

The Guide to Interactive Wireframing



UXPin

The Practical Guide to Interactive Wireframing

Copyright © 2015 by UXPin Inc.

All rights reserved. No part of this publication text 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.

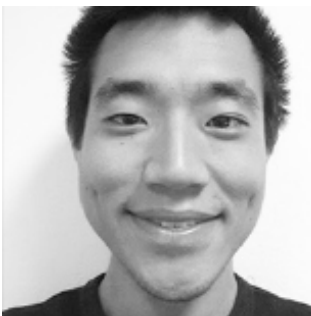
Index

How to Use Wireframes in the UX Design Process	7
What the Heck is a Wireframe?	9
The Good and the Bad Side of Wireframing	11
The <i>New</i> Wireframe	14
Creating Wireframes	15
Fitting Wireframes into the UX Design Process	17
Conclusion: How Much Should You Build?	21
From Content Wireframe to Interactive Wireframe	22
A Mapmaking Perspective of Wireframing	22
Detail Is Not Context: The Value of Content Wireframing	26
How Low Should You Go?	28
Adding Signifiers to Wireframes	30
“Interactivity” Is Not a Dirty Word	34
Conclusion	37
A Hands-On Approach to Rapid Prototyping	39
When Prototyping Is a Life or Death Scenario	39
Prototyping Your MVP	42
The Power of Low Fidelity Prototyping	44
Components of Lo-Fi Prototypes	47
A Rapid Lo-Fi Prototyping Lesson	49

How to Wireframe & Prototype Motion	55
Wireframes may be a solution	57
Wireframing motion	58
From static to dynamic	61
Simple Interactions in UXPin	63
Custom Animations in UXPin	66
Steps	67
Wrapping Up	77



Tom Green is a professor of interactive multimedia at the Humber Institute of Technology and Advanced Learning in Toronto, Ontario. He is the author of several best-selling books in the area of Flash and Flash technologies. His latest book is [Foundation Flash CS5](#) for designers, coauthored with Tiago Dias. Tom has completed DVD videos for Lynda.com, InfiniteSkills, and Adobe Systems and is a regular contributor of tutorials to [webdesign.tutsplus.com](#) and [Layersmagazine.com](#). He is also an active member of the Adobe Education Leaders and Adobe Community Professional groups, speaks at conferences and seminars around the world, and contributes regularly to the Adobe Developer Connection in the areas of Flash authoring and video technologies. [Follow me on Twitter](#)



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](#)



Marek Bowers is a UX and Product Designer. He studied Mechanical Engineering at UCLA and went on to design cooling systems for commercial aircraft. However, Marek found that his true passion was for creating web and mobile applications, leading him to become a specialist in interaction design and usability. In the past decade, his clients have included Activision, Johnson & Johnson, Dell, Lexus, Experian, and dozens of other high-profile companies.

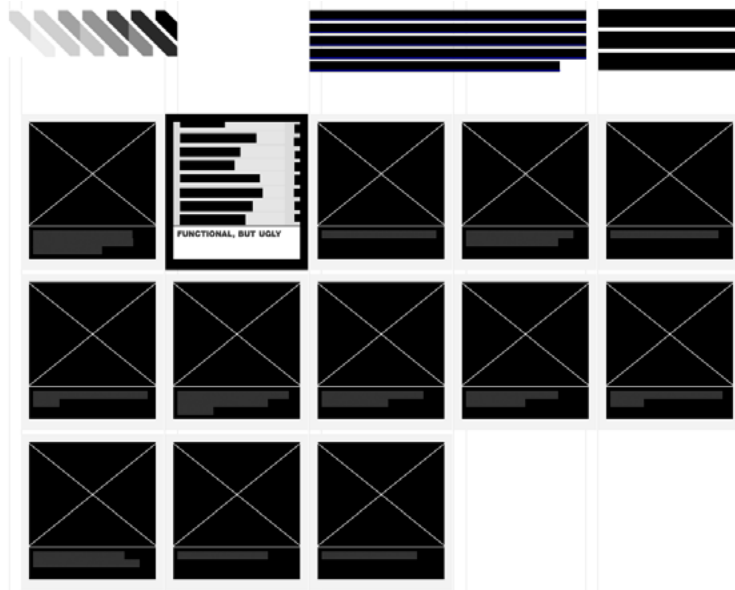
How to Use Wireframes in the UX Design Process

Given the rise of Agile and Lean UX methodologies in digital design, [many are calling into question the value of wireframing](#). They say the days of detailed throwaway deliverables are over, and that teams must prioritize interactive prototypes for their value in usability testing and living documentation.

Well, they're absolutely right.

What we want to clarify, though, is that wireframes are not what they used to be. Wireframing is not dead, it just changed. It's not about formalizing early-stage designs to soothe the panicky imaginations of stakeholders at the cost of setting unrealistic design expectations.

It's about [divergent exploration](#), creating a more structured sketch of concepts to bounce around with other product team members.



Source: [Web Without Words](#)

Wireframe for the sake of seeing intangible concepts take shape. With certain tools, you might even start building the foundation for a prototype. Don't wireframe for the sake of creating design artifacts that can easily dupe people into thinking it at all represents the final form.

Our first chapter is a celebration of wireframes. We'll explain what they are, why they're practical, the different methods of creation, and how they fit into different UX design processes.

Once we've covered the fundamentals, we'll dive into content wireframing and interactive wireframing in the next few chapters.

What the Heck is a Wireframe?

You know those black-and-white boxes with Xs and text that look like what was left over after someone scrubbed all the details off of a website or app? That's a wireframe.

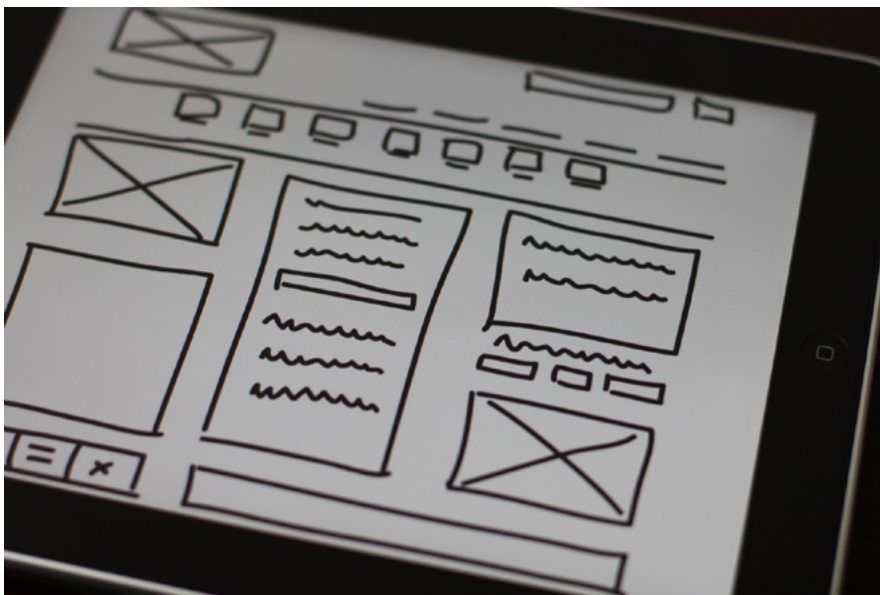


Photo credit: "Wireframe". Baldiri. Creative Commons.

Sometimes, you'll come across some confusion about terminology, with people interchangeably using terms like wireframes, sketches, and mockups to refer to rough design work. Let's clarify for a moment: sketches are the roughest, wireframes are bit more structured, while mockups are usually mid to low fidelity.

As described in the [Guide to Wireframing](#), wireframes are a framework of simple, low-fidelity designs that explore content structure during the infancy of design projects. Wireframes can range in fidelity from nothing more than boxes drawn on a napkin to something more sophisticated with actual photos and crisp typography.

In general, though, wireframes should lean more towards the lower, more simplistic fidelity – it doesn't make much sense to worry about the paint when the concrete isn't even dry yet.

The purpose for wireframes, and why we still find them helpful, is that they help you focus on the structure and outline of a site so that you know what you're building. Designers who rush into higher fidelity might easily overlook some aspect of the overall information architecture, leading to a Jenga-like redesign process. By dedicating an early phase to planning with wireframes, designers can build a blueprint of the project without being distracting by pixel perfection.

The Good and the Bad Side of Wireframing

So why is wireframing on the chopping block?



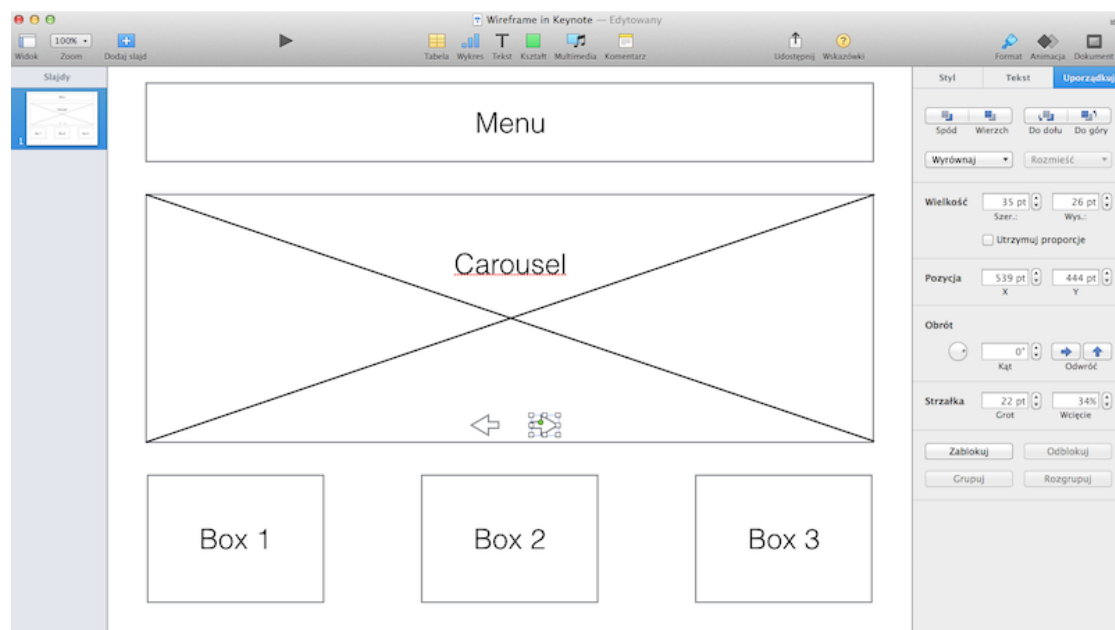
Source: "[Dawn of the Anna](#)". 55Laney69. [Creative Commons](#).

In any competitive market, it's just good business to periodically reevaluate each phase of the process. And, like every other step, wireframing certainly has its disadvantages:

- **Inaccurate depiction of final product** – Their low fidelity means that viewers must use their imagination about what the final product will look like. This hurts stakeholder presentations, and also inhibits judging the “gut” reaction the final product will give. If you want to talk about anything interactive, you'll either need plenty of notes or wild gesturing to get the point across (if at all).
- **Less suitable for usability testing** – On a related note, the best usability testing is done with prototypes since users can actually

react and play with something. While testing wireframes can hint at foundational UX issues (like if users can't find primary content), wireframes can just as easily feel like cold technical documentation.

- **Over attachment** – The more time you spend wireframing, the less willing you'll be to change any fundamental flaws in the page or site structure. It's just human nature to feel emotionally attached to work that you're proud of. More dangerously, wireframing can encourage a hand-off mentality in which you strive for greater detail so that it can be passed off better to developers (as if they're WYSIWIG machines).



Source: [Keynote](#)

In our opinion, the most valid complaints about wireframing pertain to lack of interactivity and slowing down the rapid prototyping method (which we described above).

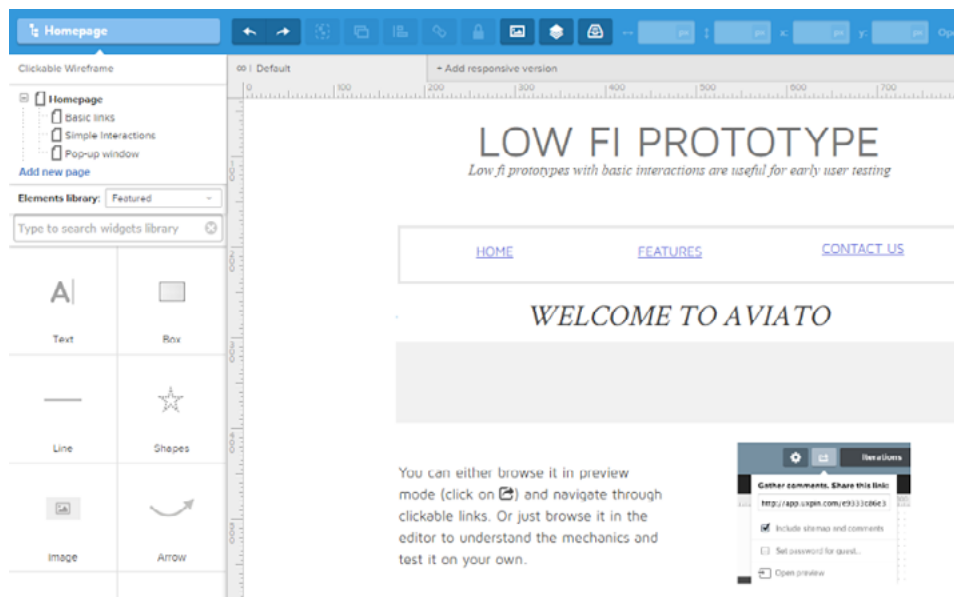
Now, let's examine a few advantages:

- **Structured outline** – This is, beyond a doubt, the main advantage of wireframes, and reason on its own to do one. A wireframe is the blueprint of the site, showcasing what goes where so that everything can be built as parts of a whole with no guesswork or “figuring it out as you go.” The alternative is steamrolling ahead into the later stages without a clear direction of where you’re going... but like building a tower on unsolid ground, it’s just a matter of time before everything comes crashing down.
- **Express unexplainable Ideas** – The design process isn’t always so easy to explain with words, and sketches can be easily misinterpreted. Building a wireframe with a tool everyone’s familiar makes communication a lot easier.
- **Promote content-first design**– All users want content, not design. When you wireframe, you need to think about how much space is required for design elements based upon the level and type of content. In fact, wireframing is best when you have some real content on hand (totally fine if it’s rough) or you can even use competitor content like we recommended in the free e-book [Web UI Design for the Human Eye](#).

While the benefits of wireframing remain the same, you can circumvent a lot of the disadvantages by adapting your process to new technologies.

The New Wireframe

The complaints in [Bermon Painter's article](#) were valid in the past – wireframes used to provide waste (especially as dead-end deliverables in Photoshop) and they used to be unable to test usability. But that view of wireframing is outdated.

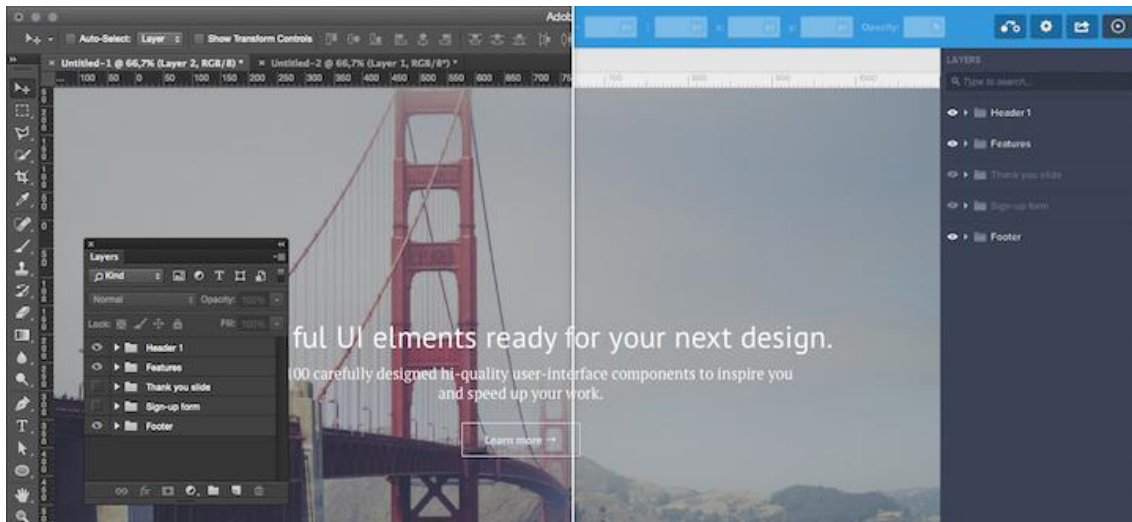


Source: [UXPin](#)

With the right wireframing and prototyping tools, wireframes **easily accommodate interactivity**. They become a kind of lo-fi prototype, able to test usability without taking up time in designing the details (yet). This, then, makes them *perfect* for rapid prototyping – the interactivity of a prototype allows them to be tested early on, but the low fidelity of a wireframe makes them easy to build.

Another common complaint is that wireframes are just another throwaway deliverable, but this, too, is no longer accurate. Again, when you use a specialized tool, your wireframes simply roll over

into the next phase – evolve, if you will – into something capable of handling higher fidelity like a mockup or hi-fi prototype.



Source: [UXPin](#) via [Photoshop](#)

Lower fidelity wireframes are the most flexible. If you create them in a wireframing or prototyping tool, you can add fidelity later in the tool or move to Photoshop if you demand visual perfection. You get a better idea of the content structure without spending tons of time either way.

Creating Wireframes

Wireframes can come in many forms, some of which aren't even digital. Choose the method that's most suitable to your product, accounting for your time and resources. Personally, we prefer sketching on paper and then wireframing digitally – we fail quickly on paper, then improve ideas as wireframes (which can be shared and made interactive).

