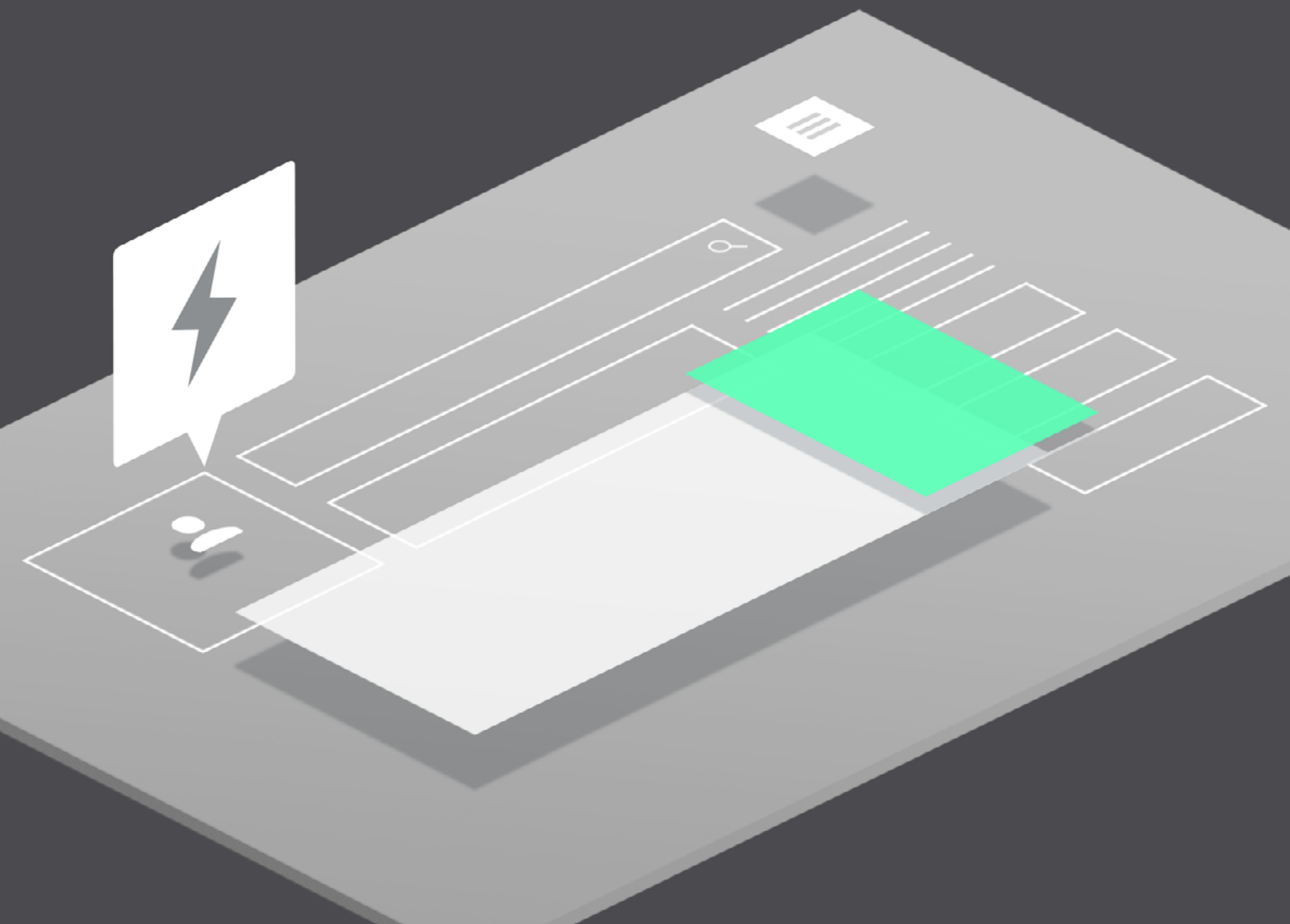


UXPin

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
5 Reasons Why You Need to Prototype	10
How Prototypes Improve Collaboration & Communication	12
How Prototypes Add Balance to Design	14
How Prototyping Makes Usability Testing Easier	16
Why Prototyping Is Mandatory for Mobile	18
Takeaway	21
2. Choosing the Right Prototyping Process & Fidelity	22
When to Start Prototyping: 3 Points of Convergence	23
How to Prototype: The Rapid Prototyping Process	29
What Is a Prototype: The 4 Dimensions	31
What to Prototype: 4 Ways to Combine Fidelity & Functionality	32
Takeaway	36
3. Traditional Prototyping Methods and Tools	37
Paper Prototypes	38
Wizard of Oz Prototypes	44
Takeaway	47

4. Digital Prototyping Methods and Tools	48
Presentation Software	49
Coded (HTML) Prototype	52
Prototyping Software7 & Apps	55
Takeaway	58
 5. Creating Prototypes for Usability Testing	 60
Knowing Your Users: Personas, Scenarios, and Experience Maps	61
Usability Tests Before the Prototype	65
The Right Users and the Right Tasks	66
General Advice for Testing Prototype Usability	69
Different Fidelities for Testing Prototypes	71
4 Content Guidelines for Testing Any Prototype	73
Takeaway	75
 6. 10 Prototyping Best Practices	 76
1. Know Your Audience and Your Goals	76
2. Prime Your Audience Beforehand	77
3. Involve the Users (Participatory Design)	78
4. Focus on Flows and User Scenarios	79
5. Keep Clicking Simple	80
6. Don't Neglect Animations	81
7. Sketching: The Prototype for the Prototype	82
8. Don't Let Coding Hold You Back	83
9. Use Prototypes for Usability Tests	84
10. Prototype Only What You Need – Then Stop	85
Takeaway	86

7. Wireframing & Prototyping:	
The Past, Present, and Future	87
Present: The Current State of Design	87
Present: The Current State of Prototyping	91
Past: A Prototyping Timeline	92
Future: The Age of Prototyping	95
Takeaway	98
8. Creating Interactive Prototypes from Photoshop Files	99
Importing from Photoshop – overview video	101
Importing from Photoshop into UXPin – step by step	102
General notes on using interactions and animations	107
Button: scrolling the page after click, changing style on hover	111
Form: triggering visibility on scroll	115
Form: interactive inputs	117
Form: interactions after signing up	118
Previewing and gathering feedback	120
9. How to Create Interactive Prototypes From Sketch Files	122
Overview Video	123
Importing from Sketch into UXPin	124
Prototyping Animations & Interactions	129
Previewing Animations & Interactions	131

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.

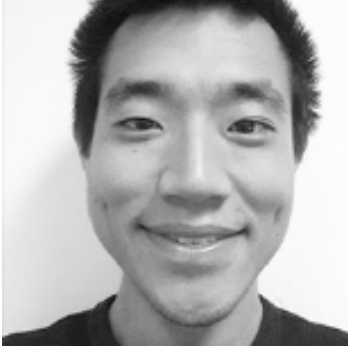
Prototyping helps unveil and explore human needs, opening the door to insightful interaction and more empathetic design solutions. If a wireframe is worth 1000 words, then a prototype is worth at least 10,000.

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

Happy prototyping,

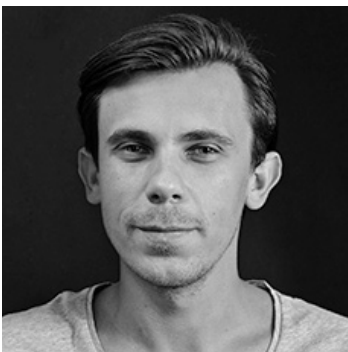
Jerry Cao

(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.

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.

5 Reasons Why You Need to Prototype

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

Just like you test drive a car before buying, you need to prototype before developing.



To better understand why you should prototype, let's look to Todd Zaki Warfel, the designer who wrote the book on prototyping... [literally](#). Warfel summarizes his 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 include:

- **Communication & 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.
- **Gauge feasibility while reducing waste** – Wireframes, mockups, and requirement documents live in paper, not reality. Prototyp-

ing allows teams to experiment, giving them the freedom to fail cheaply while learning more. Just take a look at how prototypes helped [increase estimate accuracy by 50% while reducing requests for clarification by 80%](#).

- **Sell your idea** – Prototypes can be great for pitching if you're working with skeptical clients. Experiencing the real-life website or app proves your vision more than a wordy description or mockup bogged down with notes.
- **Test usability earlier** – By user-testing a prototype, you're able to find problems and fix them earlier in the process, saving yourself a huge hassle of dealing with them when they're cemented in code. The same case study cited above saw 25% reduction in post-release bugs thanks to prototyping.
- **Set your design priorities** – We recommend prototyping early and often because prioritizing interaction design will keep you grounded in reality when you make static design decisions. 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. This works both externally – pitching to clients and stakeholders – and internally – in collaborating more deeply with your team, or rallying them to support a new idea (or at least play with it first before axing it).

Prototypes clarify internal communication in a few ways.

First, it creates a connection between designers and developers earlier, giving both the same goal to work towards. Different departments use different lingo and 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 ex-



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.

ploring 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.”

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. On a side note, prototypes also add flair to presentations – people can literally experience the “magic” of design: it’s the difference between seeing the blueprint versus exploring the model house.

Prototypes are the toys of design. They get people thinking about the crazy ideas that just might work.



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.

Philip van Allen, Professor and Designer at the **Art Center College of Design**, believes that prototypes help make decisions that **cannot be described by parameters**. For the designer, there are three main benefits of prototyping:

- **Decision-making** – Important design choices concerning ergonomics, shape, function, production – sometimes all at once – are finalized in the prototyping phase. A working prototype will give you instant feedback so you can make an educated decision (not just a heuristic one).
- **Focus** – With concrete feedback for all senses (instead of simply “guessing” what the final product will be like), prototypes ground you in user reality. UX priorities become apparent when you can experience them right in front of you.
- **Parallelism** – The design process doesn’t have to be sequential. Gathering feedback, setting requirements, and brainstorming new concepts and interactions can all happen at the same time while prototyping, and when done right will complement each other (we discuss the processes in detail later in this ebook).

Prototyping is when concepts become real,
requiring creativity balanced with practicality.



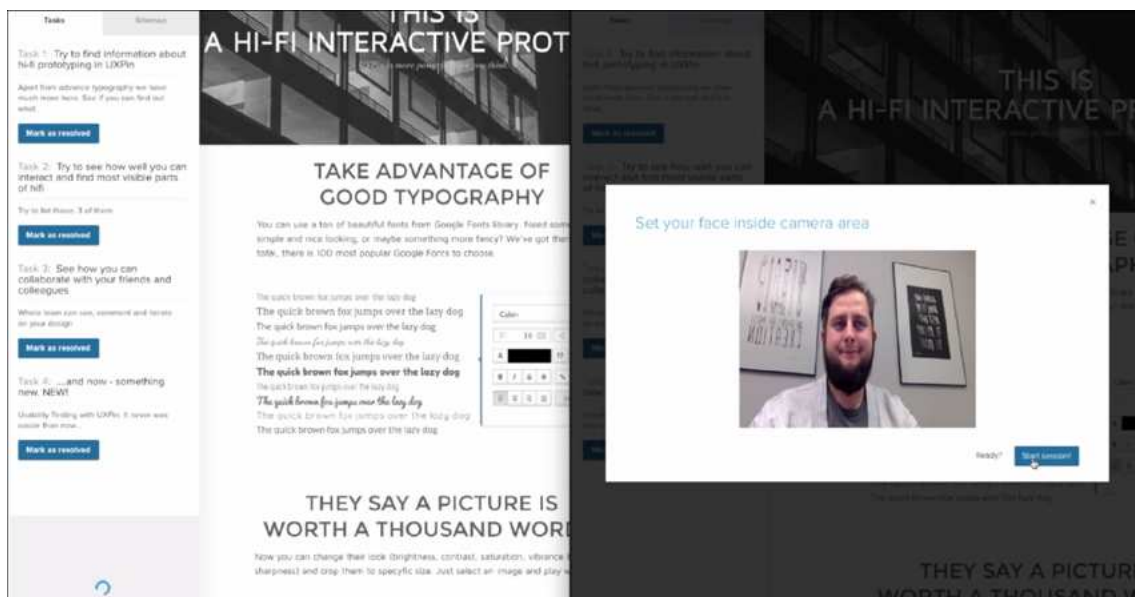
How do we know this? When we [worked on improving the Yelp website](#), we built first a [lo-fi](#) and then a [hi-fi prototype](#). Initially the interactivity was simple, but in designing it, we were confronted with decisions and obstacles that would not surface in a static design. Through prototyping, 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 (covered in [another e-book](#)), here are some broad basics to keep in mind:

- **Know the product** – Know your product well and what you want to test before you start. If you go into usability testing with a product that behaves erratically, or if you're not sure what needs to be tested, the results will be invalid.
- **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. Incorporate your findings into your iterations as soon as possible.

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.

Prototypes are to products what sketches are to wireframes & mockups.



Why Prototyping Is Mandatory for Mobile

While prototyping for desktop products is merely recommended, for mobile it is mandatory. Mobile devices are used in more environments than desktop devices, which leads to far more user scenarios and use cases. What this means, of course, is that while you might be able to get away with not prototyping for desktop, you'll have no such luck with mobile.

Rachel Hinman, the **Nokia** Research Center's Senior UX Researcher, agrees that the need for mobile prototyping far outweighs the already important need in the desktop field. Look at the chart below, [based on Hinman's slideshow](#), to see a comparison between the stability of the two:

	PC	Mobile
Environment	typically predictable	highly variable and changing
Position	often seated, relaxed	on-the-go
Input	full keyboard, mouse	limited keyboard, technologically complex gesturing
Screen	large	restrictive
Multitasking	easier with keyboard, RAM, mouse, and large screen	more difficult conducive to quick, individual tasks

Donna Lichaw, **Product Specialist at Greatnorthelectric**, [also agrees](#), and compares general reasons to prototype versus mobile-specific reasons. Gestures add a whole new dimension of interactivity that doesn't even exist for desktop, so prototyping is the only way to test the near-endless combinations of gestures and animations.

We've found the following to be the most compelling reasons for mobile prototyping:

- **Flow makes or breaks the mobile experience** – Because mobile experiences span several pages, user flow should be front and center on the list of design priorities. Just visit [UXArchive](#) to see how some of the top companies like Snapchat, Google Maps, and others architect their mobile flows.
- **Animations make the experience fun & desirable** – Animations are the core element to mobile micro-interactions, with

gesture-triggered effects like timing, velocity, and bounciness adding an element of fun while [providing visual feedback and affordance](#). Check out [this piece](#) to see the contrasting views on animations from **Google Material Design** and **Apple iOS**, then see [20 examples](#) of these best practices in action.

- **UX mistakes are more expensive** – Mobile app costs range from [\\$6,000 to upwards of \\$200,000](#), with the average cost hovering around \$6,400. Dealing with the App Store can also be a headache, so some up-front prototyping can go a long ways in saving time, money, and sanity.

UXPin allows for responsive [animated prototyping](#) and [built-in usability testing](#). Create your tasks, then watch your user's screen and facial reactions as they interact with your design. After each test, you can generate a video clip which captures every word, reaction, and click.

Mobile UX adds a new dimension of interactivity
that can only be perfected through prototyping.



Takeaway

We here at [UXPin](#) 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.

While this chapter was more of an introductory overview, in the next chapter we'll get into how to implement prototyping into your design process, and which fidelity will work for you.

Choosing the Right Prototyping Process & Fidelity

Where design and development diverge and converge

As the phase that merges creativity and feasibility, prototyping is the cornerstone of the design process. The first workable version of the product is created and features are either realized or scrapped. The product finally comes to life.

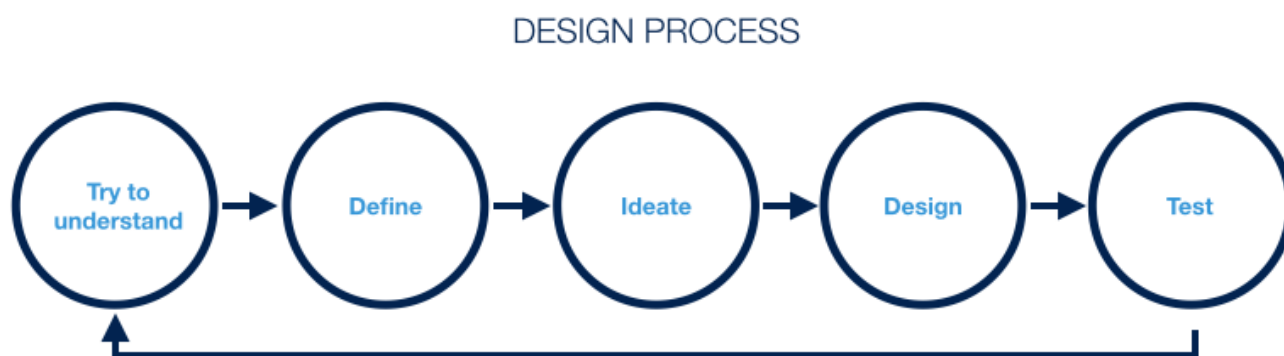


Photo Credit: Marcin Treder, [UXPin](#)

But as such a transformational step, there's lots of different styles, methods, and processes to do it. Ultimately, the nature of your project and your specific needs will determine the appropriate process and level of fidelity, but here we'll outline your choices so you can find the best fit.

We'll start by examining how prototyping should fit into your process compared to wireframing and mockups, then look at four different combinations of visual fidelity and functionality for prototypes.

When to Start Prototyping: 3 Points of Convergence

There's no green light that will magically blink when it's time to start prototyping. How and when to prototype is the subject of much debate in the design world (as you can see by the comments in this piece), and various differing theories and strategies have emerged. As discussed in [The Guide to UX Design Process & Documentation](#), the traditional linear process looks something like this:

- **Sketching** – Brainstorm by drawing quick and dirty sketches on paper.
- **Wireframing** – Start laying out the skeletal framework with boxes and rough shapes.
- **Mockups** – Inject detail into wireframes with colors, typographies, photos, and other visual design elements.
- **Prototyping** – Add interactivity to mockups by stitching screens together for basic prototypes or adding animations/interactions for advanced prototypes.

- **Development** – Code in language of choice to turn the prototype into the final product.

However, with the popularization of new ideas such as [lean UX](#) and [rapid prototyping](#), plus the school of thought that wants to [get into coding as quickly as possible](#), this traditional sequential method is becoming outdated. Below we'll look at some new variations, and explain their advantages and disadvantages.

1. Wireframing & Prototyping

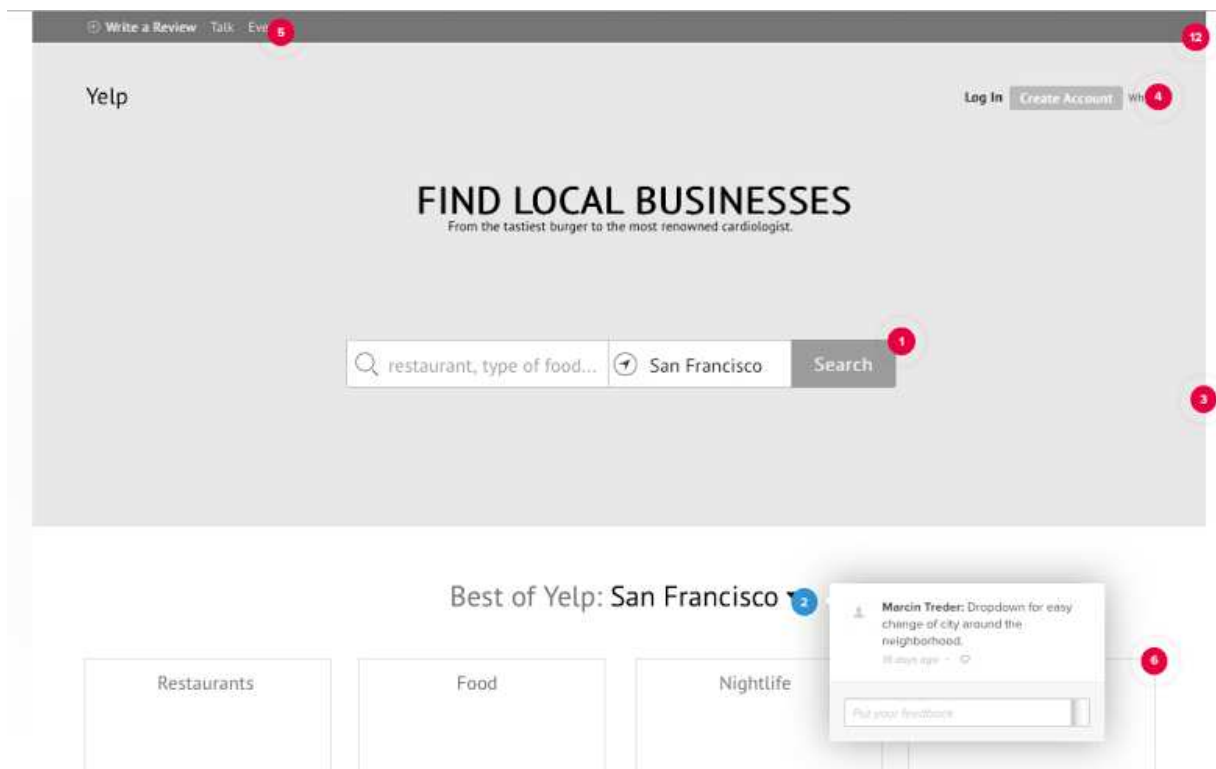
Some designers prefer to get to prototyping right away, and for good reason. This is a core concept of Lean UX, devised by Jeff Gothelf, Director of UX at **Neo Innovation Labs** (you can hear him describe it [in this thorough webinar](#)). Lean UX emphasizes prototypes as “the fastest way between you and your code.”

The advantages of a Lean UX process are easy enough to understand:

- **Faster** – Skipping and consolidating phases will get you to the end product faster, but possibly at the cost of quality.
- **More efficient** – The nature of the method is to avoid waste, so the work done will be only the essentials – no time spent on “busy work.”
- **Experience, not deliverables** – Part of “trimming the fat” is min-

imizing documentation. Teams communicate better and faster, improving collaboration in designing the experience. State the design vision, then iterate with that in mind.

One of the core processes of Lean UX is going straight from a detailed sketch or wireframe into a prototype. This can be done as simply as adding a few animations and basic interactivity to your prototype, testing with a minimum of 5 users, and iterating as needed.



Source: *UXPin Low-Fidelity Yelp Design*

This is very similar to our early process to [redesigning Yelp's website](#). Because a major downside to Lean UX is that it doesn't work well with distributed teams (sketches need to be discussed, not handed over), we adapted the process to suit our 3-office project by also drawing wireframes collaboratively in [UXPin](#).

Prototypes are the fastest way between you and your code.



