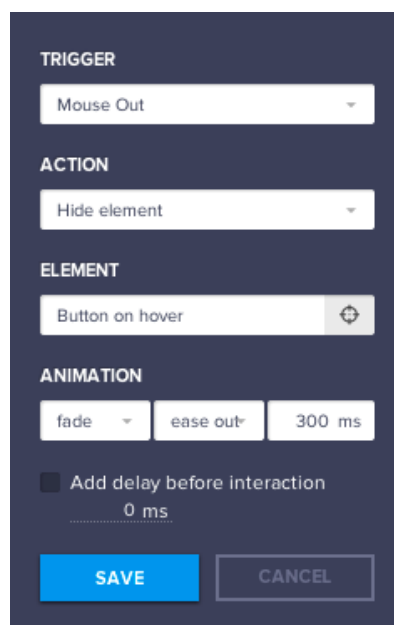
ELEMENT
Features

ANIMATION	
linear	300 ms

Add delay before interaction
0 ms

SAVE CANCEL

To hide the orange button, make sure your interactions menu looks like this:



The screenshot shows the interactions menu for a 'Mouse Out' trigger. The 'ACTION' is set to 'Hide element', and the 'ELEMENT' is 'Button on hover'. The 'ANIMATION' is set to 'fade' with 'ease out' and a duration of '300 ms'. There is a checkbox for 'Add delay before interaction' which is unchecked, with a value of '0 ms'. At the bottom, there are 'SAVE' and 'CANCEL' buttons.

TRIGGER
Mouse Out

ACTION
Hide element

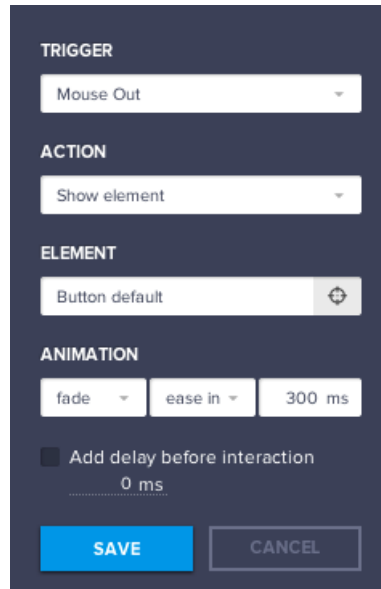
ELEMENT
Button on hover

ANIMATION		
fade	ease out	300 ms

Add delay before interaction
0 ms

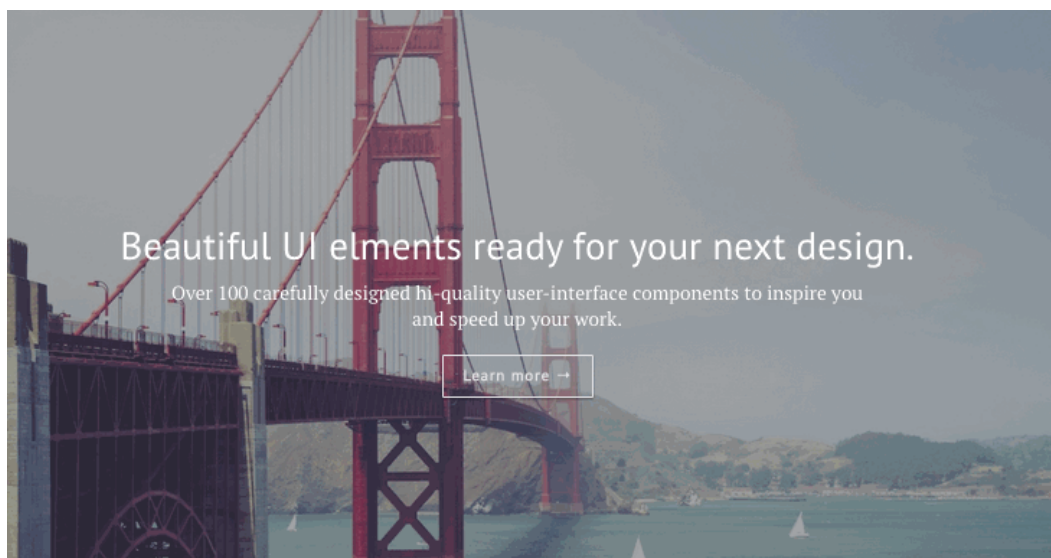
SAVE CANCEL

To make the transparent button show when you mouse out (in case you scroll back up), set up your interactions menu like this:



If you like, you can use different animations. For me, the choice was fade with ease in/ease out for mouse actions and linear animation on scroll.

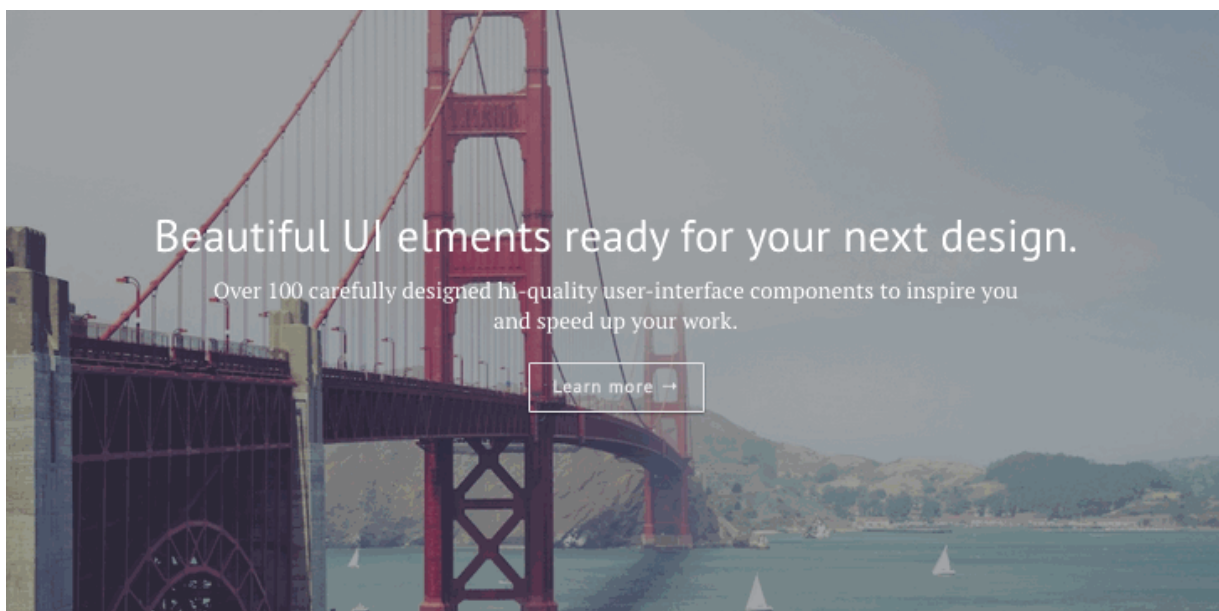
Click the first button in [our prototype](#) to see the effect, or click the image below to show the interaction as a GIF in a new window.



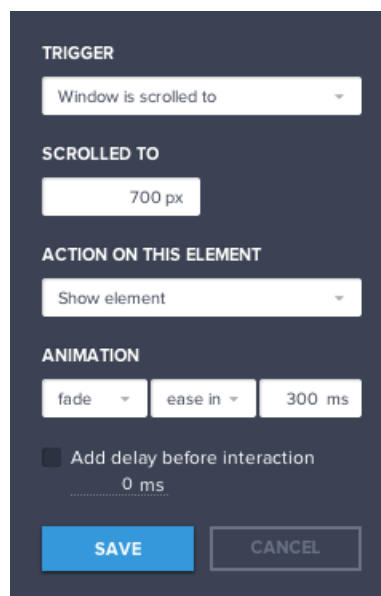
Form: triggering visibility on scroll

Next we will take care of the third section of the prototype, which is the sign up form. I'll show you how make it smoothly appear when the window is scrolled to a certain level of pixels.

