The Dev Pack

A guide to preparing design mockups for development

func-tion

Frank Maidens & Vivian Hui

Contents

Version 1.0 – January 2015

Intro

What's this guide all about?

Working face-to-face is great, especially when it comes to making awesome websites; designers and developers can collaborate openly and quickly produce really interesting stuff. But for many creative professionals, working independently of developers is the day-to-day reality.

We're a design-focused studio that's teamed up with lots of different developers on web projects over the past 10 years. For us, working remotely is the norm. That means the success of our projects is determined in large part by the method we use to prepare and deliver our designs. Feedback on our projects has generally been awesome, so we figured we'd bottle it here and share our process with you.

Our design production process leads to what we call a 'dev pack'— essentially, a ZIP of files and resources for developers to use when it's time to bring a design to life. These dev packs eliminate the need for developer guesswork during implementation *(something we're told is always appreciated).* And they provide clients with a transparent view of our production design process *(so they know what they're paying for).* Additionally, referencing past dev packs when launching a new project helps us set expectations, define the scope of our involvement, and impress clients when starting a relationship.

Because the quality of our work depends on the successful handoff of a design, we feel it's important to be as clear as possible and to anticipate any questions a developer may have. Clear definition of grid measurements, type styles, and other details can save developers from having to scrape that information together themselves. Cropping images and saving things like SVG icons before handoff saves a heap of time, and generally ensures those resources will appear as they were intended in the live product.

In a nutshell, good dev packs allow for fast, accurate implementation of visual designs, and greatly reduce the back-and-forth time typically required during a build.

We've written this short guide to elaborate our process. Hopefully it will inspire conversations about design delivery and improve the products and relationships of other lovely people working in similar situations to ours.

Who are we writing for?

This is a guide to help visual designers and developers who are working independently of one another without the benefit of sustained contact or the opportunity to collaborate regularly.

We recognize that some designers have the luxury of working in a pod of multidisciplinary talent or have a client that supports an agile approach to web design and implementation. No doubt the agile process is a wonderful thing, but it's recognized in this PDF as a "nice-to-have" in a world populated by remote teams and freelance support.

We have three audiences in mind for the pages that follow:

> **Visual designers** working independently of developers who want to better prepare a design for dev *(beyond sending a PSD file)* and wish to charge accurately for the design production phase of a web design project.
>
> **Web developers** who want to receive more usable design assets from their designers and want a resource to help start the conversation and frame the design deliverable.
>
> **Clients** who want to better understand the production design phase to improve the way their design and dev teams collaborate.

This guide is for: Visual Designers
Whether you're speaking to the client or the dev team, describing what you will deliver via the dev pack is a great way to define the project scope and set expectations.

Illustrating your production process to a client is very helpful. It sheds light on the design phase of the project and gives them a sense of what responsibilities you'll be taking on. Greater transparency here will better define the project scope and facilitate a conversation about the time required and/or value added by your process.

Working independently of developers can be challenging, especially if you've never worked with a particular individual or team before. They don't know your process or what to expect from you. From our experience, we've learned that developers like to know what's coming. Providing a clear description of the deliverables and assets contained in a dev pack will help a new team get acquainted with your delivery method and allow them to facilitate the build in line with the schedule.

You don't need to be able to write code to prepare your assets in a useful way *(though you should have a solid understanding of how web sites are constructed if you're going to design them successfully).* By following the practices outlined in this PDF, you can make the lives of developers more comfortable, create stronger relationships, and build a better final product.

A word of caution: following the recommendations in this guide may cause developers to fall in love with you. So if you don't have the spare time for extra high fives *(or referrals and repeat business)*, it's probably best to stop reading now.

This guide is for: Web Developers

At Studio Function, we've been lucky to work with a wide range of talented developers. We've come to understand that converting design mockups to a live product is one of the most important phases of a successful web design project. A good dev pack is essential during this phase.

A good dev pack allows for fast, accurate implementation of visual designs and greatly reduces the back-and-forth typically required during a build. It does not absolve the designer of their responsibility to have a general knowledge of how the site is built and to design within those means. But a good dev pack can ensure that the designer is more connected to the implementation. A connected designer helps ensure the preservation of design details during the build by providing the required assets. Careful preservation of design details usually spells the difference between good and excellent.

For the developers reading this guide: our appreciation for you cannot be overstated. Our hope is that you can use this PDF to help frame your conversations with designers about what kind of handoff you need from them to do great work.

This guide is for: Clients

Oh, clients. If not for you, we'd all be sitting around wondering what to do with our weekday afternoons and empty bank accounts. The conversation begins with you, and we know you want to understand what's required to get the job done.

After reading this guide you'll have a greater understanding of the production design process, the value of that process and an idea of some of the methods designers use when preparing their work for handoff. It's mutually beneficial if you are able to provide designers with specific requests about how to prepare their designs or initiate conversations about capabilities and delivery methods.

At the very least, you can reference this PDF when speaking with your own developers and have them weigh in on the kind of design delivery required for fast, accurate implementation.

From Concept to Approval

In addition to having strong preferences when it comes to software, most visual designers also have well worn paths within their design process. There are many ways to approach a design problem *(and each client relationship is different)* but working smarter benefits every project and improves the quality of the dev pack.

Understanding the medium

Historically, artists working in oil paint had a deep connection with their medium. They had specific ways to mix pigments and blend oils, and an understanding of resins and varnishes.

Although web design isn't as hands-on *(or toxic)* as oil painting, it still has its own list of technical requirements that every visual designer should understand. By understanding their medium, designers can challenge the conventions of typical web designs.

Designing with dev in mind

A graphic design degree alone is no longer enough preparation to create great work. Every designer should get their hands dirty and pick up some general front-end dev skills so they understand the limitations, possibilities, and depth of the digital medium. Going deeper into how the web works allows for practical designs that are more interesting and innovative.

Visual designers with an understanding of HTML, CSS, and Javascript will have better conversations with developers thanks to a shared technical vocabulary. They can discuss ems and pseudo selectors and avoid abstract feedback like "the type feels too small."