Our design vision was to increase frequency and time on site for occasional Yelp users. We took our simplified wireframe and added a few basic interactions (like transitioning to the search screen), plus commented on the features we had yet to incorporate. You can see that the [low-fidelity Yelp prototype](#) was far from complete, but far more telling than just a wireframe. Most importantly, we built our prototype quickly and let the design serve as the documentation.

If you'd like to know more about Lean UX and rapid prototyping, [UX Matters features a great discussion thread](#) about it in which 9 industry experts weigh in with their opinions.

For more information on wireframes, [download our free ebook, *The Guide to Wireframing*](#).

2. Mockups & Prototypes

Mockups are like a better-dressed wireframe. Neither require functionality, but mockups give you a better idea of what the final product will look like, and at times suggest how it will function. Usually mockups are built from wireframes, but this is not always the case, as we describe in [The Guide to Mockups](#).

The main difference between building a prototype off a mockup instead of a wireframe is that mockups automatically provide the baseline design for a mid- to high-fidelity prototype. Because wireframes are so bare-bones, prototypes built from them are still low fidelity. You'll definitely want to consider this since, while low-fidelity prototypes are great for quick collaboration and exploration, high-fidelity prototypes can be [better for product definition and estimates](#).



Source: [UXPin High-Fidelity Yelp Design](#)

This can be a helpful process for slight redesigns of existing websites since you already have the high-fidelity assets. Of course, for drastic redesigns or completely new sites and apps, low-fidelity prototyping is recommended.

In the Yelp redesign, the second half of our design process was creating a high-fidelity prototype from our mockup (which we imported into UXPin using our [Photoshop and Sketch integration](#)). Because all the layers were preserved, it didn't take much time to copy the same interactions that we did in the low fidelity prototype. In the end, [the result was a hi-fi prototype](#) that looked like the end product and had a few basic functions.

3. Coding & Prototyping

While we discuss coded prototypes in detail later in this ebook, we just want to briefly make some general comments here. As discussed earlier, introducing code early into the design process has a lot of benefits, namely in having a more solid foundation when beginning development, and in cutting down the amount of needed revisions. But it's not so much a question of should you code with the prototype, but can you.

As Andrew Fitzgerald, Senior UX Architect for **Deloitte Digital**, [points out in post for UX Booth](#), most designers have a “complicated” relationship with code. When it comes to coding prototypes, it helps to start with a sketch and then dive straight into HTML or another language of choice (just make sure you check your work on mobile and tablet). This lets you explore ideas on a whiteboard or on paper where they're easy to alter so you aren't trapped with mediocre concepts just because they're cemented in code.

Regardless of whether you choose to add coding to your prototype, make sure you get the developers involved in the process. You don't want the first time your developers see the prototype to be when you email it with a long list of notes and instructions.

How to Prototype: The Rapid Prototyping Process

Rapid prototyping is not so much a separate process as it is a filter for efficiency. It can be considered a core principle of Lean UX, but it applies to any of the prototyping processes we previously described.

Rapid prototyping is all about revising quickly based on feedback and shifting to multiple prototyping approaches based on the requirements. A rapid prototype is not designed to evolve into fully functional solutions, but is meant to help the team visualize and craft the UX of the final product.

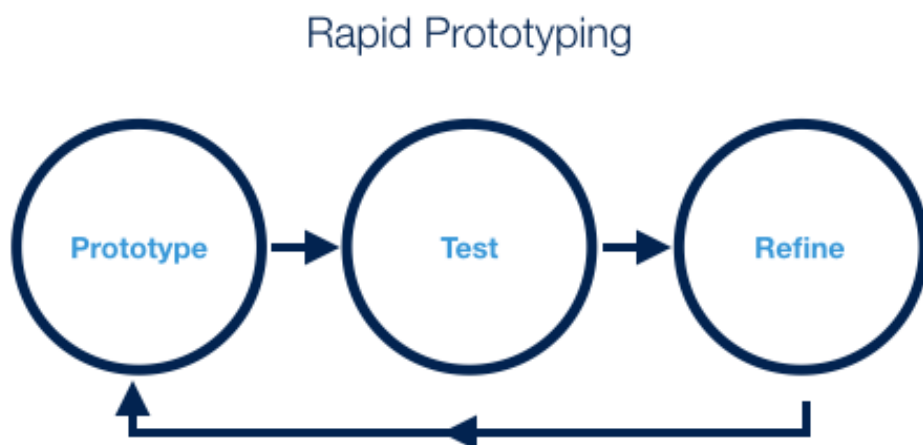


Photo Credit: Marcin Treder, [UXPin](#)

Rapid prototyping isn't so much a separate process as it is a filter for efficiency.



In an exemplary article for *Smashing Magazine*, Lyndon Cerejo, **UX Strategist for Capgemini**, explains the process of rapid prototyping. Rapid prototyping can be broken down into the following steps:

- **Scope the prototype** – Figure out what exactly you want to build. Any advanced functionalities or interactions, or changes to workflow and design will warrant prototyping.
- **Craft personas & user scenarios** – Once you know the area for prototyping, you need to know the who, why, and how of the UX. For example, if you're building a photo sharing app, one scenario might be Vain Vanessa choosing a new filter for her latest selfie because she wants to attract more followers.
- **Iterate & gather feedback** – Start with breadth, and then dive into depth. You'll usually start with a horizontal prototype (main landing pages and homepage) and then iterate into vertical prototypes based on specific user flows (like uploading, editing, and tagging friends in your photo). You might start on paper, but finish in a high fidelity tool after multiple rounds of feedback.
- **Repeat** – Do it all again until the experience you're crafting is nice and polished.

This strategy could also work well for customer feedback on MVPs, which we discuss in *The Guide to Minimum Viable Products*.

What Is a Prototype: The 4 Dimensions

Now that we've examined the processes, let's look at the physical characteristics of prototypes. We can analyze all prototypes based on four qualities: **representation**, **precision**, **interactivity**, and **evolution**. These qualities vary for each method, but offer insight into which can be helpful for you.

- **Representation** – the physical form of the prototype, whether a paper mobile device or a desktop HTML file.
- **Precision** – “fidelity” in other words; this describes the level of detail and realism, from a some rough sketches to a highly polished simulation of the real product.
- **Interactivity** – how much a user can interact with the prototype, ranging from a “watch only” presentation to complete interactivity.
- **Evolution** – measures the expected lifecycle of a prototype, whether it's meant to be quickly built and then thrown away (as with rapid prototyping), or whether further iterations will be built upon it until the final product.

As we discuss the different fidelities and functionalities below, consider how each combination rates in these four categories, then decide based on your own needs. We'll also discuss specific prototyping tactics (like paper, Keynote, specialized tools, etc.) in the next chapter.

What to Prototype: 4 Ways to Combine Fidelity & Functionality

Prototypes can be organized into 4 categories, based on how you combine high- or low-fidelity visuals and functionality. All of these types can be useful – but in different scenarios and for different needs.

Fred Beecher, **UX Lead at The Nerdery**, believes that there's no such thing as high or low fidelity – [only the right fidelity](#). While we'll discuss the different types of prototypes in the next chapter, let's explore the different combinations here so you can select the right fidelity.

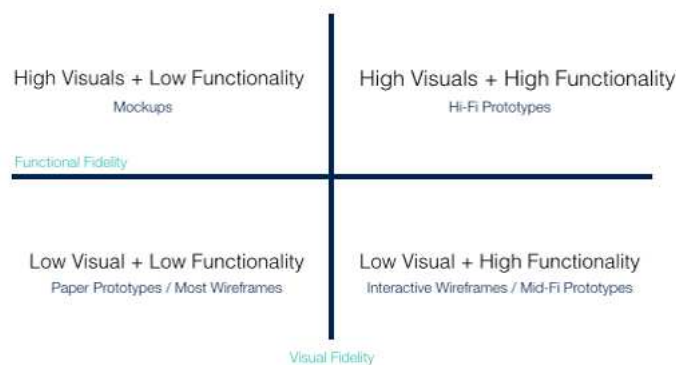


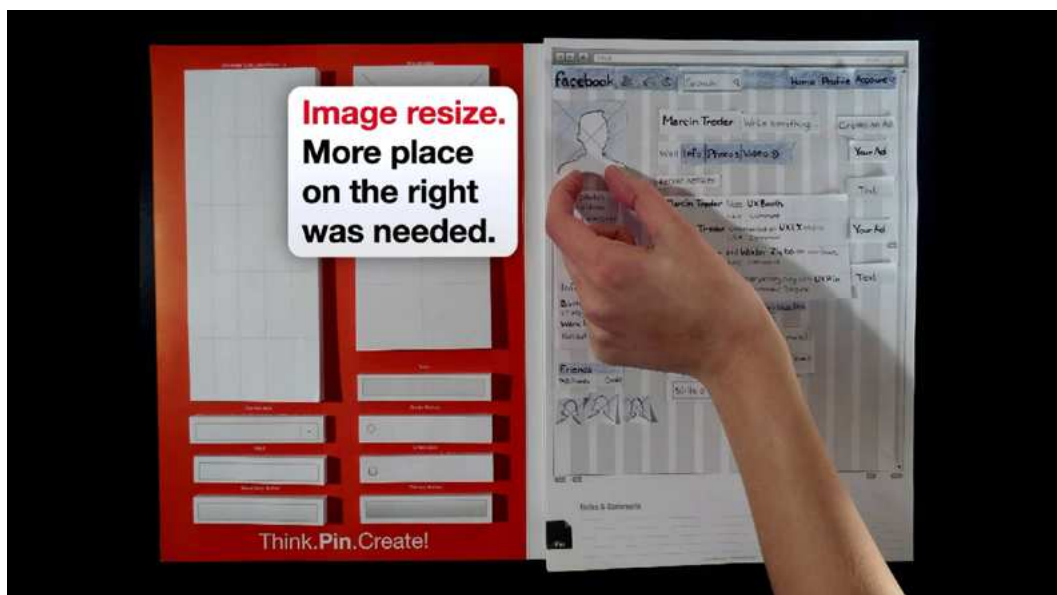
Photo Credit: Marcin Treder, [UXPin](#) Based on: [Integrating Prototyping Into Your Design Process](#) by Fred Beecher

There's no such thing as high or low fidelity – only the right fidelity.



1. Low Visual + Low Functionality

The main advantage here, at the cost of both visuals and functionality, is speed. These prototypes can be made quickly and quite easily, and can be thrown away and replaced just as easily. A paper prototype is the perfect example: it's quick to make, doesn't have much functionality, but puts something physical in your hand to answer structural questions.



Source: [UXPin](#)

These can be built over and over until some fundamental questions about UX concepts, feature completeness, and high-level page flows are answered. They are the first prototype that will likely be created as part of the rapid prototyping process.

2. Low Visual + High Functionality

Considered low to medium fidelity, these are the simplest prototypes that are usually created in a prototyping tool like [UXPin](#) or with code like HTML. Basically, these prototypes are “interactive wireframes.”



Source: Mid-Fi Prototype via [UXPin](#)

These prototypes are mainly created for:

- Evaluating and testing usability
- Testing proof of concept
- Gaining validation from stakeholders
- Supplementing formal documentation for developers

With their simple visuals, these prototypes are ideal for focusing purely on interaction design. Since they have a higher level of functionality, they can work well for usability testing. A lower level of functionality might not provide enough usability for test participants, while a higher level of visuals may be a waste of time if the interactions aren't effective.

3. High Visuals + Low Functionality

This combination works well as a “mockup +” that serves all the purposes of a mockup – finalizing visual decisions, impressing stakeholders, etc. – but with the bonus of some limited interactivity. These are typically created by adding basic animations (like clicking to another screen) to an existing mockup.

If you don’t plan on having many animations in your website, these are great for testing the flow of content. You can image map some screens together in the browser or print out screenshots for a prettier paper prototype.

4. High Visuals + High Functionality

The hi-fi prototype is just one step below the finished product. With nearly complete visuals and functionality, this style of prototype can sometimes be released publicly as an MVP to generate feedback and test usability. **Apple** actually prefers this method, and [takes it to an extreme](#) in building multiple high-fidelity prototypes that are essentially finished products.



Source: Hi-Fi Prototype via [UXPin](#)

These prototypes are a good fit if you're just tweaking an existing design, testing with users who aren't very tech savvy, or developing your site or app with outsourced programmers (less chance of misinterpretation).

Takeaway

More than the other phases of design, prototyping has a wide versatility in how it can be used and what it can achieve. Even before we get into the types of prototypes, the initial decisions (rapid prototyping vs. higher quality, basic vs. detailed visuals, coding vs. no coding) will all lead to different outcomes in the design process.

Now that we've explained the basic categories of prototypes, in the next chapter we'll explore the different methods, tools, and styles of prototyping.

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.

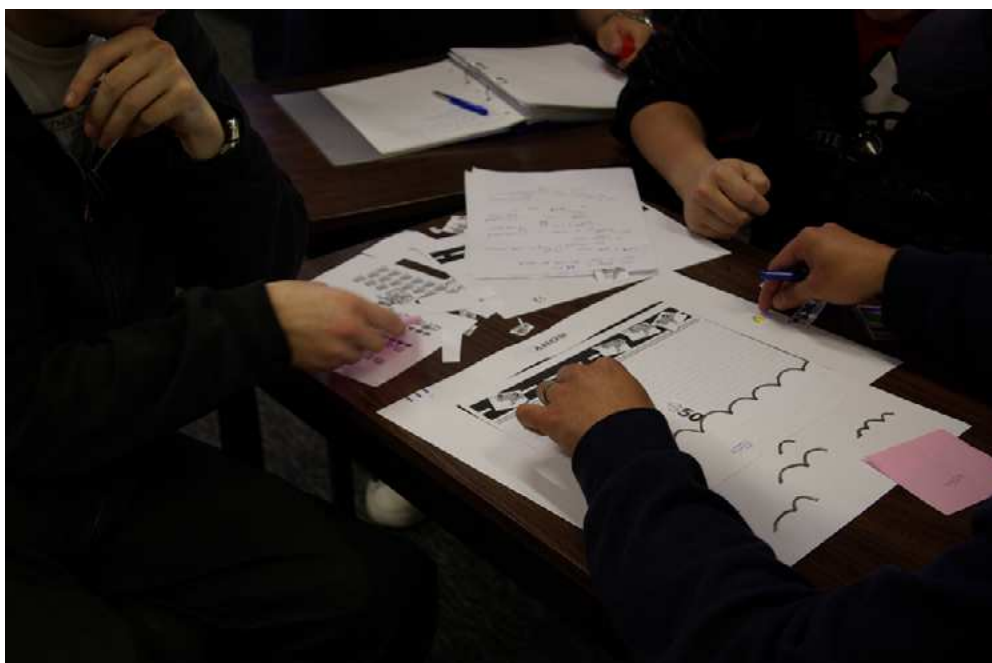


Photo Credit: [Samuel Mann](#). [Creative Commons CC BY-SA 2.0](#)

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. Shawn Medero, **UX Lead at the University of Washington**, believes that [starting on paper can be a surprisingly helpful tactic](#) as interfaces become more complex and development schedules become shorter. Here's a few reasons why:

- **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.



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.
- **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.
- **Easy documentation** – Past versions of the prototype are right there in front of you, allowing for part of the design to be the documentation. Additionally, technical notes can be attached to the prototypes themselves.

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 ex-

clusive – they can complement each other well if you play on each's strengths.



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

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.

- **They can generate false positives** – While paper prototyping can be useful for individual processes, you need to explain the

context. Otherwise, you might get feedback based on your efforts (“How creative!”), not based on your actual product (“The navigation is confusing.”).

- **No gut 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.

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

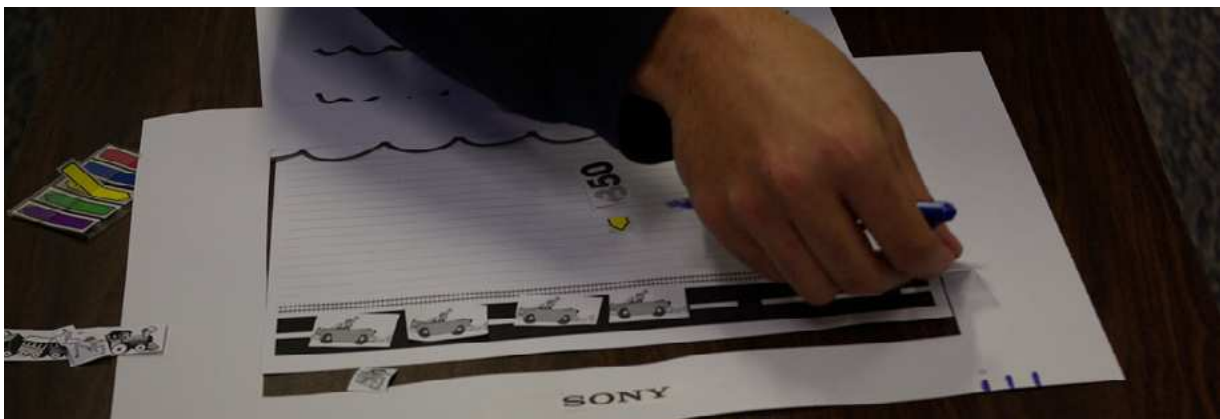


Photo Credit: [Samuel Mann](#). Creative Commons CC BY-SA 2.0

the most out of paper prototyping when you treat it more as an informal conceiving 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.

The first thing to know about paper prototyping is that someone has to play the role of the “human computer.” [In his list of](#)



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

[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

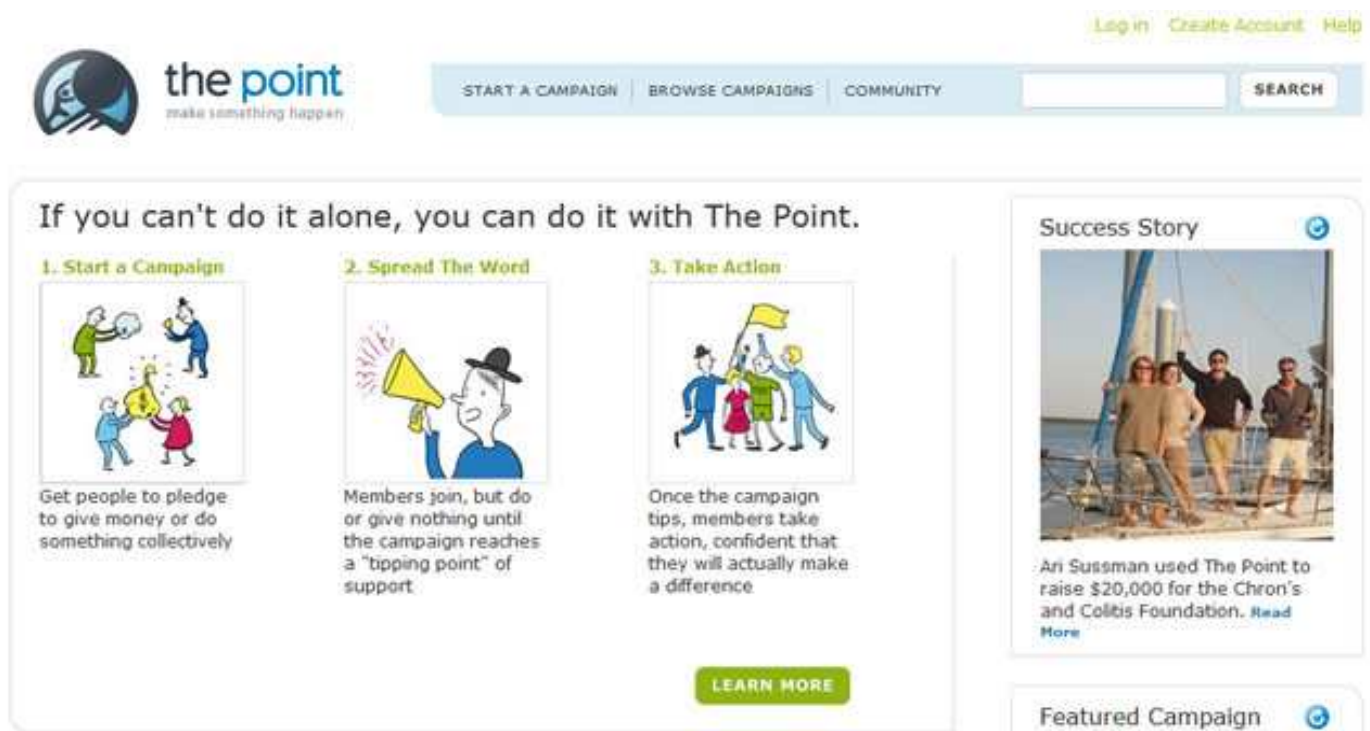
Perhaps the most successful proponents of the Wizard of Oz are David and Tom Kelley, Founder and Partner at **IDEO**. [In their](#)

[love letter to prototypes](#), they tell the story of how the IDEO team wanted to pitch their proposed “smart car” features to an automaker. 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.



Source: [10 Massively Successful Minimum Viable Products](#)

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 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.