Scroll down [in our prototype](#) to see the effect, or click the image below to show the interaction as a GIF in a new window.



Since the form has to appear on a certain interaction, it has to be hidden by default – make sure you have that step covered. Now, select the whole layer of the Form and click the lightning bolt icon for interactions. As you probably already guess, the trigger will be “window is scrolled to” and the action will be “show element”:



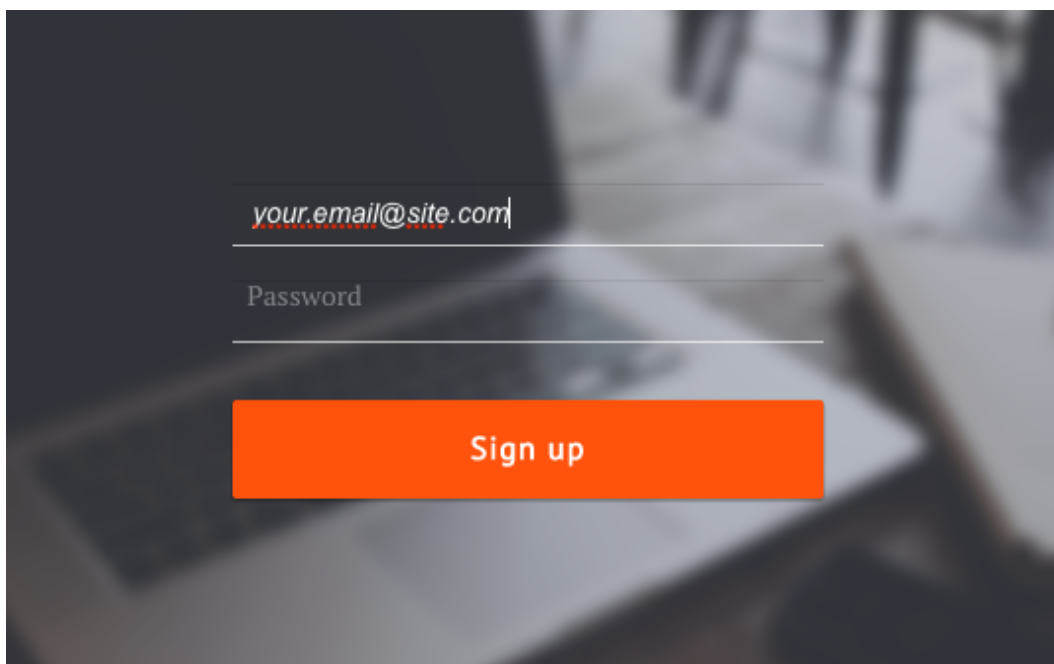
The image shows a dark-themed settings panel for an interaction. It is organized into several sections:

- TRIGGER:** A dropdown menu set to "Window is scrolled to".
- SCROLLED TO:** A text input field containing "700 px".
- ACTION ON THIS ELEMENT:** A dropdown menu set to "Show element".
- ANIMATION:** Three input fields: "fade" (with a dropdown arrow), "ease in" (with a dropdown arrow), and "300 ms".
- Delay:** A checkbox labeled "Add delay before interaction" which is unchecked, followed by a text input field containing "0 ms".
- Buttons:** A blue "SAVE" button and a grey "CANCEL" button at the bottom.

Form: interactive inputs

Since we've introduced a form in our prototype, why not make the inputs interactive to complete the experience?

Choose inputs from the library of elements (tip: hit cmd/ctrl+f and type "input"), style it accordingly and place it where you want customers to sign up. You don't need to set any interactions for allowing typing inside the inputs or switching to the next one by the tab key – that's already included. So basically, if you just want the inputs to be ready for inserting text, you're all set!

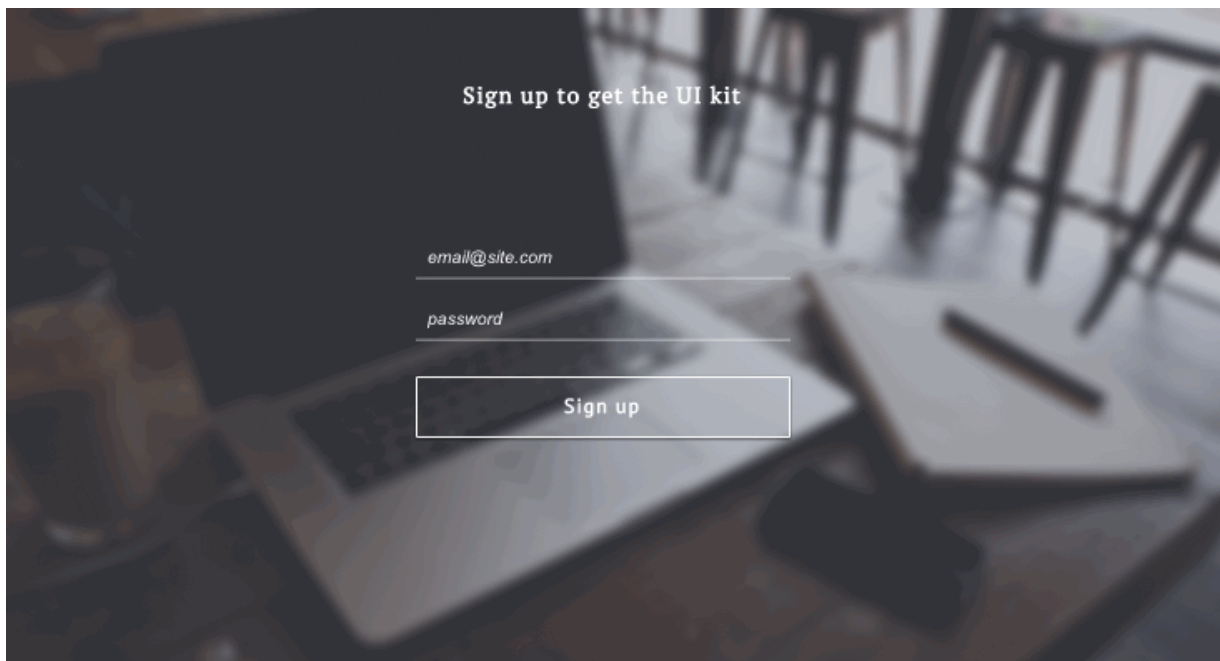


Other than that you can come with some additional interactions of your own. I added a little trick to the input, making the "email" caption disappear once you click on the input. I just placed a text element underneath the input and added an interaction on the input element.

Form: interactions after signing up

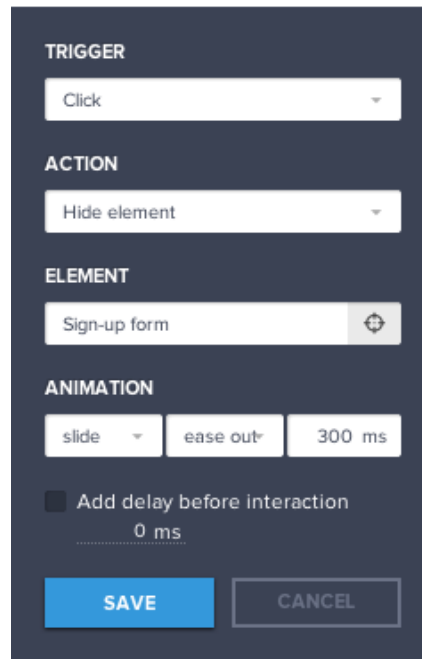
As the final touch on our prototype example we'll introduce a sign up confirmation that slides in when the user clicks "sign up" button.

