Philosophy

Why is this page in the Wiki? Why does this Wiki exist? Why do any of us exist? What is the purpose of the universe, life, and everything?

Those are some examples of topics we will **not** be tackling on this page (except perhaps that first one)! This page in the Wiki will go over the "Design Philosophy" that we follow here at Geodesic Solutions. Some of these are a little newer, and apply starting with **version 7.3.0**: the version that first added the default responsive web design.

There are going to be "special cases" within the software that these do not always apply to cleanly, but the ideas on this page go over our **overall philosophy** about certain design aspects.

Goal - Update requires Zero "Extra Work"

With any design changes in the software (including changes, additions, removals, etc – anything that affects the look of the website), our goal is to make updates require absolutely no extra work for websites that are "close to" the default design. At the same time, we make sure to use as little "HTML markup" and CSS as possible, to leave as small a download footprint as possible. The less HTML and CSS there is, the less needs to be "downloaded" to the browser, which means the faster the page will load.

When we speak of this "goal," we mean that we try to get as close to that mark as possible. With some design changes, that is just not possible[1]; in those situations, where changes to your existing templates may be required, we will make sure to thoroughly document needed changes in the full update instructions.

"Deep" Customization

We encourage anyone that *wants to* and has the budget / know how to pull it off to deeply customize the design of the site, but ONLY if you (or your client) understands what this will require long-term. As mentioned in Goal: Update requires Zero "Extra Work", we try to make updates as seamless as possible when there are design changes. That said, there is no way we can account for every unique design someone has made; by the very nature of such customizations, they just cannot be "planned for." Because of this, the more "deeply" a website is customized, the more work it will be to update the website. In summary, make sure you are aware of what you are getting into before hand, namely:

- The more deeply customized the design is, the more work it will be to update the software[2]. <
- Keep your site up to date with every release. When allowing multiple versions to pass between updates, the more versions are released, the more changes you will have to make at once. It's generally much easier to apply smaller changes more often than, say, trying to apply changes added over an entire year, from 2-3 new feature releases or more, all at once. <
- Apply changes to a test site first. That way, you can make any required design changes on the test site so that your "live" site experiences very little down time. <

Responsive Web Design VS Separate Mobile Design

There are generally two different schools of thought when it comes to making a website "mobile friendly"[3]. Either a website creates a mobile version that is displayed for mobile devices[4], or the website uses Responsive Web Design(RWD), a relatively newer design idea that has become very popular in recent years. There are very good and valid arguments for either option. It is beyond the scope of this article to go into detail about the merits of either one, but you can find much information about both options on the web.

Default Templates Use[5]: Responsive Web Design

Software Capability: Both! In version 7.3.0 and higher, if you so choose, you can set specific template set(s) to only load if being viewed on a mobile device. The default design does not make use of this feature – it uses Responsive Web Design – but it is "capable" of loading a special, mobile-only template set if you wish to go that route on your website.[6]

What that means for you: "out of the box" in version 7.3.0 and up, you will get a mobile friendly (and "future device" friendly) design that uses RWD. You also have the option to create or hire a designer to create a "mobile only" version of the design that is only used when it detects a mobile device. In other words, **the choice is yours!**

Why Responsive Web Design?

First, lets make this clear: you are free to use either option in your own custom design, and there are tools available to use either option. We believe in giving you the choice and ability to make your site work like you want. However, if you "ask us" which is better, we will tell you that Responsive Web Design is the better option[7], and here is why:

New mobile devices, tablets, and other technology that can "browse the web" come out every day. These days, you can browse the web on your favorite game console, tablet, iPhone, Android phone, Blackberry, and even some high-tech kitchen appliances! New ones are coming out every day, so where do you "draw the line?" Anything that has "touch" is a mobile device? Or anything you "hold in your hand?" The world of "browsing devices" is a very complex and growing world. By creating a "desktop" version and a "mobile" version of your website, you are basically "forcing" everyone into one of those two categories: a strategy that will probably mean a lack-luster experience no matter what device you are using. As time moves on, that will turn into a Venn Diagram where most devices are in the "overlap" – things that work partly like a desktop and partly like a mobile device.

Rather than try to "force" devices into one of two or three different categories, the default design uses RWD[8], which allows for responding to each specific device's capabilities and screen size. For example: we might build capabilities for touch devices so things work with a "swipe", to best take advantage of that capability on touch-capable devices, while also building in UI improvements specifically for use on devices with a pointer (mouse) and its ability to trigger "hover" events – an interface that makes full use of the capabilities of the viewing device. Meanwhile, both options would be built "on top" of a rock-solid basic user interface that does not rely specifically on one capability or the other. In other words, start from a basic foundation and "build up" to enhance the experience based on what the device is capable of without being "reliant" on any one capability. See Progressive Enhancement below for more information.