Here's a few common wireframing methods described in the free [Guide to Wireframing](#):

- **Sketching** – The most basic method of wireframing is simply to draw your ideas on paper. Regardless of the level of detail, sketching your wireframe is fast and intuitive, though it lacks a lot of the online benefits and problems may arise when you need to share with other team members. Of course, sharing isn't too much of an issue if you sit near your team members (or aren't afraid of taking a photo and emailing).
- **Paper-cutouts** – This form of sketching takes a lot of the guesswork out of sketching and gives a better handle on usability. However, to create one properly requires time and sometimes even artistic ability, plus they also present the same problems with sharing as sketches.
- **Graphic design software** – If you're more comfortable with your favorite graphic design software, use that. With enough experience in Photoshop, Illustrator, or Sketch, these programs can feel just as natural as putting a pen to paper. Aside from no capacity for interaction, the other drawbacks are their learning curve and always-present temptation to dive into graphic detail.
- **Presentation software** – We all know how to use Powerpoint or Keynote, so this is usually the easiest method. Just draw a few boxes on a blank slide – templates like [Keynotopia](#) can even help you along. But while presentation software is very familiar and forces us to think about page flow (since slides are presented linearly),

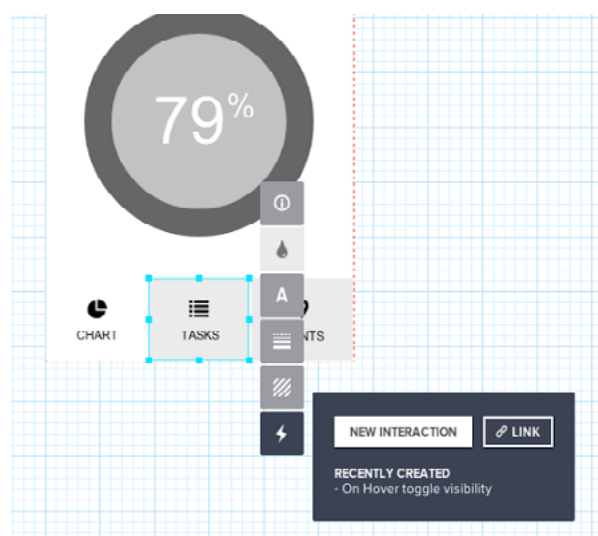
they lack substance. You won't be able to add complex interactions easily, so your wireframe becomes a dead-end deliverable.

- **Wireframing tools** – Specialized software designed specifically for wireframing facilitates the entire process and anticipates common concerns before they happen. The downside is they typically cost money – some more than others.

Choosing the best method of building a wireframe always depends on how you intend to use it, which we'll discuss next.

Fitting Wireframes into the UX Design Process

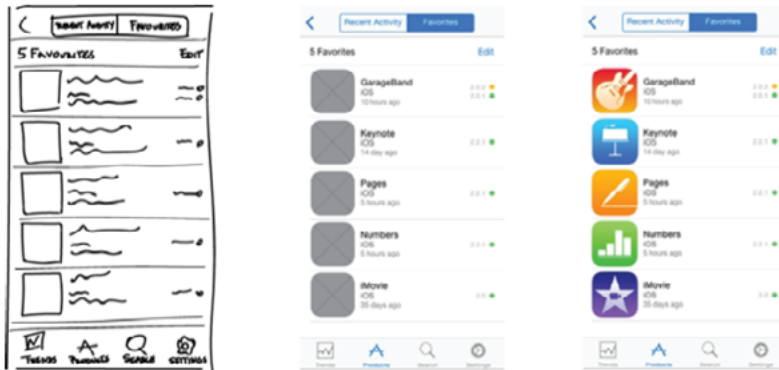
There's as many variations in the design process as there are designers, so there's no one way to use a wireframe. However, if we summarize some of the processes, we can show you how to best use wireframes under different situations.



Source: [UXPin](#)

1. Traditional Process (Wireframe ► Mockup ► Prototype)

The wireframe originates from the wireframe-mockup-prototype process reminiscent of [waterfall design methods](#). In this process, fidelity and functionality increase linearly in phases.



Source: [UXPin](#) based on [JFarny](#)

This process relies on the specialization of each phase: the wireframing phase specializes in outlining and formalizing a big picture, the mockup phase specializes on the visual details, and the prototype phase specializes on usability and interactivity. In this case, the wireframing is the backbone – as the unifying document, it provides the foundation on which the mockup and prototype are built.

If you're stuck with the waterfall method, you can still use paper, Photoshop, or a wireframing tool. Just don't spend too much time, since things always change when you least predict it.

2. Rapid Prototyping (Wireframe => Prototype)

A result of the [Agile design strategy](#), rapid prototyping forgoes the traditional method of building one product to completion, and

instead builds multiple, simpler **minimum viable products** for quick testing for smarter iteration.

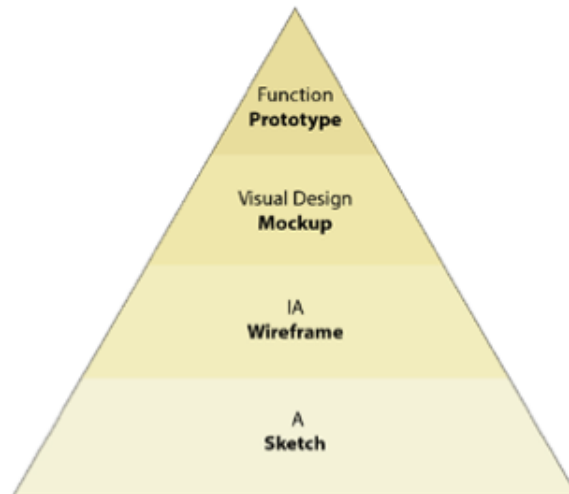
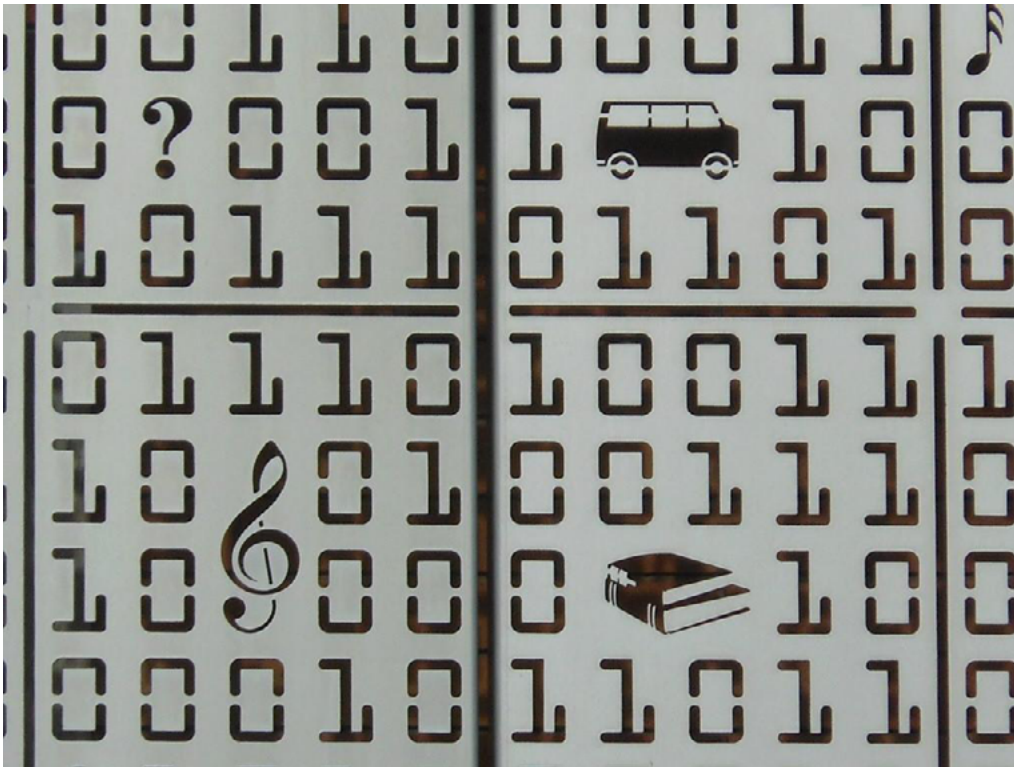


Photo credit: UXPin based on Tyler Tate via UXBooth

Despite claims to the contrary, wireframing will not impede rapid prototyping, but in fact makes it more effective. Due to the nature of quickly building, testing, and then recycling the prototypes, an interactive wireframe (lo-fi prototype) works better in this role than prototypes of higher fidelity. Interactive wireframes are easier and faster to create, but will still yield the same usability data from testing. They're so easy to build, in fact, that you can build and test several simultaneously.

3. Early Coding (Wireframes => Coded Prototype)

Alternately, you can go straight from a wireframe into an early-stage coded prototype. This method works best for projects with unique technical requirements that need to consider coding as soon as possible, but is also a preference of designers who are confident in their coding ability and want to get the ball rolling.



Source: "Binary". [Michael Coghlan](#). [Creative Commons](#).

Whatever the reason, wireframes still serve as a helpful visual guideline for HTML prototypes. In most cases, the wireframe is the only blueprint from which to design, and so anything absent will have to be addressed during the actual development, which can be problematic (unless you are extremely proficient in code).

On the other hand, a thoughtfully created outline will hold the project together and give you the chance to consider the details of layout before you're lost in the thick of coding.

If you'd like to learn more about jumping from wireframes to code, we recommend checking out Ash Maurya's piece describing his [mid-fidelity process](#). If you'd like to learn about wireframing *with* code, check out designer Matt Griffin's [article on A List Apart](#).

Conclusion: How Much Should You Build?

Serving as an early stage for planning and outlining, wireframes form the foundation on which the rest of the product is built. Because that foundation is still subject to change, don't spend days wireframing when that time is better used for building your prototype.

Think of wireframes as a design sandbox: once you've played around enough and have a basic structure in your hand, it's time to move on.

More cohesive than a sketch, and less involved than a hi-fi prototype, wireframes fill the all-important niche between abstract ideas and something you can actually see. Don't short-change your product by jumping that gap in a single bound. Take some time to conceptualize the fundamentals. And remember, with new tools and technology, you can always adapt your wireframe to be whatever you need it to be.

From Content Wireframe to Interactive Wireframe

The LoFi wireframe – is where decisions around what content goes where are made. In this chapter we explore this process and suggest that adding interactivity to LoFi wireframes is where we should also start the process of interaction design to add context to the usual boxes and arrows found in a LoFi wireframe.

Let's get started with a helpful perspective on wireframing, then we'll dive into explorations of fidelity as it relates to content wireframes and low-fi wireframes.

A Mapmaking Perspective of Wireframing

Although a content wireframe is a good first step in the UI/UX workflow, it should be regarded as nothing more than a rather broad map for the project that roughly shows landmarks on the page.

We use the word “map” deliberately because, in many respects the wireframing process – from content wireframe to hi-fi wireframe –

can be compared to the evolution of world maps. As knowledge of what made up the world increased, there was a corresponding increase in the detail contained in the maps reflecting that knowledge.

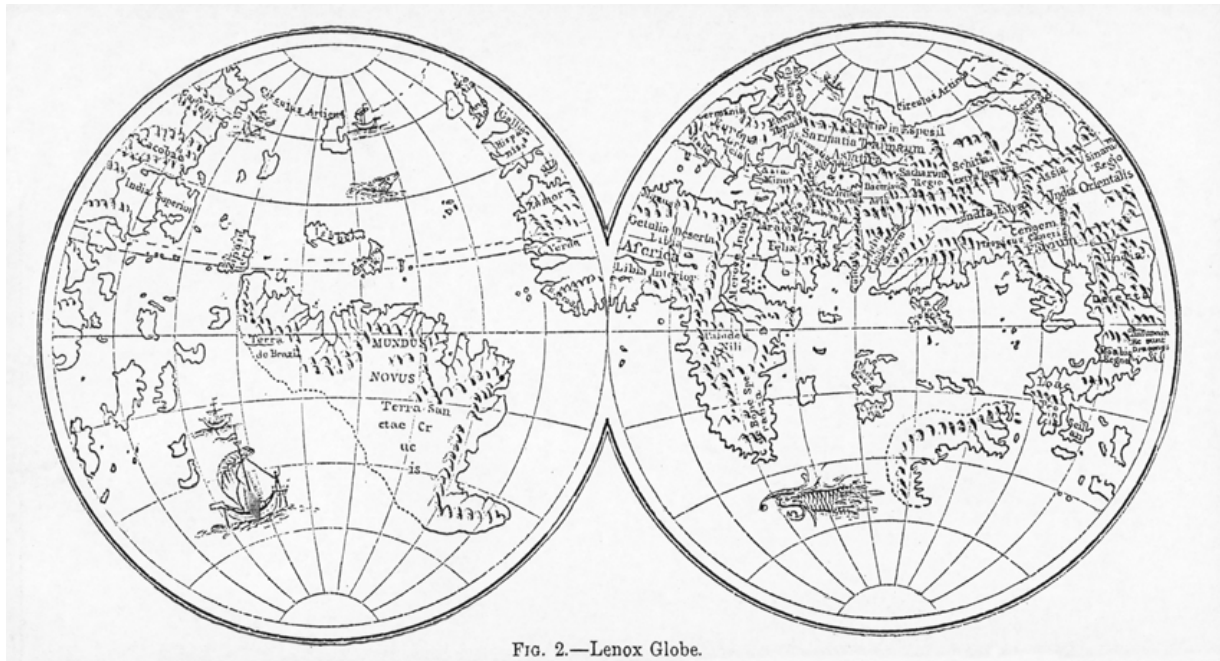


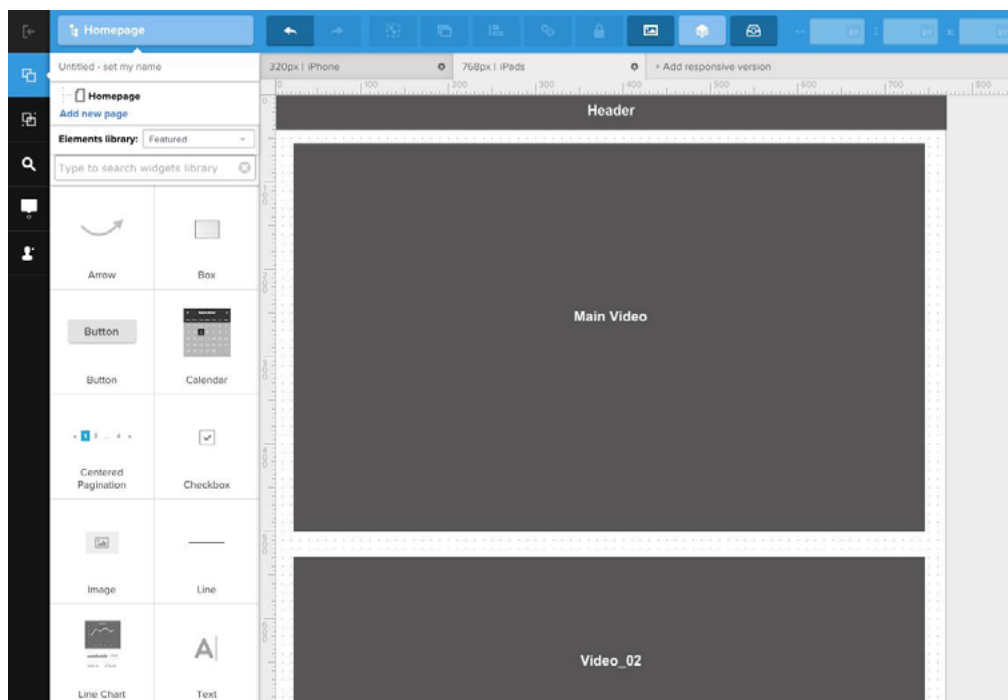
FIG. 2.—Lenox Globe.

Figure 1: “Lenox Globe (2) Britannica” by Kattigara – Own work.

Licensed under CC BY-SA 3.0 via [Wikimedia Commons](https://commons.wikimedia.org/wiki/File:Lenox_Globe_(2)_Britannica.jpg)

For example, one of the oldest globes in the world is the Lenox globe found at the New York Public library. It is a rudimentary globe that was just starting to show the continents and also contains the only known use of the phrase, “Here be dragons“, just off of the east coast of Asia.

So what does the Lenox Globe have to do with the UI/UX development process? Think of it as a content wireframe. The globe roughly points out where things are but not a lot about what will be seen when we get there.



Content wireframes are all about “blocking” out sections of the page so you can start to visualize how much space is required for each category of content.

If the content wireframe, like the continents of the Lenox Globe, roughly shows the location of where things are, then lo-fi wireframes provide the context for what is there. Think of a lo-fi wireframe like a zoomed-out version of Google Maps – you can see more context (state boundaries, major cities, highlighted geographic features), but not every person or car on the road.

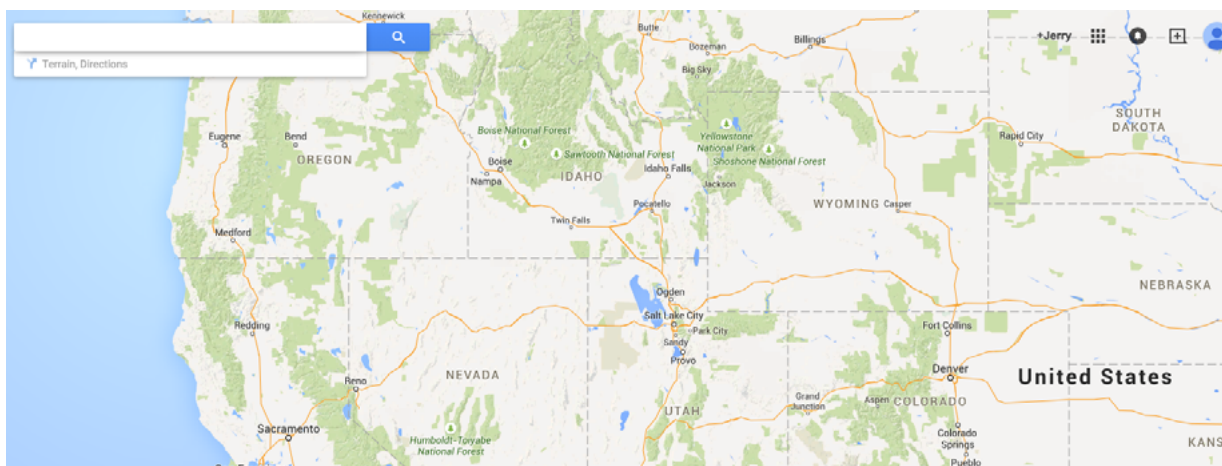


Photo credit: Google Maps

As we first described in the [Guide to Wireframing](#), wireframes (regardless of fidelity) are the blueprint for design. They're supposed to connect the underlying conceptual structure (or information architecture) to the surface (or visual design) of a website or mobile app. More specifically, they're visual representations of an interface, used to communicate the following details to get everyone on the same page:

- **Structure** – How will the pieces of this site be put together?
- **Content** – What will be displayed on the site?
- **Informational hierarchy** – How is this information organized and displayed?
- **Functionality** – How will this interface work?
- **Behavior** – How does it interact with the user? And how does it behave?