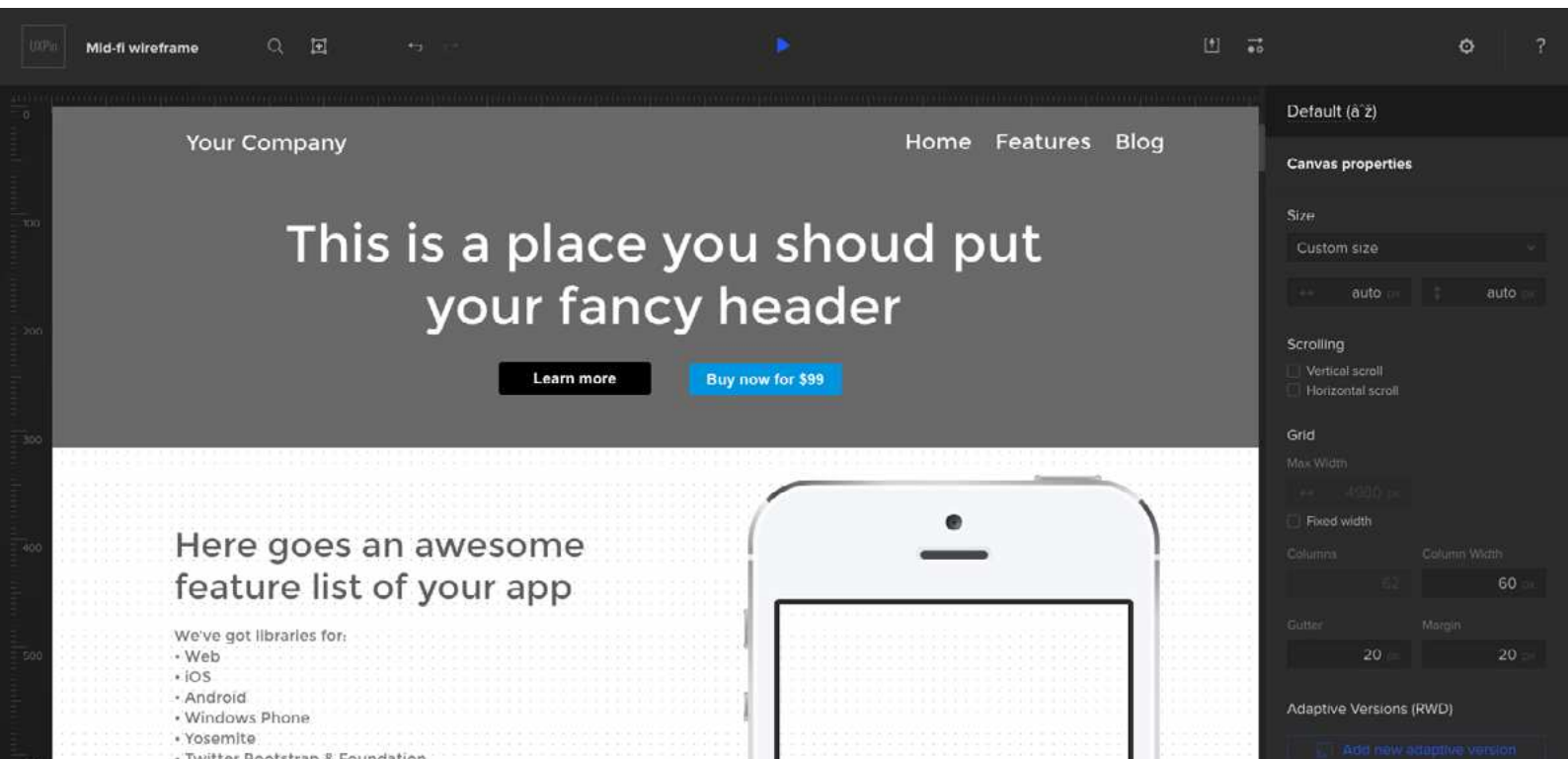


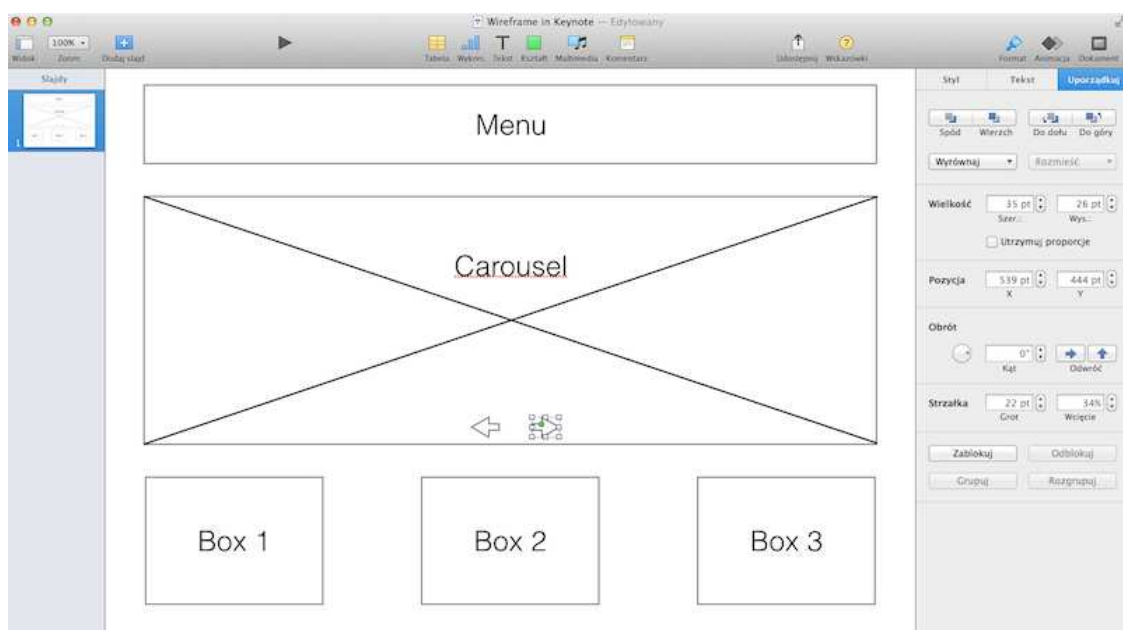
Photo Credit: www.uxpin.com

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.

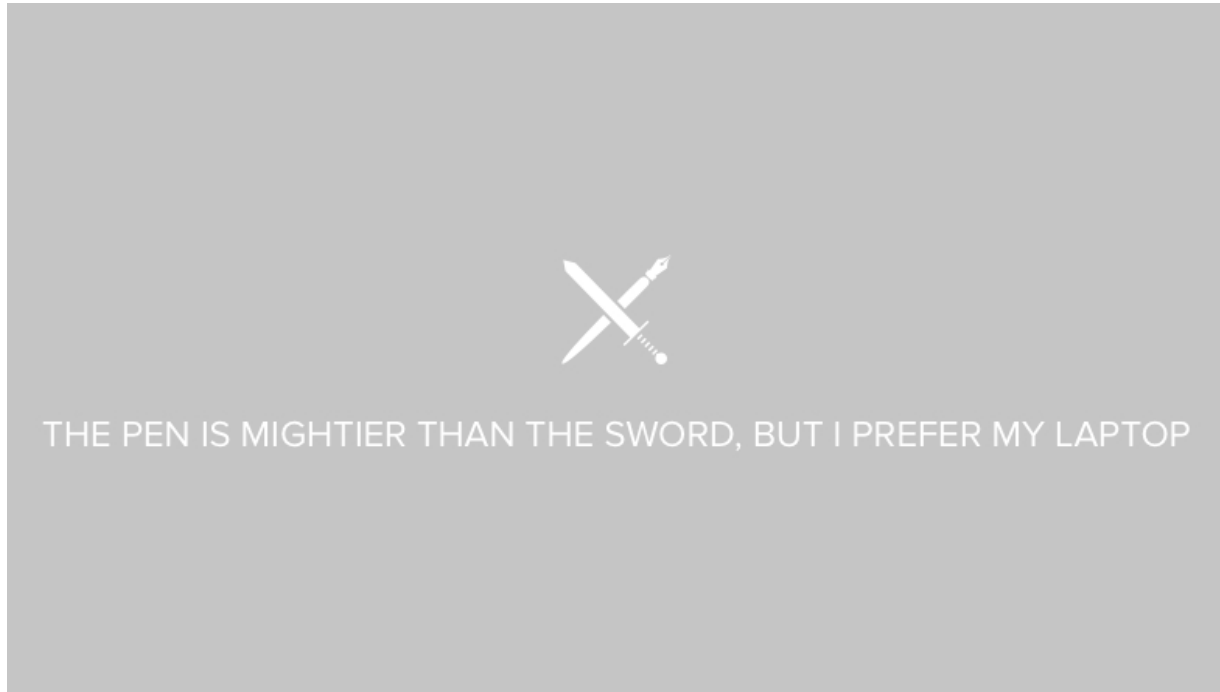
After creating the first version of the app in a little over a month, ZURB used the time they saved for final refinements.

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.



Creating Prototypes for Usability Testing

How to Develop & Test Prototypes for Optimum Usability

Why worry about usability testing so early when prototyping already has a big enough to-do list? Because unless your prototype is usable, then all your testing will tell you is that generally people don't like terrible products.

Usability isn't something you just can cook up in any one phase of design, but must be developed and refined throughout the entire process. If you want the best end product, you have to anticipate

real user scenarios from the beginning. Usability testing should be the *last* place to start thinking about usability.

With that in mind, let's look at how to keep usability in mind as you build the prototype, how to test usability before you have a prototype, and tips for testing with prototypes.

Personas act as another person in the room when making UX decisions.



tweet this

Knowing Your Users: Personas, Scenarios, and Experience Maps

The first step in delivering the perfect product to your target audience is knowing your target audience. While you can't really go out and meet each and every future user, you can use strategies to predict who they are and how they'll act.

1. Personas


To reiterate our description in our free ebooks [Web UI Design Best Practices](#) and [The Guide to Usability Testing](#), a persona is a fictional character with behavioral and psychological characteristics drawn from your target audience. They help to serve as another person in the room when making UX decisions.

For example, you might ask “What kind of navigation system would Sally the Seasonal Shopper prefer?” or “Do you think Pete the Power User would be too busy to see this link in the corner?” By having a “real” human being in mind, it's easier to keep you focused on just who exactly you're building this for, and what they would appreciate.

If personas are the who, then user stories are the how.



The trick to personas is to make them as detailed as you can. Product Manager **Alan Klement** [suggests treating them more](#)



JONATHAN VIZZIER

"Design isn't just how it looks, it's how it works."

Demographics

- 27 years old
- Masters in Visual Design
- Visual Designer
- Single
- Earns \$85K per year

Behaviors & Beliefs:

- Obsessive over visual quality
- Hates when product managers use the word "just" before describing feature tasks
- Wants to be as involved in the design process as possible
- Loathes jargon, wishes people would get to the point

Characteristics & Attributes (0 to 5)

- Design experience: 3
- Education: 4
- Tech Savviness: 5
- Ambition: 5
- Workload: 5

Goals:

- To build a strong portfolio, regardless of whatever job I'm at
- To start mastering UX design by the end of this year for a career transition
- To rise up in his company and start getting assigned larger-profile projects
- Wants to help the product team see the value of emotional design, not just "core <P>Pis"

Photo Credit: www.uxpin.com

as “characters.” Flesh out your personas with personal details about their jobs and life stories, their personal preferences, their daily anxieties, and their main motivations. The more realistic you make your personas, the more able you’ll be to predict their behavior with your UI, and the closer you’ll get to a final product that fulfills their needs.

2. User Stories/Job Stories

If personas are the *who*, then user stories are the *how*. A good user story takes the persona and gives them both a task and a context surrounding it. After all, interactions and animations aren’t just for flair – they need to help users achieve their goals.

If personas are the who, then user stories are the how.



Author of *Digital Project Management* **Kristofer Layon** stresses that the right level of specificity is the key: as an example, “As Power User Pete, I need to be able to sort my email quickly so I can get back to my job.”

You can take these a step further by using job stories instead (popularized by **Intercom**), which favor motivation over implementation.

Instead of the “As a ____, I want to ____ so that ____.” framework, the above job story removes some of the ambiguity of a persona by focusing on causality instead. Job stories can be more actionable because the “when” focuses on the exact context of use. “When spending a month in Honolulu, Donny Designer suddenly remembers to sort his email on his cell phone, so that he won’t return with 1000 unread emails.”

User stories and job stories don’t just help you anchor the product in real human behavior, but can also provide context to participants during usability testing. When it comes time to test your prototypes, make sure you describe the scenarios for each task.

3. Experience Maps

Experience maps take user stories to a whole new level with more details at each level of the user's experience. For our purposes, they can show you how the final product fits into the lives of the user – and so how the prototype can evolve into the product that's the perfect fit.

As Chris Risdon, Design Director at Adaptive Path, [explains in a post for the UX agency](#), an experience map can display a minutiae of data, or could just be a simple timeline of thoughts and emotions during each stage. Whether you choose a simple timeline or a fully illustrated map, you need to hit these points:

- **Uncover the truth** – Scour your company for quantitative and qualitative data on the experiences you want the prototype to provide. Look at a variety of sources like web analytics, call center logs, and customer surveys/interviews. Triangulate your data so you fill any knowledge gaps.
- **Chart the course** – Experience maps should contain the lens (persona through which the journey is viewed), the journey model (touchpoints across all channels), and takeaways (design principles and insights from the mapping process).
- **Tell the story** – Your map needs to have a beginning, middle, and end. Identify which insights are important to the narrative and which are nice-to-haves. Like a good poster, your map needs

hierarchy (what stands out immediately versus what sinks in later).

For an exhaustively thorough resource, check out [Adaptive Path's free ebook on mapping experiences](#).

Usability Tests Before the Prototype

Usability testing doesn't have to start with prototyping – in fact, if you have the resources to start sooner, you certainly should. While mostly conceptual, these tests can pinpoint the best way to structure your prototype's navigation and information architecture. The most common pre-prototyping tests include:

- **Card Sorting** – Simple and steadfast, this test reveals how users would prefer your product's information architecture. All the elements of your product are written on cards, and the test-takers are asked to organize them under predefined categories (“closed”) or under ones they've thought up (“open”). For details, read Donna Spencer's [Card Sorting: A Definitive Guide](#).
- **Tree Testing** – The “sister test” to card sorting, tree testing evaluates the effectiveness of existing information architectures. Users are given a basic, stripped down map of the site/app/etc. and asked to click through to complete certain tasks. The test monitors if they choose the correct or easiest route, and if not,

what got them lost. Founder of **MeasuringU** Jeff Sauro [explains the details](#).

- **Interviews** – Sometimes the best way to understand your users is to simply ask. It sounds simple enough, but the nuances and strategies for user interviews are endless. Kate Lawrence, UX Researcher at **EBSCO Publishing** [gives some tips](#) on how to run these specifically for usability testing.



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

The Right Users and the Right Tasks

While usability tests are all different, all of them need users, and most of them involve tasks. Since these two elements are prominent in all usability testing, we'll briefly explain how to best deal with both. (If you'd like more information, we explain the specifics in our free [Guide to Usability Testing](#) and [User Testing & Design](#).)

1. Recruiting Users

After all the work with personas, by now you should have a clear idea of your target users. It also helps to segment your users based on behavior.

In fact, you shouldn't obsess over demographics. The biggest differentiator will likely be [whether users have prior experience or are knowledgeable about their domain or industry](#) – not gender, age, or geography. For example, [when we ran usability tests for improving Yelp's website](#), we divided our group of target users into two segments: those with accounts and those without.

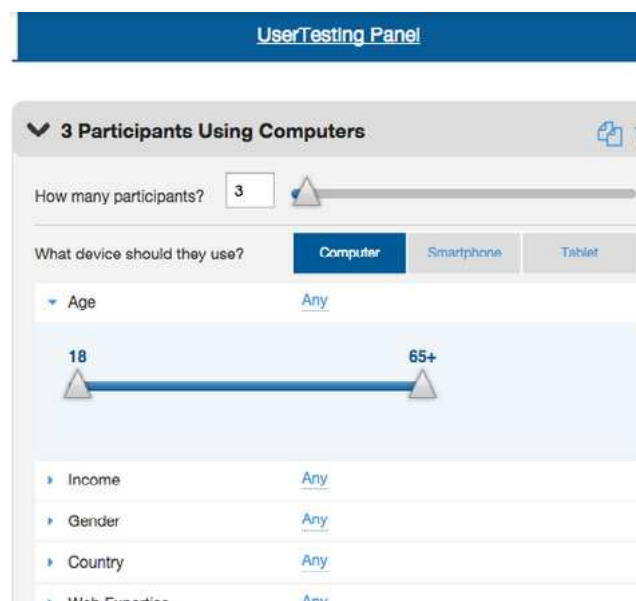
The image shows a screenshot of the 'UserTesting Panel' interface. At the top, there's a blue header with the text 'UserTesting Panel'. Below that, a grey bar indicates '3 Participants Using Computers'. The main form allows users to specify recruitment criteria. It includes a 'How many participants?' field set to '3' with a slider. A 'What device should they use?' section has three buttons: 'Computer' (selected), 'Smartphone', and 'Tablet'. Below this, there are expandable sections for demographic filters: 'Age' (set to 'Any' with a range from 18 to 65+), 'Income' (set to 'Any'), 'Gender' (set to 'Any'), 'Country' (set to 'Any'), and 'Web Experience' (set to 'Any').

Photo Credit: www.usertesting.com

Knowing who to recruit is just the first step. The more involved part is finding and recruiting them. Jeff Sauro [outlines the 7 best ways to locate the ideal users for your testing](#), ranging from Craigslist to outside agencies. You can also get this free guide from Jakob Nielsen, which includes [234 tips for recruiting users](#).

2. Writing Tasks

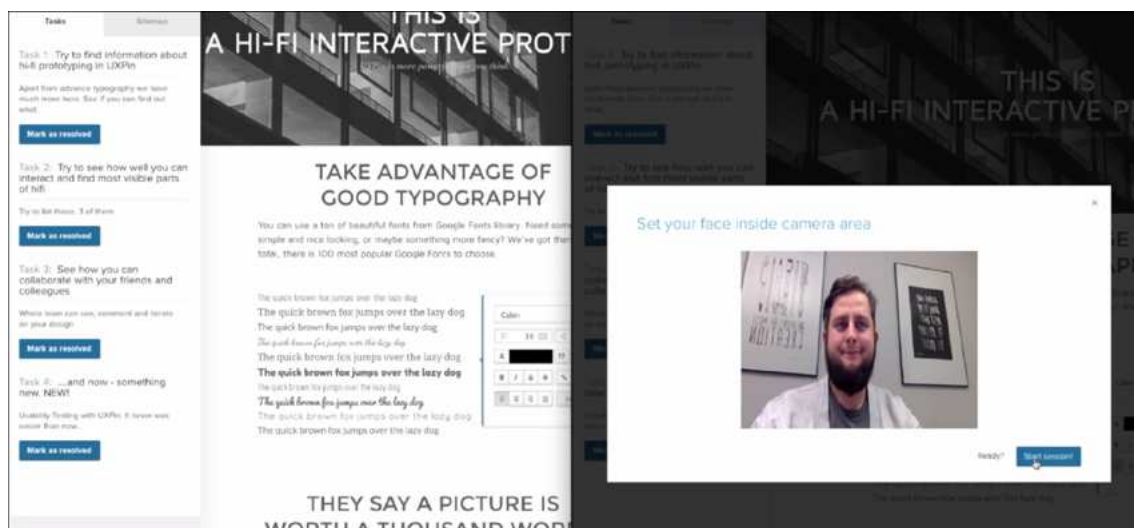
Tasks determine what the user actually does during the test, and therefore determines what usability factors are being examined. Naturally, they should be crafted with thought and care based around what your goals are. Tingting Zhao, Usability Specialist for **Ubuntu**, [describes some distinctions to keep in mind when designing task](#). There are 2 main decisions for creating tasks:

- **Direct vs. Scenario** – A direct task is one that is strictly instructional (“Search the website for a Tandoori chicken recipe.”), while a scenario task comes with context (“You’re hosting a dinner party for some old friends, and you need a Tandoori chicken recipe because of a special memory you all share.”). Direct tasks work best if you’re testing technical data, while scenario tasks are better in all other cases, as they help the user forget they’re in a testing environment.
- **Closed vs. Open-ended** – A closed task has clearly defined success criteria, while an open-ended task can be completed multiple ways. Closed tasks check specific functionalities, while open-ended tasks are better for understanding how your users’ minds work. A closed task would be: “Your friend is having a birthday this weekend. Find a fun venue for up to 15 people.” An open task would be: “You heard your coworkers talking about the iWatch. You want to learn how it works.”

Don't underestimate the importance of task design, even down to the wording you use. To learn more about writing tasks that work, **UserTesting** has a [free ebook on usability testing](#).

General Advice for Testing Prototype Usability

One of the first questions usability testers ask is whether or not it should be moderated. [While there are a lot of good reasons for unmoderated tests](#), for prototype tests we recommend moderation. Given the “incomplete” nature of prototypes, chances are that users will have questions about the UI that a moderator will have to answer.



In UXPin, you can actually test your prototypes quite easily. Download the Chrome plugin, set up your tasks, and start testing with your users. As you can see in the [testing overview](#), UXPin generates video clips that let you see every click, understand user's thoughts, and watch their screens and faces.

Furthermore, the level of complexity will also determine whether the moderator should be remote or in-person. Considerations about

mobile and web differences should be taken into account, as some mobile use cases are better suited for a lab setting (for observing gestures/body position/etc.) For more information on the differences between mobile and web usability testing, download the [*Guide to Usability Testing*](#).

Another common mistake in testing is to stop or alter the test if the user experiences difficulty. Since the goal of usability testing is to find and solve difficulties, this situation could actually make the test a success. If, for example, the user strays off onto paths that haven't been developed yet in the prototype, you could ask them why they went there and what they would have liked to accomplish. A few follow-up questions about the obstacles may yield more valuable feedback than a user with a "perfect run."

When testing your prototype, it's more important to note what isn't going to plan.



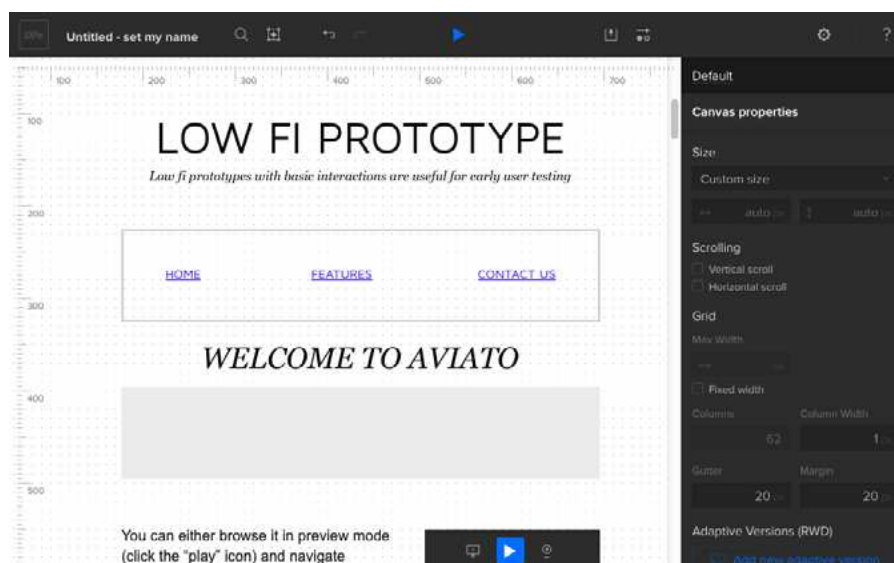
tweet this

Different Fidelities for Testing Prototypes

While some believe in testing early with rough prototypes and others advocate testing higher fidelity prototypes, we believe the best approach is to test at every fidelity possible – and as often as possible. Chris Farnum, the Senior Information Architect at **Enlighten**, [explains the pros and cons of each type](#). As we'll describe below, lower fidelity tests are better for testing concepts while higher fidelity tests are more suitable for testing advanced interactions.

1. Low Fidelity

Lo-fi prototype usability tests, including paper prototypes, can work at the early stages of development, but become impractical later on. As we explained in Chapter 3, lo-fi prototypes have the advantage in minimal time and resources. Lo-fi prototypes also encourage more honest criticism, since it's immediately clear that it's just a work in progress.