Designing with development in mind doesn't mean you're dependent on straightforward, easy to build layouts. Keeping the dev in mind means approaching visual design from the perspective of a developer and understanding how the end result can be built. It means finding efficiencies in the design process by:

- avoiding almost-the-same-but-different typography *(does anyone really need both a 24px and 25px font size?)*

- using global colour swatches to ensure consistency

- setting up logical responsive grids and baseline grids on separate layers so they're easy to call out and reference in the future

- planning and designing how the variable widths of the site might affect the layout, navigation systems, and content

- incorporating screencaps *(and saving code snippets)* of live widgets in mockups whenever possible *(e.g. custom maps and styled plugins)*

Presenting web design concepts

We consider the presentation of web design concepts a labour of love. Each project and client is different, and each comes with their own set of expectations and goals. When it's time to share the first round of design work, we prefer to show 3–5 divergent concepts that demonstrate key areas of the most important pages/modules. We do this because we're looking to get their feedback on an art direction, rather than approval of specific solutions to pages or sections. Focusing on the most essential parts of the design system at this stage accomplishes two things:

1) it shows the client enough of each concept so that the feeling and effectiveness of those designs can be directly assessed, and

2) it limits the deliverable on our end so we're not tempted to waste time solving all of the site design problems until an initial direction has been established.

The result is an array of concepts that present an exciting range of possibilities. It allows us to table concepts and advance the conversation quickly. We can get approval on initial art direction before investing wholeheartedly in addressing all of the site's design challenges.

Presenting web design concepts

Good
Show flat mockups of each concept.

Share notes in an email about each concept, or have a meeting to review the mockups.

Pros: This approach is easy and cost-effective to execute and allows many different concepts to be explored quickly.

Cons: Site functionality and effects are left to the imagination of the client or communicated through annotations, which can be easily overlooked. The presentation of flat mockups may also be underwhelming.

Awesome
Make an online presentation of flat mockups that show basic functionality (e.g. fixed navs, transitions, scroll effects) that the client can click through.

Meet to present the concepts and share a ZIP of all flat images and scroll demos to illustrate desired functionality.

Pros: This approach is fairly easy to build with basic to intermediate front-end skills and can help the client visualize the concepts in the browser. The added functionality brings each concept to life and makes them feel more 'real.'

Cons: It can be time consuming to slice mockups and build web presentations. Certain effects may be difficult to code and require more advanced technical expertise.

Mind=Blown
Create live prototypes of each concept so client can click around and see the interactivity in addition to how the designs look/feel.

Meet to present the concepts, and share a ZIP of all flat images for reference.

Pros: This approach is extremely detailed and easy to understand, since the client will be reviewing and using the actual site design in the browser.

Cons: Unless each concept is very visually similar, this approach means a lot of wasted code and time. It could be seen as total overkill and a poor use of the client's budget to jump into code of multiple design concepts when only one needs to be developed.

Another Thought
Design mockups vs. design in the browser

Clients want to see multiple concepts. As designers, we want to present multiple concepts *(different visual interpretations of the brief and content strategy)*. We have built some fantastic client relationships thanks largely to our design process. This process requires us to fully explore design approaches from both ends of the spectrum to facilitate collaborative conversations with our clients.

Though we are capable front-end developers, it is by no means our core competency. It's much easier for us to use design software to create mockups of 3 or 4 separate concepts that look and behave nothing alike. This approach saves the time required to code those divergent concepts in the browser. Using design mockups also facilitates the revision process; we find it faster to make large structural changes to a concept in a design program, rather than overhaul HTML and CSS.

But most importantly, we use design programs because they enable us to focus on the design of the piece without being tempted by the forces of "convenient implementation." We have a good understanding of how our designs will ultimately be constructed. That's always in mind. But having the freedom to explore placement, size, and other style options visually, rather than in numbers or lines of code, is a boon to our creative process.

Successful designs need to be simple and meaningful, but they must also be memorable. Memorability is the most often overlooked characteristic of excellent design today. With so many systems of convenience in place, there is little wonder why much of the contemporary web looks the same. The same grid systems and large header 3-column template-like designs blanket most websites in a forgettable haze.

A truly excellent design is memorable. For us, this means consciously departing from conventions in order to elicit a sense of wonder and discovery in the viewer. The design must still be simple, legible, and effective at delivering its messages — but a sense of comfort should wash over the viewer only after they have been greeted with a sense of unexpected delight.

We find it much more difficult to conceive and implement a divergent web design on the fly in the browser than to explore and arrive at such a solution in a layout program.

Wireframes, paper prototypes, or other content strategy docs can provide the designer/developer with an idea of the content that needs to be included in the page design. But without a proper design phase outside the restrictions of the browser, it may be difficult to arrive at a solution that is truly memorable.

Section 1
Important Notes & Demos

Think of this folder of notes and links as a MedicAlert bracelet: it can speak for the designer when they aren't around.

The dev pack can help keep a project moving by giving everyone something to reference when discussing a particular point. It's key that the developer actually reads the notes so nothing is overlooked in the build *(though we still haven't figured out how to make this happen).*

What is it?

This is a folder of important notes, example links, webfont and breakpoint info, code snippets, and suggested plugins that the developer needs in order to build the approved visual design and functionality of the site. It includes any client decisions that may be useful to the developer, like "the approved functionality is similar to *coolexamplesite.com,* except reverse direction and twice as fast."

Why is it included?

Sometimes the designer doesn't have a direct line with the developer and they're working through a project manager on the client side. This folder of notes arms the dev with everything in the designer's head *(and what the client has approved).* Even when designers are able to directly collaborate with the dev team, this folder of notes is still handy to reference as the build progresses.

Sharing important design notes

Good
Annotate the design mockups (could be a commented PDF, a folder of images with notes in the sidebar, uploaded or shared via an online collaboration service).

Pros: This approach is easy, cost-effective to execute, and each note is visually in context with the design.

Cons: Annotations are dangerously easy to overlook or ignore.

Awesome
Make a folder of TXT files, code snippets, plugins and demos with clearly labeled file names (e.g. “Modal Info.txt”, “Webfont license.txt”, or “Dropdown plugin” folder).

Annotate design mockups and include diagrams pointing out specific functionality.

Pros: These notes, links, and code snippets can be collected throughout the design process, and they’re easy to understand and use.

Cons: This approach may be tedious to penetrate and read through. It’s hard to ensure each important note is read.