Wireframes are not supposed to represent the visual design, contain any graphic elements, or convey the brand or identity.

What you can gather from this is, as Sergio Nouvel so succinctly put it in [UX Magazine](#), “lo-fi wireframes don't ask for much content. They are meant to communicate a visual idea and explore possibilities rather than document a design.”

Detail Is Not Context: The Value of Content Wireframing

What we suspect Nouvel was getting at is the lo-fi wireframe adds context to the content blocks in the content wireframe. It answers the inevitable question: “What’s in those grey boxes?”

In modern design processes, lo-fi wireframes are helpful precisely because they force you to step away from overcommitting to static designs. Because wireframes are a stepping stone to prototypes, the more visual detail you add to a wireframe without testing with users, the more time you’ve spent working on a design that might not even work.

Wireframe only as much as necessary to lay out the structure of your prototype.

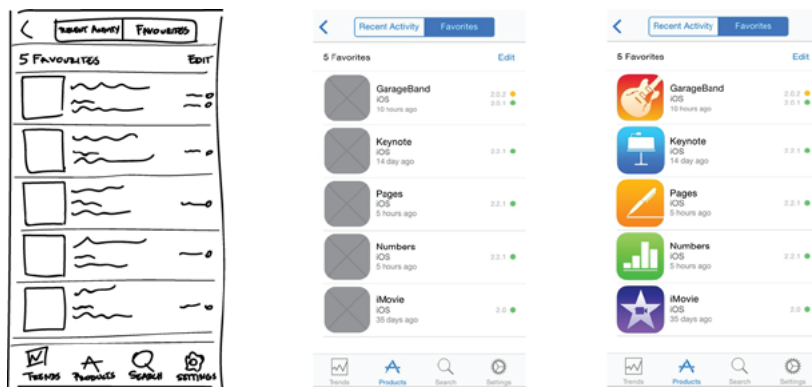


Photo credits: [UXPin](#) via [iTunes](#)

To learn how to see in low fidelity, [Wirify by Volkside](#) is an excellent tool for designers of all experience levels. Wirify is a Firefox widget whose sole purpose is to reduce a web page to a lo-fi wireframe. Just open a web page, click the Wirify link on the Bookmarks toolbar and

the site is reduced to its page structure by hiding the content and showing a lo-fi wireframe in its place.

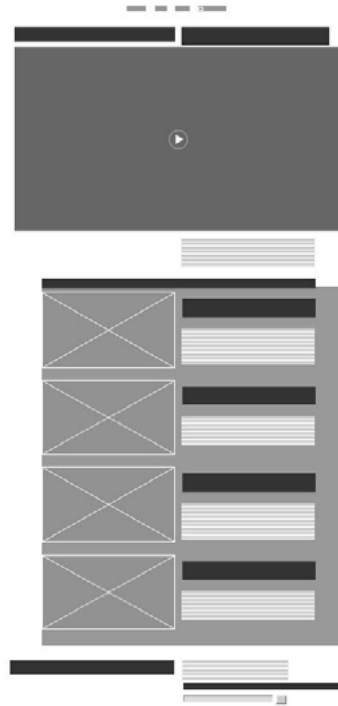


Figure 2: thisismadebyhand.com rendered by [wirify](#)

Of course, Wirify won't actually help you build your own wireframe. But it is an excellent reference for understanding the right level of detail for lo-fi wireframes.

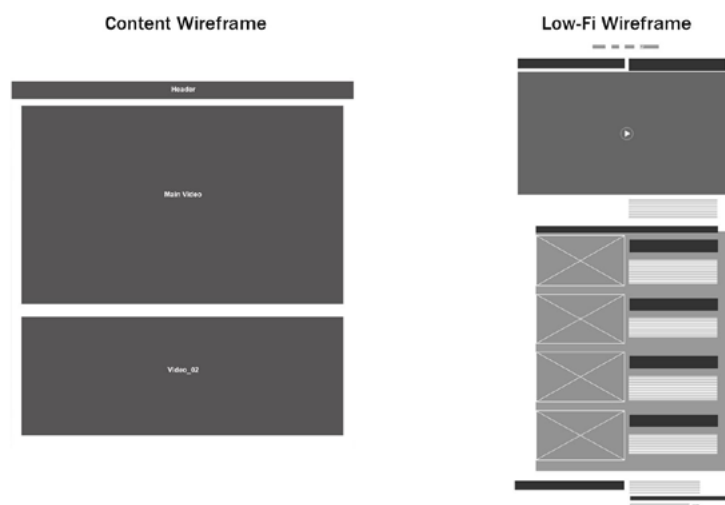


Figure 3: lo-fi wireframes introduce “context” to the process. Both created in [UXPin](#).

In the above example, you can see how we carve a content wireframe (invented by designer [Stephen Hay](#)) into a low-fi wireframe. It helps to first create a content wireframe because it helps “section off” parts of a site. Since you can quickly create content wireframes, you can move these sections around like large puzzle pieces. Once you feel comfortable with the rough visual hierarchy created from all the sections, you can then start fleshing out the structure of each section (at this point, the wireframe becomes a lo-fi wireframe).

In this case, the content wireframe helps us understand how we’d lay out a page to support a main video and secondary video. As we then sculpt it into a low-fi wireframe, we further define the primary/secondary visual hierarchy – we’ve now iterated the layout to support up to 4 secondary videos along with text descriptions.

The content wireframe helps us play with space, while the lo-fi wireframe helps us make that treatment sustainable across a whole site.

How Low Should You Go?

For lo-fidelity wireframes, you can get away with greyed out boxes, X-ed out images, and repeated lines to represent text.

In the lo-fi Wireframe we showed before, the header at the top of the page is nothing more than four pieces of text. The main video now contains a header and sub head with a short text block under

it and the remaining content is nothing more than images and text blocks. What we don't need to know yet is what text, what photos and which videos will be used (as well as other details like fonts, colors, etc.). They just add unnecessary detail which is best left to the mid-fi wireframe described in [Guide to Wireframing](#).

Nonetheless, we've still identified where all these items will go.

To refer to the “Contractor” metaphor used in the previous piece, this is the point where the contractor opens his laptop or tablet and presents the floor plan. The conversation is, “I see a fridge here. The stove here. A window here...” What you don't see is paint color, appliance make and model, countertop material or anything else that makes up a kitchen.

Even the use of color in the wireframe should be kept to a minimum since they add detail, (which can distract or even add confusion).



Photo credits: UXPin

A frequent issue I've seen with wireframes and prototypes is the variety of colors or grayscale shades, line weights, font types, and element sizing or layout – all without much thought, if any. This adds confusion as you don't know whether these slight variations should translate to variations in the end-product and, if so, what they will communicate. It's as if someone did graphic design without a style guide, or some-one spoke the English language without ever reading a page of the Oxford English Dictionary.

For lo-fi wireframes, the goal isn't depth so much as breadth – you're creating a skeleton of the site, but don't forget to include all the important bones.

Adding Signifiers to Wireframes

It never ceases to amaze me how, in a digital universe, that we revert to a flat, non-interactive medium when it comes to lo-fi wireframes.

As pointed out earlier in this article, wireframes are a communication vehicle. One thing they need to communicate is behavior which means showing how the wireframe interacts or behaves with the user. Once we move from the sketch to computer, there is no reason why interactivity can't be added to a lo-fi wireframe.

This is where adding signifiers to your wireframes comes in.

For those of you encountering the term “signifier” for the first time, let’s rewind a bit to the free ebook *Interaction Design Best Practices* and clarify some terminology.



Photo credits: “cobra chair”. thom gill. Creative Common

First, let’s define affordances. A chair, for example, affords sitting. A broom affords gripping. As I tell my students, these things are instinctive. Why? Because of the signifier. We see the chair’s flat surface, and we instantly understand it’s affordance of sitting. Likewise, we see the broom’s straight handle, and instinctively know that it afford gripping by our hands.

We just have to look at them to know what they do. When it comes to web design, designers use signifiers all the time to communicate affordances. If you see a button or hyperlink on a web page (both of which are signifiers), you know it affords clicking. The jog controller on the YouTube player affords sliding. Designers do this intuitively and instinctively based on the design patterns they have seen.

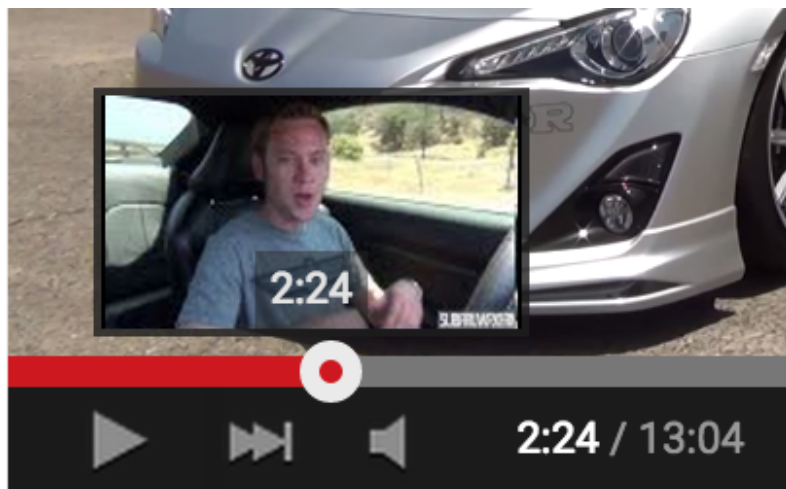


Photo credit: [Youtube's slider control](#)

When it comes to lo-fi wireframes, affordances are neither intuitive nor instinctive... or are they?

Obviously a grey box doesn't exactly scream: Clickable Button.

This is where an Explicit Signifier – adding the text “Home” to the grey box, for example – makes sense. Now that the user knows what the grey box does, it isn't that difficult to add interactivity to the wireframe.

Adding interactivity to your wireframes helps bridge the gap between signifier and affordance. Annotations can certainly help explain how a dropdown might work (and the signifier hints at it visually), but nothing beats actually demonstrating it within your wireframe.

In fact, in UXPin, you can easily turn flat wireframed pages into interactive pages. Alternatively, if you use Sketch App or Photoshop to create wireframes, these files can be [imported into UXPin](#) via drag and drop and all layers are preserved for interactions.

Regardless of how you do it, keep in mind that detail in a lo-fi wireframe needs to be kept to a minimum. Still, most sites are composed of a number of pages with an information hierarchy and adding navigation between the pages of a lo-fi wireframe hints at the experience of navigating the project.

For example, in [UXPin](#), you can also add tooltips that are tied to mouse events. For example, let's assume a user moves a mouse over an image placeholder in a wireframe. You could add a tool tip explaining, "This is a poster frame from a video." One other advantage to using tooltips is they can be used as the Annotations commonly seen in lo-fi wireframes without the confusion of lines and legends.

In the case of a "Home" navigation button, added in UXPin for example (follow along with [live link](#)), you can create the box with the text and add a tooltip that explains what the button does. From there you would turn off the visibility of the tooltip and, using the Properties for the button element, add a rollover event that makes the tooltip visible and a click event that navigates to that page in the project.



Figure 6: UXPin allows you to add tooltips and interactivity to lo-fi wireframes.

When tested in a browser, Figure 6, the user not only sees the text in the tooltip but there is a visual clue- the cursor changes to a Pointer Finger – indicating interactivity.

Now let's take this a step further. Once you've actually built out the homepage, you could then click on the "Home" button and add an interaction so that the button actually works in your wireframe. Your wireframe is now interactive with a backup annotation (in case some additional explanation is required for non-designers on your product team).

"Interactivity" Is Not a Dirty Word

Adding interactivity to a lo-fi wireframe may strike some as being "overkill". Others may regard adding it at this early phase of the project as being pointless simply because the interactive elements are not as robust as they will become in the high-fidelity static design or prototyping phases of the project. These are valid objections but what they also do is avoid answering the inevitable question, "How does this work?" The answer can only be provided by adding "light interactivity". By that we mean very simple code that moves the user from page-to-page or answers questions such as "What does this do?".

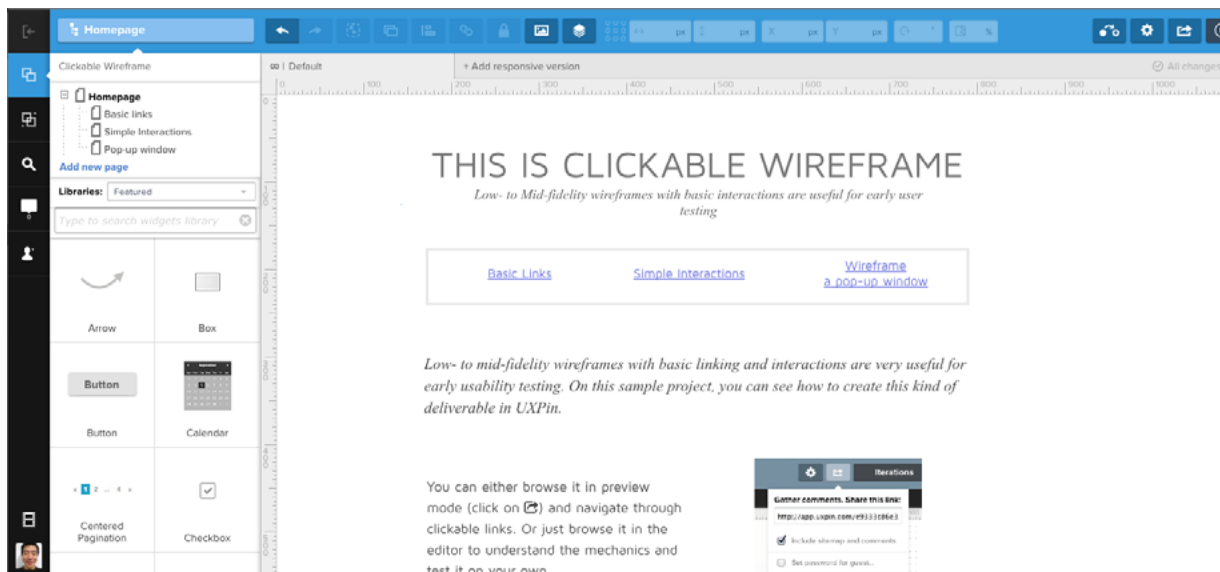


Photo credit: [UXPin](#)

Another major sticking point, which is becoming more noticeable in our current age of specialization, is, “Who will do it?”.

As Nick Bewley pointed out in his article, “[How to Decide Between Static, lo-fidelity and hi-fidelity Prototypes](#)” the multi-disciplinary approach to design could add confusion, not cohesion, to the process. “Another potential problem”, writes Bewley,” is the over-fragmentation of design ideas, which can result in a poorly conceived design that lacks unity... The over-fragmentation of responsibilities without a unified approach to product creation often results in a poorly conceptualized product. The product could be crippled by creating interaction, interface, experience, and visual design in separate vacuums.”

This is a very real pitfall but I tend to regard that scenario as being more of a project management failure than anything else. In a collaborative field where many disciplines are brought to bear upon the project, if the team does not have a clear understanding of the

goals of the project and their roles in bringing it to life, the project is doomed to failure before a pencil even hits a piece of paper.

Though the focus of the lo-fi wireframe process is “where” content will be placed, it is also an iterative process where content is shifted around to establish an information hierarchy and to create the design patterns that establish the user experience. A lo-fi wireframe is also a great opportunity to establish “how” the user will interact with the content. At this stage, light interactivity is all that is needed. Even so, the importance of adding “light interactivity” can’t be understated.

Intrinsic to “how” interactivity is applied is the story around “what” happens when interactivity is triggered. You might ask yourself the following questions:

- Is there a color change?
- Is there a dialog box?
- Does the interactive element grow or shrink in size?

There are dozens more questions that can be asked and, depending on the complexity of the project, the lo-fi wireframe stage of the process is a great place to start exploring them with interactions. These interactions could be added to:

- Navigation menu items
- Call-to-action buttons
- Pop up windows or modals
- Alert or dialog boxes

As an iterative process you have the freedom to play what I call low risk “What if... “ games as the story around that interactive element changes or is refined. Once that story is locked in at the end of the lo-fi process, the focus can turn to making that interactive experience deeper and richer as more and more detail is added to the interaction and the interactive element itself.

Conclusion

I started the piece in the last place you would consider: a globe in the New York Public Library. In many respects it can be regarded as a content wireframe for the world maps that succeeded it. It roughly showed the placement of the content- the continents – but as knowledge of the world increased, detail was also added to the maps that followed.

Lo-fi wireframing is no longer a “rough idea of where stuff goes”. Creating a lo-fi wireframe is the best way to start the visual design stage of a web project, as it allows everyone from the design team to non-design stakeholders to focus exclusively on the suggested layout – without the distractions of colour, type, or other design elements.

By concentrating solely on what is required, what goes where on the page, and the information hierarchy, the lo-fi wireframe allows the basic layout of the pages to become the skeleton upon which the project will be constructed.

At the same time we are concentrating on the content, we also need to start the process of addressing interactivity and user experience. There are some pretty powerful tools that allow us to add “light interactivity” and to communicate to the entire team the purpose of an interactive element in the wireframe.

Just keep in mind that lo-fi wireframing is an iterative process. Iterating interactivity, therefore, is just as important as iterating content placement because you start exploring how the user will interact with the content at the start of the process rather than making it up as you go along.



A Hands-On Approach to Rapid Prototyping

Now that we've reviewed the benefits and the process behind creating low-fidelity wireframes, let's take a look at how to link everything together in an interactive wireframe (also known as a low-fidelity rapid prototype).

In this piece, we'll explain why rapid prototyping leads to better design, what you need to consider when prototyping, and then show you how to get started.

But first, a prototyping success story from the unlikeliest of places: 5.5 miles above sea level.

When Prototyping Is a Life or Death Scenario

In September 2008, outdoor outfitter [Eddie Bauer](#) worked with two world-class, high-altitude mountaineers—[Ed Viesturs](#) and [Peter Whittaker](#)—to develop and test the first prototype of the [Katabatic Tent](#).