Source: [UXPin](#)

However, at the later stages, when usability tests check advanced functionalities, lo-fi prototypes stop becoming helpful since you've hit the fidelity limit. This is especially true for paper prototypes, since you need a “human computer” to manipulate all the parts, and that can become extremely difficult as you add menus, interactions, pages, and elements.

2. High Fidelity

Hi-fi prototype testing gives the user a near-realistic experience of what the final product will be like. Hi-fi prototypes are ideal for testing complex interactions and your solutions for usability



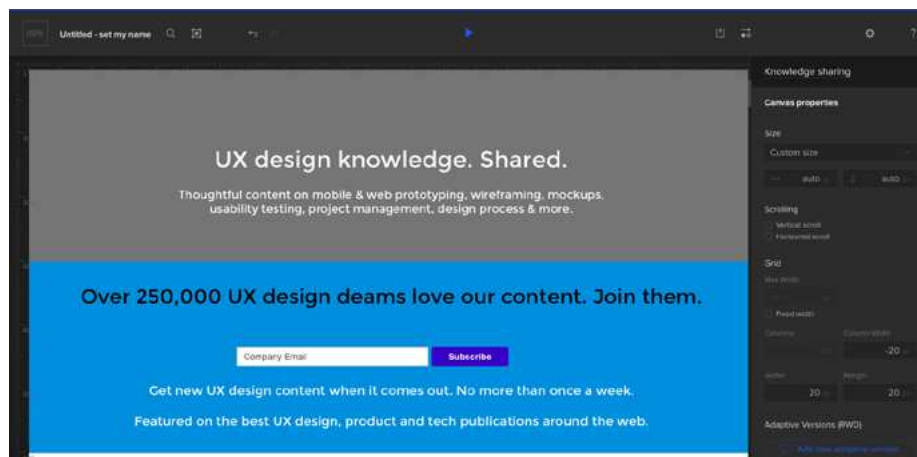
Source: [UXPin](#)

issues discovered in earlier rounds of testing. However, unlike lo-fi prototypes, these are costlier to make. So if you'd like to test conceptual and abstract fields, lo-fi prototypes would work best.

3. Medium Fidelity

Can't decide between high or low fidelity? Mid-fi prototypes work best when you need a balance between fidelity and cost. If you're only going to run one round of usability tests, go medium fidel-

ity. The limited visuals at least give the users an impression of what the final product will be like, creating a more realistic environment. At the same time, the UI is advanced enough to give accurate indications of what works and what doesn't, so you can confidently assess the basics like task flows and interactivity.



Source: [UXPin](#)

4 Content Guidelines for Testing Any Prototype

When you start building the prototype, it's not only acceptable to gloss over minor details in lieu of the essentials, it's at times recommended. But when it comes time to test your prototype, make sure you've filled in some of these details that may get overlooked in lower fidelity.

Jim Ross, Senior UX Architect for **Infragistics**, [outlines the important subtleties](#) of testing a prototype versus a fully functional app or website. In our experience, these are the most helpful tips for preparing your prototype for testing:

1. **Avoid Lorem Ipsum** – Distracting, confusing, and lacking meaning, lorem ipsum text does not fully capture your product's message. Actual content, or at least a relevant placeholder, will better simulate your end product. **Google Ventures** [agrees with the advice](#).
2. **Use generic names** – Tests may be more fun with silly or celebrity names, but fun isn't the point. Any distractions will bias the results, so keep names generic and realistic, like "Michael Schumacher" or "Joan Price." If needed, use a [Random Name Generator](#).
3. **No placeholder images or icons** – Boxes with Xs may work during wireframing, but not in testing. Images and icons play a large role in UX, so these should be implemented by testing time, even if only with temporary sketches. The exception is if these images are purely decorative and don't help to understand the UI.
4. **Use believable data** – Don't fill data like phone numbers or addresses with Xs or jokes – these are distracting. Realistic and believable data here will give your user test the most accurate results. For example, if you're building an ecommerce app, Xing out the credit card number might confuse test users about whether the app displays information like that.

Test participants may become fixated on details that you thought were negligible, so be careful what you *don't* say. These small steps to reduce distraction and confusion can go a long way toward cleaner test data.

Takeaway

It almost goes without saying, but you're designing the product to be used by real people. In order to prepare it for real people, it should be tested on real people. Prototypes are built for experimentation, so it only makes sense to test them on real users. Brainstorming sessions and endless speculating will only get you so far; taking your prototype for a test drive, so to speak, will show you the areas you need to focus on going forward.

Once you're ready to start usability testing with your prototype, download the free [Usability Testing Kit](#), with 5 testing templates created by UXPin CEO Marcin Treder.

10 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 top experts in the field, plus added some spice from our own trial and error. What we came up with are 10 guidelines that can help you prototype, no matter what method you choose.

1. Know Your Audience and Your Goals

Knowing why and for whom you're designing will keep you on track and aid you in making the right choices. What may work well for a brainstorming session with your team may not be as suitable for a high-visibility meeting. As Todd Zari Warfel states in his book [*Prototyping, a Practitioner's Guide*](#):

Prototype fidelity is a sliding scale. Don't be concerned with hi-level or lo-level fidelity. The level of fidelity that matters is

whatever is needed to help you accomplish your goal with the audience for your prototype.

Lo-fi prototypes can help with presenting conceptual or abstract solutions in the earlier phases of design. On the other hand, hi-fi prototypes can be easier to understand for non-design folks since they resemble the final product. Perhaps a paper prototype might work for a senior executive who just needs to get a quick grasp of the system, or maybe you need a mid-fi prototype to share and collaborate on. You won't know until you talk to your team to understand exactly what they need.

2. Prime Your Audience Beforehand

For someone who has been knee-deep designing the prototype for weeks, it may seem strange that new users don't know every nook and cranny. You may need to remind yourself that not everything is obvious, since you'll likely have to explain features to stakeholders or outside users.

[Later in his book](#), Warfel explains a case study of prototyping for enterprise software company IntraLinks: while half of the audience discovered some features on their own, the other half was clueless. But after he pointed them out, these users took a quick liking to these features as well.



Photo Credit: Marcin Treder, [UXPin](#)

As you prototype, you should periodically review your work as if it's the first time you're seeing it, just so you're able to give the right instructions to new users. We discussed a similar concept in our free ebook, [The Guide to Mockups](#): setting the context is the key to any successful design presentation. Explain the key features of the prototype, their desired business impact, and any remaining features still on the to-do list. If people can predict what they're going to see, they'll react more productively.

3. 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*.

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](#).

4. 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 of the *The Guide to UX Design Process and Documentation*.

5. 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 above 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.

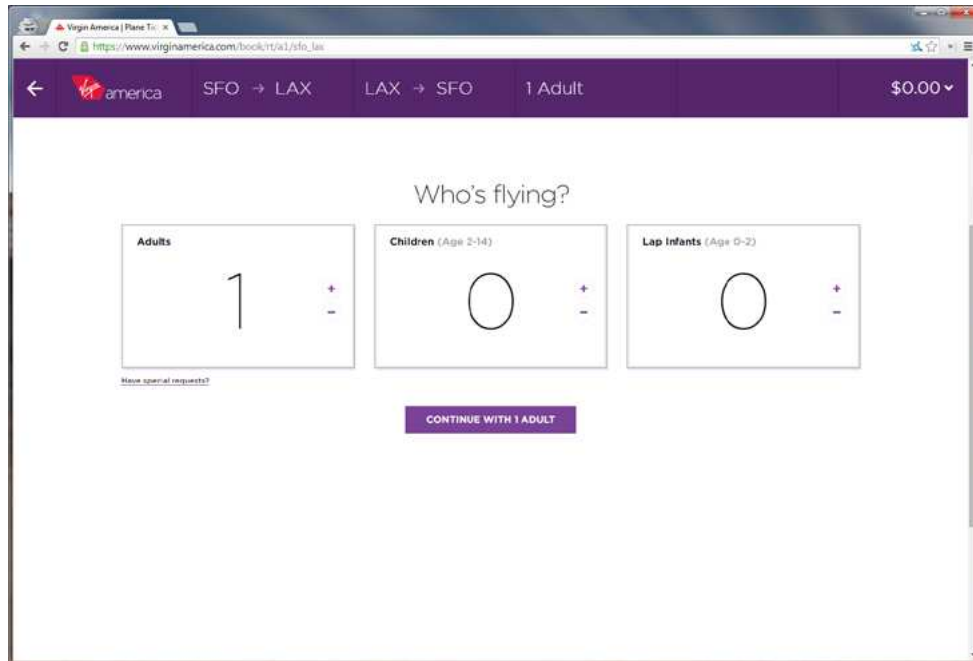


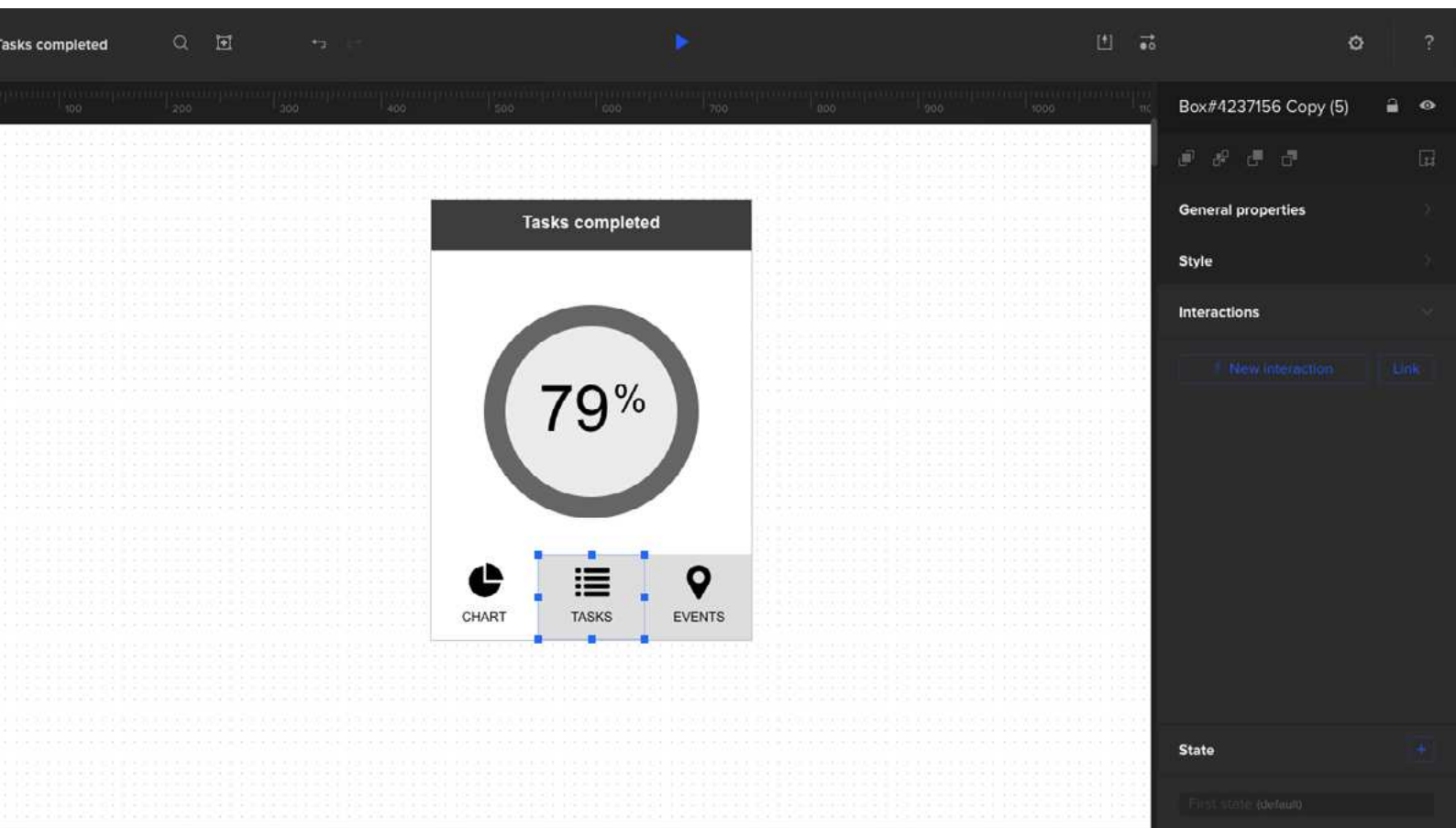
Photo Credit: www.virginamerica.com

6. 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.

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.



Source: [Advanced Interactions](#)

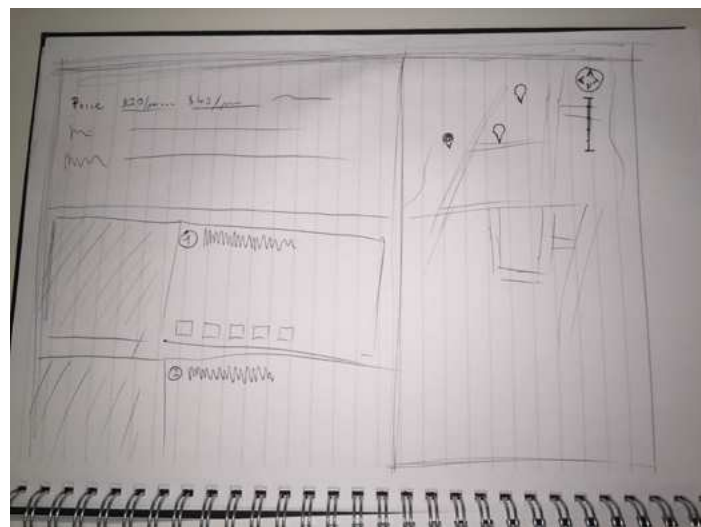
7. Sketching: The Prototype for the Prototype

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*](#).

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



Source: [User Testing & Design](#)

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.

8. 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.

9. 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.

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.

10. Prototype Only What You Need – Then Stop

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.

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.

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 chapter, we put everything into context by looking at the past, present, and future of wireframing and prototyping.

Wireframing & Prototyping: The Past, Present, and Future

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

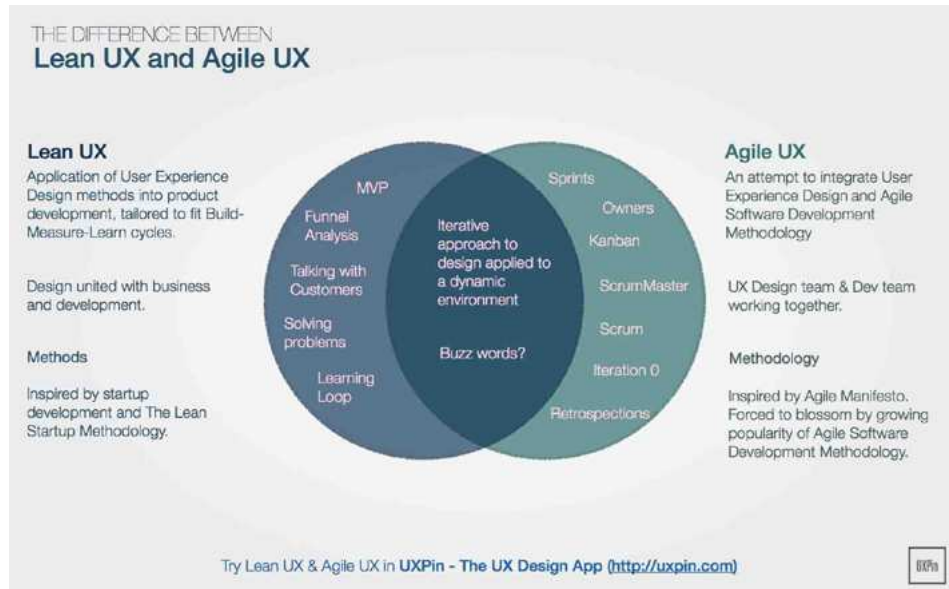
Knowing the origins of wireframing and prototyping will help you put everything into a historical context, allowing you to comprehend the practice and predict how it will evolve in the future. This chapter will be somewhat of a modern history lesson, covering digital UX design in its past, present, and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

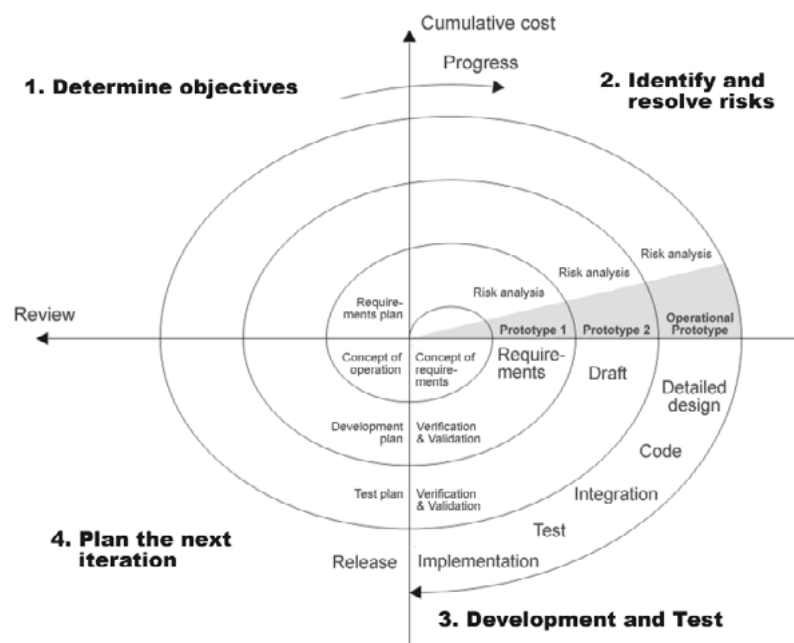


Photo Credit: "Spiral model". Wikimedia. Creative Commons.

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.

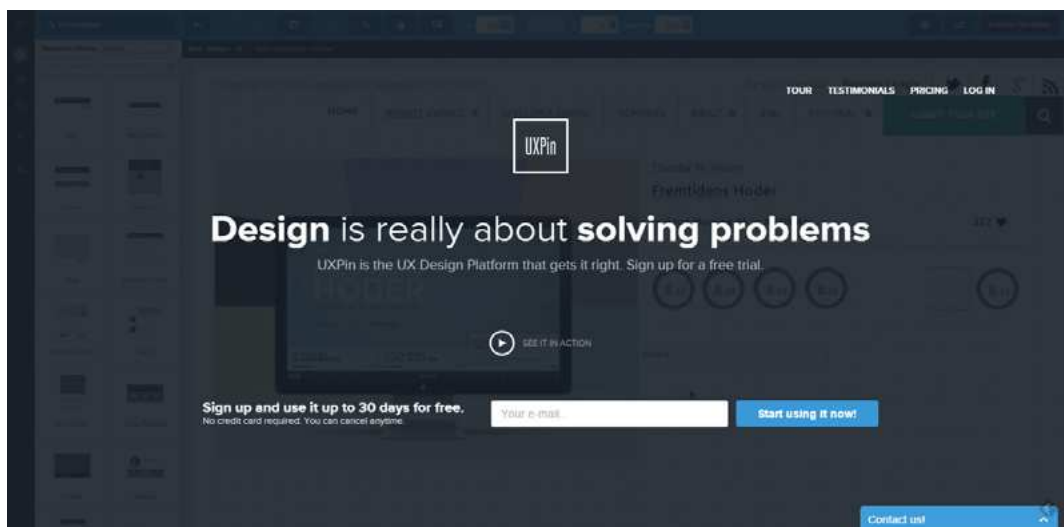
2010 ▶ Technological advances enable high-fidelity SaaS prototyping without coding.

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, humanizing 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.

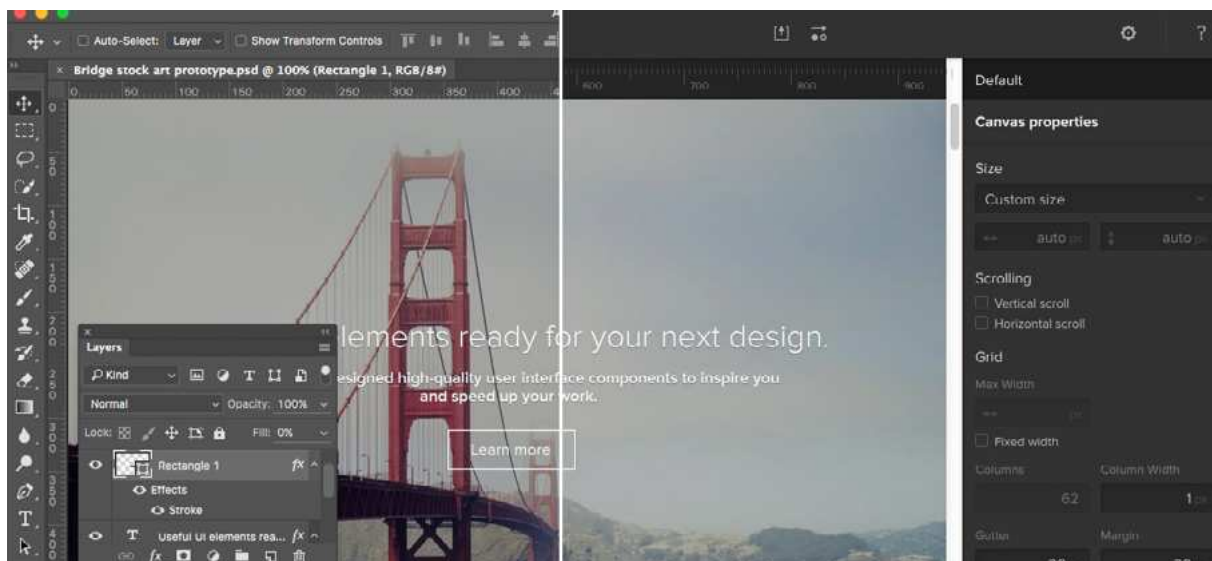
Now, let's start prototyping.