Mind=Blown
Build a webpage of all notes with reference images, live examples, working plugins, and links consolidated in one place.

Pros: This approach is extremely clear, organized, and skimmable. By seeing working examples of the desired functionality, implementation is more likely to match the designer’s intent.

Cons: A detailed live note file like this takes more time to produce, organize, and annotate. It requires more technical expertise to code working demos.

Another Thought
Notes for your future self

Don’t wait until the end of the design production phase to gather notes. It’s a lot easier to collect them as you work. If you have a lengthy discussion with the client about how some interactive element will work, make a note right after the meeting and include any diagrams about positioning, timing, and example links. It’s better to do this while the thoughts are fresh in your head, rather than 3 weeks later when the design is finally approved. Work smart and collect notes, links, embed codes, JSON custom map settings, and other styled plugins for the developer as you go.

Your future self will thank you.

Section 2
Design Mockups

As we've already mentioned, it's an important part of our visual design process to have static mockups approved before heading into front-end dev.

Flat mockups allow us to quickly present divergent concepts to a client within a visual range defined by the creative brief. We're not limited by a need to code anything or tempted to rely on obvious uses of out of the box CSS frameworks *(which can often make for generic-looking web designs)*.

What is it?

This is a folder of static JPG or PNG design mockups that have been approved by the client, along with mockups demonstrating the design at different widths. In addition to a visual design, these mockups should include detailed information on how the different areas of the site actually work and move.

Why is it included?

Depending on the developer's involvement up to this point, these mockups may be their first exposure to the design. These images are included as a blueprint of what has been approved and what we're all trying to build.

Showing interactivity in design mockups

Websites aren't static. Part of the interaction design process is to consider how things will actually work and move. Designers can't just sell a painting of a website anymore. How a site functions needs to be approved by the client, as those functions are just as important as how a site looks. Developers need to fully understand the intended functionality for each part of the site, so they can help execute the whole approved vision, not just the visual design.

Showing interactivity in design mockups

Good
Annotate mockups with descriptive notes on how areas should function or scroll.

Have conversations with the client using hand motions to describe effects or directional information.

Share other example sites with notes on functionality.

Pros: This approach is easy and cost-effective to execute. No additional technical expertise is required.

Cons: It can be extremely confusing for the developer since it's not specific, and a lot can be lost in translation. This approach also requires the dev to remember and care enough to implement abstract hand motions.

Awesome
Make a "scroll demo" slideshow of flat images with stop motion-like keyframes showing incremental movements (e.g. folder of same sized JPGs previewed through Finder).

Share other example sites with notes on functionality.

Pros: This is our preferred approach as it's fairly easy to build in Illustrator or Photoshop, and can be very detailed, leaving the developer with fewer questions. It is also easy for the client to review before making a decision on a design concept.

Cons: Depending on the complexity of the functionalities, it can be time consuming to save out each slide. Timing and certain transitions are difficult to convey in a manual slideshow.

Mind=Blown
Build a live responsive prototype where you take the time to code specific scroll effects or other interactions.

Create After Effects animation videos of the desired interactions, movements, and timing.

Pros: This approach is very impressive, and easy for both the client and developer to penetrate – nothing is left to the imagination or interpretation of the developer. Timing, transitions, and any additional effects are easy to capture.

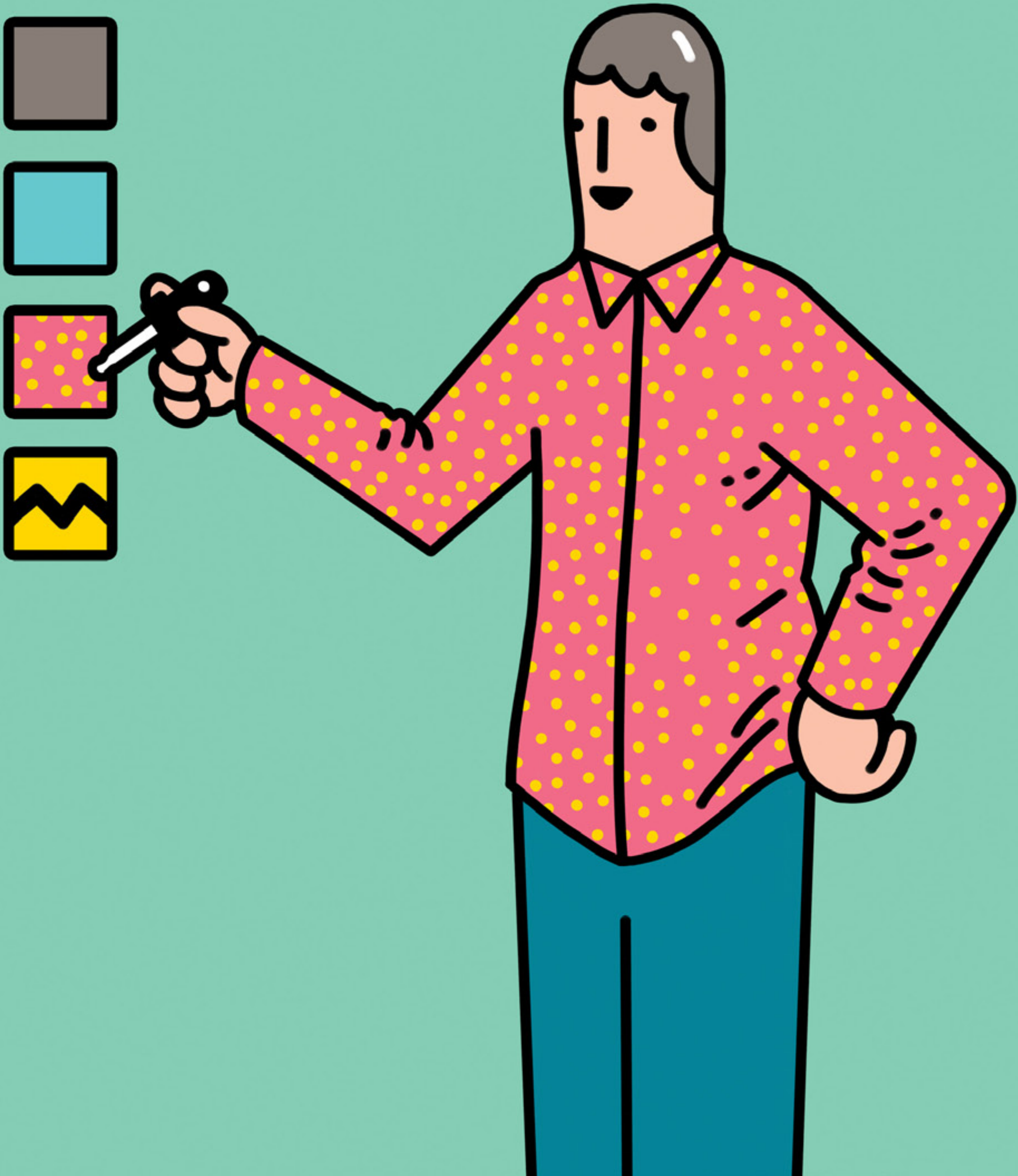
Cons: It can be very time consuming to produce and requires certain technical ability. This approach can also be overkill for easy to grasp functionalities, and may not be the most budget-conscious option.