A rugged expedition tent, six prototypes were iteratively tested on the world's most grueling peaks and terrain, including Mt. Rainier, Aconcagua, Everest and Antarctica.



Photo Credit: “[everest base camp - ebc](#)” by [ilkerender](#) is licensed under [CC BY 2.0](#)

Each prototype was rigorously tested in the field by Viesturs, Whittaker and their team of guides, and then refined and improved based on their feedback.

As Viesturs and Whittaker put it, “Nothing goes to market until the guys who will be using it are happy with the product.” Eddie Bauer’s stance was the same: “If we’re going to sell it, it has to work.” Hence, after three years of prototyping and countless cold nights, Eddie Bauer finally had an MVP of the Katabatic Tent in 2012.

Now, you might be asking yourself, what does this story have to do with digital UX design? A lot, actually. Think about it... if Eddie Bauer had simply gotten approval to sell V1 of the Katabatic Tent from a room full of executives based on 2D and 3D renderings that looked pretty cool, *without developing iterative prototypes tested by actual users*, what could have gone wrong?

- **Best case scenario:** Some weekend warriors have a chilly night car-camping in Yosemite.
- **Worst case scenario:** The product fails to protect its inhabitants at Camp 4 on Everest, which is 26,000 ft above sea level (aka the “Death Zone”) where, from an emergency support perspective, is basically on the moon.

Designing software isn’t typically a life or death situation. However, the implications of a product failing as it is providing a critical need is the same.



Photo Credit: “*Climbing through the Yellow Band, Mt. Everest, -May 2007 a*”
by [Lloyd Smith](#) is licensed under [CC BY-SA 3.0](#) via [Wikimedia Commons](#)

So, let's talk about taking the leap from static wireframes to interactive wireframes, or what we'll refer to as "low-fidelity prototyping" in this article.

Picture this: You followed the [Guide to Wireframing](#) and you now have solid, low-fidelity wireframes that represent all the functional features of your product. Great! Now, how do you ensure those "functional" features are actually functional to your key personas? How do you ensure that you're going to market with your "guide team's" buy-in?

Prototyping Your MVP

The answer is... prototype... prototype.... prototype... and keep prototyping, iteratively, until you reach an MVP ([minimum viable product](#)).

This is called rapid prototyping, which [Smashing Magazine](#) defines as the process of quickly mocking up the future state of a system, be it a website or application, and validating it with a broader team of users, stakeholders, developers and designers.

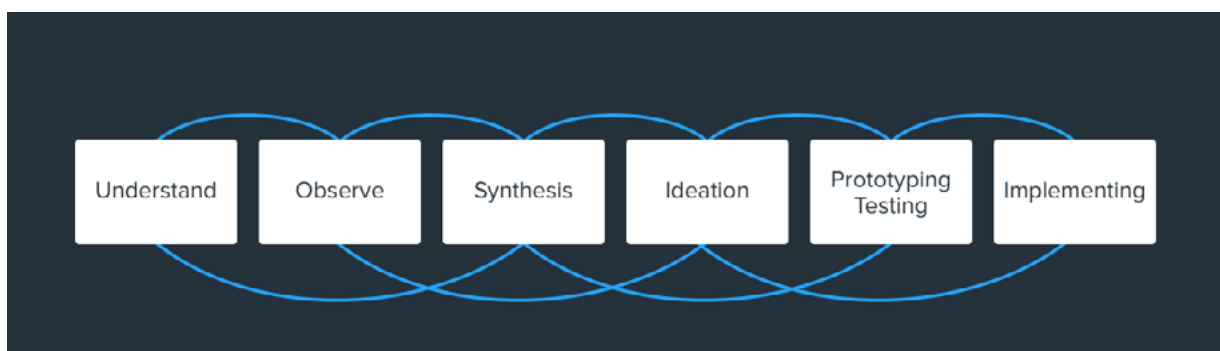


Photo credit: [Wikimedia](#). [Creative Commons](#). Edited from original.

In an article published on [UXmatters](#), [Robert Reimann](#) (Lead Interaction Designer at Sonos, Inc.; Past-President, Interaction Design Association (IXDA)) was quoted as saying:

I think, without question, that rapid prototyping, when properly employed, can make user experiences better. Getting prototypes in front of colleagues, stakeholders, and target or existing users is a great way to get quick feedback addressing your design direction, business needs, customer needs, and usability

While Reimann does admit that feedback from rapid prototypes may not yield the type of results that in-depth, in-context usability testing uncovers, he follows up by stating:

[Rapid prototyping] can certainly identify gross interaction, navigation, and presentation issues with a design, as well as areas to watch more closely in follow-on testing.



Photo Credit: “Office, Meeting, Business Partners”
by [Unsplash](#) is licensed under [Creative Commons Deed CC0](#)

Using tools that are widely available, you can easily turn your low-fidelity wireframes into low-fidelity prototypes and engage in the rapid prototyping loop of building, reviewing and refining. The goal of your low-fidelity prototype is to accurately simulate your final product's actual interactions for a group of target users, which is very similar to what Eddie Bauer did with the “First Ascent” team.

The Power of Low Fidelity Prototyping

What do you think Viesturs and Whittaker asked themselves and their team of guides when testing the Katabatic Tent prototype?

We can only speculate. But the following questions seem reasonable:

- How easy was it to set up the product and how long did it take to set up?
- What was the experience of actually using the product to meet our immediate needs?
- Did the product fail in its usability? Where did it fail?
- Is there room for improvement and, if so, where?

These questions not only apply to testing the Katabatic Tent prototype, but also apply to your low-fidelity prototype. By putting your prototype in the hands of actual users, you can gain critical insight into how the product is performing (or underperforming) against its [UX value proposition](#).

1. A Practitioner's Prototyping Tale

Here's a real story from a UX Designer working for a Fortune 500 tech company that's attempting to break into the healthcare space:

At [my company], we've always been great at consumer tech. But now we're trying to break into the healthcare space. One of my first projects at [this company] was to build a brand new [HIPAA-regulated](#) portal for physicians, nurses and technicians to easily view radiology images in the cloud. My focus group wasn't responding to my wireframes. They didn't get it because they didn't understand the interactions between screens. I decided to build a low-fidelity prototype using my wireframes so that the group could see how clicking a link here would take me to a screen there, or how selecting a drop-down option on this screen would show me an option on that screen. After showing my focus group the prototype, they got it and I received valuable product feedback."

But that wasn't the final step for this particular UX Designer.

He told me that once the project team saw the low-fidelity prototype in action, they were able to make insightful observations that impacted the future design. Because the low-fidelity prototypes were so fast and easy to create, the UX Designer was able to crank out a bunch of them with new features and enhancements to test simultaneously. Each prototype yielded valuable usability data from his group's testing.

2. Using a Low-Fidelity Prototype to your Advantage

Let's briefly review some of the key advantages to low-fidelity prototyping:

- **Get quicker and potentially more feedback:** When you give test users a beautiful visual design, the first thing they may assess are the colors, typography, imagery and general visual “feel” of the design. With a low-fidelity prototype, your testers can jump right into the nitty gritty of the functionality, without being distracted by what makes the design pretty.
- **A/B testing can be done faster/ easier/ cheaper:** When you can crank out 10 low-fidelity prototypes in the time it takes to create 2 high-fidelity prototypes, you can see that testing feature and content variations can be done much faster in low-fidelity. You can pivot much quicker.
- **Focus on flow:** Low-fidelity prototypes allow you to link up multiple wireframes to create flows. In this way, you can test the effectiveness of the order of things, rather than just the elements that the user sees on-screen. You can validate that sequences of interactions / actions make sense for users.

Components of Lo-Fi Prototypes

Now that you have the “why” behind building a low-fidelity prototype, let’s talk about the “how.” There are two main components of lo-fi prototypes:

1. Interactions:

What interactions will you be showcasing in your prototype?

Each interaction includes a trigger (i.e. “What triggers the action?”) and an action (i.e. “What actually happens?”). A trigger can be a tap, click, hover, swipe, key press, page-load, etc; whereas, actions can include show element, hide element, go to page/URL, scroll, disable/enable, check/uncheck, etc.

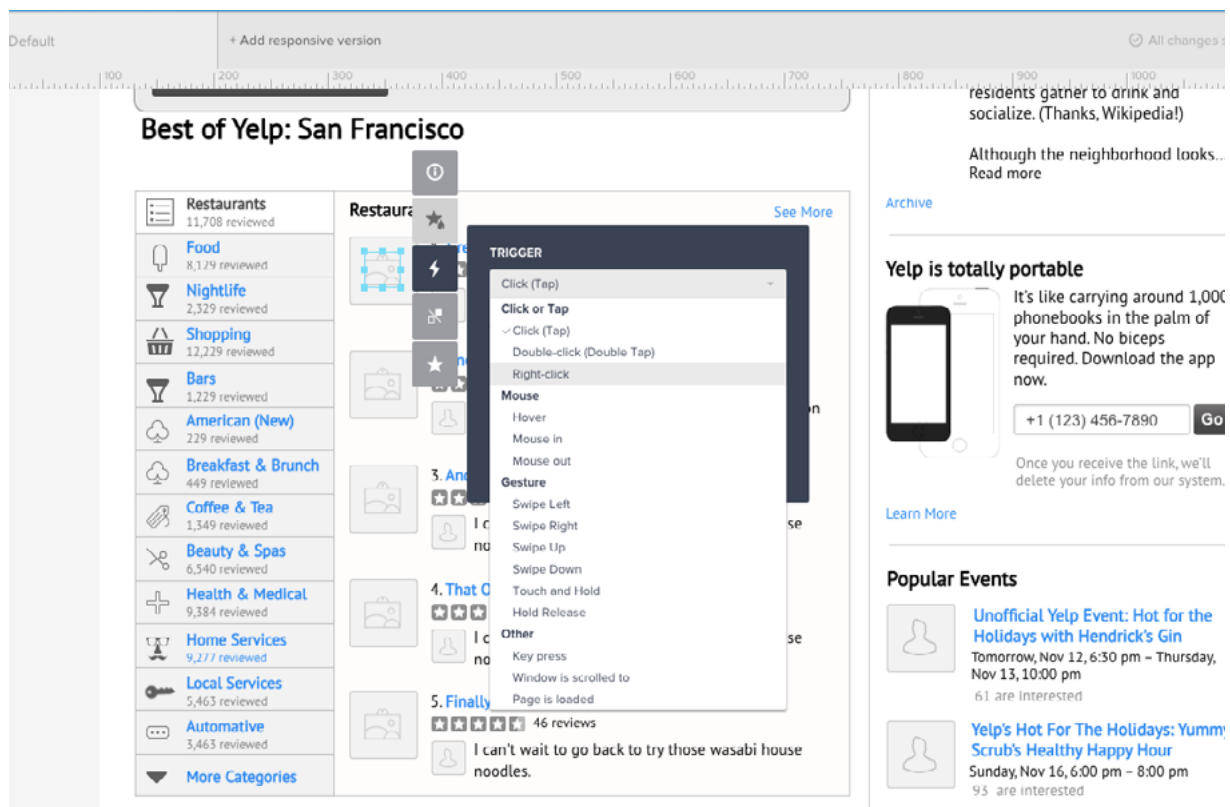


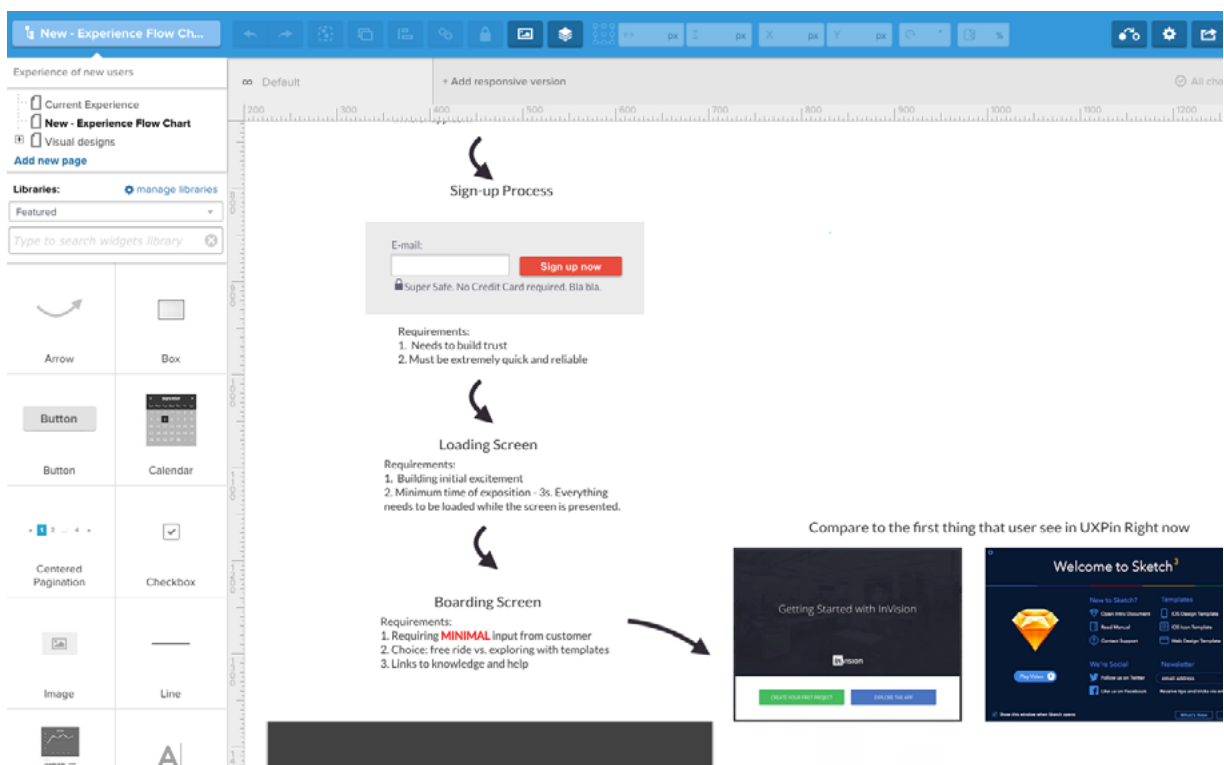
Photo credit: UXPin

2. Flow

What are your user flows?

If you have an idea of which personas will be testing the prototype, think about what happens before and after a user is on a particular page. Like we first described in *Interaction Design Best Practices*, you can link up your pages and create as many flows as you need to, especially if you have several personas.

For example, for an application that does reporting and analytics, you may have one flow for the Executive who will be reading the reports and an entirely separate flow for the Business Analyst who will be building the reports. The experience is vastly different for those personas, so think about how they will be interacting with your prototype.



A Rapid Lo-Fi Prototyping Lesson

[UXPin](#) offers some excellent tools for building low-fidelity prototypes, including [responsive breakpoints](#) that can help you simulate site behavior in different screen resolutions, setting up [interactions](#) between UI elements and pages, creating [multiple element states](#), and more.

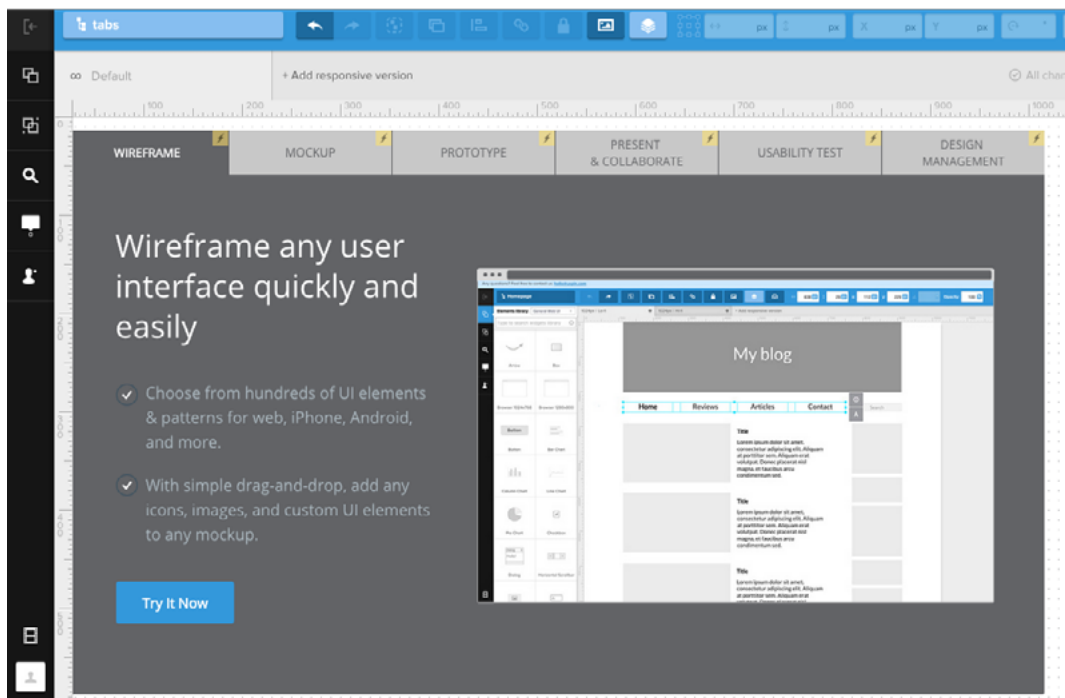
In this hands-on example, we'll create a multi-state prototype to simulate the interactions a user has with a tabbed container. Each tab contains a different value proposition, which is a combination of feature benefits and imagery.

If you haven't already, go ahead and create a [free UXPin account](#) so you can follow along. The live preview of the project is [available here](#), so feel free to open the project for quick reference.