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

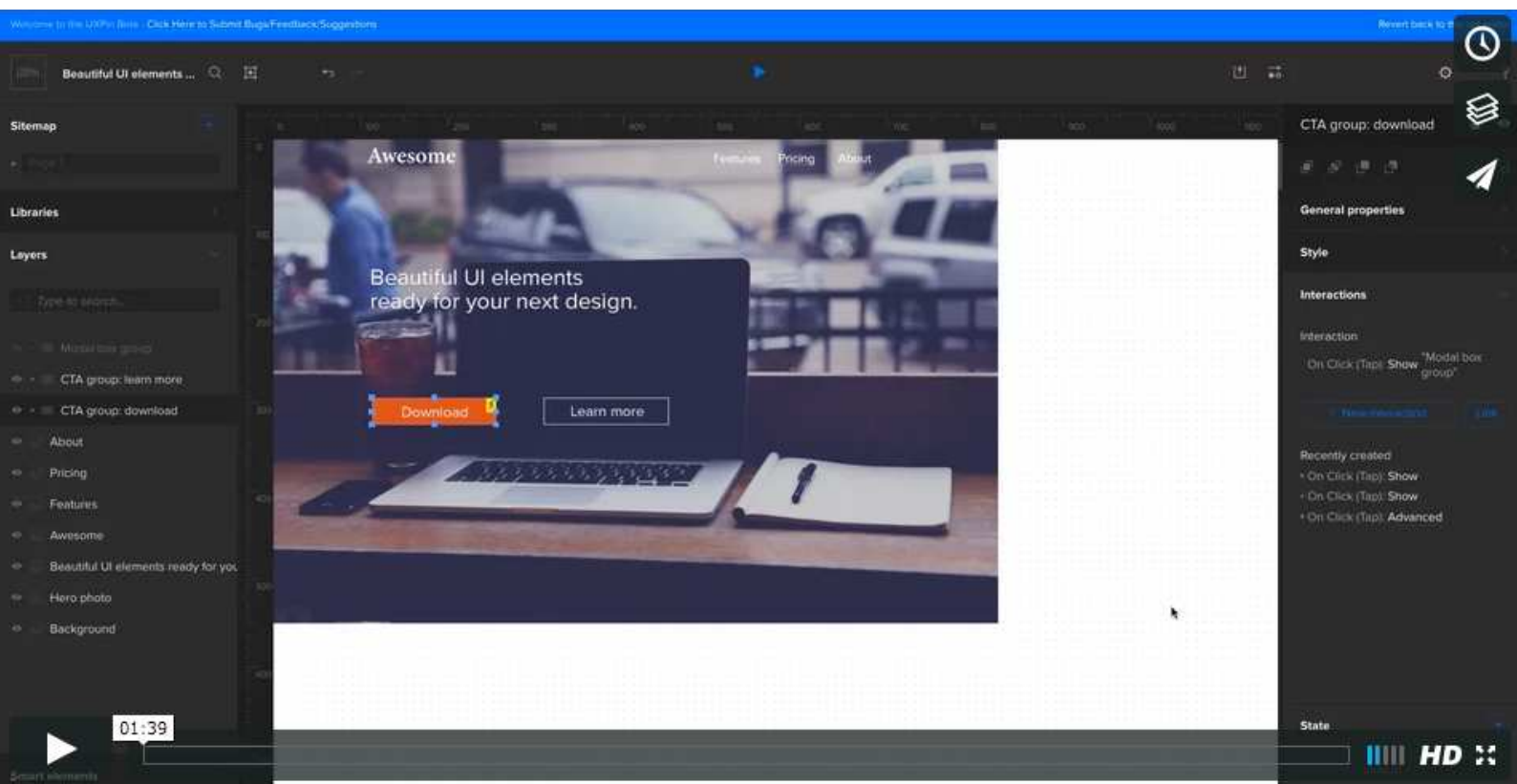


Source: [UXPin](#)

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 [for photoshop](#) and [for Sketch](#), and then check out our tutorial below or [on our blog](#).

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



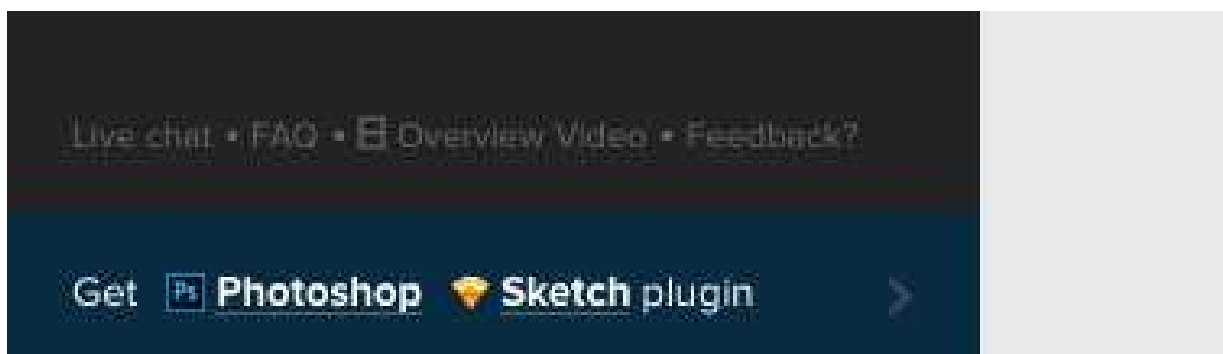
[Overview Video \(click to play\)](#)

Importing from Photoshop into UXPin

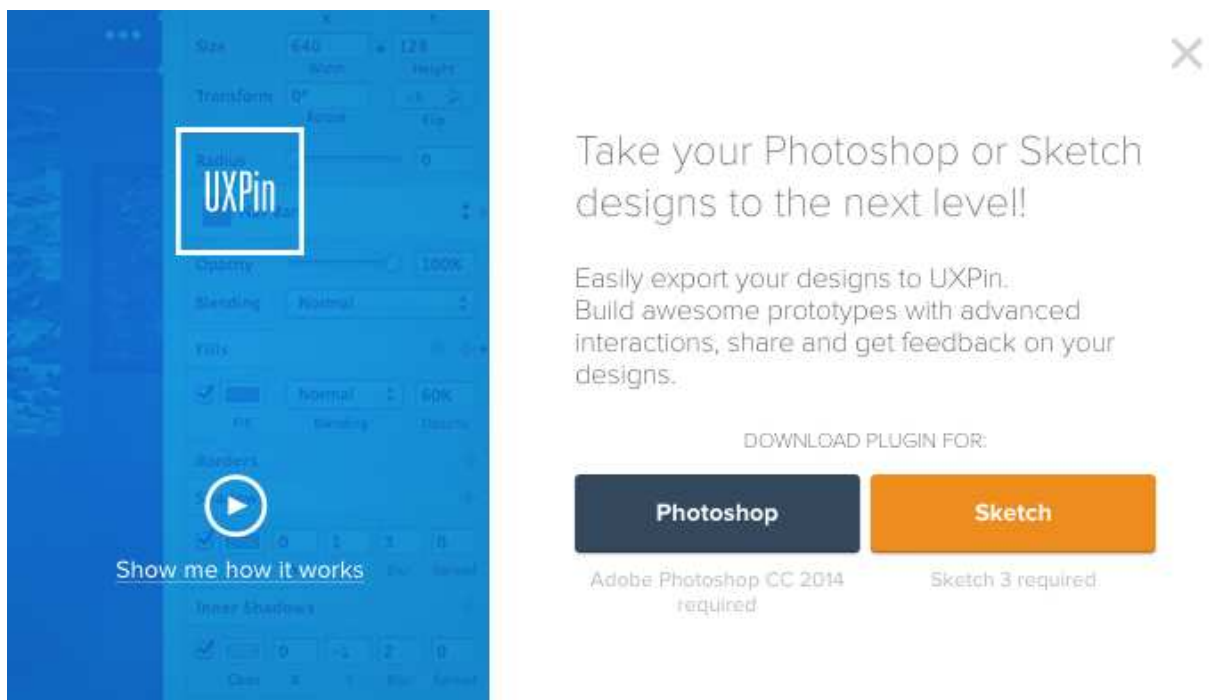
For this example, we will import a Photoshop file from our free [Web UI Kit](#) into UXPin.



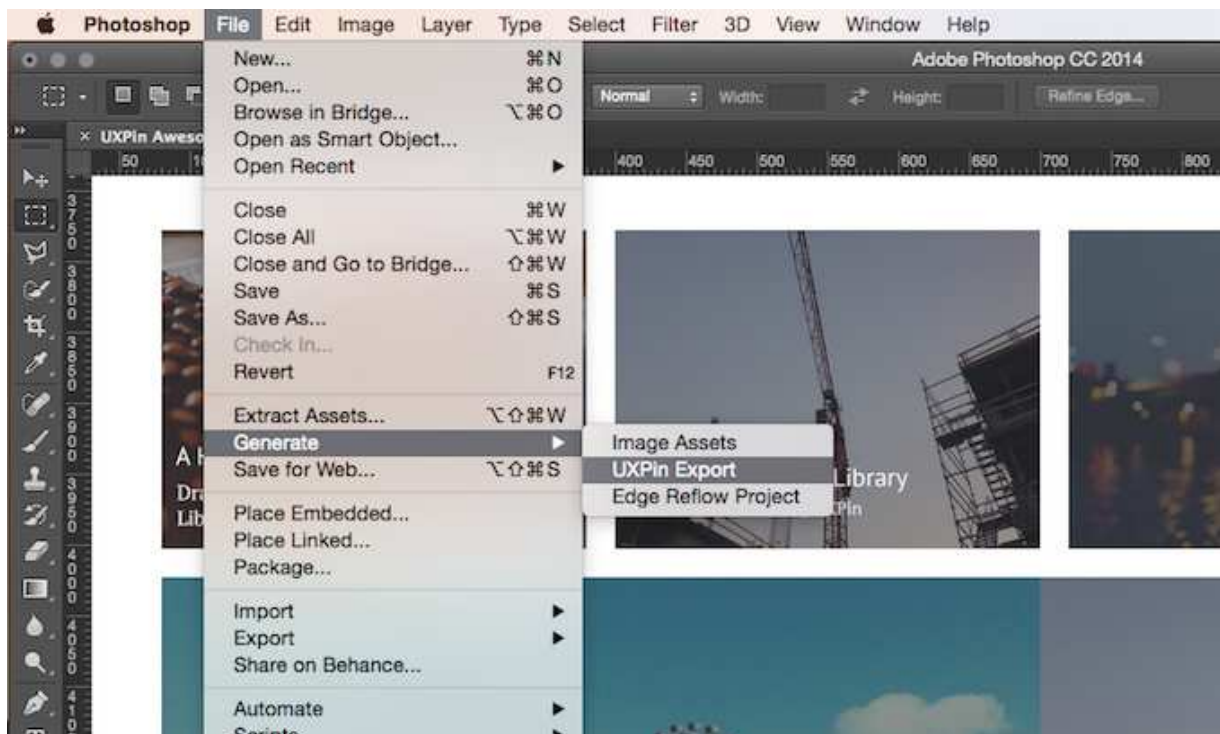
1. Sign in to your existing UXPin account (or [sign up for a free trial](#))



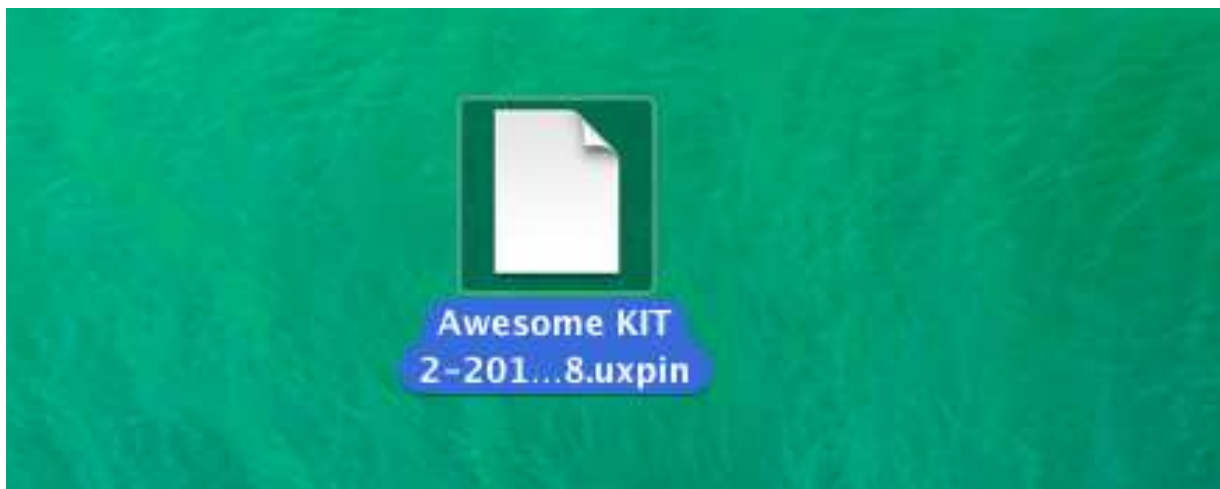
2. Click the plugin icon in UXPin (lower left hand corner of the dashboard)



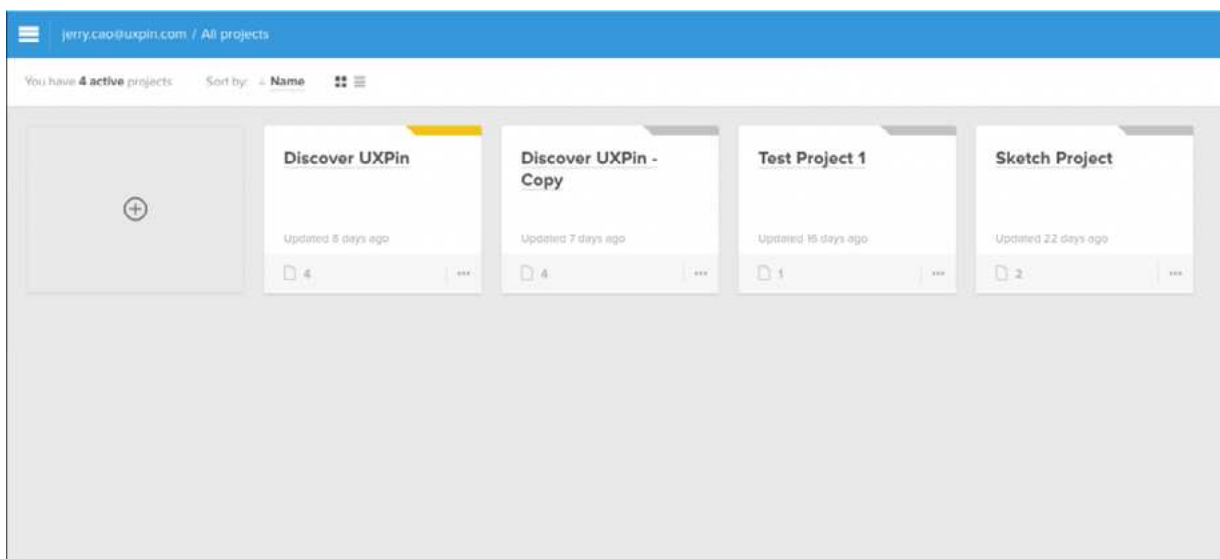
3. Download and install the Photoshop plugin



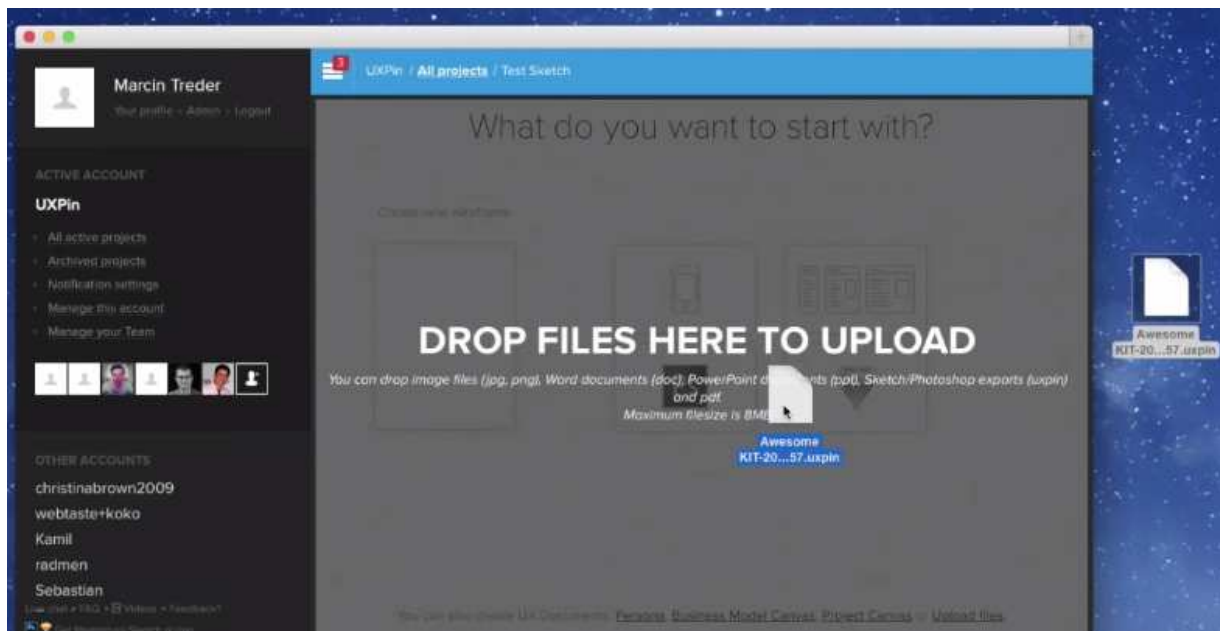
4. In Photoshop, select your elements, click File, Generate and then UXPin Export. Our file is from the [Web UI Kit](#).



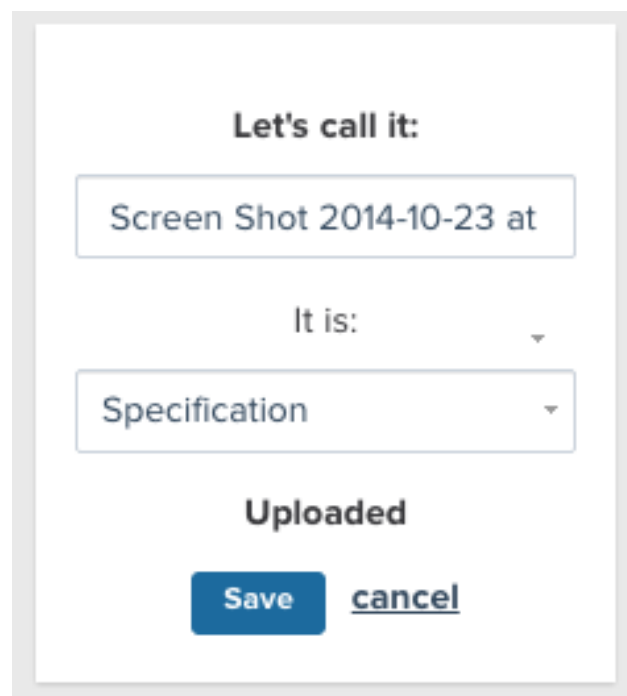
5. The file will likely export to the desktop. Your new file has a .uxpin extension.



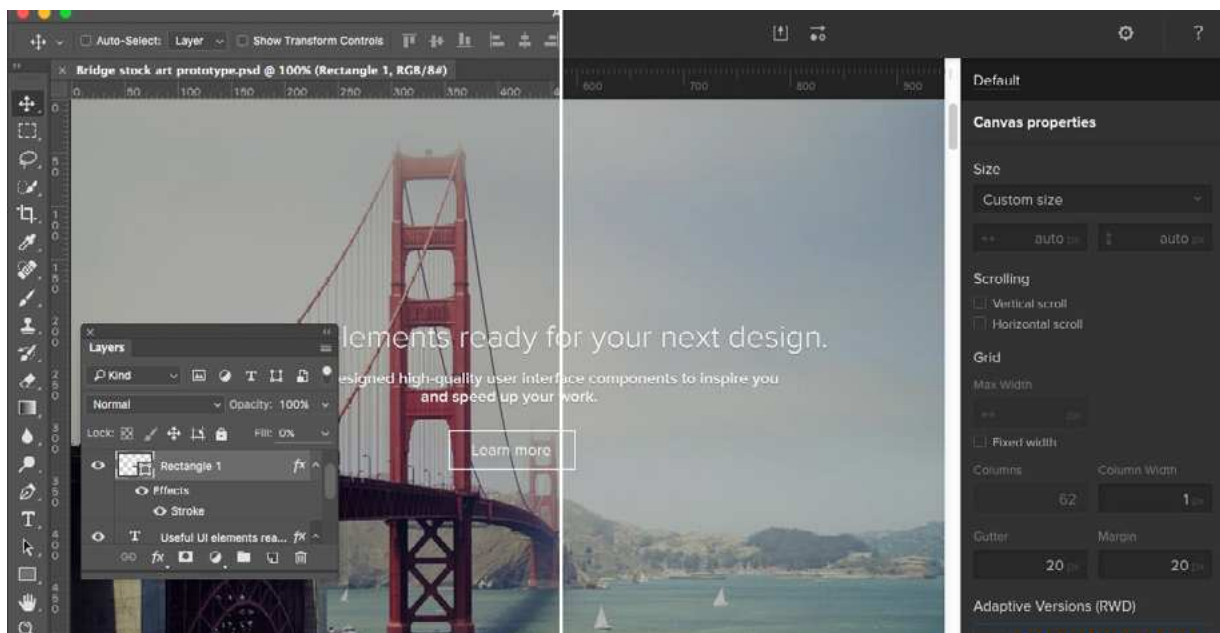
6. Return to [UXPin](#) and either create a new project or click into your existing project.