Fill out the form [in our prototype](#) and click "Sign Up" to see the effect, or click the image below to show the interaction as a GIF in a new window.



We'll be setting interactions between three elements: the sign up button, the signup form and the "Thank you" slide. However, the interaction will be set only on the signup button, because it triggers actions for the signup form and "Thank you" slide. When clicked, the signup button will hide the sign up form and show the "thank you" slide. See the instructions below.

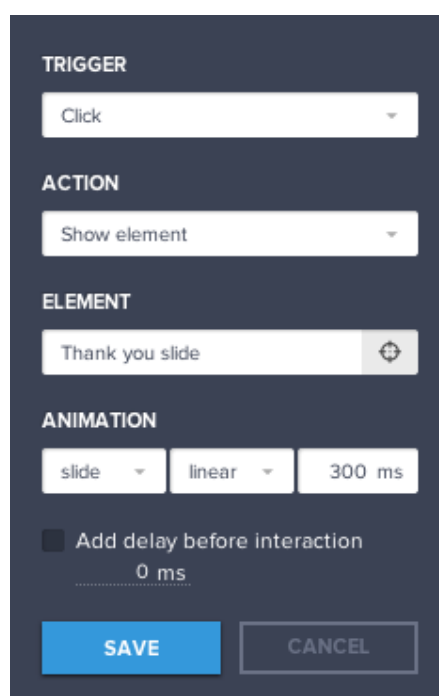
To hide the signup form when the signup button is clicked, make sure your interaction menu looks like this:



The image shows a dark-themed interaction menu with the following settings:

- TRIGGER:** Click
- ACTION:** Hide element
- ELEMENT:** Sign-up form
- ANIMATION:** slide, ease out, 300 ms
- Add delay before interaction: 0 ms
- SAVE** (blue button) and **CANCEL** (grey button)

To show the “thank you slide” when the signup button is clicked, make sure your interaction menu looks like the below. Then you’re all done!



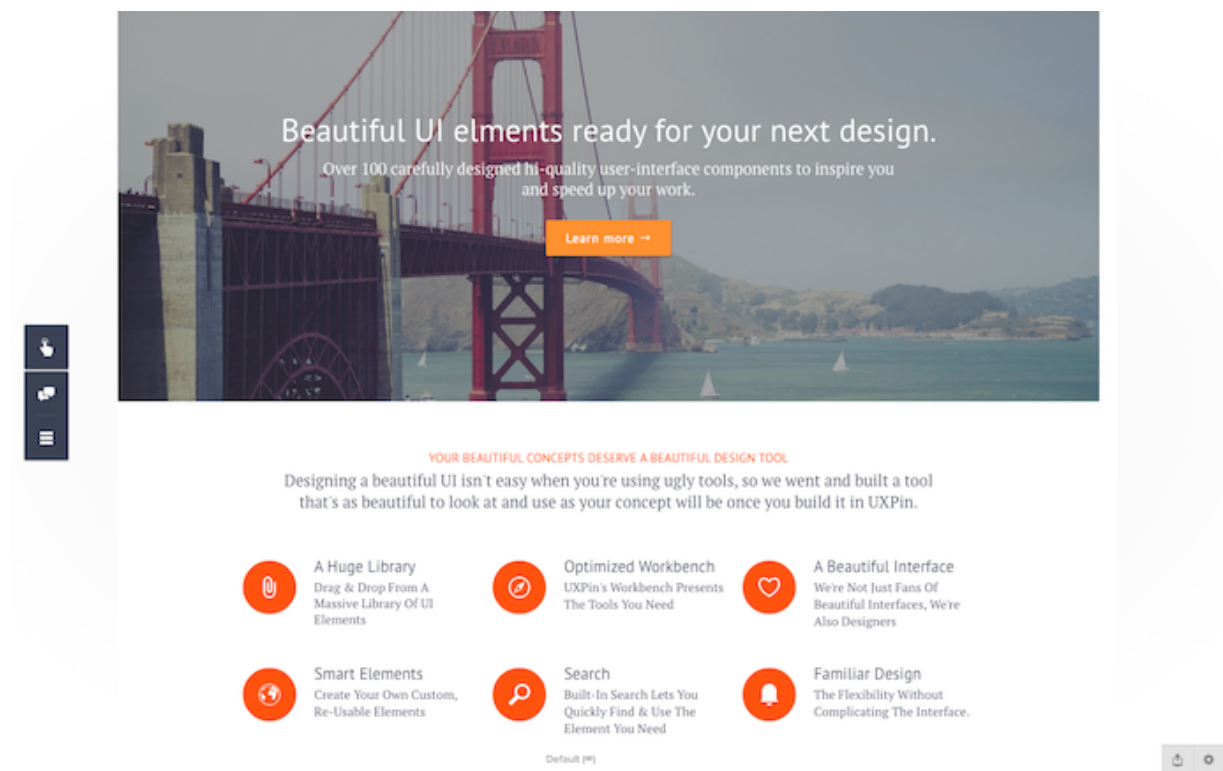
The image shows a dark-themed interaction menu with the following settings:

- TRIGGER:** Click
- ACTION:** Show element
- ELEMENT:** Thank you slide
- ANIMATION:** slide, linear, 300 ms
- Add delay before interaction: 0 ms
- SAVE** (blue button) and **CANCEL** (grey button)