1. Step 1: Create low-fidelity wireframe in UXPin

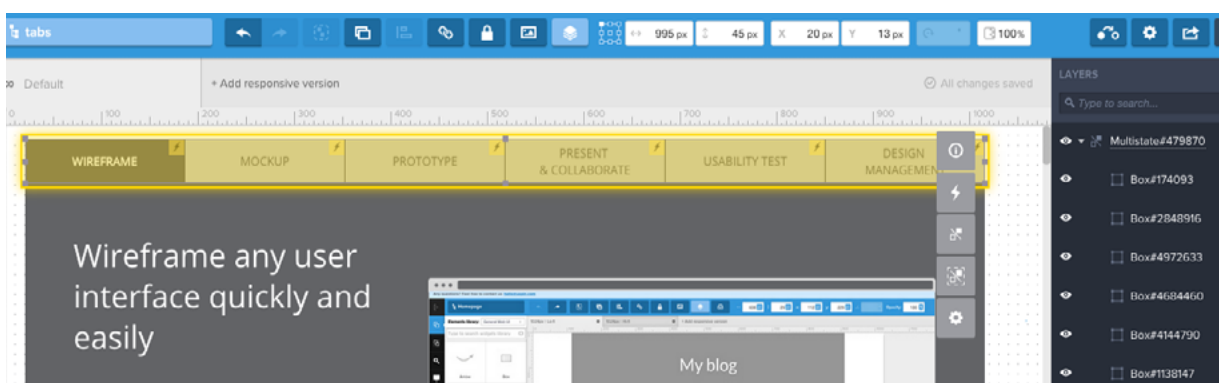
Using UXPin's wireframing tools, we created a low-fidelity wireframe with the following elements:


- Boxes
- Text Elements
- Icons
- Image Placeholder
- Button

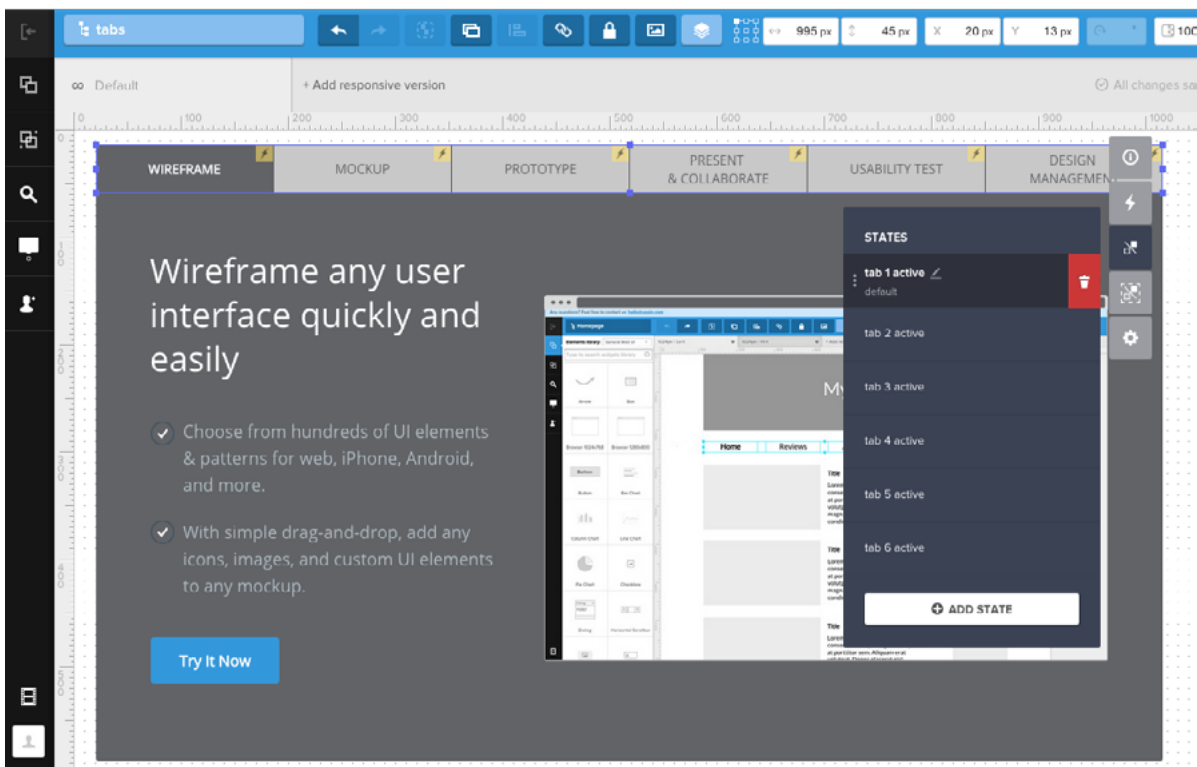


2. Step 2: Create multiple states for the tab group

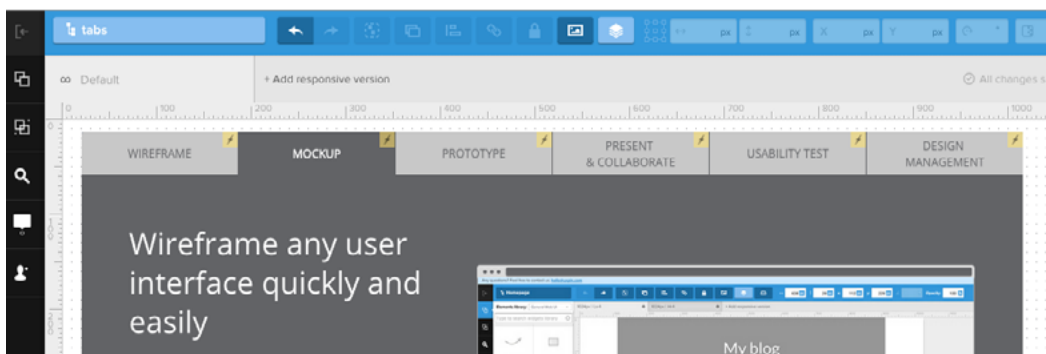
To create a prototype that allows us to interact with the tabs on-screen, we need to create multiple states for the tab group. Below, we selected all 6 tabs, right-clicked the selected elements and choose Grouping > Multistate.



Then, we selected the newly formed multistate tab group and clicked the States icon: . Here, you will see six states shown: each state represents one of the tabs being selected (and the others de-selected).



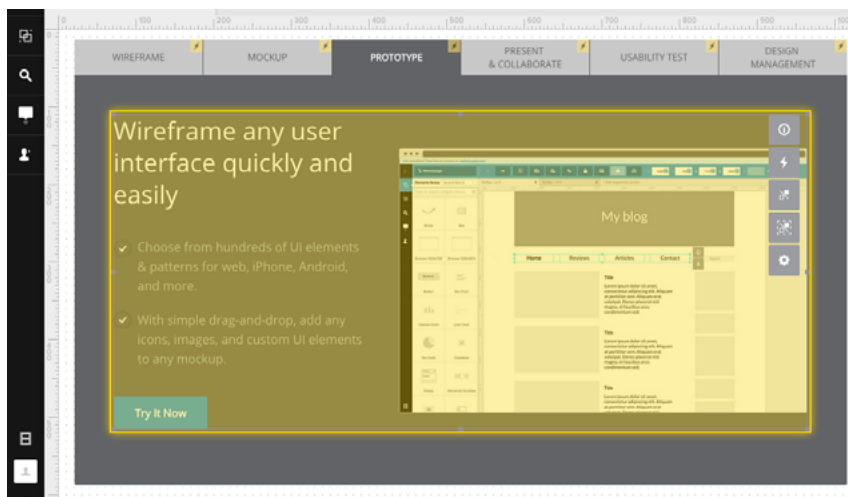
The state of the second tab when activated is shown below.




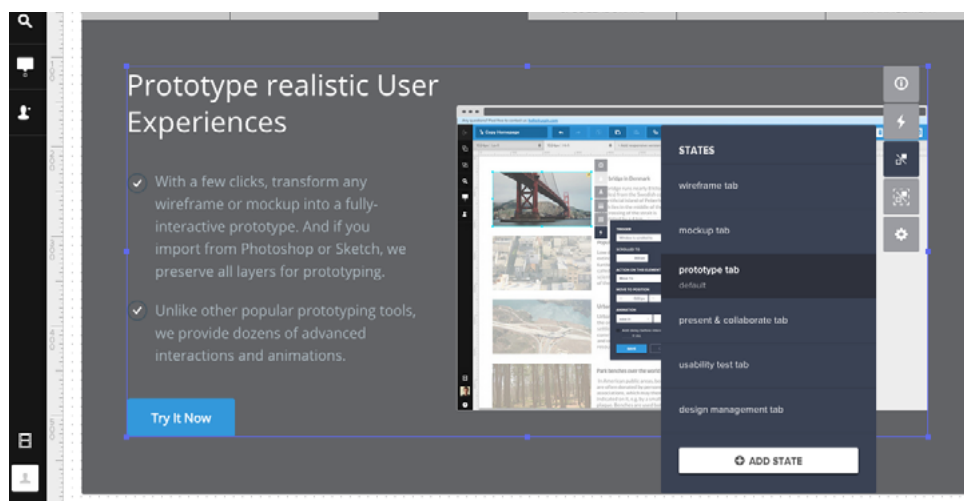
We proceeded to do this for all 6 tabs.

3. Step 3: Create multiple states for the content inside the container (content and image)

Following the same process as we did for the tab group, we selected all the content inside the container, right-clicked the selected elements, and choose Grouping > Multistate.



Then, we selected the newly formed multistate content group and clicked the States icon (). Again, you will see six states shown: each state represents the content changing when a different tab is selected.

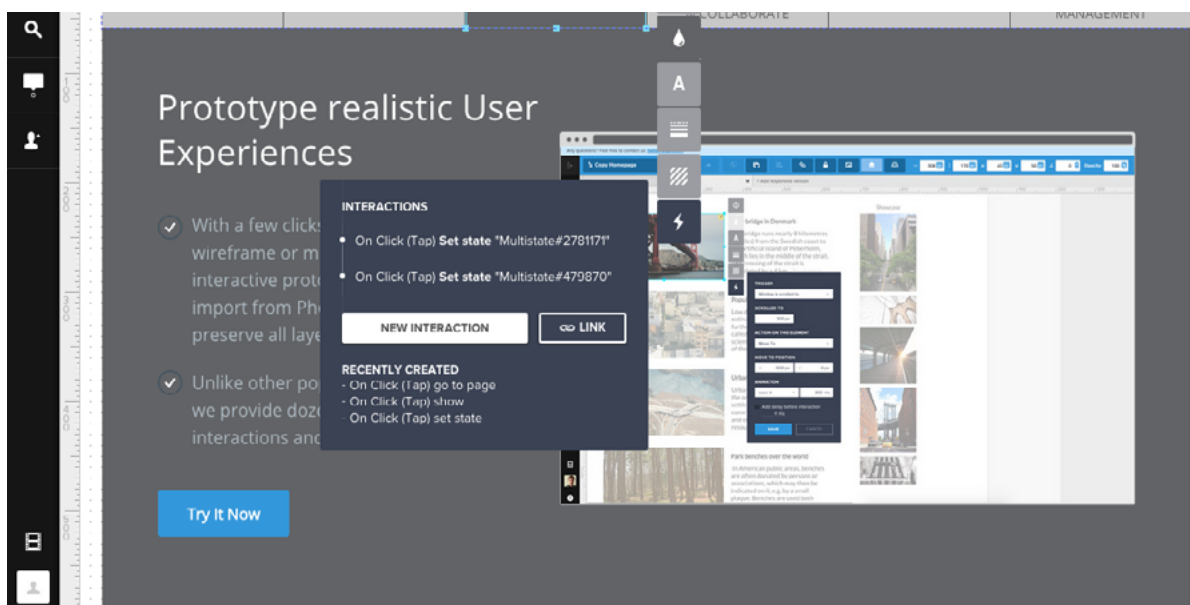


We proceeded to do this for all 6 tabs / content areas.

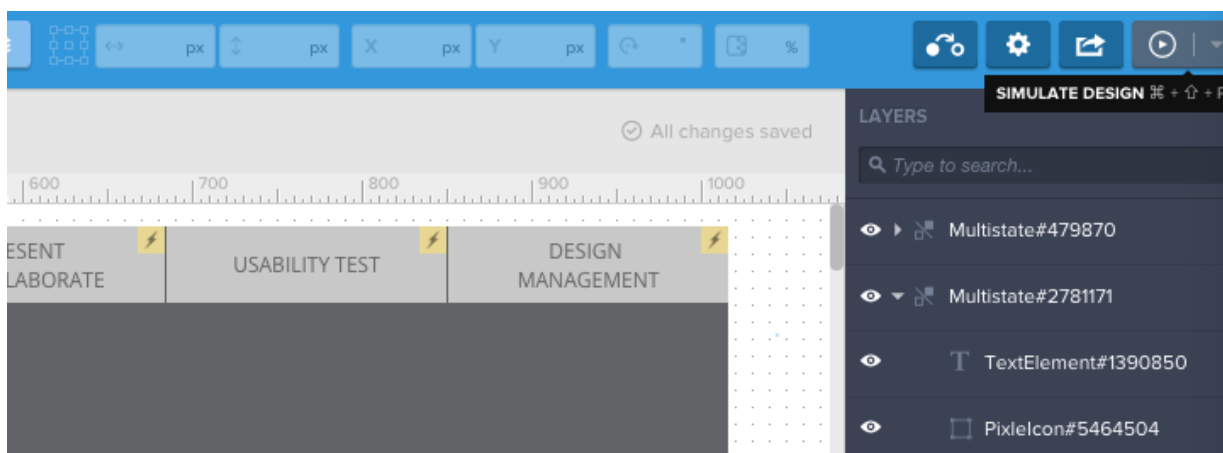
4. Step 4: Add the interactions of clicking tabs.

Lastly, we added the interactions of clicking the tabs by selecting a tab and clicking Interactions ⚡. Each tab has two interactions:

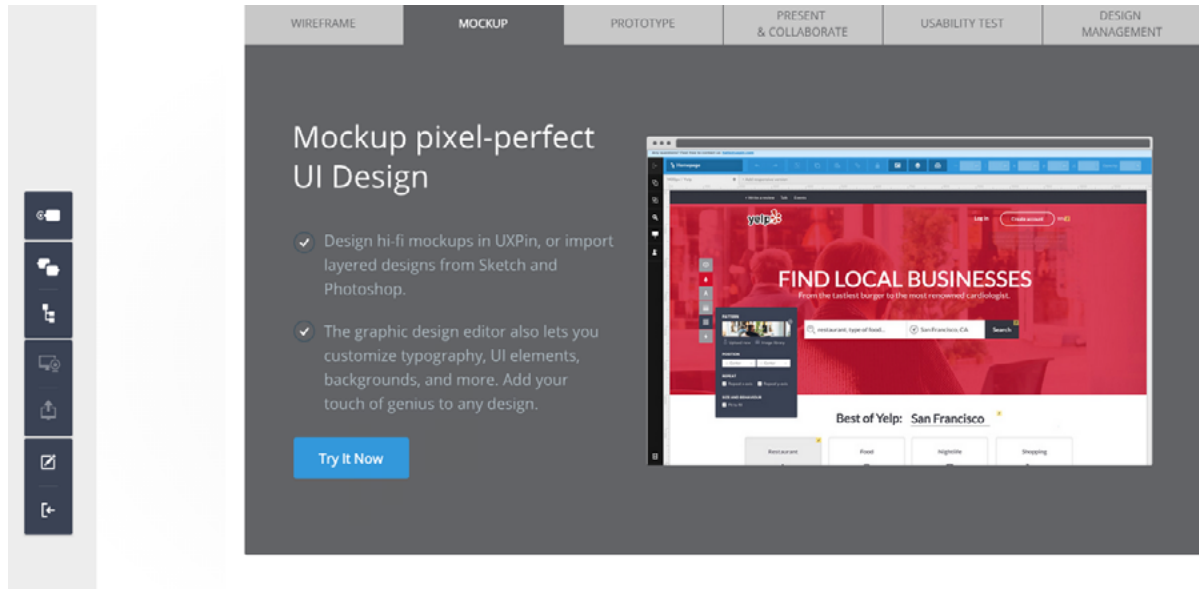
- On Click action, set the state of the tab to the selected state.
- On **Click** action, **set the state of the content container** to the content matching the tab.



Not that we're done with that, we can preview our clickable, low-fidelity prototype by selecting the Simulate Design button in the upper right-hand corner of the screen.



And wa-lah! Our new prototype! Check out the [live preview](#) to play with the functionality we just created.



In this article, we’ve covered why we create rapid low-fidelity prototypes and how.

Now that we have our low-fi prototype (i.e. interactive wireframe), we can iterate, iterate, iterate on our prototype to put our best digital MVP out there, much like the Eddie Bauer “First Ascent” team did to achieve their Katabatic Tent product win.

There is, of course, a new wrinkle to the workflow. We can’t avoid the importance of motion, especially in the mobile space. We’ll look at this aspect of the interactive wireframing process in the next chapter.

How to Wireframe & Prototype Motion

Let's start with a typical scenario.

Imagine me holding a ball. Now imagine me dropping the ball.

What did you see?

I am sure there will be as many answers to that question as there are people reading this.

None of us have met which means I can assume there were varying ideas as to how tall I am. I didn't tell you what type of ball I was using so the ball I dropped could have ranged in size from a golf ball to a basketball. I didn't tell you how far the ball would fall which means your vision could have ranged from shoulder-height to the floor to maybe an inch or two above a tabletop.

All these differing visions for a simple ball drop.

This example may be simplistic but it raises an important question: Are we not asking our clients to imagine us dropping a ball when it comes to describing motion in a wireframe?

This is an important question because trying to explain an interactive dynamic process in a static medium leaves us open to misinterpretation. For example, when it comes to mobile, there is always an interaction before there is motion: Touch a button and something moves.

We usually use the techniques shown below to demonstrate motion and movement in a wireframe:



Figure 1: Showing motion in UXPin.

As designers, we instinctively understand what is being communicated.

In the top image, the interaction is a “Tap”. Tap the button and the next screen appears. The middle one is a bit more explicit by adding

the word “TAP”. In the bottom example the interaction is a “Swipe”, as indicated by the arrow. We all understand that.

What we don’t understand is how we get from screen-to-screen with a tap or a swipe. Does the next screen magically appear? Does the next screen move from right to left? Does it zoom in? Does it zoom out? Does the image shown in the swipe move as well? How fast is the motion?

Now comes the fun part: Try explaining that to the client or others not involved in the process. Each person will “see” it differently.

Wireframes may be a solution

Wireframing is an iterative design process, which is exactly how wireframing interactivity and motion should also be approached.