Another Thought
Static mockups for the responsive web

In an ideal world, talented visual designers would sit hip-to-hip with their equally talented front-end dev counterparts, sharing root beer floats, blissfully collaborating on responsive site designs and working agilely in the browser. When that's not possible, and you have two teams working separately *(sometimes in different time zones),* a good alternative approach is to provide as much detail as possible.

For us, this means including content strategy and design mockups for small, medium, and large breakpoints for any website design project. We do this so the developer doesn't need to guess how the navigation system might change as the site gets narrower or when a 6-column grid gets too skinny and starts to freak out. This elaboration adds additional expense to the design portion of the project, but will ultimately save the client time and money in the development phase and make for a stronger, better thought-out design.

Section 3
Style Guides



Visual designers can't expect web developers to notice or care too deeply about every single nuanced design decision, so it's up to the designer to ensure that their letter spacing, line heights, and specific border radii all make it into the live site as planned.

Short of coding the entire site, there is no better way to do this than to provide the developer with a detailed style guide of all the site elements at work in the approved design.

What is it?

A style guide is a *(static or live)* collection of all site typography, colours, and key graphic elements.

Why is it included?

Style guides illustrate the elements of a site and organize the different type styles. It's easier for a developer to reference a single document than to go digging through an 80mb Photoshop file to find a particular colour or font treatment.

Producing usable style guides

Any style guide is better than no style guide, even if it just means organizing all the type and colour into a separate Photoshop or Illustrator document. The process of pulling together the elements helps the designer clean up any annoying design redundancies *(like almost-the-same-but-different type sizes or colour swatches/opacities that should be consistent, but for some reason got shifted in the artwork).*

Producing usable style guides

Good
Make a separate artwork file with all site typography pulled out onto one page (e.g. TypeStyles.ai).

Make a separate artwork file with all site colours pulled out onto one page, with labeled rgba and hex values (e.g. SiteColours.psd).

Pros: This approach is easy to penetrate, fast and cost-effective to compile, and the developer doesn't need to dig around multiple design files. No additional technical expertise is required.

Cons: There is still the need for design software, leaving room for error in interpretation. Hover states take more effort to show, and this approach also doesn't account for transitions or timing. It is also difficult for the client to review.

Awesome
Build a webpage of coded type styles with classes that make sense, accounting for all typography and colour swatches used in the approved design.

Make a legend of all colour, typography, links, buttons, dividers, and annotate every area of the design mockup.

Pros: This approach is very clear and a great reference that can be reviewed in the browser by the client and developer (no need for design software). Hover states, transitions, spacing, and type rendering are easy to review.

Cons: For larger sites with many styles it can be time consuming to write all the CSS. It can also be challenging showing how the type responds at different widths. This approach requires technical expertise.

Mind=Blown
Create a website style guide and full pattern library.

Demonstrate colour, typography, links, buttons, dividers, lists, icons, forms, tables, tooltips, and additional sample layouts – paired with annotated mockups to correspond with a style.

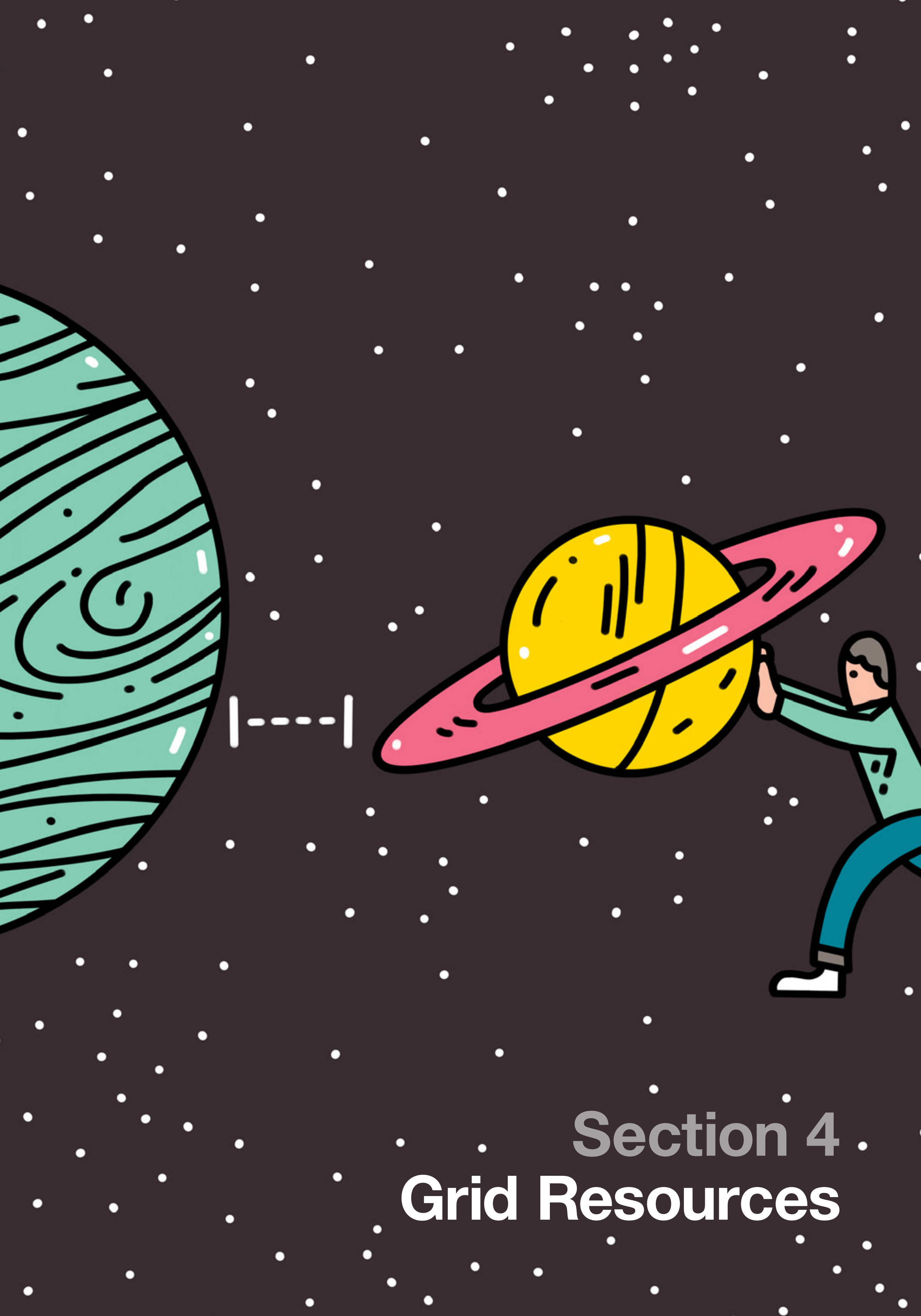
Pros: This approach is extremely detailed and future friendly. It's impressive, and leaves nothing to the imagination, ensuring that future updates will subscribe to a master plan.