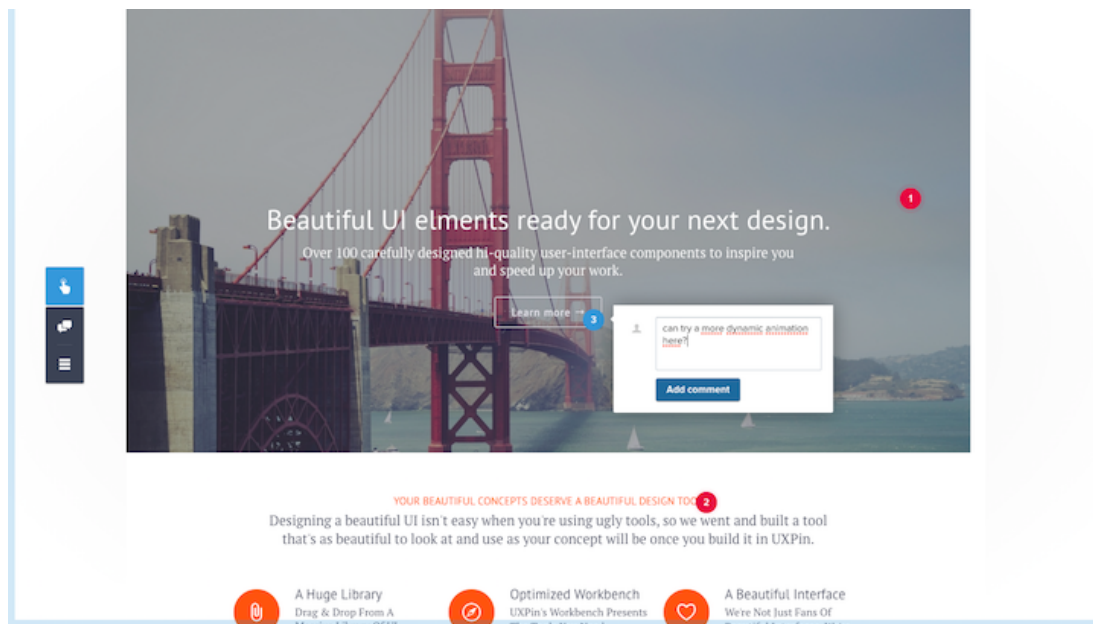
Previewing and gathering feedback



1. To see the interactions, click the play button in the upper right hand corner. This will open the [Preview Mode](#).

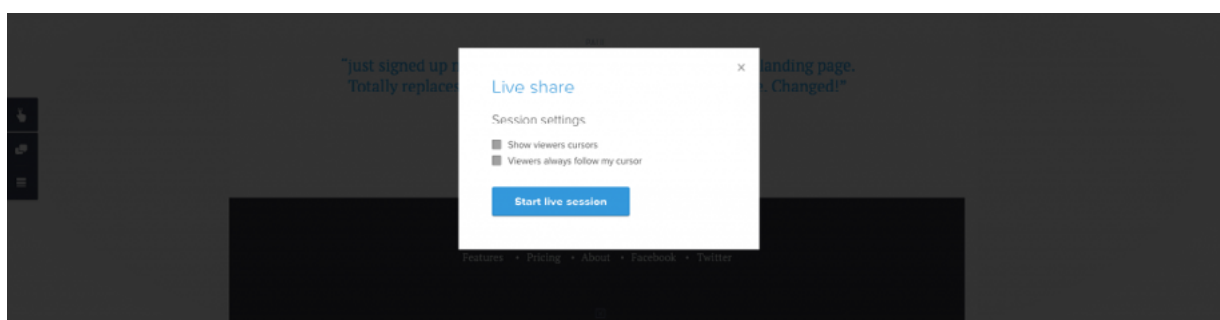


2. You're in the preview mode, so check out the results of your work:



3. To comment, simply use the left-side vertical menu. To start a [Live Presentation](#), click on the lower right hand corner.

So there you have it, a step by step process to turn your current Photoshop file into a fully interactive prototype. Plus, your entire team can comment and collaborate.



We only showed one example of an animation. In UXPin, you can make your Photoshop prototype come to life with different animations from the 11 triggers and 20 action events.

Feel free to get started and play around in [UXPin](#).

How to Create Interactive Prototypes From Sketch Files

Closing the gap between static and interactive design

You're deep in Sketch working on your design and now want to take it to the interactive stage. But you want to do it quickly, without code, while preserving all of your layers. Your team also needs to be able to comment directly on the prototype. Sounds tough, right? It's actually quite easy.



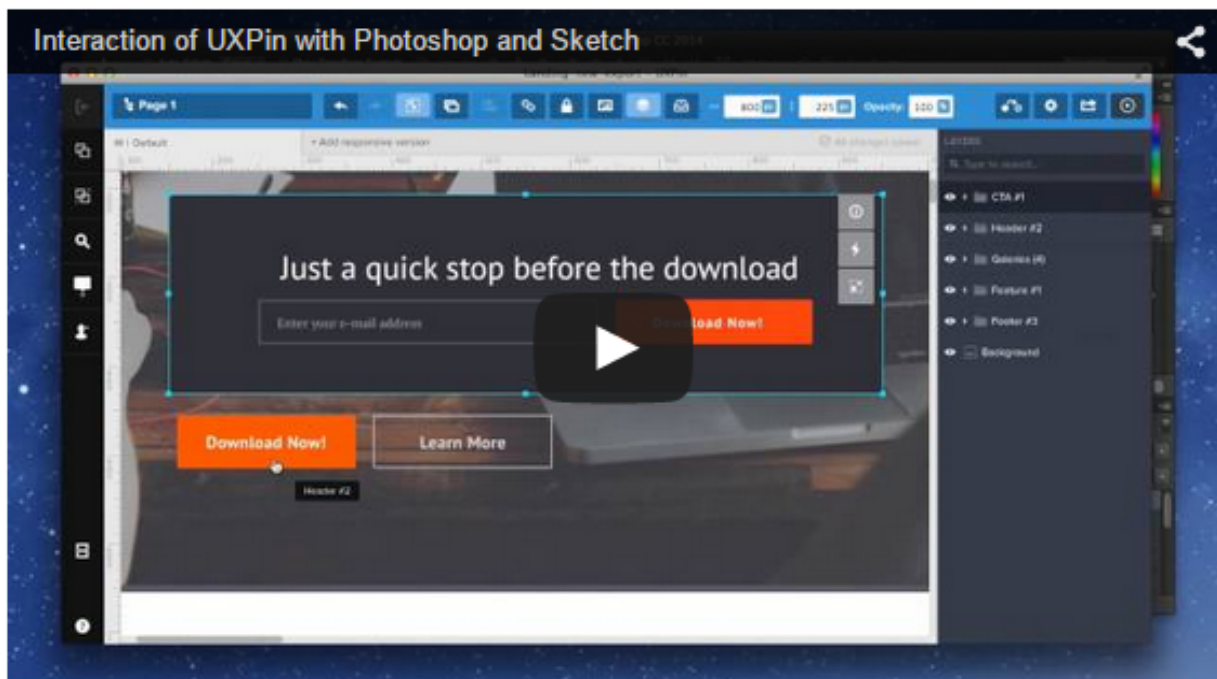
Source: [Photoshop & Sketch Integration](#)

We'll show you how simple it can be to turn your Sketch files into a fully interactive prototype with a simple drag & drop. Within UXPin, your team can also comment directly on the design.

Once you've imported your Sketch file into [UXPin](#), it's easy to add interactions & animations and then the [Live Presentation tool](#) to host a screenshare meeting to unveil your new design. There are currently 11 triggers and 20 element actions, allowing for many custom advanced interactions.

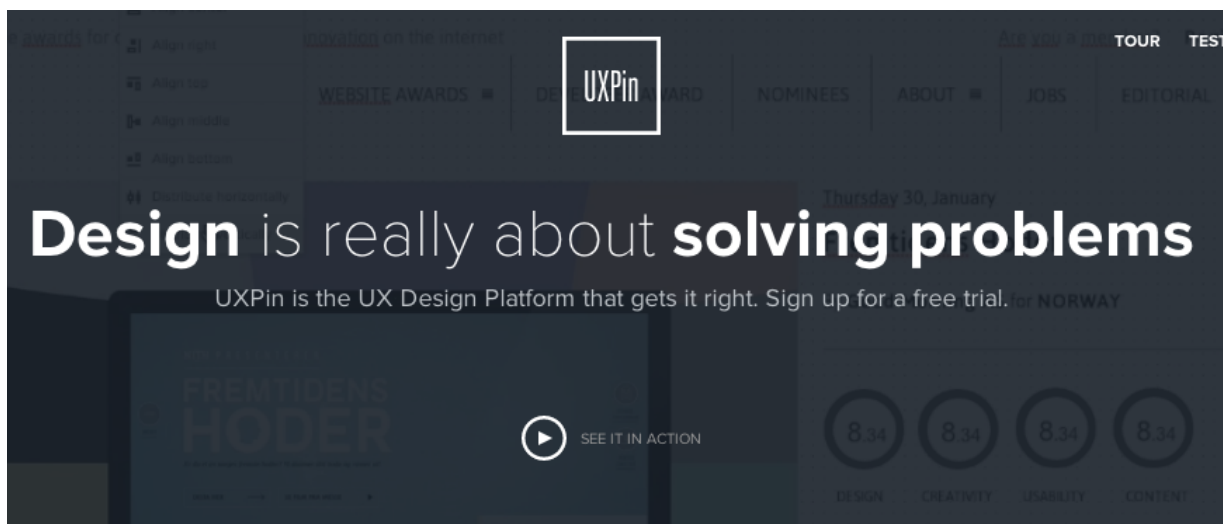
Take a look at the [overview video](#) and then check out our tutorial below (or [follow along on the blog post](#)).