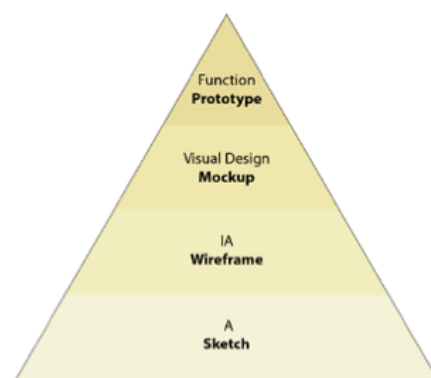


Photo credit: UXPin based on Tyler Tate via UXBooth

If the purpose of a Content Reference Wireframe is to focus on content placement and Low-Fi wireframes focus on context, there is no

reason why interactivity and motion can't be approached from the same perspective. Starting with a static Content Reference Wireframe, then progressing to a Lo-Fi wireframe, and finally working towards a functional prototype is a common UI/UX workflow.

As we move through the process we start with a simple “Here's how...” and finish with “Here's what...”.

With interactivity and motion the “Here's how...” starts with initial motion concepts and, as we move toward the “Here's what...”, the motion becomes more refined as time, speed, distance, easing and effects such as zooms and opacity changes are added and tweaked.

Does this mean extra work? Not really.

Motion is consistent and limited in any mobile project whether it be aimed at Android or iOS system or devices ranging from smartphones and tablets to desktop monitors. The work is more in finding that consistency and then applying it than anything else.

Wireframing motion

A great place to start thinking about motion, especially in the mobile space, is to use the boxes used in a Content Reference Wireframe to move things from “here to there”.

At this stage of the process, you don't need to consider just yet whether the project will be used on the Android, iOS or other operating system. What is important is how things move.

As Google so brilliantly points out in its [Material Design Guidelines](#):

“Sometimes, it is difficult for a user to know where to look or understand how an element got from point A to point B. Carefully choreographed motion design can effectively guide the user's attention and focus through multiple steps of a process or procedure, avoid confusion when layouts change or elements are rearranged, and improve the overall beauty of the experience. Motion design should serve a functional purpose.”

Adding a motion element to a Content Reference Wireframe meets the “carefully choreographed” requirement and showing how things move from here to there helps “avoid confusion when layouts change or elements are rearranged”.

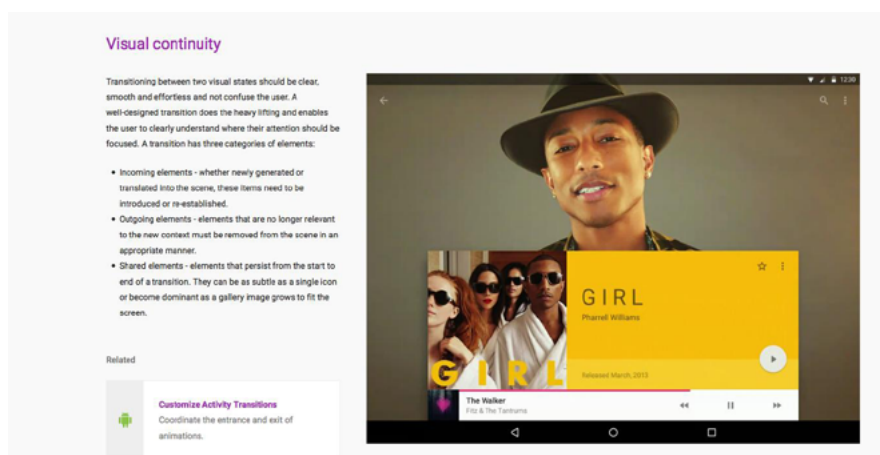


Figure 2: Showing how things move from “here to there” through Google’s Material Design Guidelines. Photo credit: [Material Design](#)

Remember, we are not talking final product or even a LoFi wireframe. At this stage of the process, we only need to show how the static finger and arrow icons, indicating a swipe, in the wireframe roughly work. This can be shown through the motion of a box or, in the case of a tap, a circle where the screen is tapped by a finger.

Even with something as simple as the swipe, there is a lot to consider. For example:

- How long will it take to move from “here” to “there” ? To the average person one second is a miniscule unit of time. On a device, that perception of one second can seem to be an eternity. In fact a lot of motion on a device occurs in fractions of a second.
- Is the motion fluid? The motion has to appear to be smooth. The word “appear” is deliberate because even a slight one-pixel “bump” will be seen.
- Is the motion natural? Nothing starts and stops abruptly. When things move they accelerate and decelerate- a technique referred to as “Smooth In and Out” or “Easing”.
- Is the motion even necessary? This is a crucial consideration. Just because you can “spin” something or move it from here to there is not a compelling reason to add it. You are just showing off rather than adding motion to “serve a functional purpose”.

From static to dynamic

The issue with using a Content Reference Wireframe to describe motion is that you are trying to communicate a dynamic process through a static deliverable.

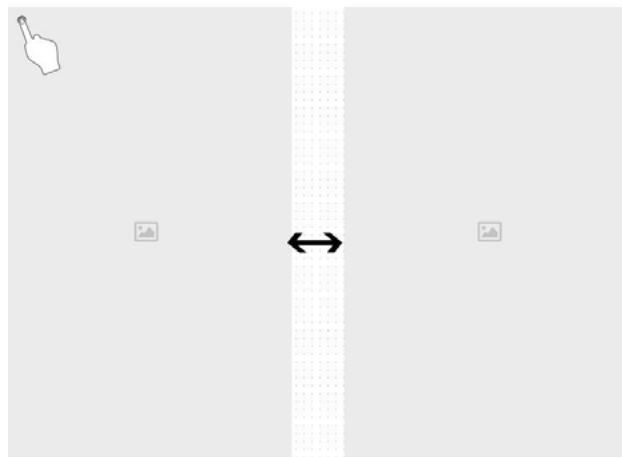


Figure 2: A tap slides one image over the other.

In the above image a tap indicates one image slides into place and replaces another image. What is not being communicated is the “How”. You can’t, for example, tell if the right image slides over the left image or if the right image moves to the left at the same time the image on the left is sliding out of view.

Though you could use Notes in [UXPin](#) or other software to communicate the motion, how that motion unfolds is determined by the user’s perception of the message.

Just as a Content Reference Frame starts the process of focusing on content, providing a Motion Content Reference Frame also starts the process of focusing on how content moves from “here to there”.

In fact we are starting to see this happen. For example, Javi Pérez a product Designer out of Huelva, Spain, regularly posts [his examples and experiments](#) around interaction and motion in the iOS space on Dribbble.

In the [this example](#), the two dots indicate the position of the fingers on the screen and the white box is an open app. Using a gif animation, Javi shows how a pinch will close the app.

Another example would be demonstrating how one transitions between icons. Jordi Verdu, a UX Designer at Google, explores this using an [animated gif](#). Trying to explain the transition between, for example, the magnifying glass and gear icons is open to misinterpretation.

When exploring motion, the goal is to start with a “Here’s how... “ conversation and iterate your ideas. Once you settle on an approach, the process can then move into one where the motion concepts are refined as they evolve into a “Here’s what...” conversation.

We’ll start with [UXPin](#), which is the wireframing and prototyping app I’m most comfortable with.

Simple Interactions in UXPin

If the intent of motion is to move things from “here” to “there”, then UXPin may be a good starting point for the process of adding motion to a project. To put something in motion you need to know four things:

- What triggers the motion.
- Where the motion starts.
- Where the motion ends.
- How long it takes to get from “here” to “there”.

This is where UXPin’s Triggers and Interactions can get the process started.

In this example I start with a box. The plan is to have the box move downwards by 200 pixels when it is touched. To accomplish this you draw a box and, using the Properties Panel, fill it with a dark gray. With the box still selected, click the lightning bolt to open the Interactions panel. When you click and hold on the Triggers a list of possible triggers appears and one of them is Click(Tap).

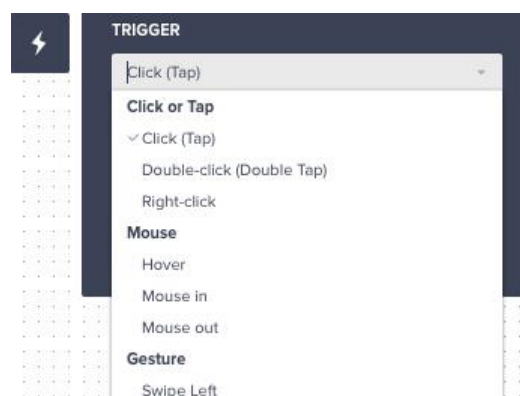


Figure 6: UXPin has a list of possible motion triggers.

The next step is to open the Actions pop down and choose from the Actions in the list.

In this case I selected Move To.

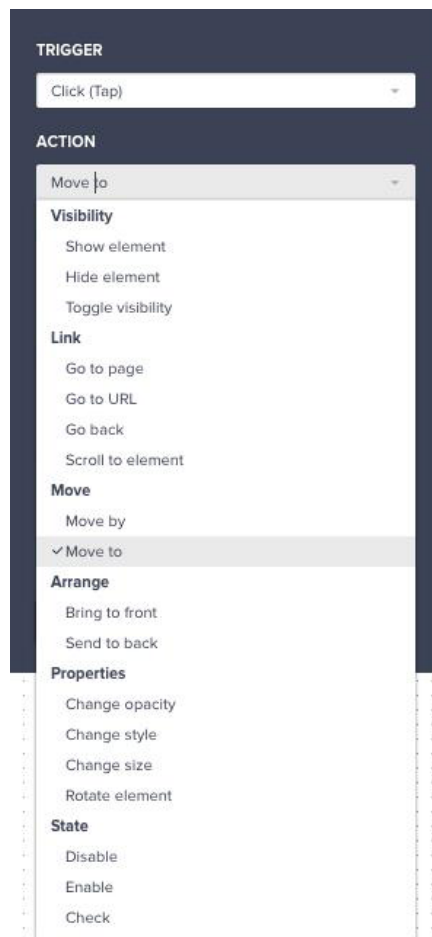


Figure 7: The list of UXPin Actions that are tied to the trigger.

Having selected the Action, you next identify the element to put into motion and also moved to “where”, which, in this case, is downwards by 200 pixels on the Y axis.

The animation area also contains a number of easing choices. Not only do you get to choose how the selection eases into position, but you also get to set how long the object takes to move from “here” to

“there”. This is measured in milliseconds but don’t get hooked on precision. Timing is dependent upon a number of factors including network speed. The final choice is to decide if there is to be a delay between the Action and the motion.

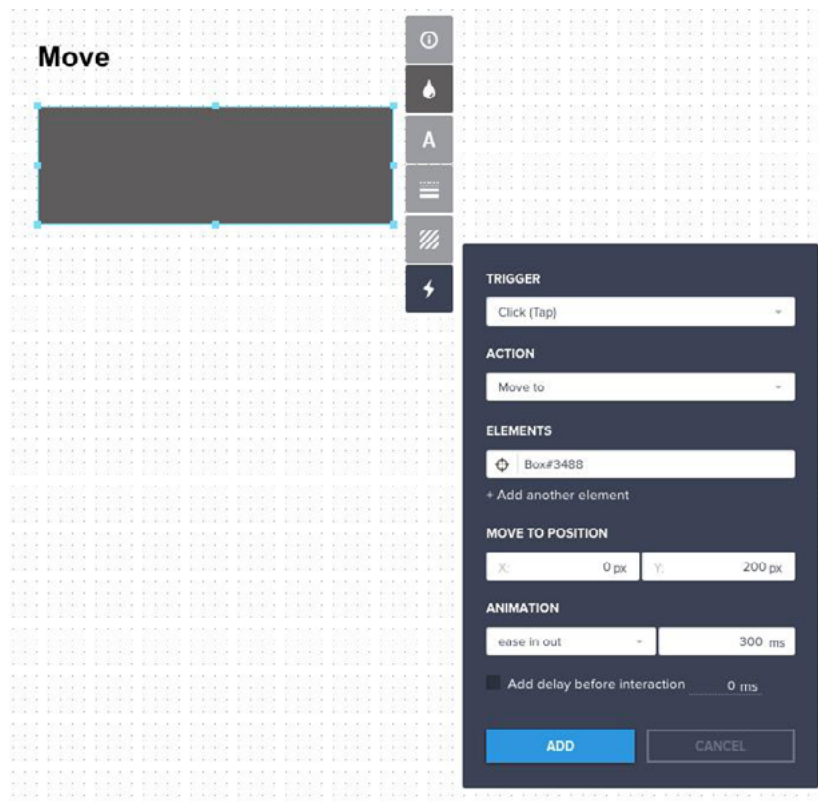


Figure 8: Adding motion to an object in UXPin.

With the motion in place you can test it by clicking the Simulate Design (which previews your project and allows for comments).

If the timing or final position needs tweaking, simply click the Edit Document button to return to the interface. Your object will have an Interactions icon in the upper left corner. When you select the object and click on Interactions, the motion, shown in Figure 9, will appear in the Interactions panel. Roll over it and the object is highlighted. Click it to open the motion properties, which can be changed or deleted.

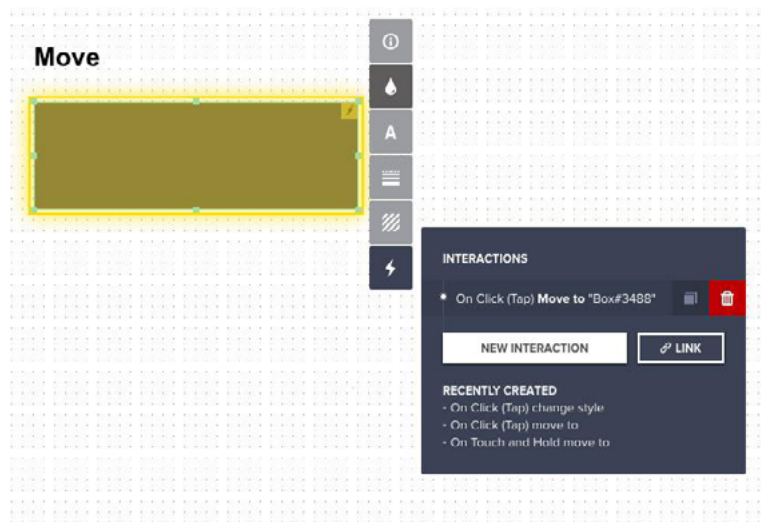


Figure 9: Click on an Interaction to edit or delete the Interaction.

Now, if you wanted to go a step further, you could animate your wireframe. We usually don't see this done too often since wireframes should be fairly lightweight with just enough motions and interactions to prevent people from relying solely on imagination.

However, if you know your site or app requires complex animations, you could certainly explore the simpler animations earlier in the process.

Custom Animations in UXPin

In this demo, you'll learn to make an interactive iPhone wireframe with working, animated icons. The end result will feature a button that, when clicked, transforms into a new app view.

Feel free to follow along with the [live preview](#) of this project.

While this tutorial assumes you have some [UXPin](#) experience, we'll cover common concepts including:

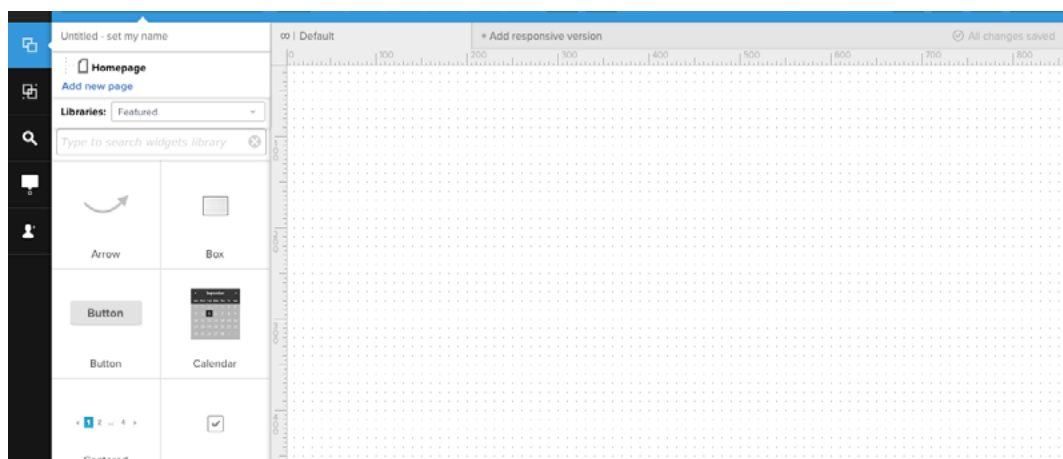
- Creating elements
- Ruler guides
- Corner radius
- Colors
- Alignment
- Groups
- Advanced animation

If you haven't already, go ahead and [start a free trial](#), then follow along below. Feel free to click into the [live project preview](#) so you can see how each step unfolds.

Steps

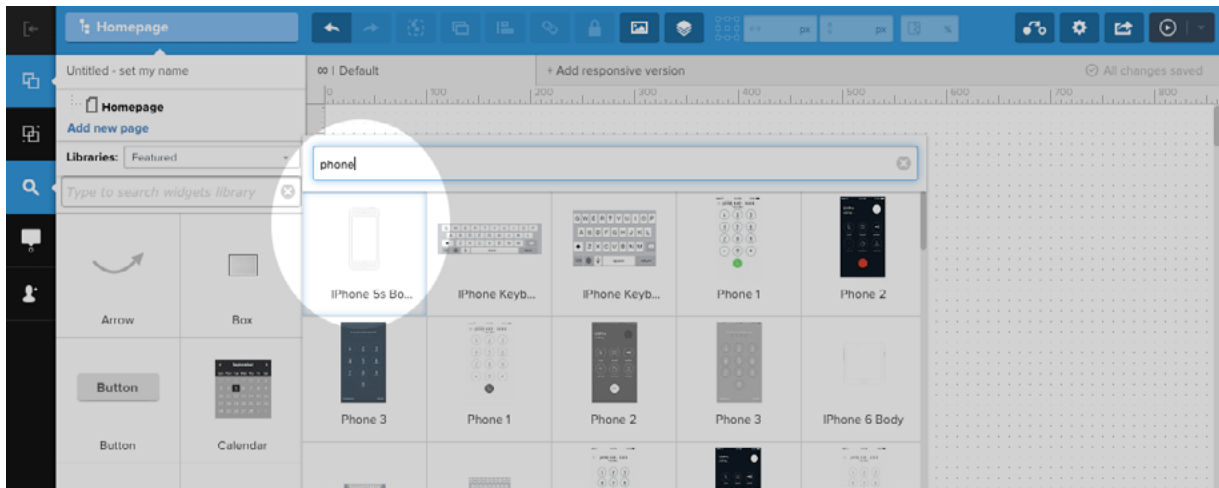
1. Create a New Wireframe

Make a new wireframe like you did in the previous examples.