Cons: It can be time consuming to produce and may be overkill for smaller sites, with wasted effort planning for elements that will never be used.

Another Thought
Designers: start coding your own CSS

There is so much to be gained by visual designers coding their own styles, since no one is more familiar with a design than its creator. When a designer writes the CSS — either for the dev's reference or for the live site — they can make on the fly decisions based on what's happening in the browser: font smoothing or text rendering options, colour tweaks, or font weight updates to ensure maximum legibility. There are certain CSS declarations that developers love to ignore *(like letter spacing or selection highlight colours)* which can be easily included when a designer composes their own styles.

The best part of writing CSS as a visual designer is watching the site styles come to life before your eyes. It gives you a chance to weigh in on your button styles, hover states and transitions, and detail things like space between paragraphs. You can privately debate how a 3px border radius compares to 4px, and no one will judge you. The goal is to leave no guesswork for the developer and have the staging site come back *(close to)* flawless.



Section 4
Grid Resources

Understanding and wielding the grid is an essential skill for any designer.

This is especially true when it comes to the creation of responsive web designs and dynamic layouts. By striving to fully understand the grid system prior to handoff, designers ensure the choices they make in layout are achievable in the developed product. It's the designer's responsibility to clearly describe the grids used in their work. If possible, they should do so with the vocabulary of the coding language used for its construction.

What is it?

This is a folder of resources that clearly labels the different grid measurements required by all site pages, sections, and modules. Grid resources should be presented visually, and should include percentage breakdowns and pixel measurements for clarity. They should also include all viewport max- and min-widths, all margins, column widths, padding values, and other divisions of space required to preserve the appearance of the approved visual design.

Why is it included?

Designers working without a complete knowledge of the required grid system are effectively building castles on sand. It's their responsibility to know how a particular page or section will be constructed if they expect the final product to appear identical to the mockup. Including a clearly defined grid resource in the dev pack should eliminate any structural guesswork on the part of the developer.

Clearly conveying your grid

Good
Clean up the working artwork files so rulers and columns are clear.

Take screencaps of each page mockup with the rulers showing so the developer can manually measure each column.

Pros: This approach is easy to execute and gets the point across. It allows the developer freedom to build the grid in their preferred manner.

Cons: There is a lot of work required by the dev to decipher and measure out the grid, meaning there may be more alignment issues to fix. Sweeping structural problems may be difficult to address later in the build.

Awesome
Overlay your grids on each static mockup, label the percentage widths and pixel measurements.

Provide a folder of annotated mockups that have grids overlaid in addition to detailed notes about the padding, widths, max-widths, baseline grid, and breakpoints.

Pros: This is an efficient, detailed approach that is clear and easy to produce and eliminates a lot of guessing by the dev, since less manual measurement is required.

Cons: Since this is still a static approach, it can be difficult to convey responsive grids and important information can be left to interpretation. Depending on the size of the site and number of different column structures used, this can also be time consuming to pull together.

Mind=Blown
Code a responsive grid framework for all of the various site grids at work to include with your style guide.

Provide a folder of annotated mockups that have grids overlaid as a legend to show which grid is in use.

Pros: This leaves nothing to the imagination and the developer and client can see all the site grids at work.