Overview Video (click to play in new window)

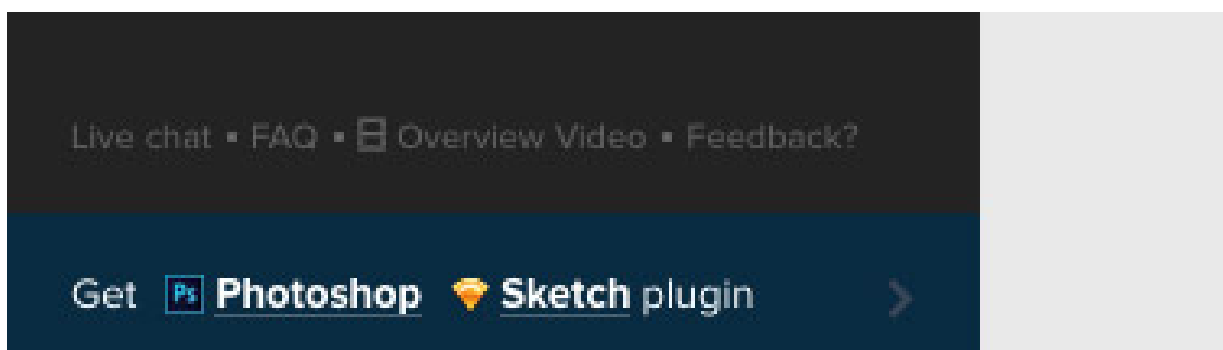


Importing from Sketch into UXPin

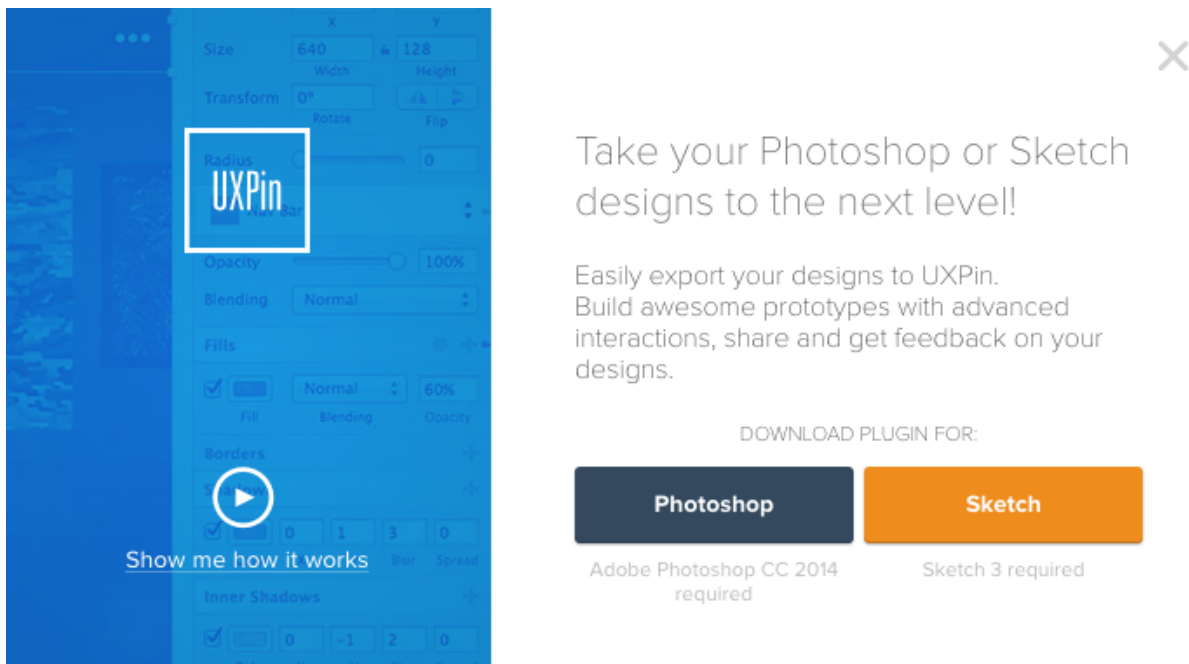
For this example, we will import a Sketch file from our free [Web UI Kit](#) into UXPin. As an introductory tutorial, we'll just work on adding a simple scrolling interaction. Once you get the hang of it, you'll be able to add interactions to the rest of your elements in no time.



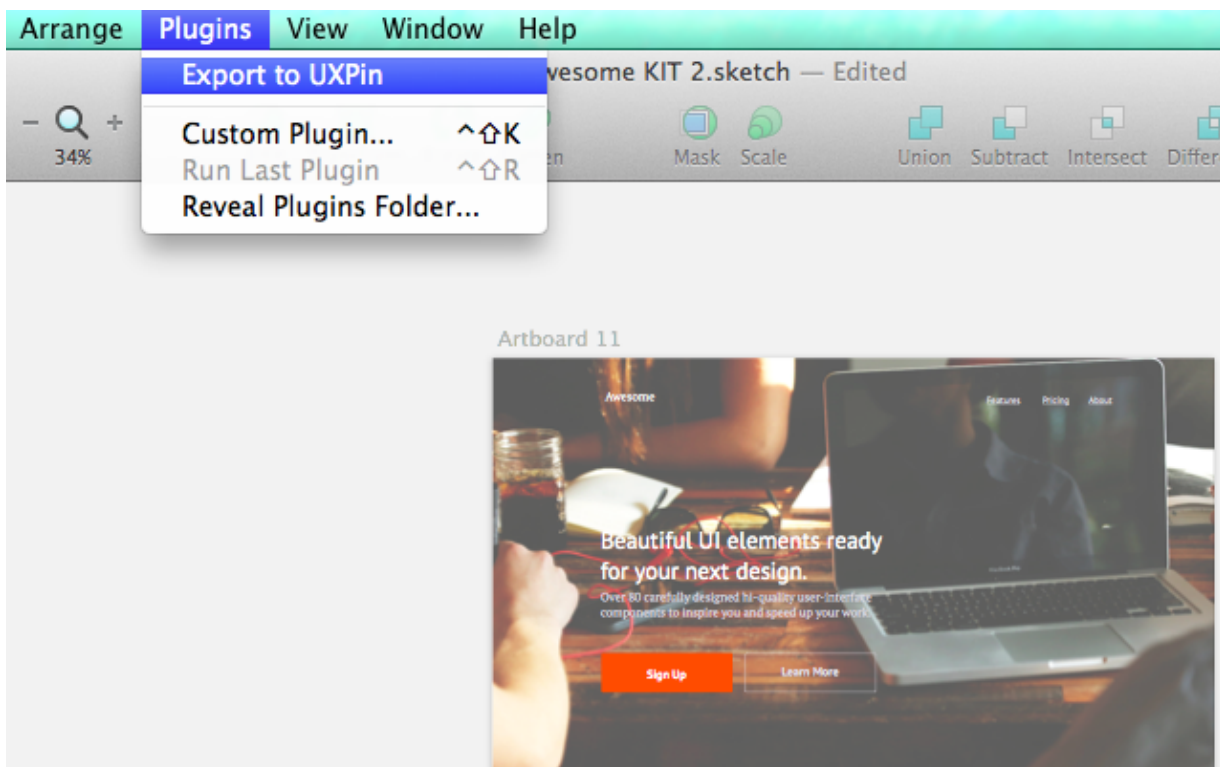
1. Sign in to your existing UXPin account (or [sign up for a free trial](#))