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



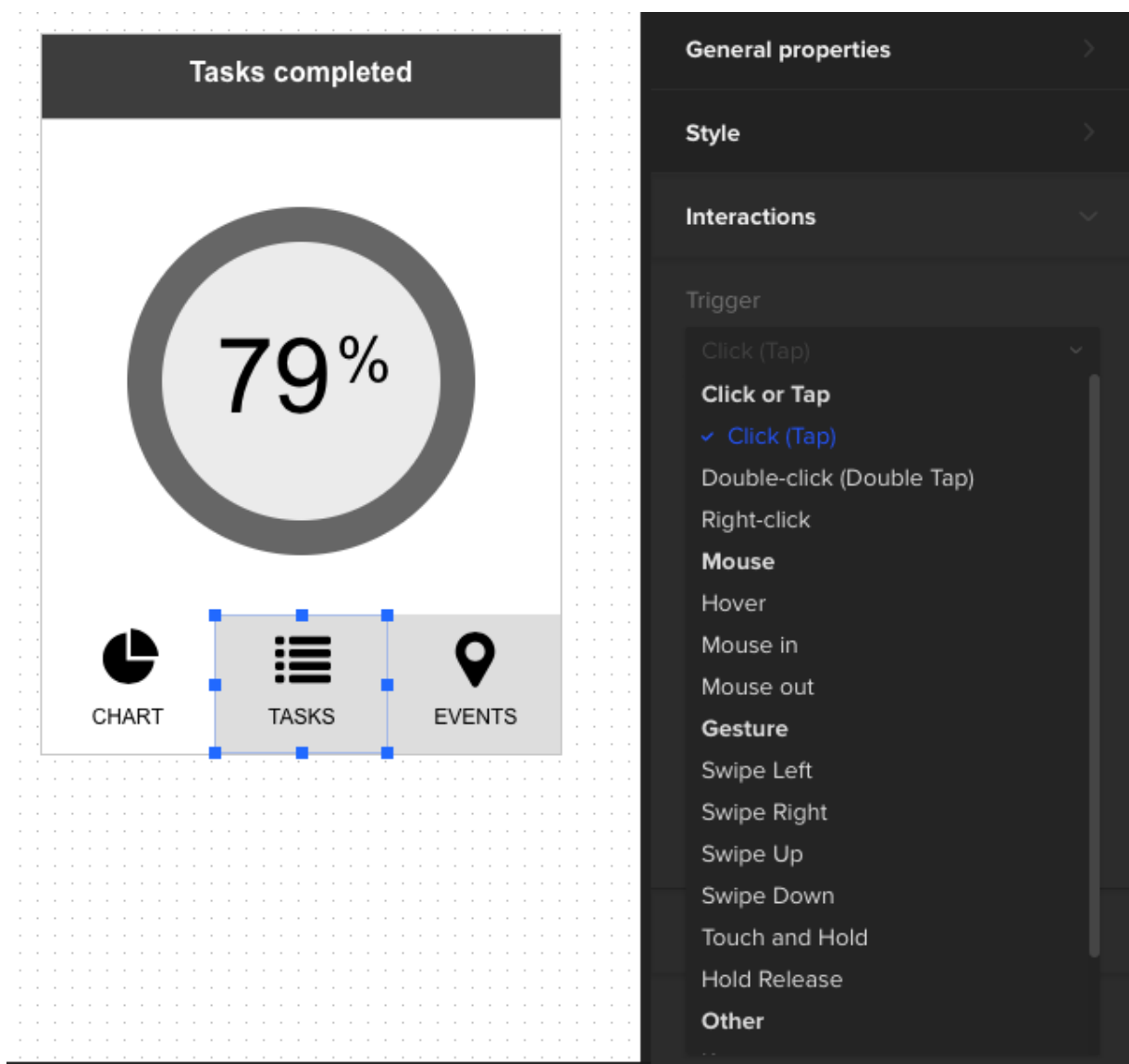
8. Once the file is loaded, feel free to rename and then click Save.



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

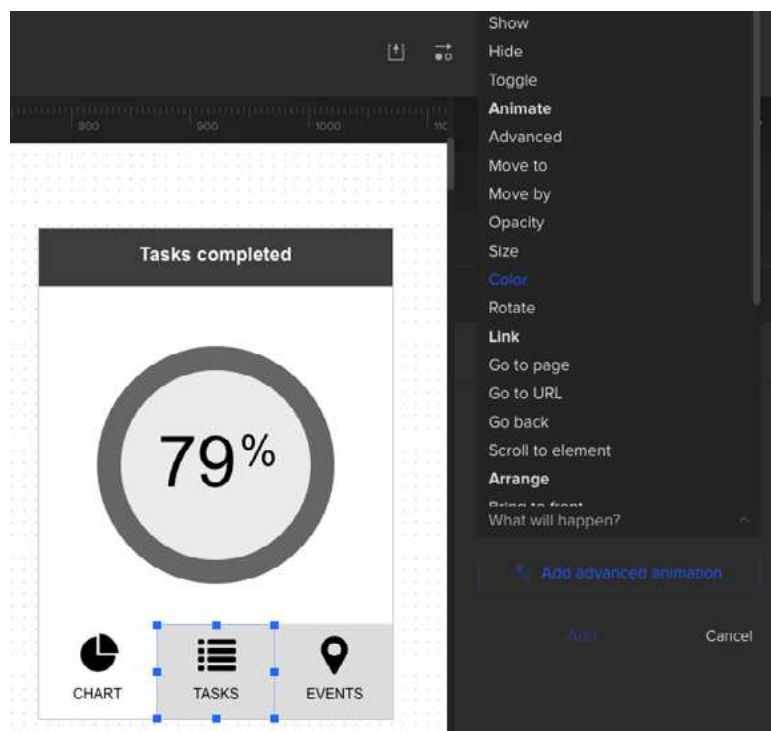
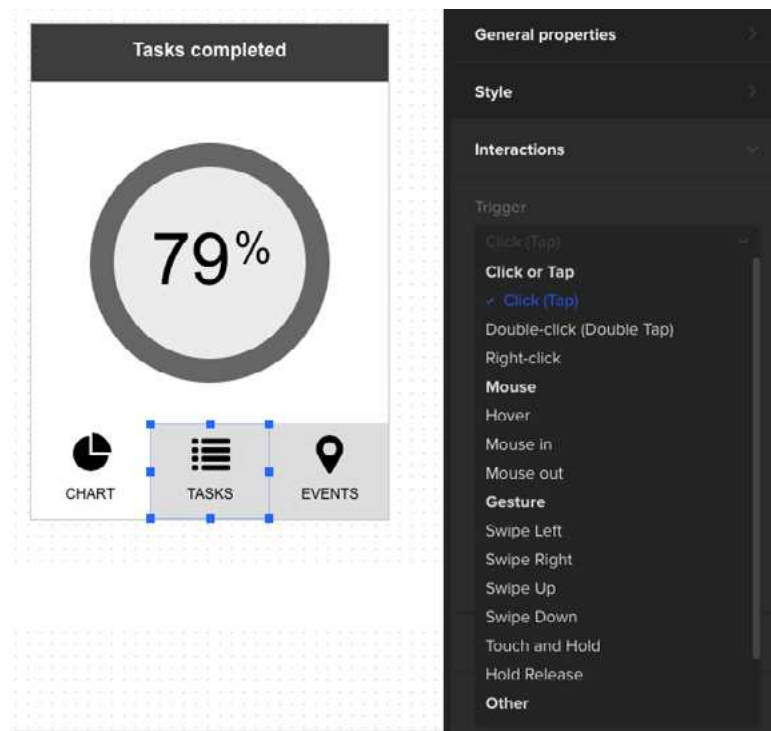
General notes on using interactions and animations

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

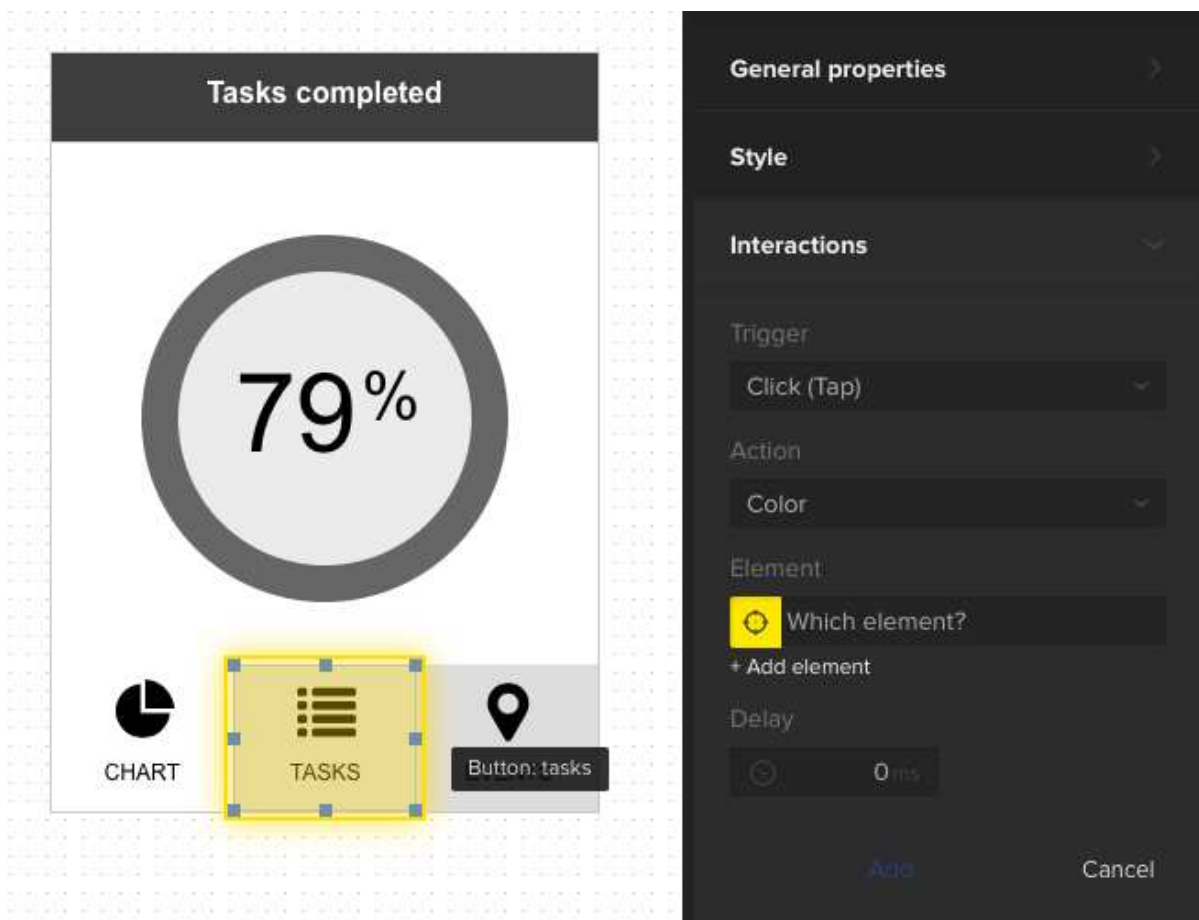


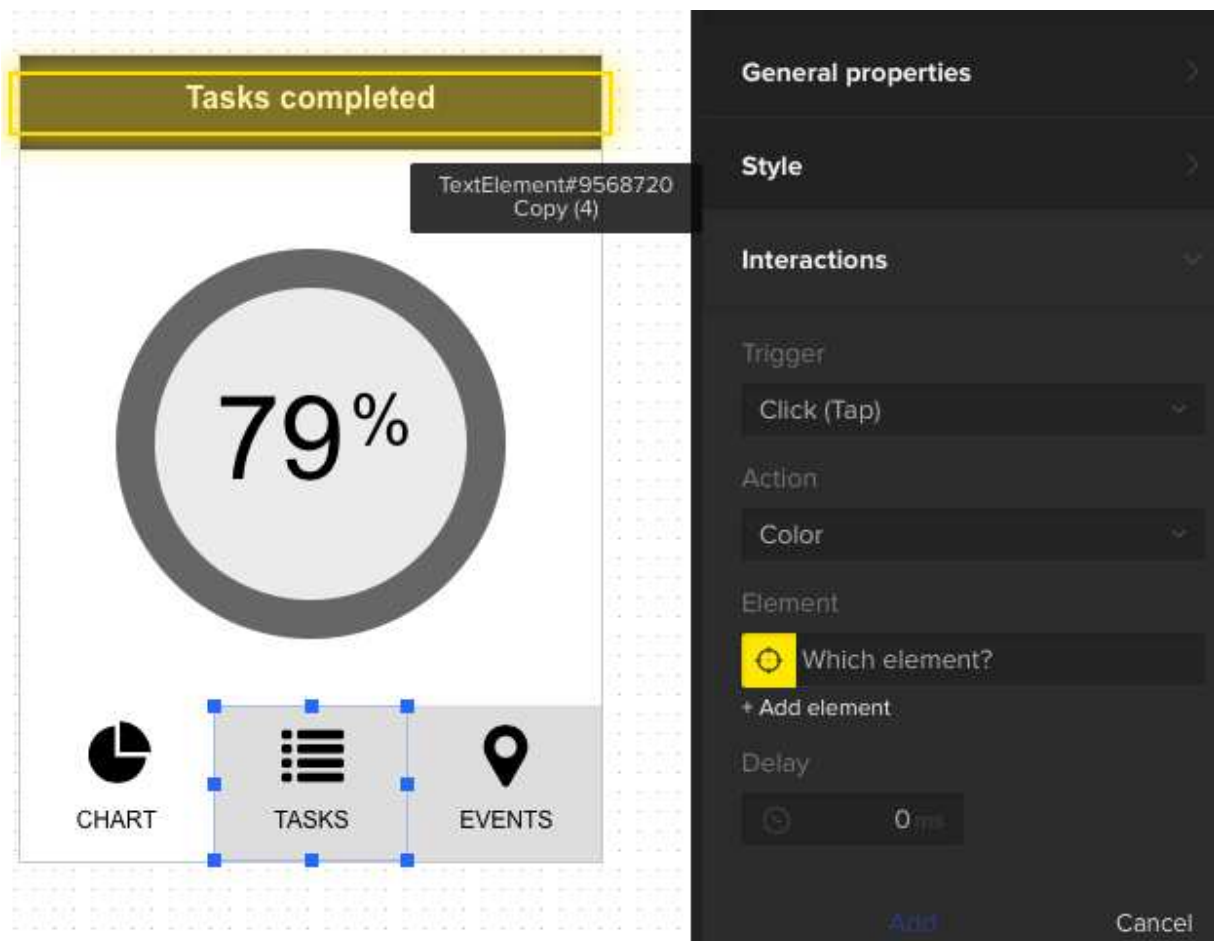
To set your first interaction, click to open the “Interactions” menu on the right side of the editor. Below that, “Recently Created” is a list that shows you any recently created interactions in your design.

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 to pick an element that the interaction will affect. You can either choose it from the list of elements in the Interactions menu, or click the “target” icon, then click the element you want to change in the canvas. Notice that when using the target icon, elements in the canvas will gain a yellow highlight as you hover over them, indicating which item you’re about to target.





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

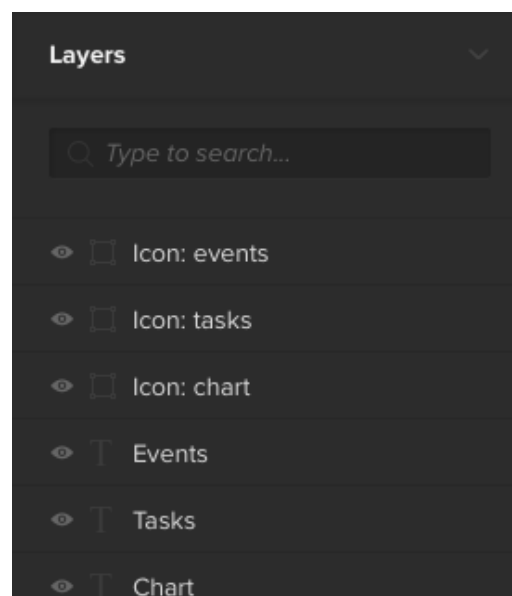
- **Triggers:** click, double, click, right-click, hover, mouse in, focus, blur, key press, window is scrolled to, page is loaded
- **Actions:** show element, hide element, toggle visibility, go to page, go back, scroll element, state: enable, state: disable, state: select/check, move by, resize element, 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

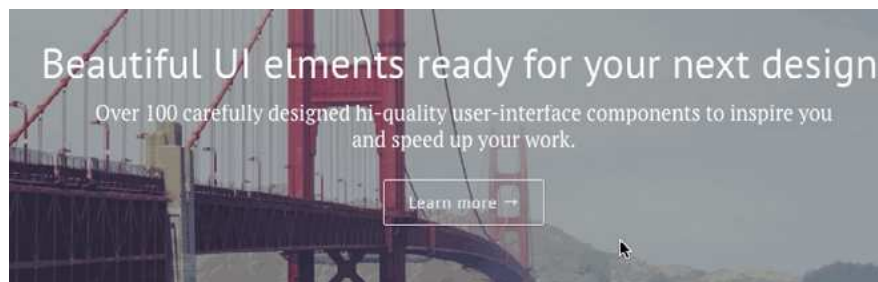
I start by making sure that I have two different buttons that will appear after specific user actions: one for the inactive state and one appearing on mouse out.



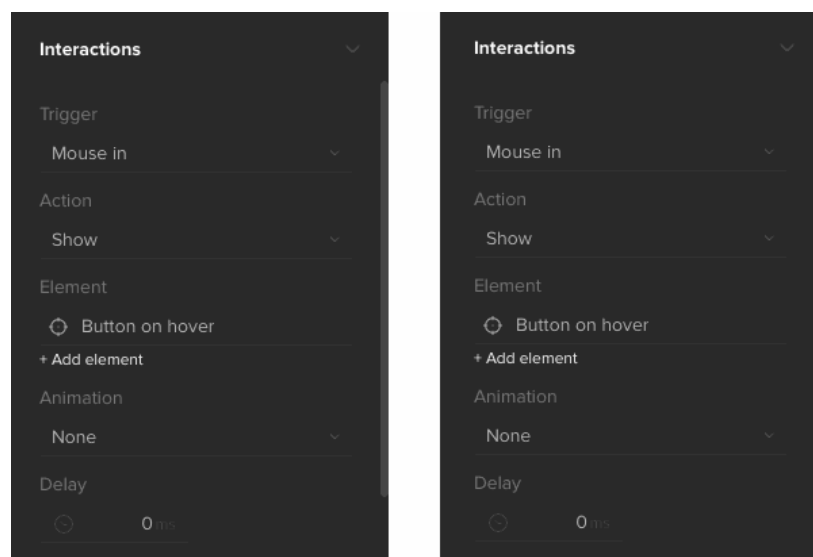
As you can see on the next page, the buttons will switch with interactions, however they will remain in the same position, so that's how we have to put them on the prototype. Using the layers editor, you can stack them in the right order – keep the one that shows by default on top, and the one showing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



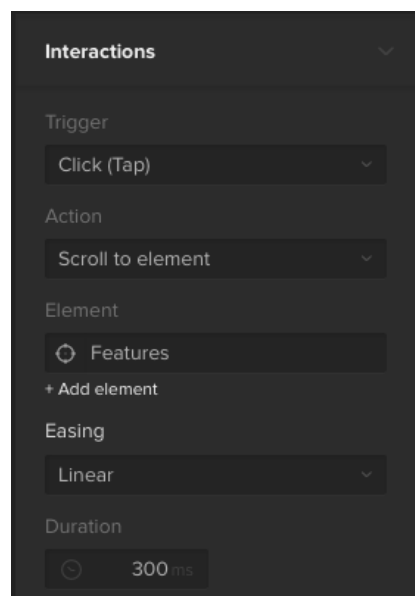
Now on to setting up interactions. For the first button we'll need a total of two interactions: showing the orange button upon hover (left panel below) and hiding the orange button on mouse out (right panel below). Here's how the setup will look:



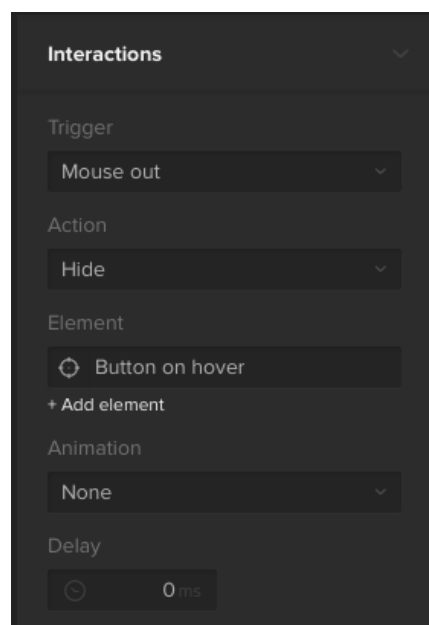
Now we just need to make the page scroll to the next section upon clicking the orange “Learn More” button. And let's not forget that the orange button needs to be hidden on mouse out so that the default transparent button can reappear (in case you scroll back up).

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

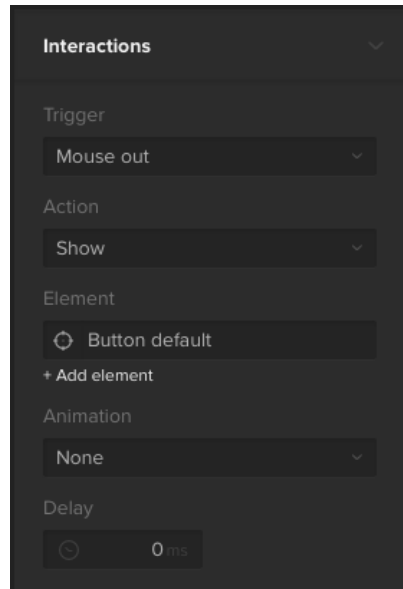
To scroll to the next section when you click the orange button, make sure your interactions menu looks like this:



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

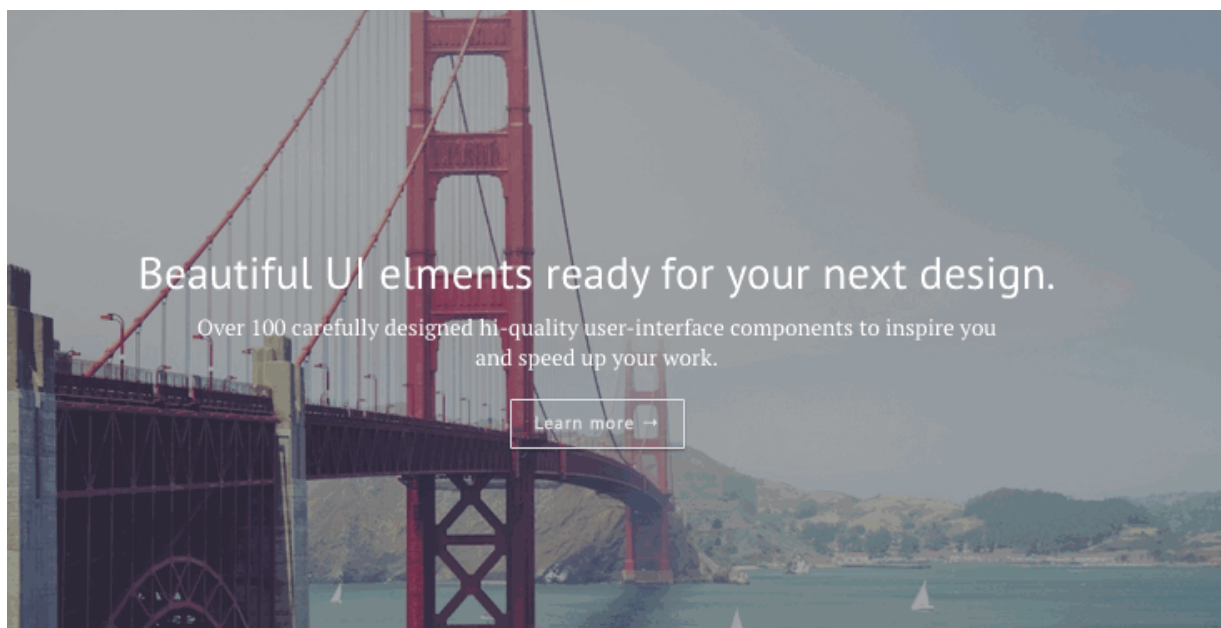


To make the transparent button show when you mouse out (in case you scroll back up), set up your interactions menu like this:



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.