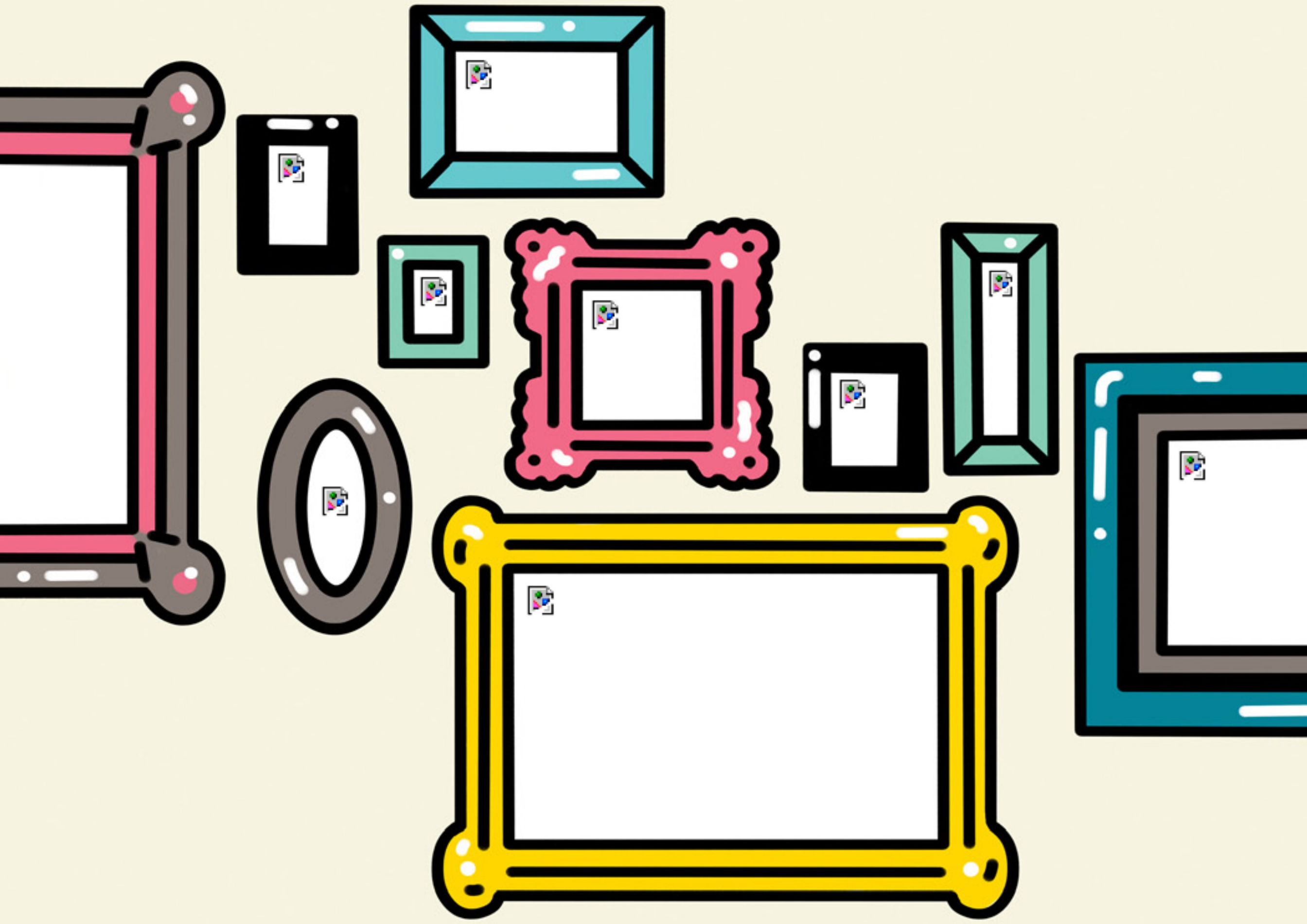
Cons: Coding a responsive grid framework can be time consuming and requires advanced technical knowledge. The developer may have a preferred approach to the build, which could render your code unnecessary.

Another Thought
Get to know the baseline grid

For some reason, most conversations about grids seem to be limited to columns and widths. Perhaps it has something to do with the unpredictable and infinite nature of vertical space on the Internet. As a result, few designers or developers ever consider the opportunities for standardization and visual harmony provided by the baseline grid.

Put simply, a baseline grid divides vertical space with a series of regular intervals. These dividing lines provide a foundation for vertical measurements onto which a designer can place various design elements.

Start by choosing a small whole number like 3px to create a range of useful sizes and measurements *(e.g. 9, 12, 18, 24, 36, 102, 360, ...)*. Use this base number as a key when deciding on elements like type size hierarchies, line heights, image heights, section heights, and margin bottoms. The result may not be immediately obvious to the average viewer. But visually and programmatically, baseline grids can make designs sing. They also provide developers with another means to ensure the fundamentals of a design are preserved in implementation.



Section 5
Images & Icons

The whole idea behind the dev pack is to make the developer's job easier.

A designer can save out visual assets *(e.g. PNG, JPG, SVG, PSD templates)* fairly quickly, and it saves everyone time if these are included in the pack, even just for use as placeholder content.

What is it?

This is a folder of background images and patterns, cropped photos, and Photoshop templates for any images with built-in overlay effects. They should be provided @2x *(for high pixel-density displays)* unless otherwise requested. SVG icons, favicons, and loaders are also included.

Why is it included?

These are included to help the developer work more efficiently, freeing them from the need to dig through source artwork files to find a photo or icon when populating the site. The designer should also be managing the crop of a photo whenever possible in order to maintain consistency in the art direction.

Supplying images & icons

Whenever possible, consider the use of CSS instead of images to achieve desired effects. Shadows, blurs, gradients, and shapes are all easily created with styles. Also, though small, favicons shouldn't be overlooked since they help add brand recognition and polish to a website.

Supplying images & icons

Good
Make a separate artwork file with all site images pulled out onto one page (e.g. Images.psd).

Make a separate artwork file with all icons pulled out onto one page (e.g. Icons.ai).

Design and provide a favicon.ico.

Pros: This approach is fast and easy to compile, and the developer doesn't need to dig around multiple design files. No additional technical expertise is required.

Cons: There is still the need for design software with this approach and it leaves the image and icon handling to the developer.

Awesome
Save out optimized versions of all cropped photos, background images and patterns @2x.

Save out SVG files of each icon at consistent size.

Make PSD templates @2x for any images with built-in overlay effects such as gradients.

Include a folder of Ajax loaders and favicons generated at various sizes.

Pros: By providing cropped images @2x and separate SVG icons, the developer is able to use them directly in the build. They can also construct any icon fonts to their preferred specifications.

Cons: Save for Web optimization from Photoshop still creates somewhat bloated image files.

Mind=Blown
Prepare all cropped photos, background images and patterns @1x, @1.5x, @2x. Save for Web and run additional image compression for faster load times.

Save out SVG files of each site icon and generate an icon font file.

Make PSD templates @2x for any images with built-in overlay effects.

Include a folder of favicons generated at all sizes, social media tag images, and Ajax loaders.

Include CSS code snippets for styled elements and gradients.

Pros: This approach provides optimized, lightweight images and icons for the developer.

Cons: So many different types of assets can be time consuming to produce.