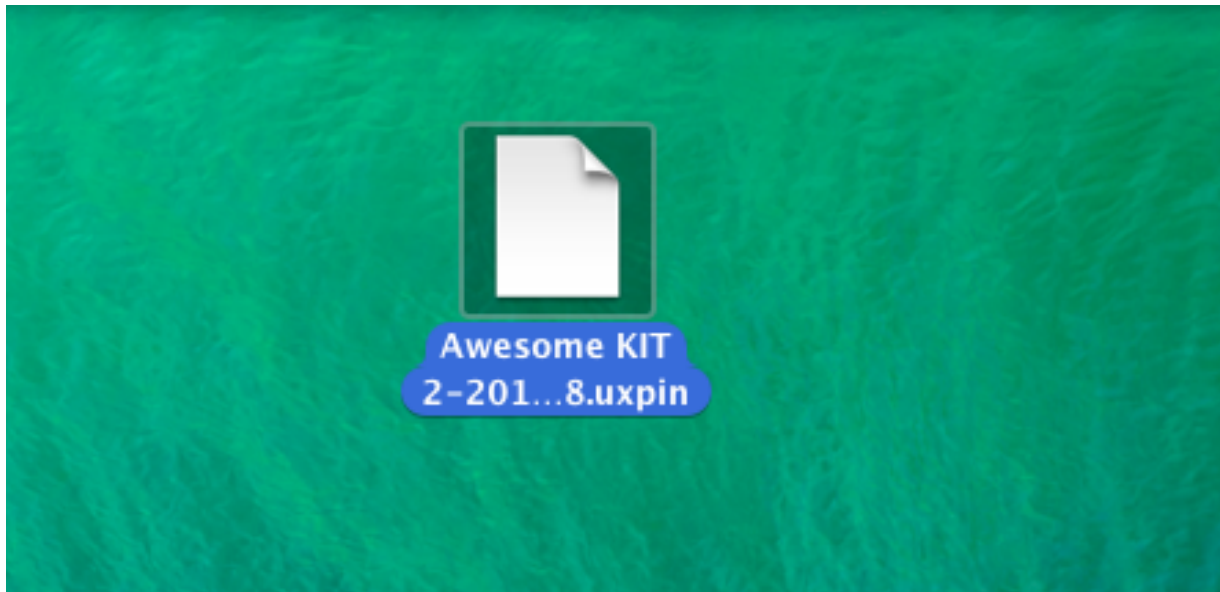
2. Click the plugin icon in UXPin (lower left hand corner)



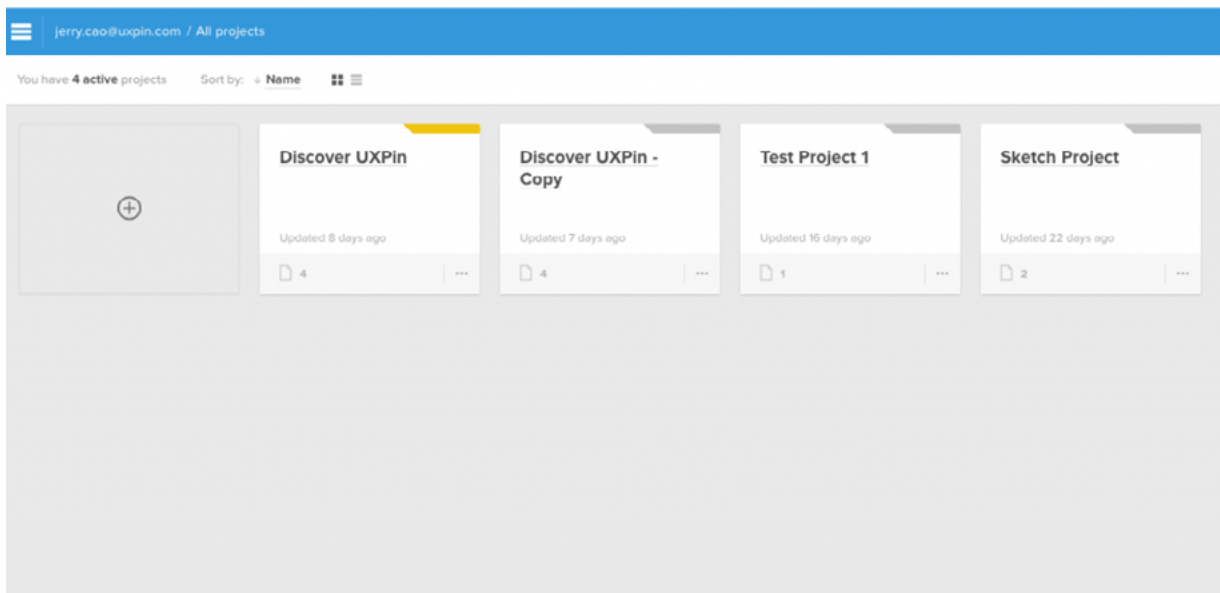
3. Download and install the Sketch plugin



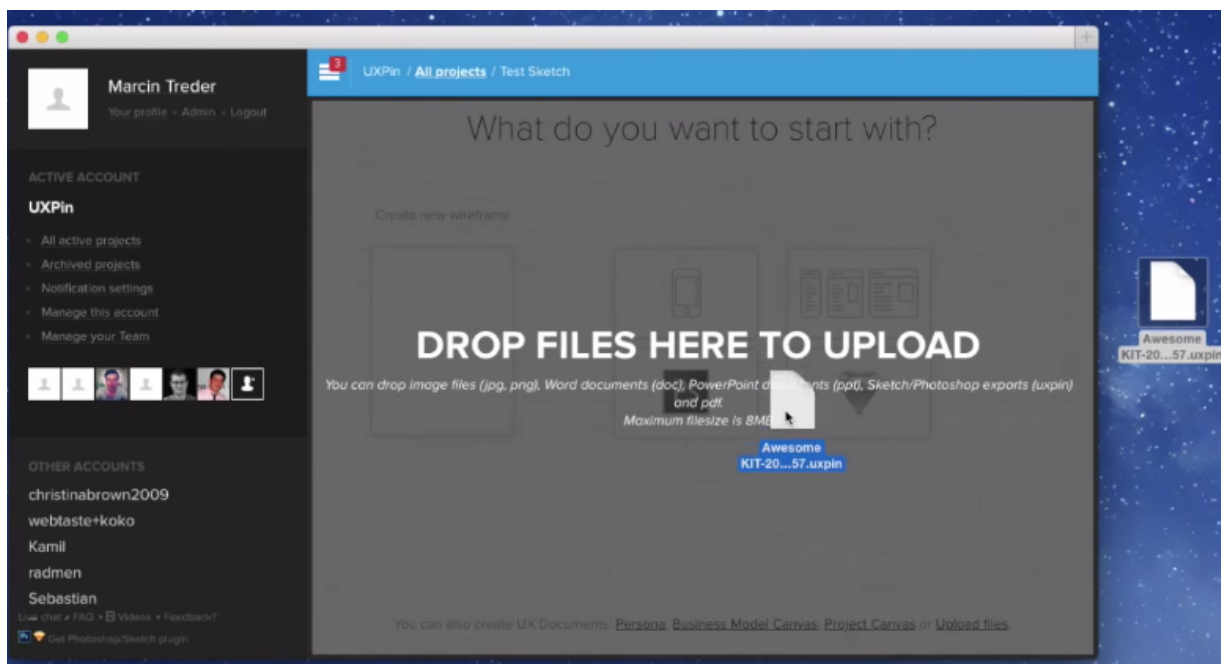
4. In Sketch, select your elements, click *Plugins* and then *Export to UXPin*. Our file is from the [Web UI Kit](#).