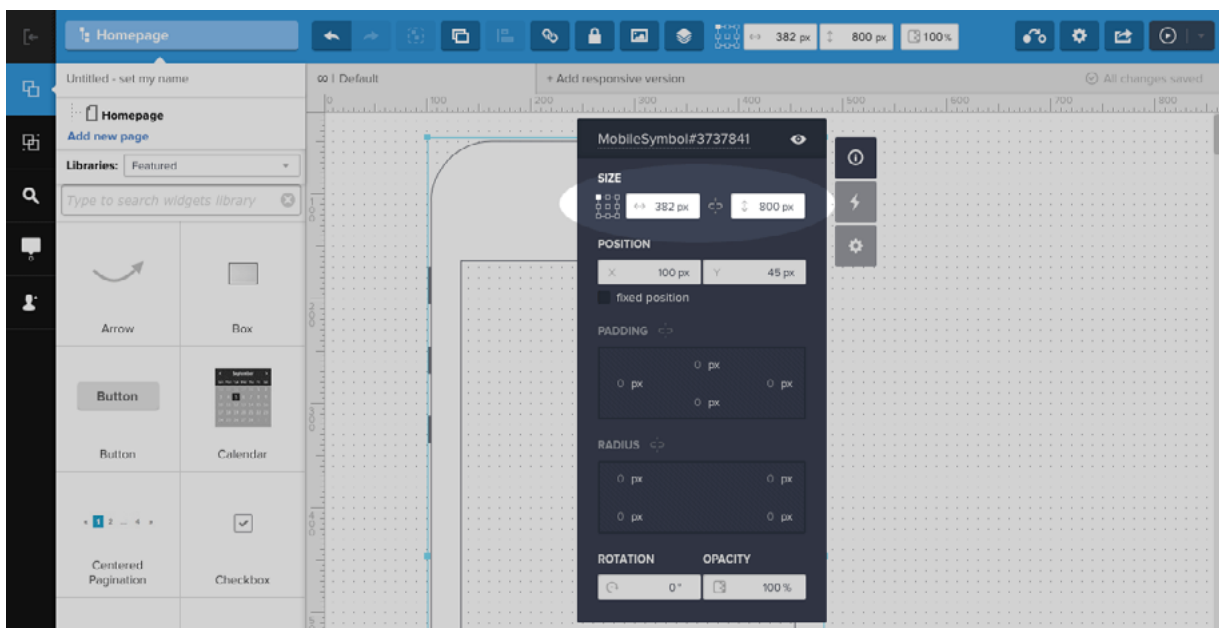
2. Add a Phone Element

You can scroll through UXPin's library of elements – or type cmd-F (Mac) ctrl-F (Windows) to bring up the “find an element” search field. Find and click the phone body.



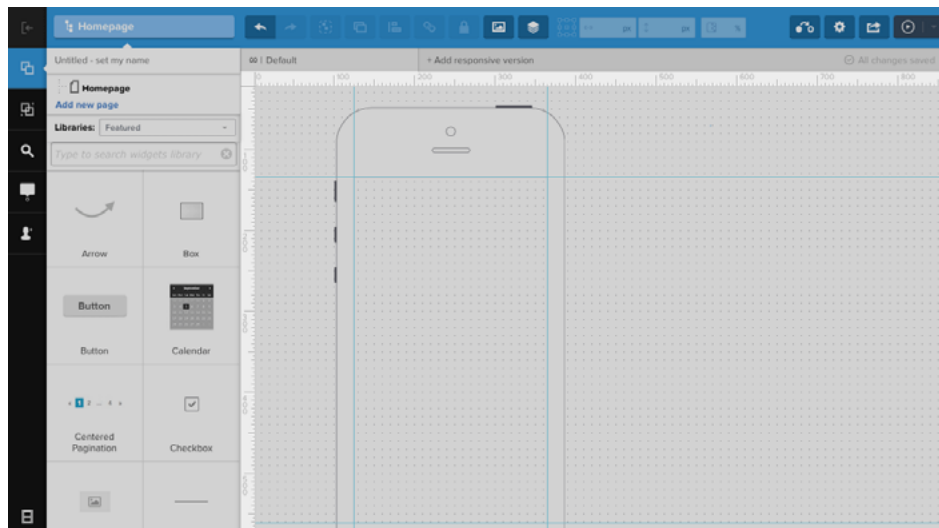
3. Resize the Phone

The phone is pretty big by default, and may not be fully visible on your screen. If so, tap the “properties” icon (circle-i) to enter exact measurements based on whatever fits your screen.



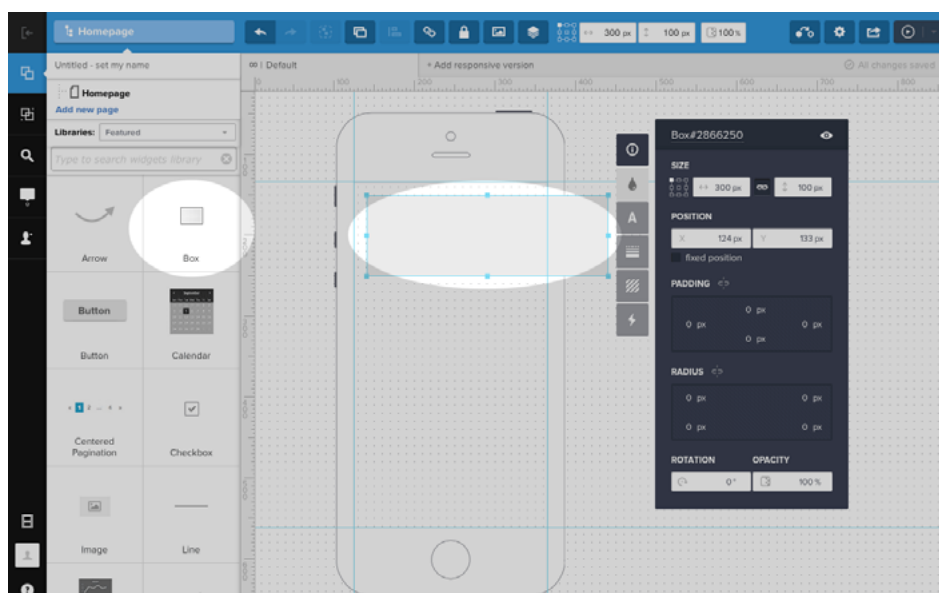
4. Define a Consistent Working Area

Drag guides out from the rulers to the edges of the phone's screen. We'll use these to help us resize the button's background exactly where we want it.

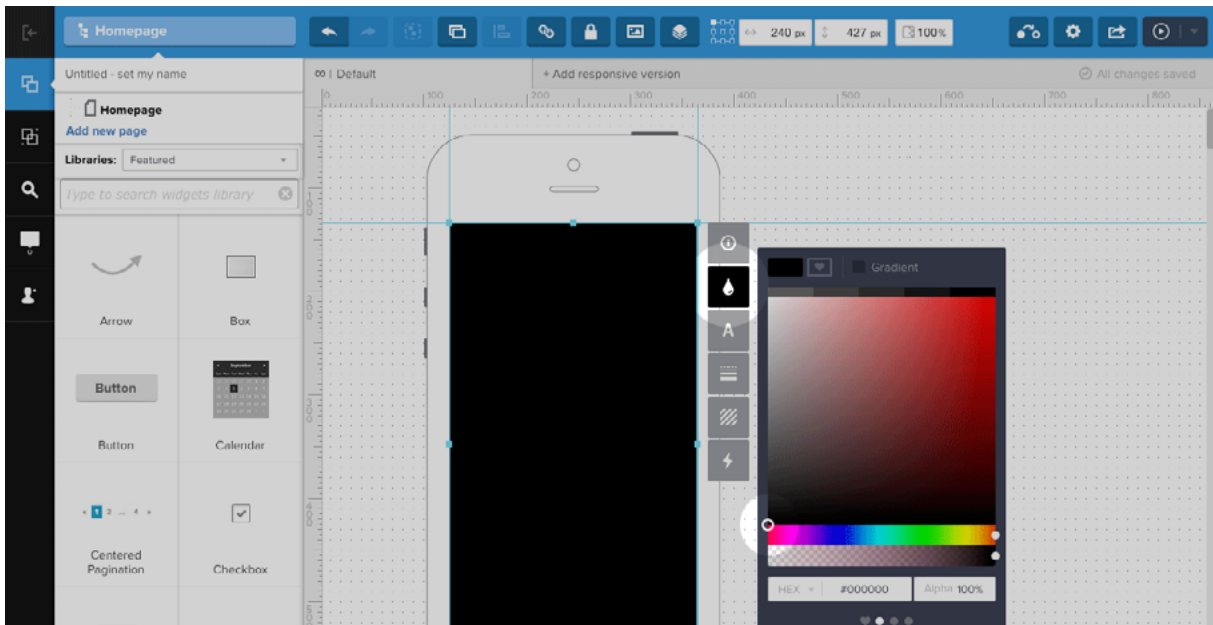


5. Give the Phone a Background

The phone element's "screen" is transparent by default. We want to make that black, which will contrast well with the animated background to come. Add block element to your wireframe...

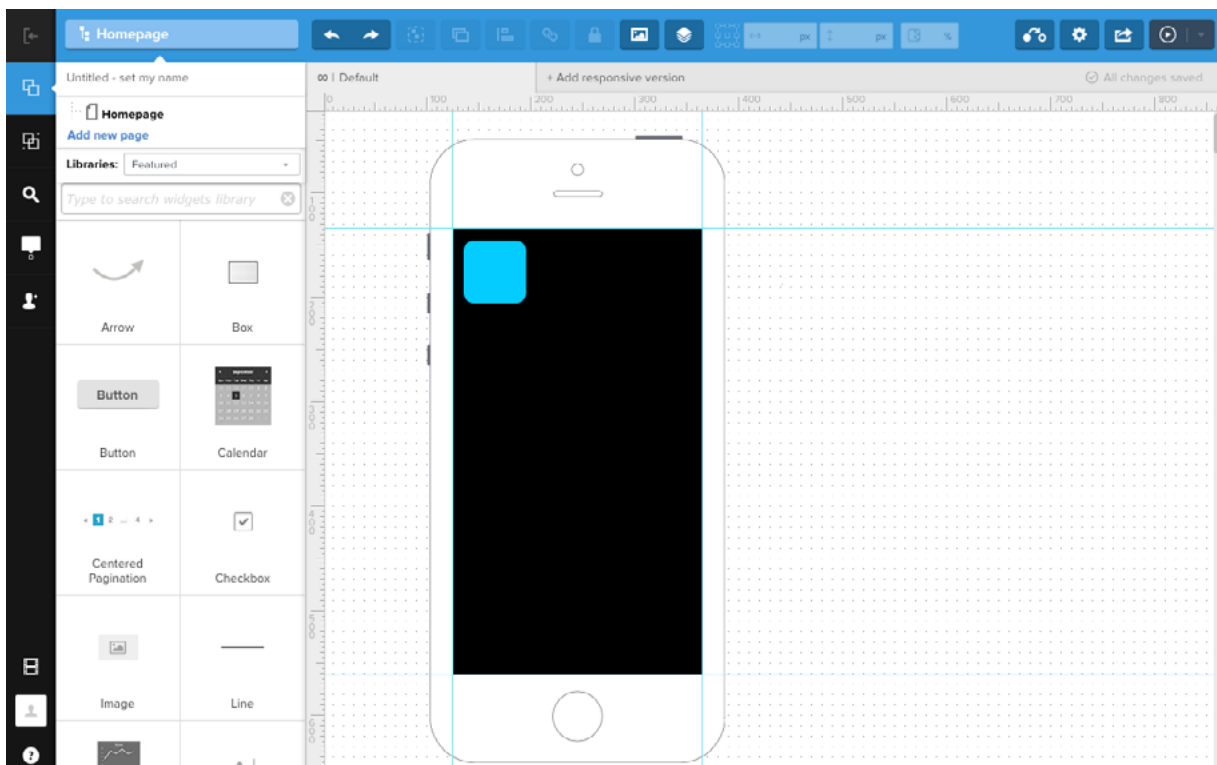


...then resize the box to fit within the guides, and color it black.



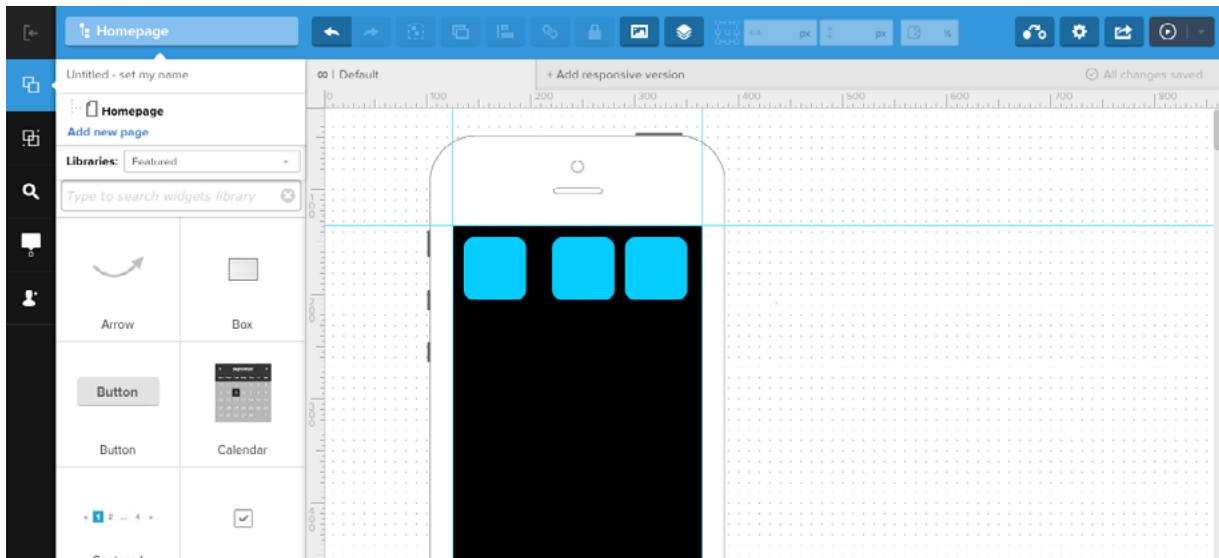
6. Make a Button

Add another box, size to 70×70, corner radius to 10px (in the color popup). Color it light blue.



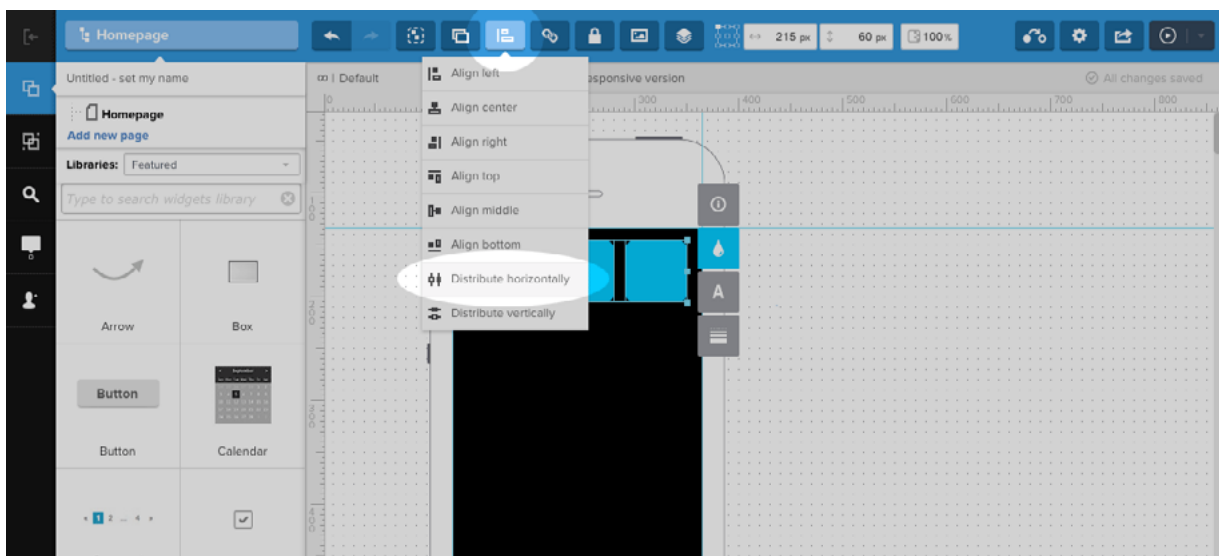
7. Make More Buttons

To duplicate the button, hold down the option/alt key when you drag the icon. Do so three times to make a row of buttons.



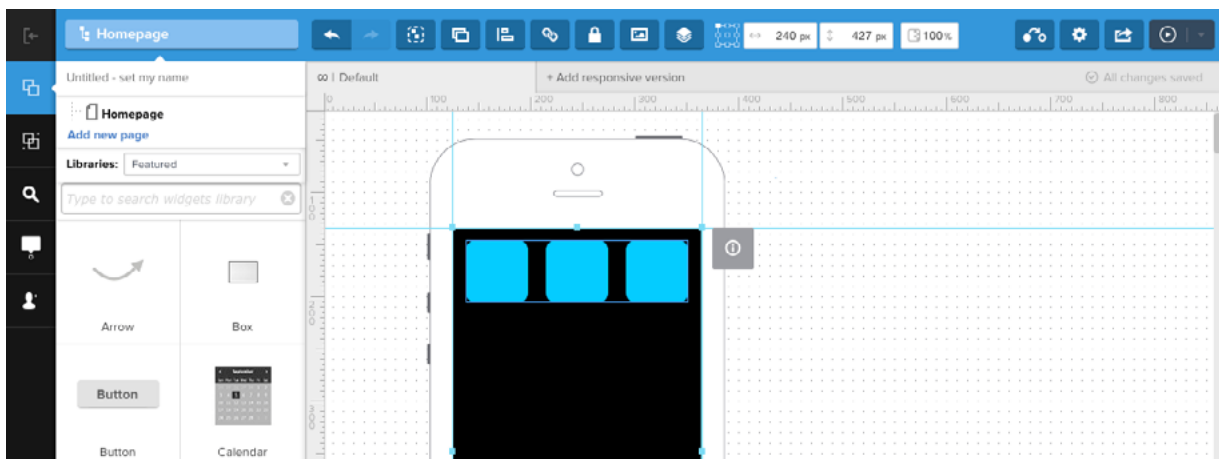
8. Distribute the Icons Horizontally

Getting icons perfectly even isn't easy by eye, so select all three buttons and use UXPin's "distribute horizontally" function, in the alignment toolbar at the top of your screen, to put the same space between all three buttons.