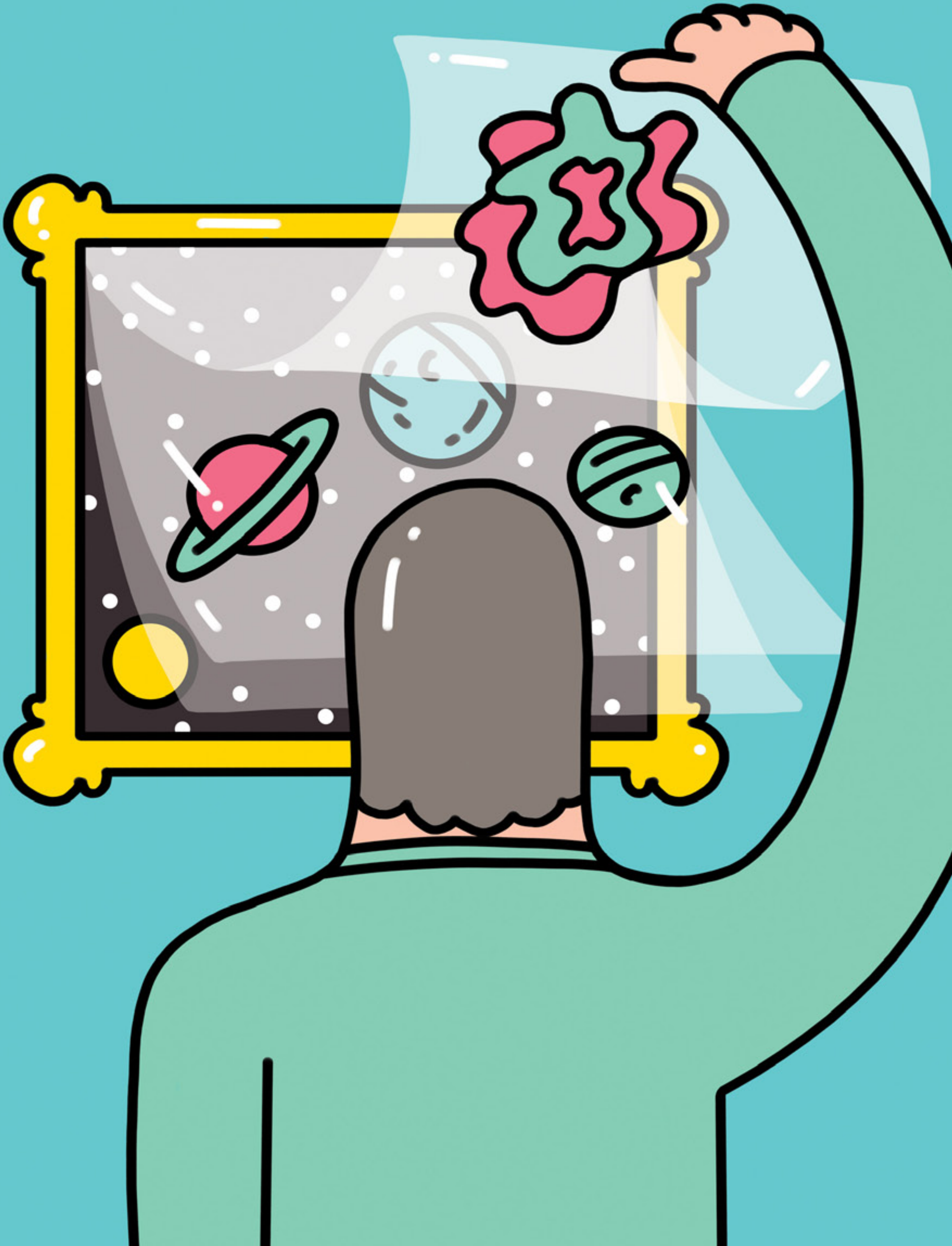
Another Thought
The case for custom icons

Icons are little workhorses that can efficiently communicate a message without the use of words. They are powerful visual metaphors that connect a symbol with its related object or idea, allowing big ideas to be communicated in small graphics.

A custom icon set guarantees that each and every icon needed for a build is available *(even that one for midnight camel rides).* And the feeling created by a well-crafted icon set that suits the creative voice of the design collateral is invaluable to the overall brand experience.

A custom icon set is an opportunity to extend the graphic language of a brand while delighting viewers in unexpected ways. Like a bespoke suit, they are tailored to fit the project and are completely unique.

Section 6
Source Artwork



Let’s face it: developers are used to receiving janky Photoshop mockups from visual designers who don’t know any better.

Devs have gotten really great at deciphering multiple *Layer 13 copy 9s* and filling in the gaps. A good dev pack will transform the deliverable at the end of the web design phase from a painting of a website to a collection of super useful things that empower the dev to work faster and smarter. If a developer can avoid loading Photoshop or Illustrator, then the dev pack has done its job.

What is it?

This is a folder with all of the source artwork of the approved design layouts *(think PSD, AI, or SKETCH files)*. Photoshop templates of images with built-in overlay effects and unmerged, unoutlined master icon vectors can also be included.

Why is it included?

These files are great for developers to reference and for the client’s internal design team to have for the future. If any element is overlooked in the dev pack, such as an icon or font style, the source artwork is available for the developer to review.

Sharing the source artwork

Though they’re always up for the challenge, most developers despise loading up the designer’s source artwork. Adobe programs are memory sucks and design files can be heavy loads. Whenever possible, trim the fat and clean up the artwork so things are obvious and easy to figure out for beginner/intermediate level Adobe users. *(e.g. in Illustrator, unlock everything on the artboard, so the developer isn't confused about why a certain graphic is 'glued down' and inaccessible).*

Sharing the source artwork

Good
Make a ZIP of your working PSD, AI, or SKETCH files with images embedded.

Don't bother cleaning the layers, rulers, swatches, or artboards.

Don't make layer comps.

Pros: This approach is better suited for small sites, and is easiest for the designer, since zero additional effort is required post design approval. If the developer never needs to open the artwork, then it doesn't matter if it's messy.

Cons: This creates bloated, large file sizes that are hard to penetrate and annoying to work with. It can leave a bad impression with the developer and the client, and any future designers that have to work with your files.

Awesome
Clean the working artwork file: delete extraneous layers and artboard objects, link cropped images from Images & Icons folder, organize rulers, organize global swatches and symbols, layer comps, name each layer/group properly.