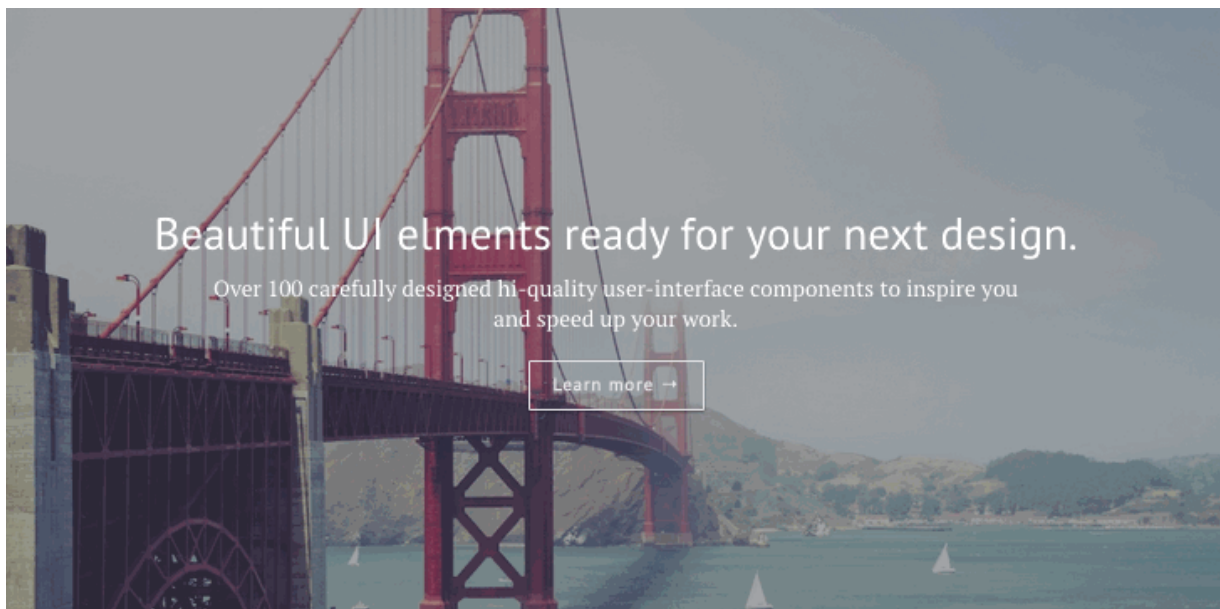
Click the image below to see the interaction at work.



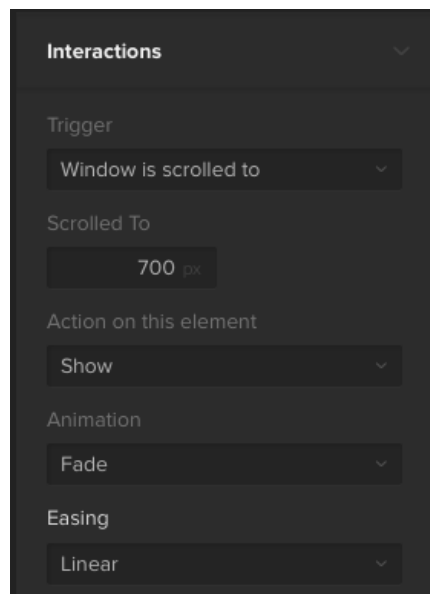
Form: triggering visibility on scroll

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.

Click the image below to see the interaction at work.



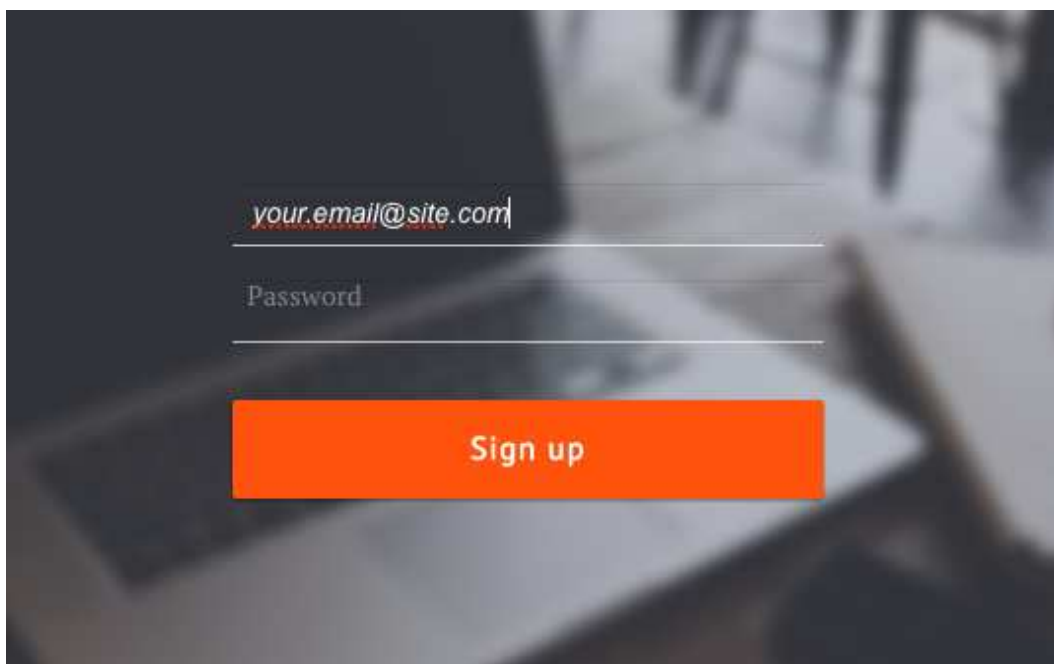
Since the form has to appear on a certain interaction, it has to be hidden by default – make sure you have that step covered. Now select the whole Form layer and look to the Interactions menu on the right. As you probably already guess, the trigger will be “window is scrolled to” and the action will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



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!

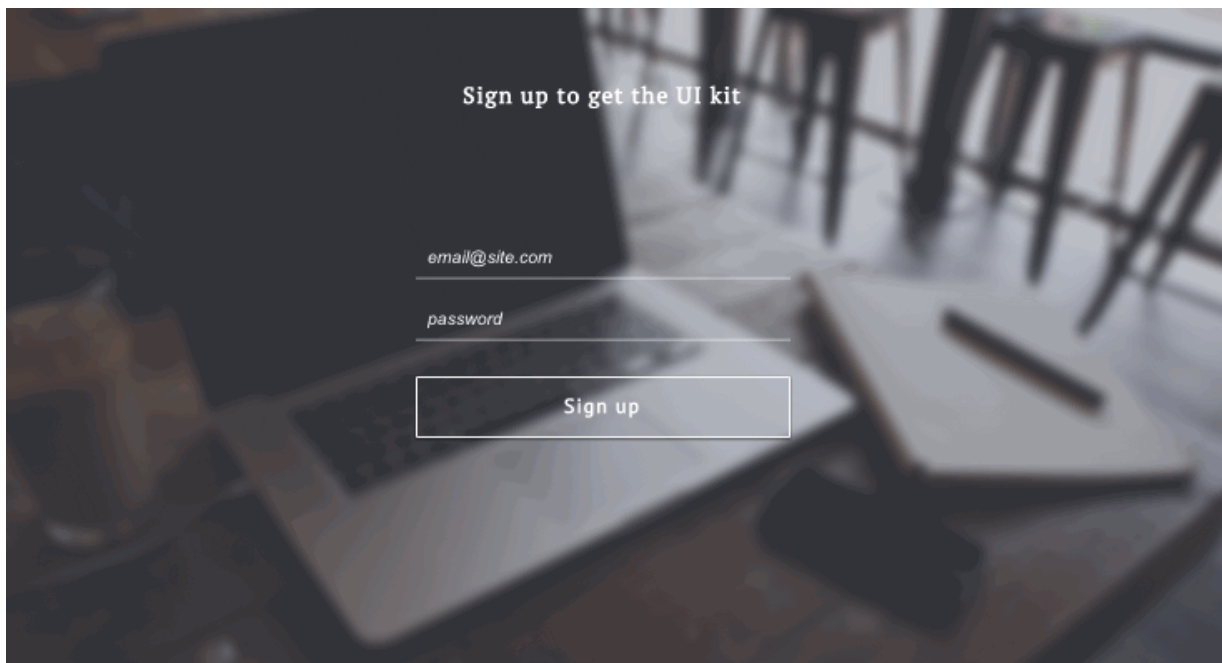


Other than that you can come with some additional interactions of your own. I added a little trick to the input, making the “email” caption disappear once you click on the input. I just placed a text element underneath the input and added an interaction on the input element.

Form: interactions after signing up

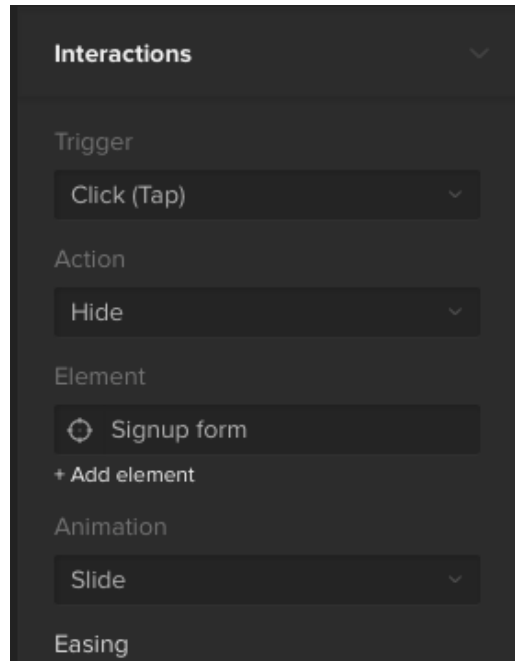
As the final touch on our prototype example we'll introduce a sign up confirmation that slides in when the user clicks "sign up" button.

Click the image below to see the interaction as a GIF in a new window.

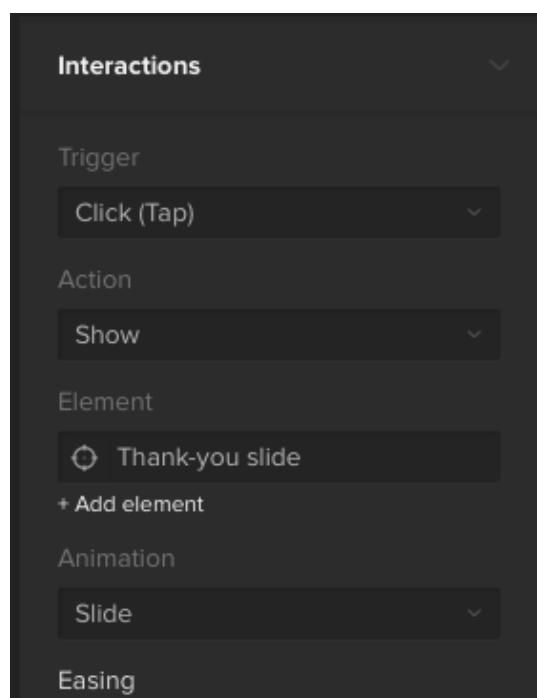


We'll be setting interactions between three elements: the sign up button, the signup form and the "Thank you" slide. However, the interaction will be set only on the signup button, because it triggers actions for the signup form and "Thank you" slide. When clicked, the signup button will hide the sign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

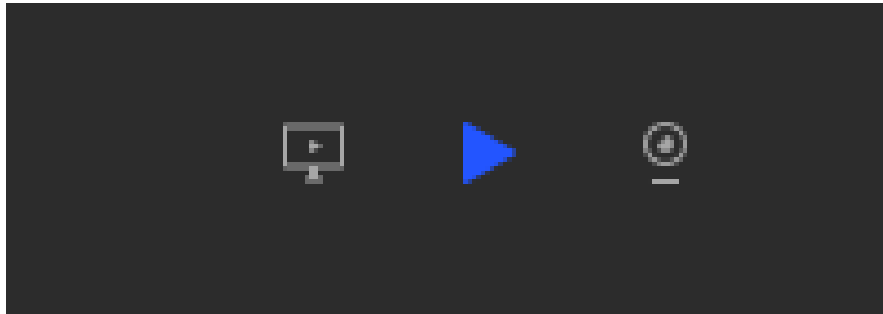
To hide the signup form when the signup button is clicked, make sure your interaction menu looks like this:



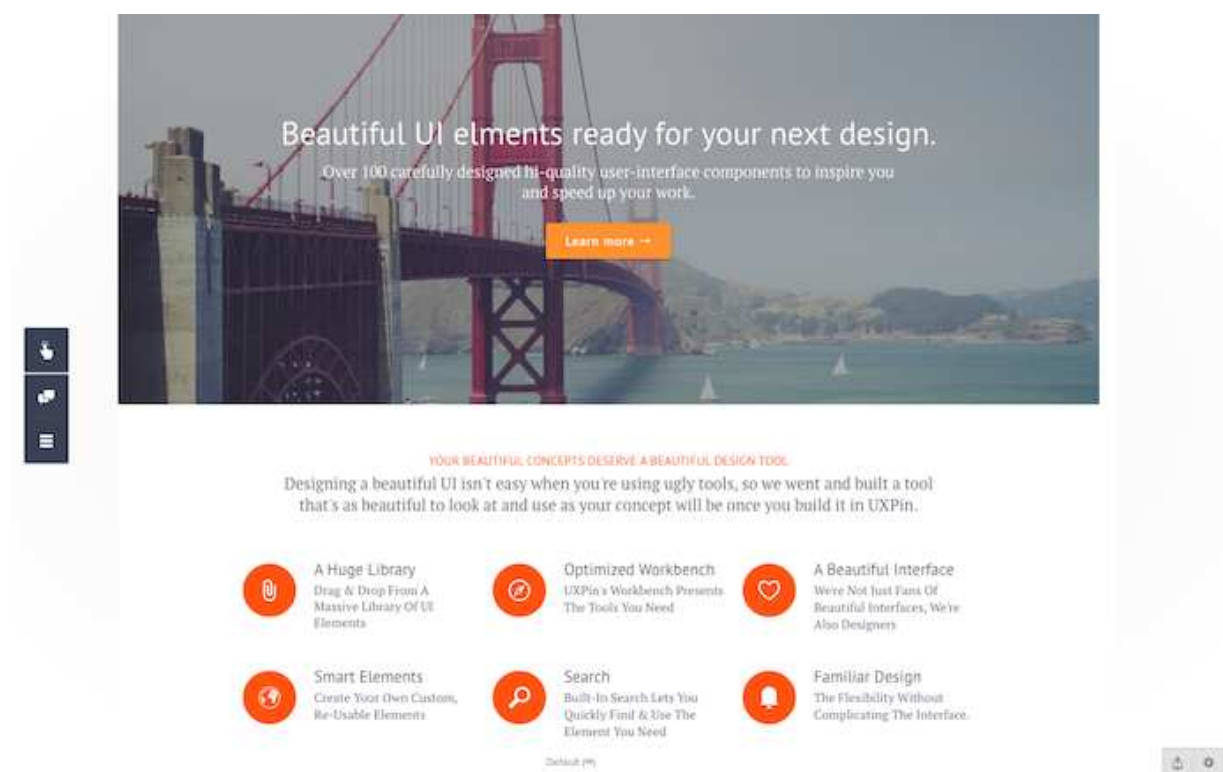
To show the “thank you slide” when the signup button is clicked, make sure your interaction menu looks like the below. Then you’re all done!



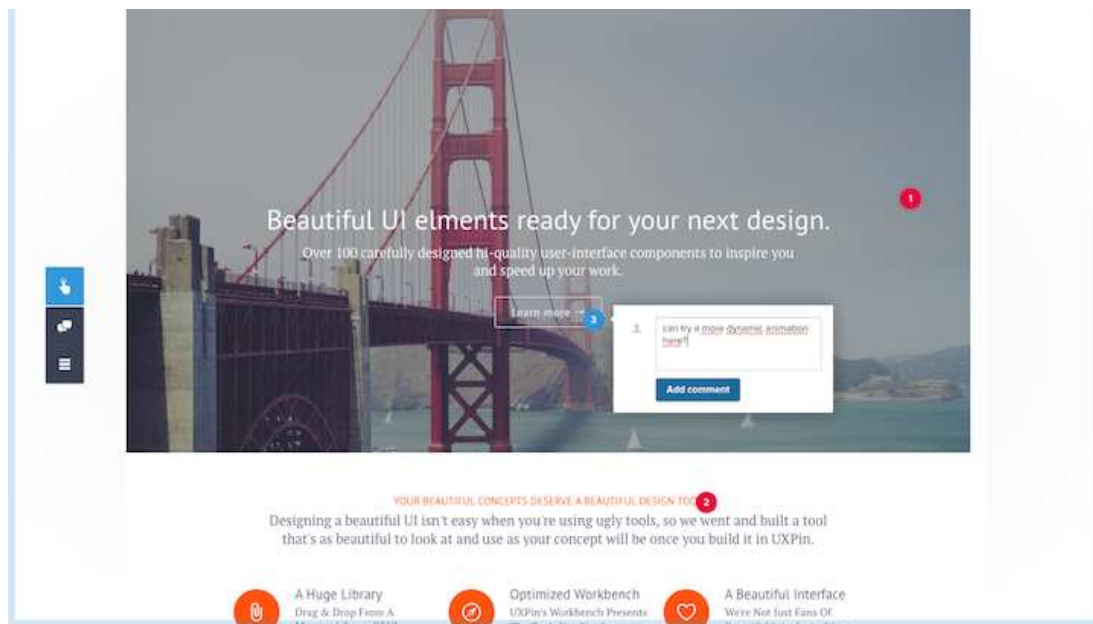
Previewing and gathering feedback



1. To try the prototype yourself, click the “play” button above the canvas. 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.



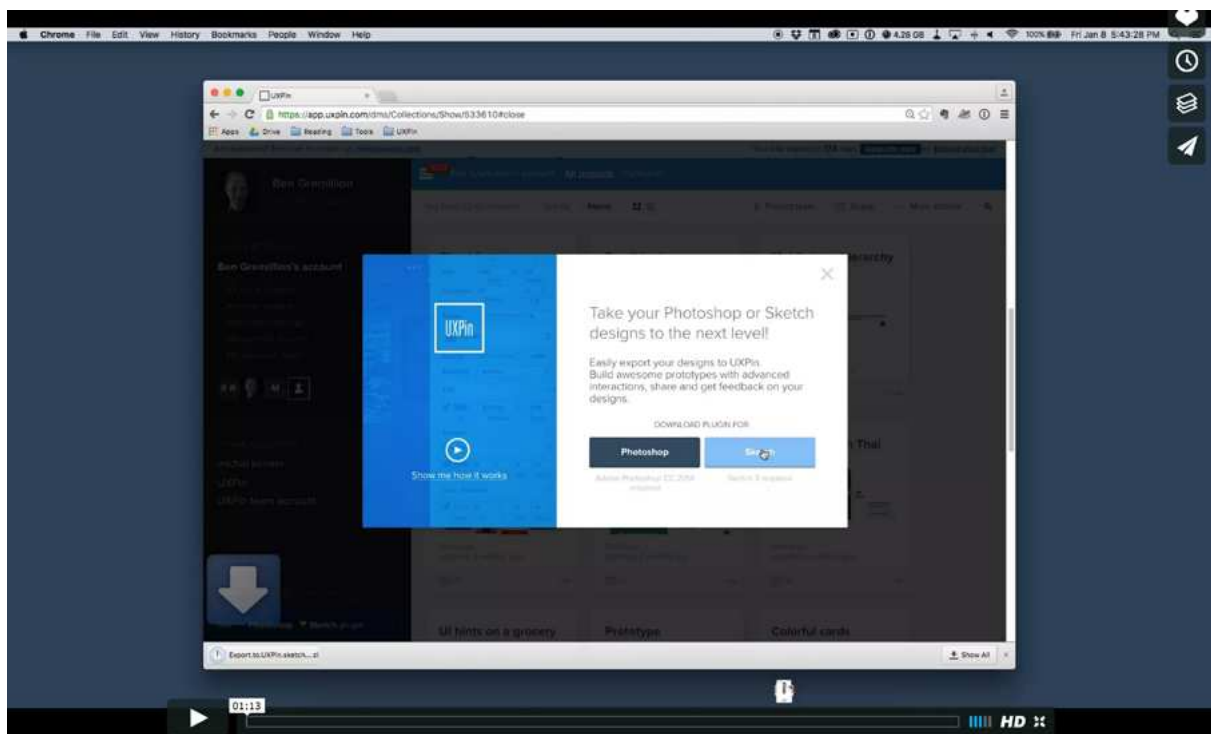
Source: [Photoshop & Sketch Integration](#)

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)

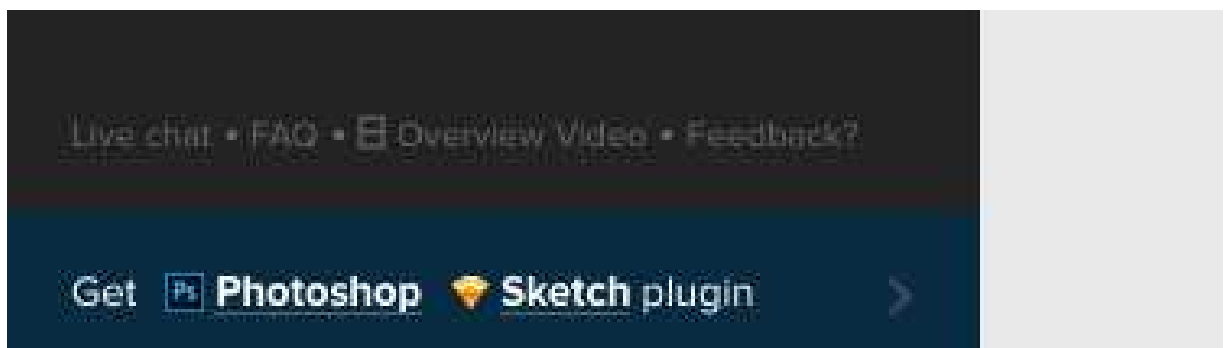


Importing from Sketch into UXPin

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.