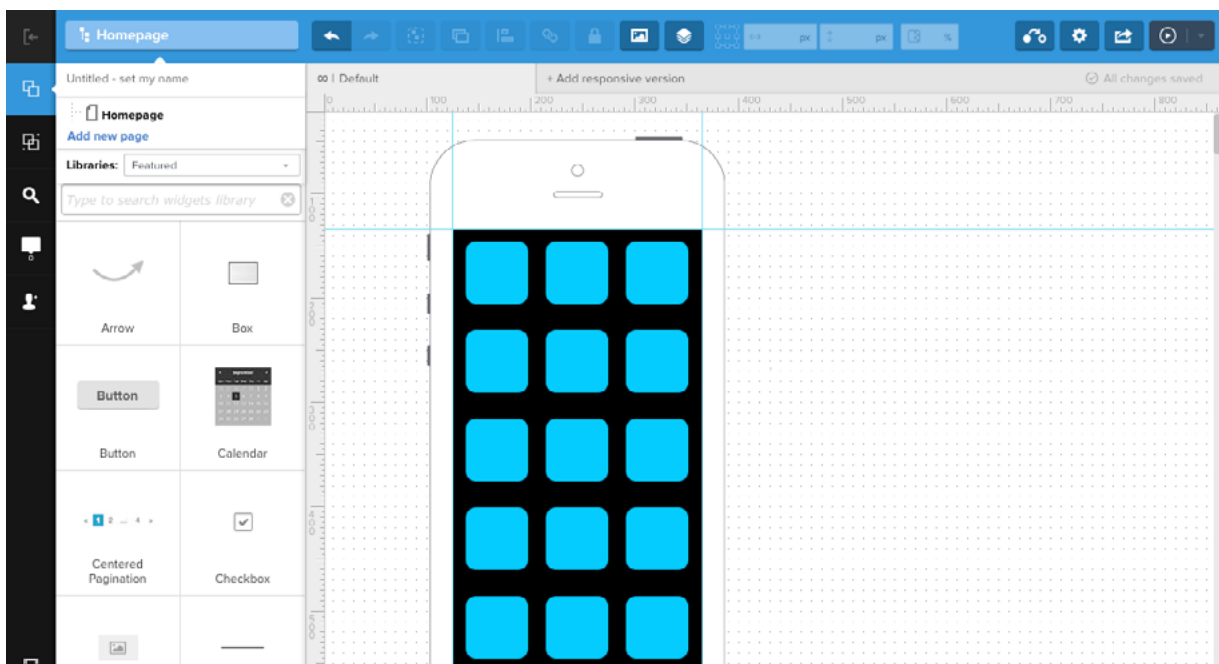
9. Center the Buttons

Let's get picky with our pixels and center the row of buttons in the black screen. To do so, group all three buttons, select the group and the black screen, and finally choose the "horizontal center" function.



10. Make Many Rows

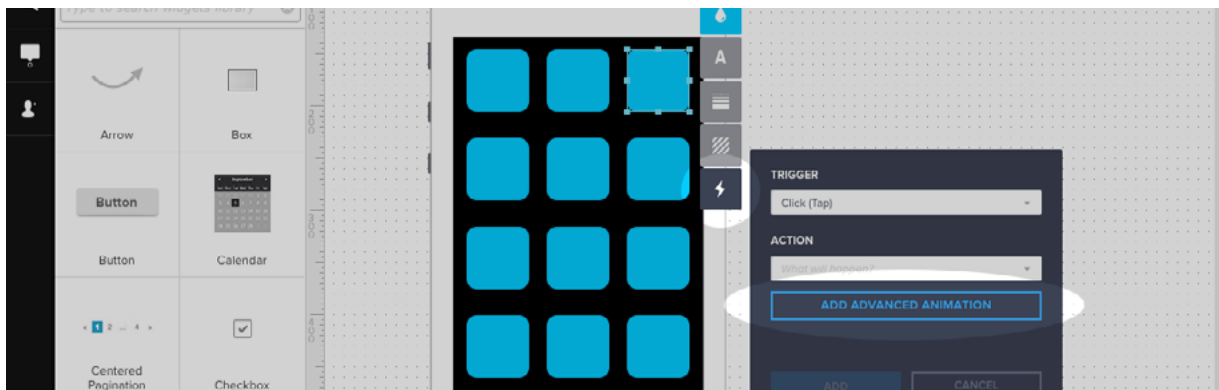
Like option/alt-dragging one button to make a row of three, you can option/alt-drag the group of three to make a series of rows.



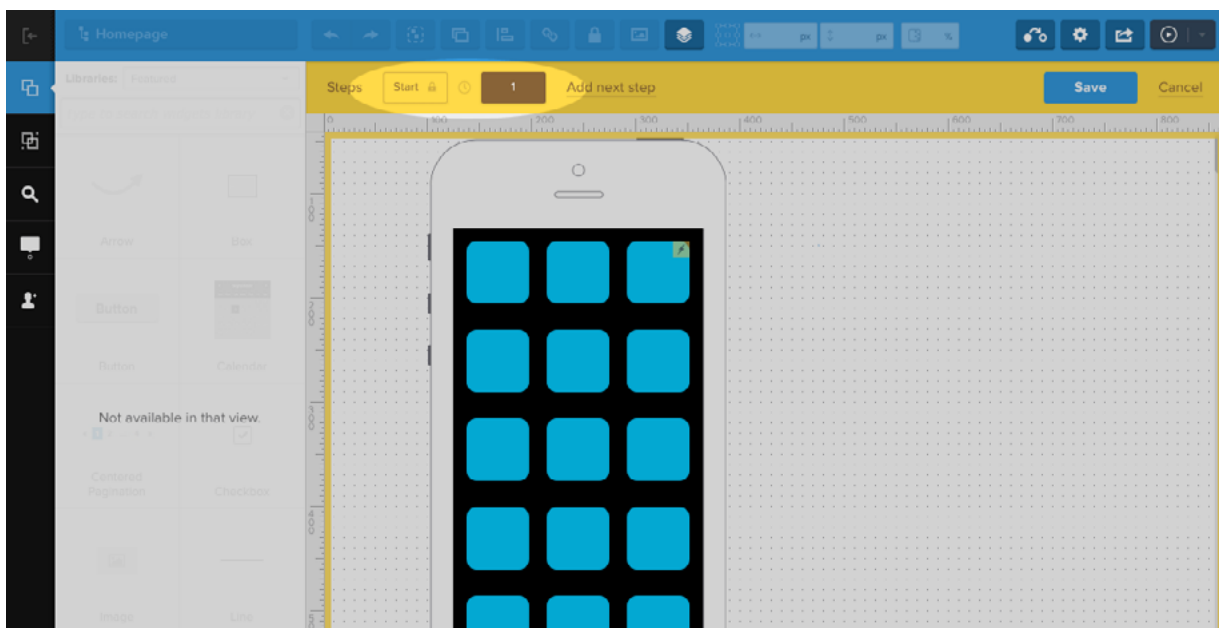
11. Add an Action

In this demo we'll make the upper right button animate. At this point, it helps to follow along with the [live preview](#).

Start by ungrouping the top row and selecting the right-most button. Then tap its “interaction” icon – the button with the lightning bolt. Finally, click “add advanced animation.”

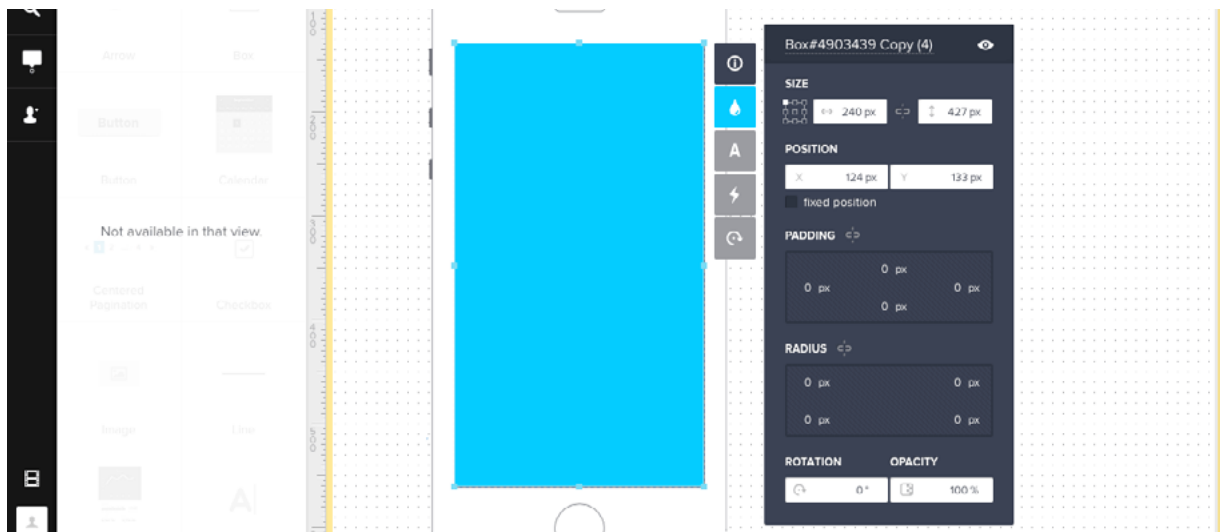


The advanced animation editor comes with a “start” state (what the screen looks like before the user taps a button) and a step one (what happens after the user taps). Make sure you're in step one.



12. Make the Change State

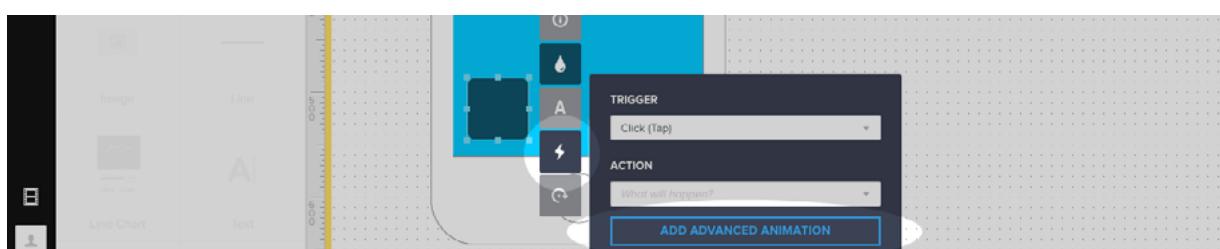
Changes in each step indicate what will happen when users click the button. In this case, the button itself will change its size and shape: remove the button's border radius and resize the button to fill the guides.



13. Make the Close Button

The best way to return to the default state is to make a new advanced animation *within* the button's advanced animation.

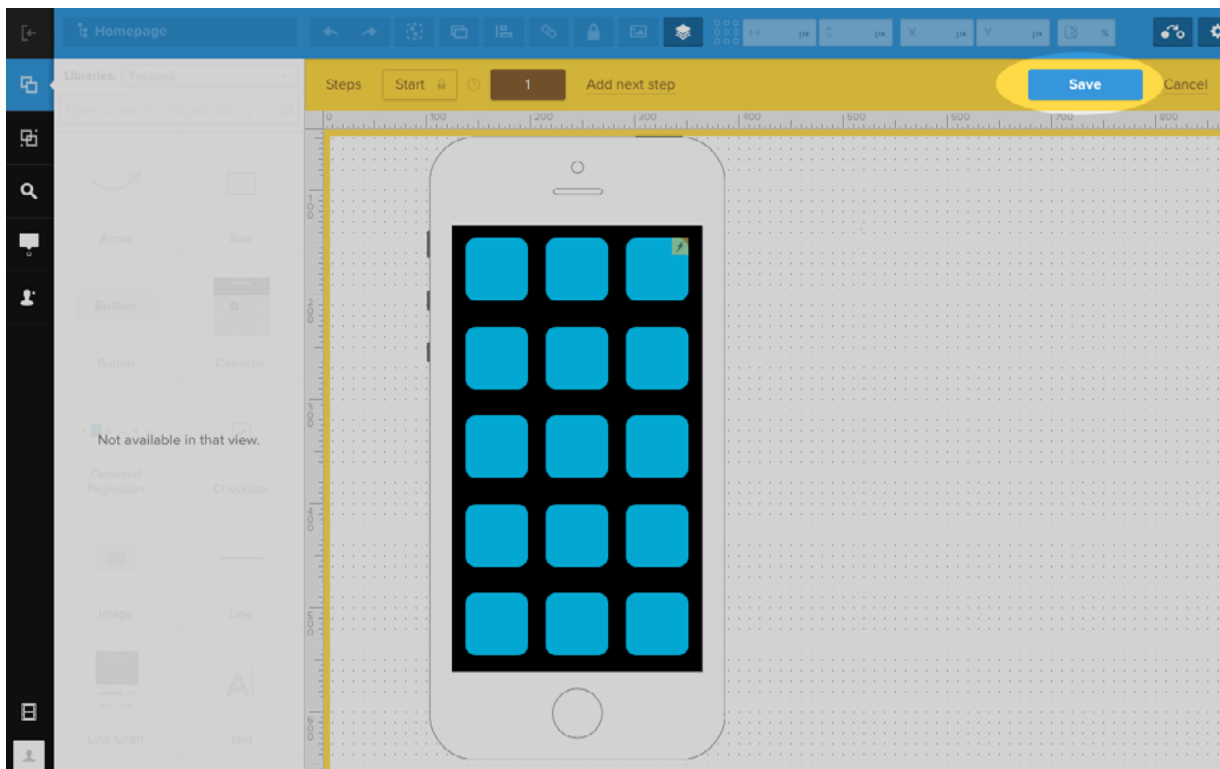
To do that, find the bottom left button – that's tricky because it's the same color as the new background, but if you click in the right spot, its handles will appear. Fix that by making it a darker shade of blue.



14. Give the Close Button an Action

This part's easy.

Click “advanced animation” in the close button’s interaction menu. Then click to step one. Notice how the screen has reverted to its icon-full, black-background state? That’s exactly what we want. Just click save – and that’s it.



In this demonstration, we’ve created an animated iPhone wireframe in which one button expands to fill the screen, and another button resets the demo.

[Try it yourself](#) and you’ll see why animations are cool: they let users see exactly what their actions do.

When you're done and want to share the interactive wireframe with someone on their device, just click the "share" button



on the top, right hand side of your screen and then click the SMS tab. Type in the phone number, and go ahead and send the interactive wireframe.

A screenshot of a 'Share' dialog box. The title is 'Share' with a close button 'x' in the top right. Below the title are five tabs: 'URL', 'E-mail', 'SMS', 'QR Code', and 'Export'. The 'SMS' tab is selected and highlighted with a blue underline. Below the tabs, the text reads 'Send a SMS with a link to this prototype'. There are two checked checkboxes: 'Show comments' and 'Show sitemap'. Below these is a phone number input field containing '+1' in a dropdown and '(555) 555-5555'. To the right of the input field is a blue 'Send SMS' button. At the bottom of the dialog, there is a checkbox labeled 'Require password to view this prototype' which is currently unchecked.

Share x

URL E-mail **SMS** QR Code Export

Send a SMS with a link to this prototype

Show comments
 Show sitemap

+1 (555) 555-5555 Send SMS

Require password to view this prototype

Alternatively, you could also generate a QR code or send the design to the person's email.

Wrapping Up

Wireframing motion is not time consuming but whether to add it to your workflow depends upon the budget, complexity and scope of the project.

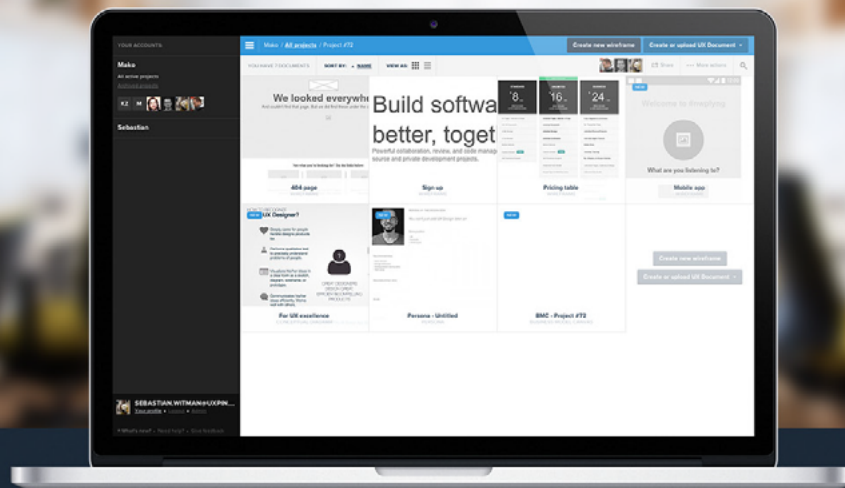
There are, however, a few advantages to adding this technique to your workflow:

- **It is iterative** – You can easily change everything from timing to the very nature of the motion without having to start from scratch. For example, maybe the images should move from left to right or from top to bottom. All you need to do is change the positions in the UXPin Interactions panel rather than starting over.
- **It is visual** – Being able to show motion is a lot more precise than describing motion.
- **It can be refined** – By adding this technique to your workflow you can concentrate on how things move throughout the entire process from Content Reference Wireframe to prototype. As you move through the process and actually start adding content, you can “tweak” the motion established at the start of the process rather than adding it as an afterthought later on.
- **You can play low risk “What If...” games** – The very nature of this technique is it is iterative. By using boxes and circles you can quickly answer such questions as “What if the images bounced into place?” or “What if a button shrinks or expands when it is

tapped?” Once you like what you see you can then ask “What if we made the button larger when it expands?” or “What if the image spins into place when it moves from here to there?”

Finally, this technique clears up any misconceptions around, “Imagine me dropping a ball...”

Create interactive wireframes in UXPin (free trial)



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

UXPin

www.uxpin.com