Break out each page mockup into a separate file for a faster load (e.g. Home.ai, About.ai).

Include any fonts used in the design (depending on the font license).

Pros: This approach is easy to produce and understand. It leaves a better overall impression.

Cons: File cleanup and organization can be time consuming depending on the size of the site. This may all be unnecessary work if the developer never needs to open the source artwork.

Mind=Blown
Clean the working artwork file: delete extraneous layers and artboard objects, link cropped images from Images & Icons folder, organize rulers, organize global swatches and symbols, layer comps, name each layer/group properly.

Break out each page mockup into a separate file for a faster load (e.g. Home.ai, About.ai).

Include any fonts used in the design (depending on the font license).

Export all pages to PSD, AI, and SKETCH file formats.

Pros: By providing PSD, AI, and SKETCH file formats, the developer (and any future designers) can open up the artwork in the software of their choice.

Cons: This approach is extremely time consuming to produce and may be unnecessary work. It may seem like overkill to the client depending on the project size.

Another Thought
Photoshop vs. Illustrator vs. Sketch

You’ve read all the blog posts around this conversation already, so we won’t bore you with feature comparisons. This is just a ranty note to profess our undying love for Adobe Illustrator. We have been using Illustrator since 1999 *(yeesh, we’re old)* and it is the still the most powerful tool out there for vector-based layouts. In our opinion, using Photoshop for web design is confusing. It feels like a broken system, especially with the move towards SVG on the web. Though Sketch is a definite step up from Photoshop and has some nifty web-focused features, it just seems like a heavily watered down version of Illustrator.

Maybe we’re missing something? Tweet @studiofunction

The Dev Review

The final stretch before any launch should include a design review of the developed staging site to catch any visual issues or functionality bugs. This careful sweep ensures that the approved visual design was properly translated to code. If the dev pack has done its job, then this review will be a painless affair.

The dev review is an important step for the designer to be involved in since they're the ones equipped to spot exacting details in size and alignment.

Providing feedback on the coded site

Good
Have a meeting or a call to discuss general site revisions.

Send a follow-up email listing the different areas discussed (e.g. “header text on the home page needs to be larger, refer to mockup”).

Pros: This approach doesn’t require any coding knowledge, but still gets the point across. It is easy for any designer to compose feedback by visually comparing a screencap of the site in the browser to the approved mockup.

Cons: Revisions may be overlooked when they are unspecific. They may also be annoying to implement since they require the dev to basically research each fix that is listed.

Awesome
Use dev tools in the browser to provide numerical feedback and CSS declarations to specific styles (e.g. “h1.home – increase font-size from 2em to 2.25em”).

Organize feedback in a shared spreadsheet noting the page, viewport, issue, suggested solution, priority, status, and any additional notes, reference images, or assets required to complete the fix.

Pros: This systematic approach provides the developer with a clear-cut list of areas to address. By providing CSS fixes, nothing is lost in translation and there is no guessing required.

Cons: Depending on the site complexity and number of revisions, this spreadsheet can be very time consuming to build, with the danger of suggested fixes interfering with other styles. It also requires technical expertise to manage.

Mind=Blown
Fork the site CSS and edit all the styles as required.

Provide a spreadsheet with additional bugs that require dev attention to resolve, noting the page, viewport, issue/bug, priority, status, and any additional notes, reference images or assets required to complete the fix.

Pros: This approach takes the time that would’ve been spent building a list of fixes into tangible results on the staging site, and frees the developer to focus on bigger bugs and issues.

Cons: A level of trust needs to be established before allowing a designer to push code live. It also requires more advanced technical expertise to manage.

One Last Thing

Thank you for reading! We hope you found something use-
ful that you can carry into your own process. If you found this guide valuable, please share it with a colleague. We made this resource pay what you want so more web professionals have a chance to read it and incorporate some of the ideas. If you feel this guide is valuable to your process or business, consider making a contribution at *studiofunction.com/dev-pack-guide/*

About Studio Function
Our studio is built on three key ideas. We want the freedom to ask the right questions. We want to produce meaningful, func-
tional, beautiful designs. And we want to partner with clients that are solving important problems.

Work with us
Why should clients consider us for partnership? Because we're designers, first and foremost. We're focused on the problem and we want to understand your unique viewpoint. We want to find solutions using strategy and collaboration. We also want to make things that look as good as they work. For us, those two attributes are inseparable.

Our studio was founded in 2011. It was the best decision of our lives *(so far)*. We love what we do and want to share that with you.

Learn to code at HackerYou

For visual designers who want to improve their front-end dev skills, we highly recommend HackerYou's hands-on approach to web development. They offer part- and full-time web development courses all year round. Visit *hackeryou.com* for more info.

Special thanks

All illustrations in this PDF are by Sam Island. Check out more of his work at *samislandart.com.* Copy editing and general life advice provided by the talents of Beth Martin and Caleb Sylvester.

Colophon

This PDF was created using Adobe InDesign. The text is set in *Crimson Text* by U of T engineering student Sebastian Kosch, available via Google Fonts. Headlines and other chunky bits are set in *Nimbus Sans Novus* by URW++.

Here's some of the feedback our dev packs have gotten lately:

Christina Truong **@christinatruong**
Kudos for making your instructions & organization of design assets so easy to follow for the redesign! I wish all designers were like this!

Wes Bos **@wesbos**
Having @studiofunction deliver site fixes in actual and proper padding and margin pixels makes my life 100x easier

Kevin Walsh **@kev_walsh**
Best part about @studiofunction's dev packs is that I never need to open the master design file. Makes me feel spoiled as a front end dev!

Brenna O'Brien **@brnnbrn**
As usual, the dev pack looks great and will really help us hit the ground running on the new site!

Drew Minns **@drewisthe**
I love @studiofunction because their design packages include style guides built in HTML and CSS. A developers dream designer.

Thom Leigh **@trolleycrash**
@studiofunction's designs demonstrate a deep understanding of how they will be put together in code. Developing them out is a breeze.

Help us improve this guide

The dev pack has been a continual work in progress for us over the last decade. We would love to hear your feedback and recommendations. Tweet @studiofunction or email us at info@studiofunction.com and let's continue the conversation.