RWD in Layman's Terms

Way back, more than a decade ago now, when CSS 1 was first coming out and started to get used, the big "advantage" was the complete separation of style and "markup". At the "purest form", you would create your HTML markup semantically. For instance, use <h1> tag for something that is a heading level 1, NOT for the purpose of style, but because you want to say "this is a heading level 1". Or you would use <p> tag for a paragraph, etc. Then you use CSS to style those tags, perhaps give all headings a fancy shadow, and perhaps make the entire page use one font, and headings use another. One of the big "demonstrations" of the day, was to show the exact same page, using different CSS files to completely and utterly change the entire look of the entire page. At the HTML level, the HTML was exactly the same, the entire design of the page changed simply by using a different CSS file to style the HTML. And this was back in **CSS 1** days, using a single HTML markup and using CSS to change the look **is nothing new**.

Now enters Responsive Web Design (RWD). RWD is the **next logical step** for CSS: before you could easily change the entire design by "swapping out" CSS files. Now we have something called "media queries", which allow specifying CSS that is used only when the screen size is a certain size. So now instead of swapping out a CSS file, you have different CSS defined in the same CSS file, but entire sections of the CSS are only used based on the screen size. This allows us to do the "totally different design" based entirely on what the screen size is, rather than swapping out the CSS file. That's all there is to RWD, it's just a fancy way to make CSS apply only for certain screen sizes[9)], so that we can change the design and layout to respond to the size of the screen.

RWD Principals We Follow

First, note that these are principals we follow for the default design, and tend to be considered "best practice" in the industry in a lot of cases, but ultimately you do have full control over the design of your site so you can do anything you want on your own site.

Do Not Restrict Functionality

One huge reason for using a RWD based design, is that everyone has the same tools, the same stuff on the page, no matter how they are visiting the site. We have seen some "mobile optimized websites" that actually remove / restrict entire sections of the site because the people behind the site think that visitors on mobile phones don't want to use that stuff. The *theory* is that if you are on a mobile phone, you want to just quickly look something up or quick access to something. The *reality* is that more and more people are using mobile devices for day to day things that they may have used a desktop computer in the past. There are many people that **only** have access to a phone, they have no other means to access your website.

That is why we do not believe in limiting, "dumbing down", or otherwise restricting access to things on a website just because you happen to be visiting on one device or another. We think everyone should be able to do anything they can do on a desktop, they should also be able to do on a mobile phone. You will not find anything in the default design that limits access to mobile or desktop views.

Exceptions that prove the rule

There are a few exceptions in the default design, at least they may appear to be exceptions at first glance. Below we'll mention the different parts:

Sections *hidden* in smaller view: On the home page, you may notice it does not show the full top menu when you first view the site. It also does not show the full list of categories, and a few other sections are hidden when you first view the site. These are actually not entirely hidden, they are just "tucked away" so that you tap on the section title to display the section. This is done for one main reason: on smaller screens, we want to show the "important part" of the page first, and get rid of any "clutter" that visitors typically don't need. But rather than getting rid of or hiding it entirely, it is actually just "collapsed", for instance on the home page to see the list of categories, you would just tap on the **All Categories** title.

Fading Graphics in Search Box Hidden: On the desktop view, you will see images that fade in and out on the right hand side of the fancy search box on the home page. Those images were entirely hidden from the mobile view[10)], as a **design decision**, as those graphics are 100% "cosmetic" to the site. They serve no functional use. They are not images of actual items on the site, they are just basically background images to make the search box look neat in the desktop view and draw your attention to the search.

Having the images display in the mobile view tends to be very "choppy" on most mobile devices. Even on the high end devices that the fade is smooth, in our opinion it looked out of place on a small mobile screen. They are meant partly to draw your attention to the search feature, but on mobile devices, that is basically the main thing on the page before you scroll, there is no need to draw the user's eye to that part of the page.

And most importantly, removing them did not remove any functionality from the mobile view, as mentioned the images are only cosmetic. It speed up the page load time since the mobile did not have to download those images, and in our opinion in the mobile view it looks better without fading images in the background. Had that not been the case, if those fading images were not merely cosmetic, we would have found a way to still show them on the mobile view.

Semantic Standards Compliant HTML

All HTML in the default design, is fully W3C compliant HTML 5. We also make the HTML as "semantic" as possible.

We avoid solutions that require complicated HTML markup that does not make much sense when you look at it.

When it comes to images serving as links, we avoid making those images a background image defined in CSS, instead having the image tag directly in the HTML. There are times when this may still make more sense for optimization reasons, but is avoided for the most part. Doing so misses out on an opportunity to use the alt attribute to let search engines know what the image is for (and humans as well).

Less (HTML) is More

This one goes hand in hand with Semantic Standards Compliant HTML - we try to use as "little" HTML markup as possible, while still having semantic markup. It accomplishes 3 things:

- Easier for **humans** to read and understand the HTML markup.
- In theory, should be easier for browsers to interpret the markup, and therefor speed the page up.
- Less HTML means smaller download, and faster loading page.

This means we avoid complicated HTML markup. If we can find a way to accomplish something in a semantic way, while reducing the HTML, we will do so. One good example of this is the new [11] way that we use to mark up navigation links.