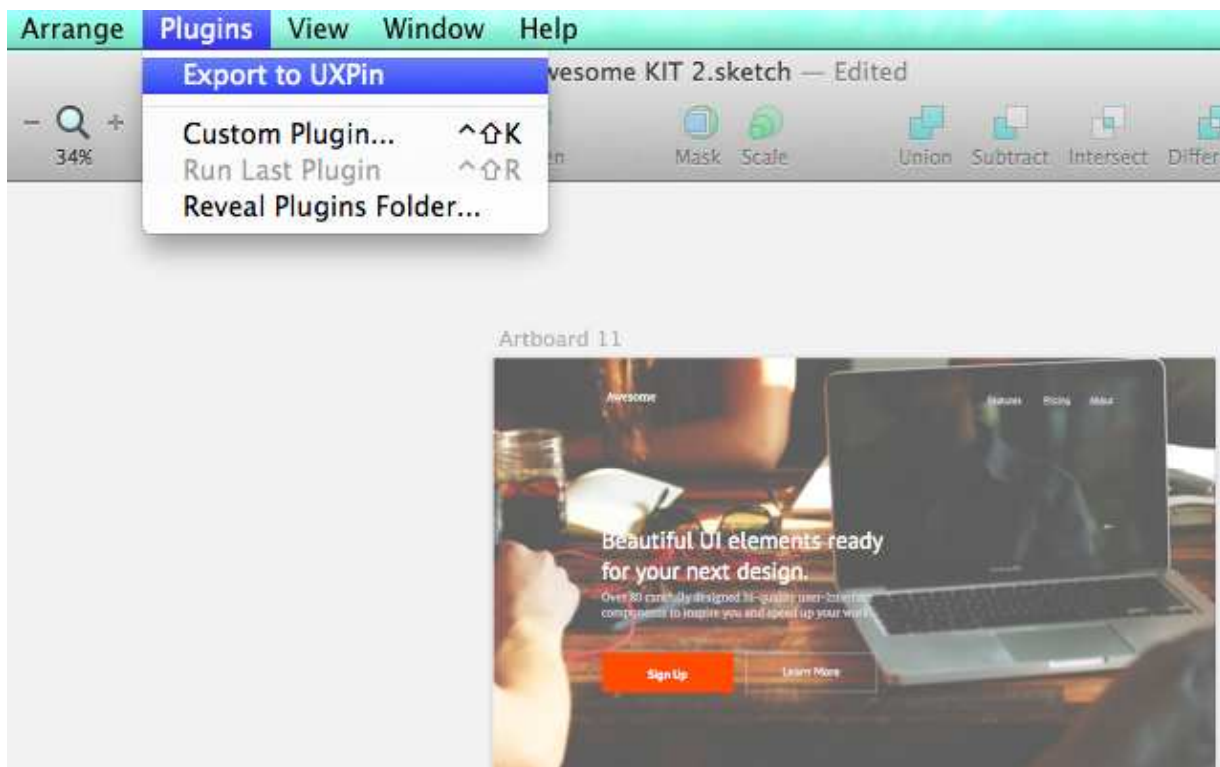
1. Sign in to your existing UXPin account (or [sign up for a free trial](#))



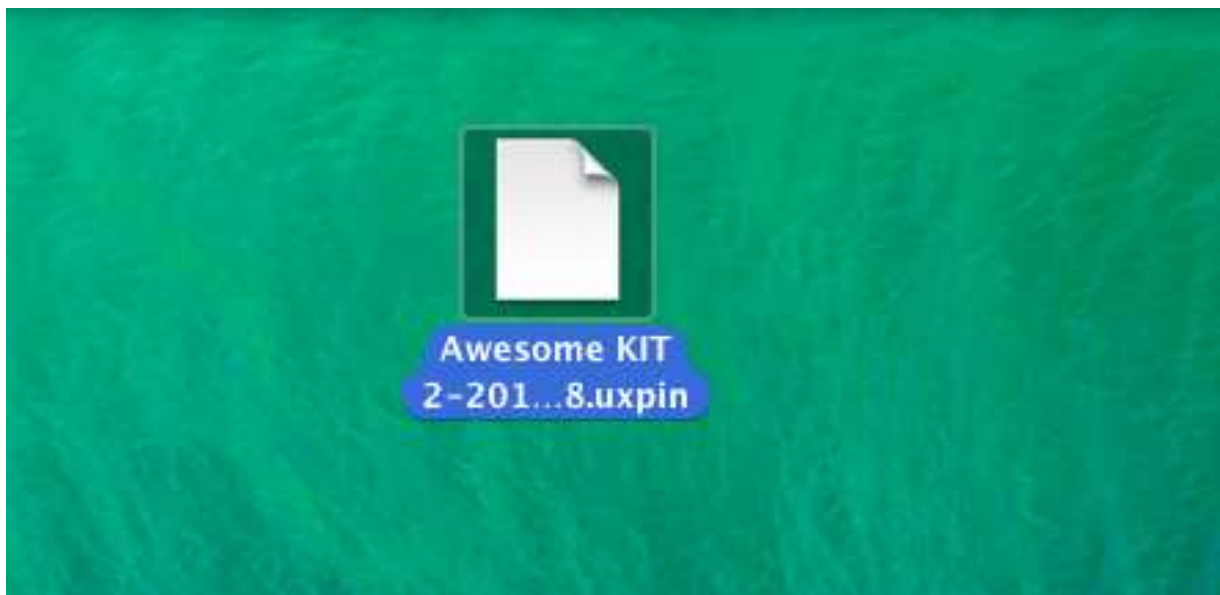
2. Click the plugin icon in UXPin (lower left hand corner)



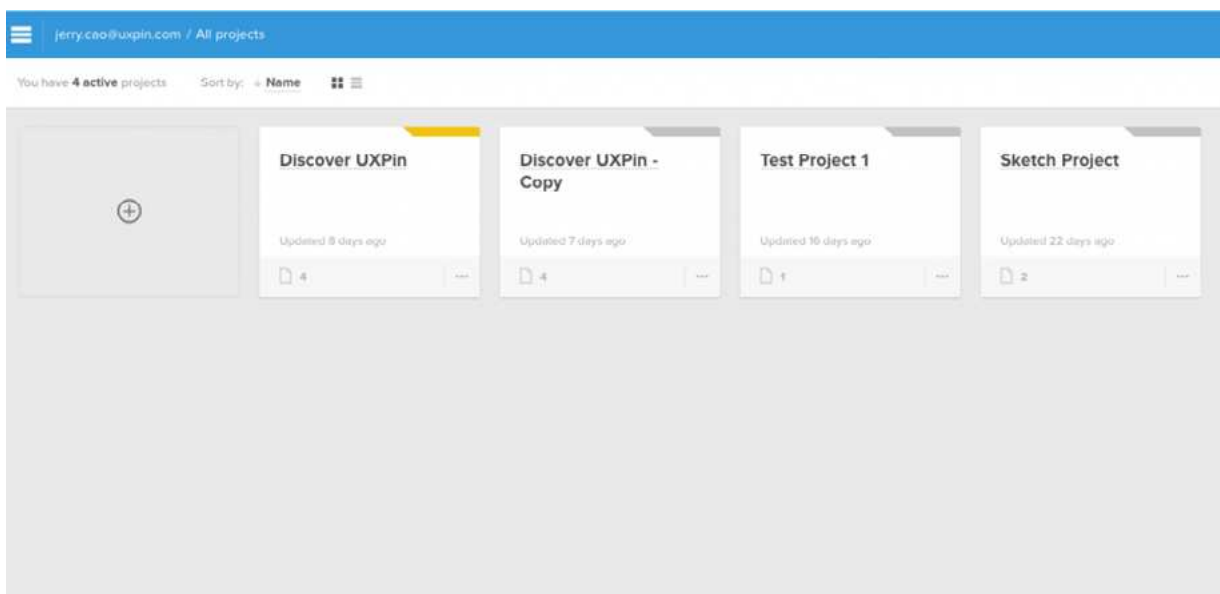
3. Download and install the Sketch plugin



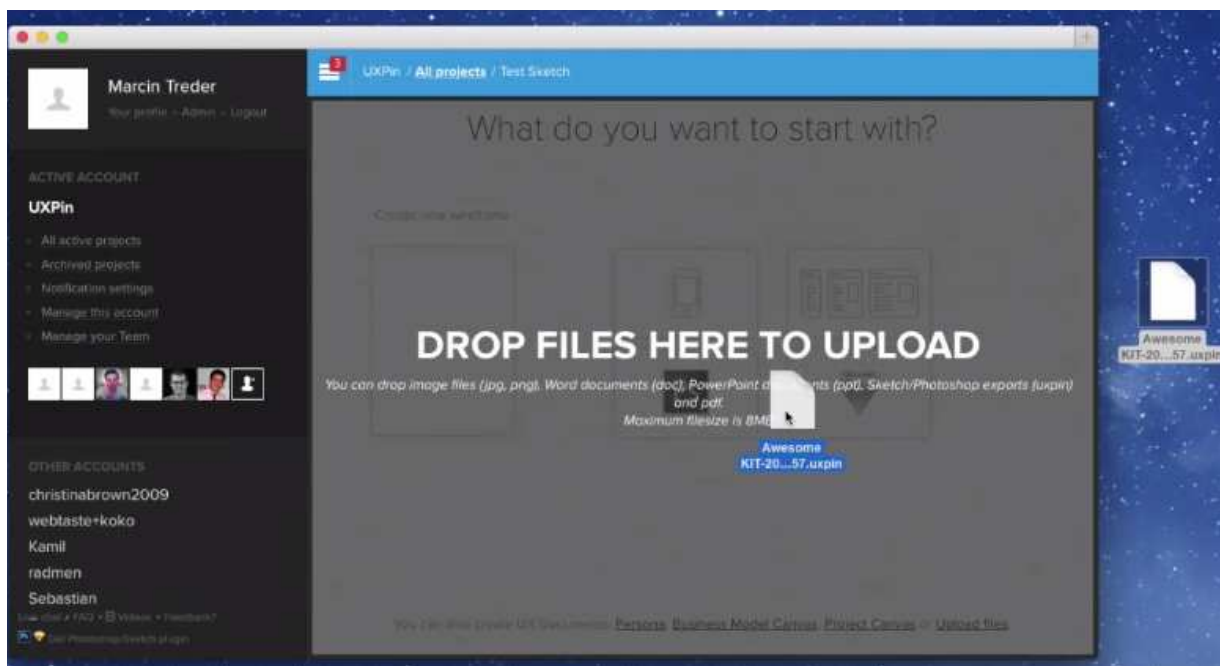
4. In Sketch, select your elements, click *Plugins* and then *Export to UXPin*. Our file is from the [Web UI Kit](#).



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.

Let's call it:

Screen Shot 2014-10-23 at

It is:

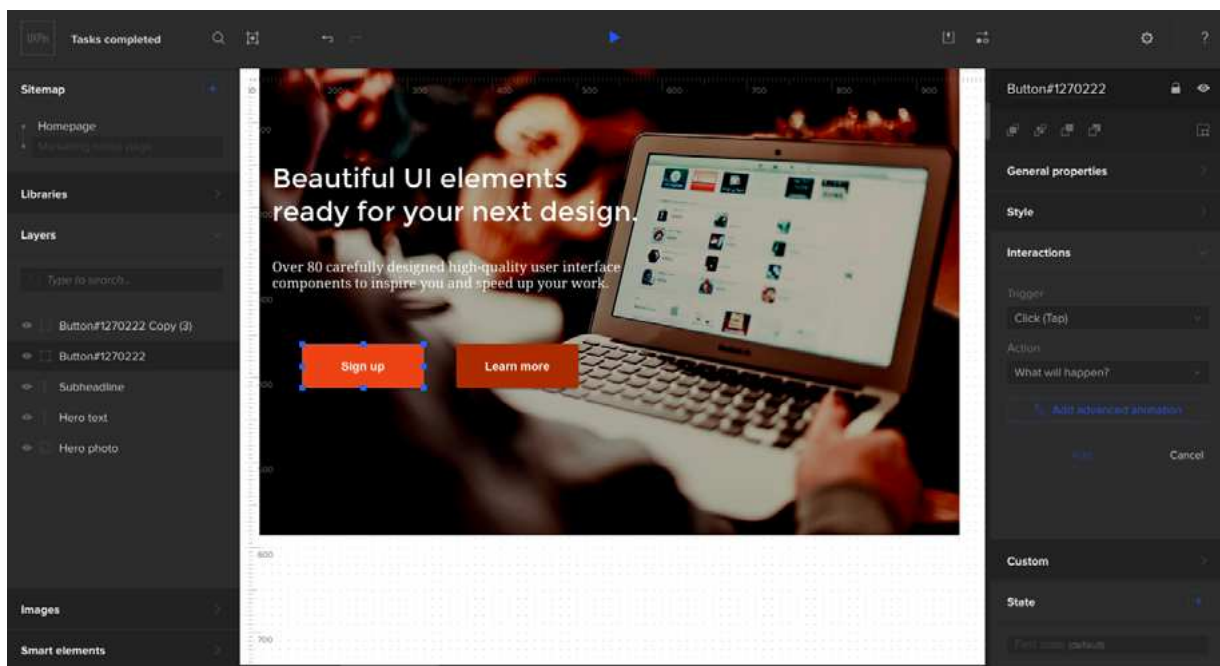
Specification

Uploaded

Save

[cancel](#)

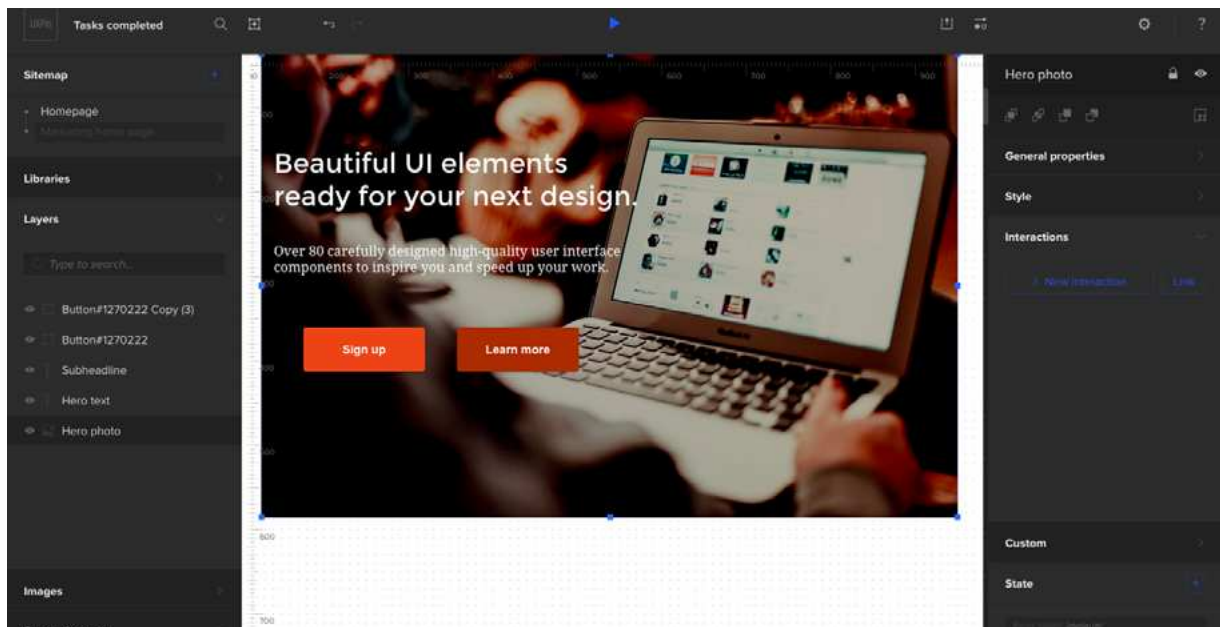
8. Once the file is loaded, feel free to rename and then click Save.



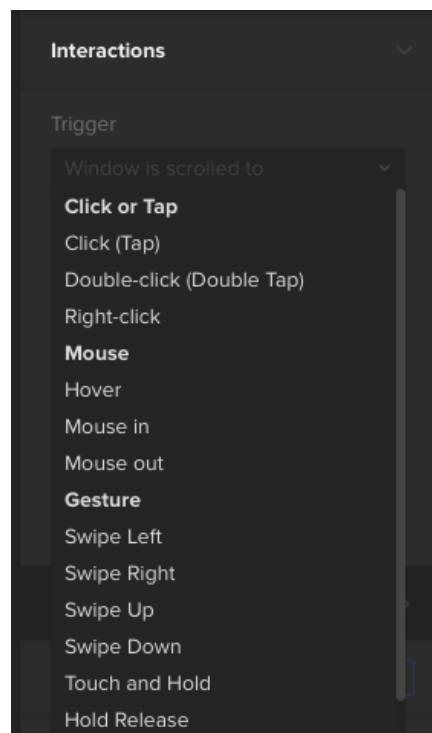
9. Done! All the elements of your Sketch file are preserved. Feel free to click around and add interactions.

Prototyping Animations & Interactions

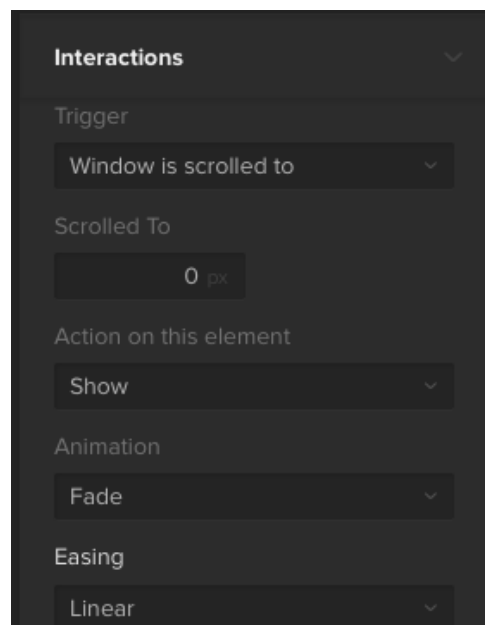
With 11 triggers and 20 element actions, UXPin allows for many custom advanced interactions. For this example, we will set a scroll trigger to reveal a display header. To learn about more interactions, check out our post on [Advanced Interactions & Animations](#).



1. Click on the display header, then click the “Properties” menu to the right. Click the eye icon to hide the header, and check “fixed position” on.

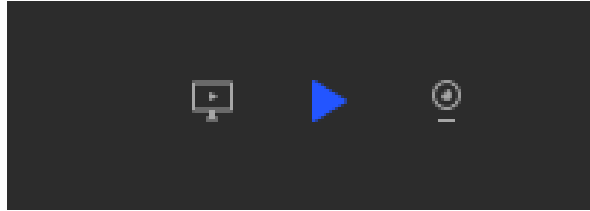


2. Click the Interactions menu, then click “New Interaction.” For the trigger, select “window is scrolled to.”

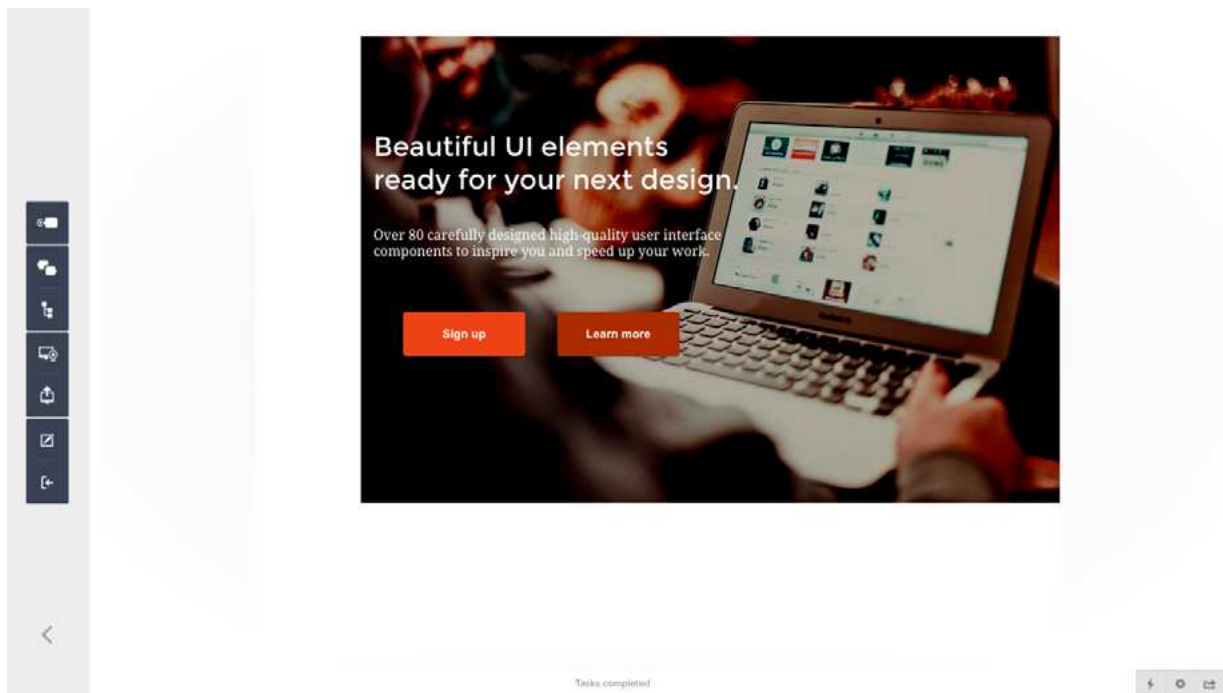


3. After you click *Window is scrolled to*, set the pixels to *100px*. Then, select Show element as the element action. Finally, make sure your animation settings are *fade*, *linear*, and *300 ms*.

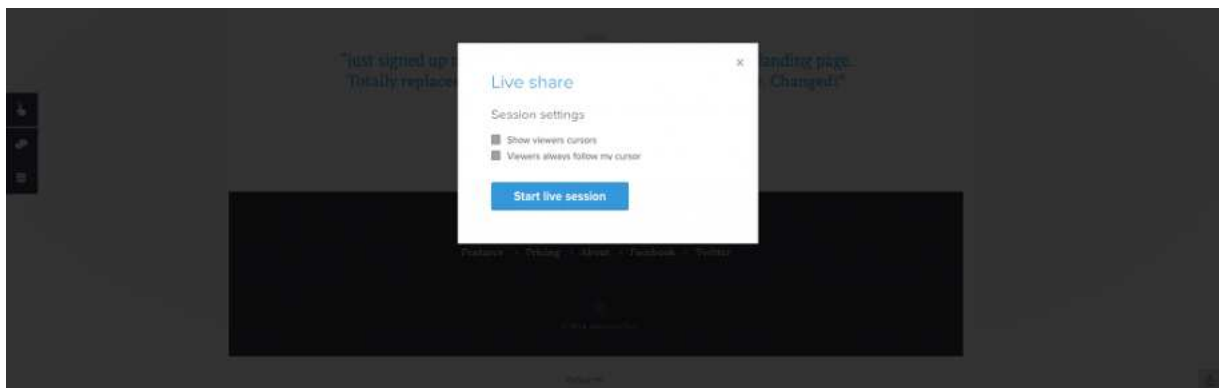
Previewing Animations & Interactions



1. To see the animation, click the “play” button above the editor’s canvas. This will open the [Preview Mode](#).



2. As you scroll down, the header will animate and appear.



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.

Everything you ever wanted in a **UX Design Platform**

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

Start using it now!