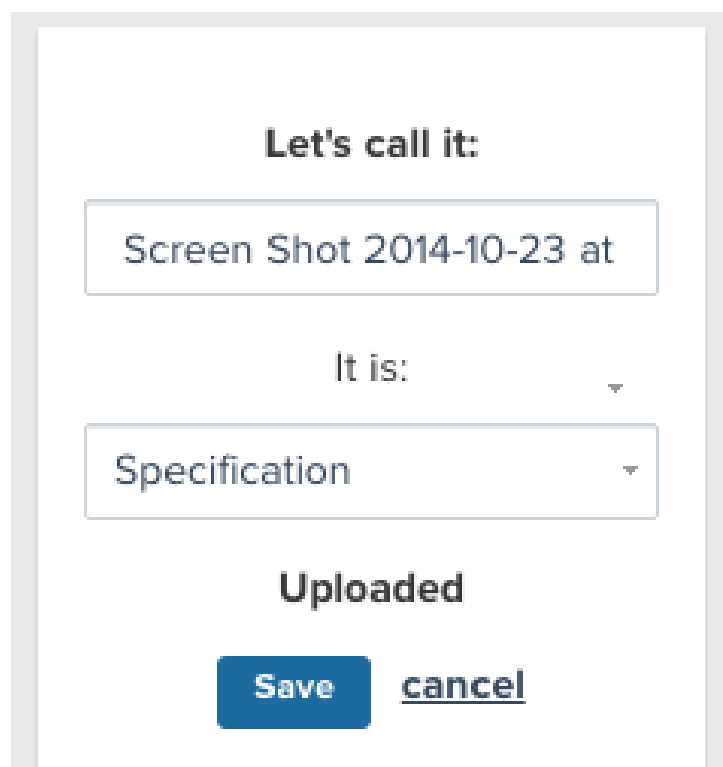
5. The file will likely export to the desktop. Your new file has a .uxpin extension.



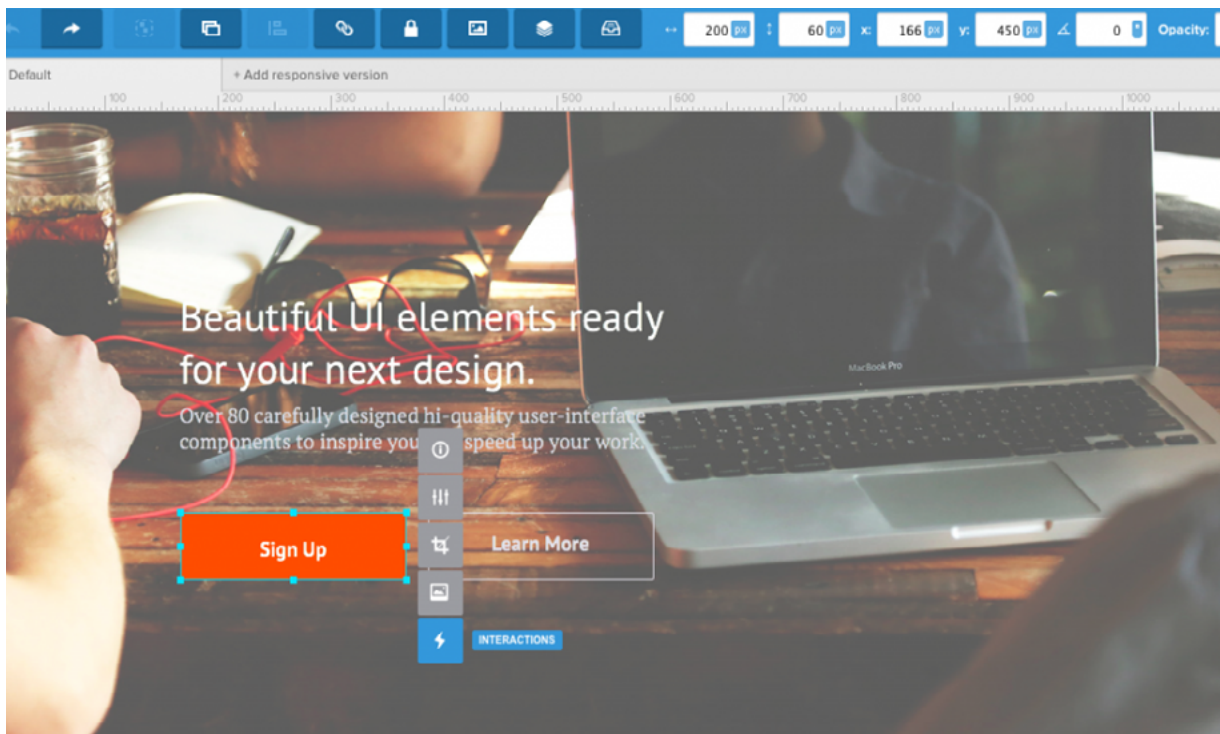
6. Return to UXPin and either create a new project or click into your existing project.



7. Once you're in the project, Drag and drop your .uxpin file into UXPin.



8. Once the file is loaded, feel free to rename and then click Save.



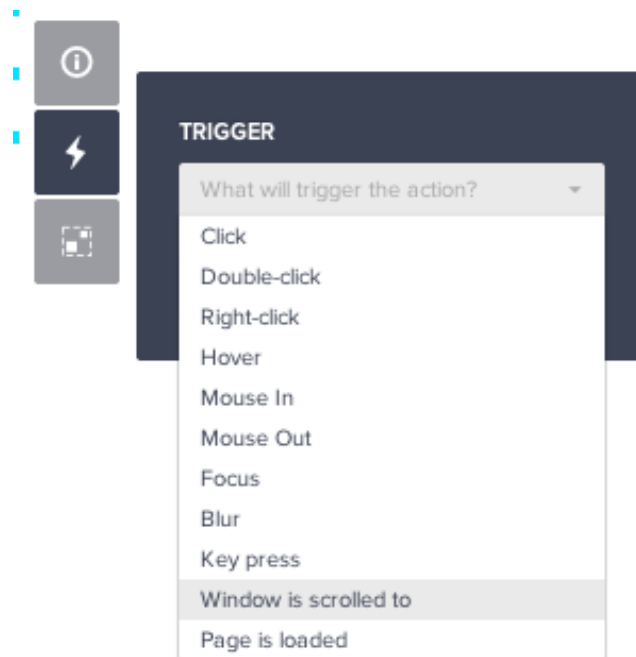
9. Done! All the elements of your Sketch file are preserved. Feel free to click around and add interactions.

Prototyping Animations & Interactions

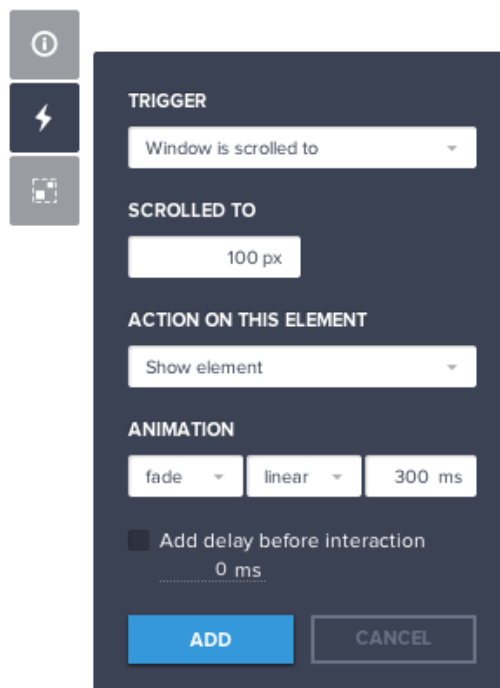
With 11 triggers and 20 element actions, UXPin allows for many custom advanced interactions. For this example, we will set a scroll trigger to reveal a display header. To learn about more interactions, check out our post on [Advanced Interactions & Animations](#).



1. Click on the display header, then click *Properties*. Once the menu shows up, make sure you click the eye icon to hide the header and check fixed position.