This is one common way to represent navigation links in HTML 5, we do NOT do it this way anymore as of version 7.3:

```
<!-- OLD way -->
<nav>
    <ul>
        <li><a href="#">Link 1</a></li>
        <li><a href="#">Link 2</a></li>
    </ul>
</nav>
```

Note that this is simplified to just the "markup only", CSS classes and such are stripped out.

That way is indeed very semantic, first you know these are navigation links because of the <nav> tag. And you know it is a list of things by the <ul> and <li> tags. But it does use a lot of HTML. We took that, and basically removed the list tags, since when you look at it, with the <nav> tags alone, it should already be pretty obvious that it is a list of navigation links. We felt that also including the list tags were a little redundant, and un-needed since the HTML 5 <nav> tag already acts to "encapsulate" the list of navigation links.

The reduced version that we use in the default design uses this format:

```
<nav>
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
</nav>
```

Progressive Enhancement

The "old way" that developers have used for years was something called **graceful degradation**. What that means is that if some fancy bell or whistle does not work on a device, it has "workarounds" built in so that the basic functionality still "works". For instance, a fancy slider that might not work on an old version of IE might just display the contents in rows instead of using a slider to slide between them. This development / design approach "starts with" all the bells and whistles, then "degrades" into "sort of working" on older technology. And because it starts with all the bells and whistles, it

tends to be bells and whistles based around a very specific set of capabilities, usually for desktops. This is a good practice to follow, especially compared to no "fall back" solutions at all, but it does have it's limitations.

The new kid on the block, so to speak, is **Progressive Enhancement**. The new kid looks at the "old ways" and asks the room in general, "But why?" It flips the old methodology on its head by starting from the basic experience and focusing on that for the main part of the design, then later adding bells and whistles based on what is available! It starts by using a solid base to lay the foundation, then adds up on top of that based on what the device can handle.

The end result is a design built from a solid base that is fully accessible to anyone, on any device! Rather than degrading gracefully (or sometimes not-so-gracefully) to older or different browsers or devices when a particular, fancy bell or whistle isn't available, that capability simply won't be there, but it will **not hinder the user experience**.

Something we stumbled on that sums it up very well is the Modernizr 3 Workflow. One line, in particular, stands out:

*Care about the **quality** of each browser experience, **not** the differences between them.*

RWD CSS Media Query Breakpoints

In the software, we base the media query break-points[12)] on the specific design and contents that are affected rather than the typical device screen sizes. Meaning that the width that CSS "switches" to use alternate CSS is based on what width it starts to "look best" with the change. What this results in, is a layout that looks great and works best for each device, whether or not that device happens to fall "neatly" into the standard screen sizes or not.

In other words, you won't see one huge gigantic "portrait mode" section or "tablet" section in the CSS; instead you will see smaller sections that offer alternate CSS based on the width break-point that works best for whatever is being changed.

This fits in with the concept of progressive enhancement, in that the design elements that need to adjust when the screen size is large enough, happens at just the right size. If you have a device with width of 480px, you'll get a specific layout. If you switch to landscape that gives you a little more horizontal space, some of the layout adjusts to take advantage of any extra pixels. If you have a device that is slightly larger resolution, you might have room for another layout change that "looks best" once the screen gets that big. So the layout / design "progressively enhances", the more pixels your device can fit horizontally, the more "enhancements" it will have to take advantage of that extra screen real estate.

What about Mobile Apps?

What about improvements to the current iPhone app? Is an Android app coming? These are common questions. For now, we are going to "wait and see." If there is still a large "demand" for specific mobile apps after the RWD implementation, we may create them. Generally, though, we would prefer to have a design that makes a mobile app unneeded and redundant. As we mentioned before, there are so many different devices out there (and so many more added every day) that the old philosophy of creating an app for each major type of device is becoming more and more unfeasible.

1)

Especially in situations where there has been a significant change to the overall design, which happens from time to time.

2)

Note that if the changes are made the "right way", it will greatly minimize what needs to be done in each update, but for some design customizations, depending on the changes there is just no getting around needing to do extra work when you update. The best example of this is any time you customize system or module templates in your own custom template set.

3)

Other than creating an app for every device...

4)

A mobile version of the website that is basically a "mini website", a much smaller version that is laid out for the small screen, with a lot of the functionality stripped out as well.

5)

as of version 7.3.0

6)

Note that we do not provide a mobile-only version of the templates. You would need to create that yourself in a custom template set if you wish to use mobile-only templates instead of / in addition to the responsive web design.

7)

As opposed to creating a separate mobile version of a website

8)

As of version 7.3.0

9)

Actually there is a lot more that media queries can do, but the screen size is how it is used for RWD.

10)

And by the way, they were done in such a way so that most mobile browsers will not un-necessarily download the extra graphics when they are not displayed, so it speeds the page up.

11)

As of version 7.3

12)

The point at which something in the CSS changes based on screen width - for instance at what "break point" does it go from a single column layout to a double column layout