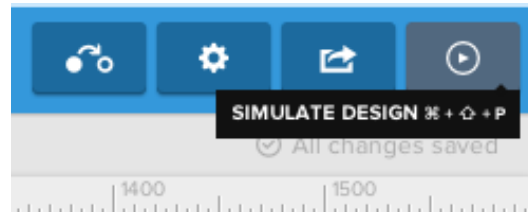


2. Click the lightning bolt icon, then click *NEW INTERACTION*. Once the dropdown menu loads, click *Window is scrolled to*.

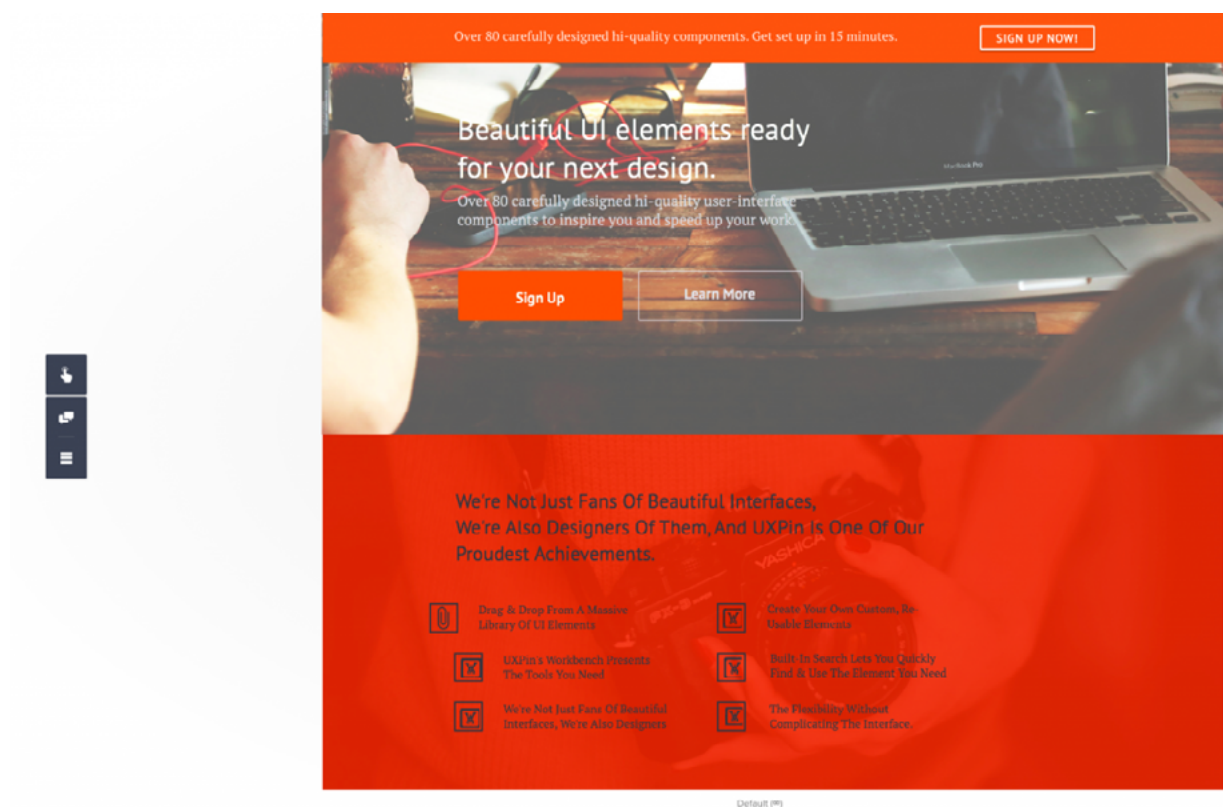


3. After you click *Window is scrolled to*, set the pixels to *100px*. Then, select *Show element* as the element action. Finally, make sure your animation settings are *fade*, *linear*, and *300 ms*.

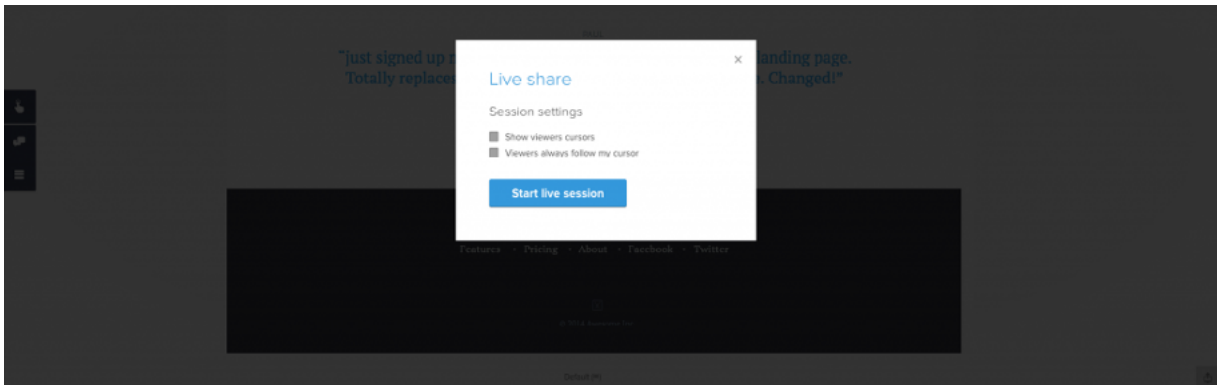
Previewing Animations & Interactions



1. To see the animation, click the play button in the upper right hand corner. This will open the [Preview Mode](#).



2. As you scroll down, the header will animate and appear. You can try it here [in this preview](#).

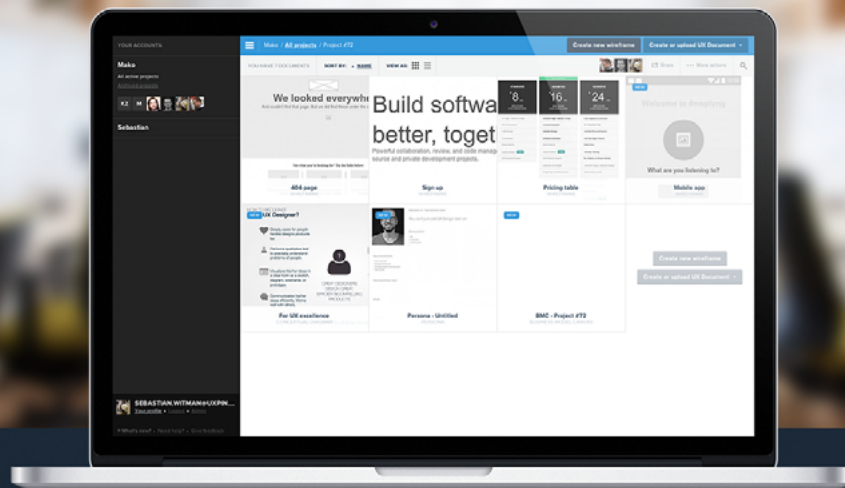


3. To comment, simply use the left-side vertical menu. To start a [Live Presentation](#), click on the lower right hand corner.

So there you have it, a step by step process to adding a quick interaction to your Sketch file. Feel free to repeat the process with different elements and interactions to create a fully animated prototype. Your entire team can also comment and collaborate.

We only showed one example of an animation. In UXPin, you can make your Sketch prototype come to life with different animations from the 11 triggers and 20 action events.

Feel free to [get started](#) and play around in UXPin.



- ✓ Complete prototyping framework for web, mobile, and wearables
- ✓ Collaboration and feedback for any team size
 - ✓ Lo-fi to hi-fi design in a single tool
- ✓ Integration with Photoshop and Sketch



www.uxpin.com