
# RESPONSIVE DESIGN

## KEYHOLE SOFTWARE TUTORIAL

WRITTEN BY DAVID PITT

KEYHOLE SOFTWARE

VIEW CODE SAMPLES


# Responsive Design

## A Keyhole Software Tutorial

### Table of Contents




Tel 877-521-7769 https:/keyholesoftware.com


# Responsive Design

## A Keyhole Software Tutorial

This tutorial covers:

- Responsive design in the enterprise
- Mobile first or one-size-fits-all
- How responsive design works
- Responsive design frameworks
- Bootstrap.js, a responsive design framework
- ResponsiveUI layout ideas

Enterprises are feeling the pressure of the need to develop applications that allow users to use their own devices to access enterprise applications. Most devices will have a browser application, just like a desktop device, so current web applications are accessible without doing anything but providing connectivity to the corporate network.

However, odds are that these “built-for-desktop” browser applications will not be fully useable, especially if any kind of data entry is required. Why, are they unusable? Because the screen sizes differ so much, and even though the application will run in a mobile browser, the user will constantly be scrolling and expanding trying to comprehend, navigate, and interact with the application. Not to mention, they are not built for mobile touch screen interfaces as they assume a standard keyboard.

What options do enterprises have to solve this problem? Of course, native, non-browser applications can be created to provide an excellent user experience. However, going down the native mobile application path also introduces deployment, security, and a big learning curve. It requires commitment to application development efforts. If an organization has one homogenous device, then this could be an option. But most enterprises would have to develop capabilities in multiple platforms (for example, Android, iPhone, and Blackberry), which is both expensive and time consuming.

Enterprise SPA offers a solution by employing responsive design for user interface implementation. This tutorial will describe what it is and how a responsive user interface can

be implemented with HTML5 and CSS3.

## What Does Responsive Design Mean?

Simply, the term “responsive design” can mean a lot. But for this tutorial we will take it from the perspective of enterprise application development. Generally speaking, the term means that an application user interface is “responsive” to the accessing device.

If done correctly, when a user accesses an enterprise application, the user experience is as clean and consistent as if the user interface was built for the specific accessing device. This is accomplished not only by CSS3 browser features, but also by making appropriate user interface design decisions.

Screen resolutions vary significantly by device. Here are some common screen sizes for mobile and desktop devices, as shown in figure 1.1.

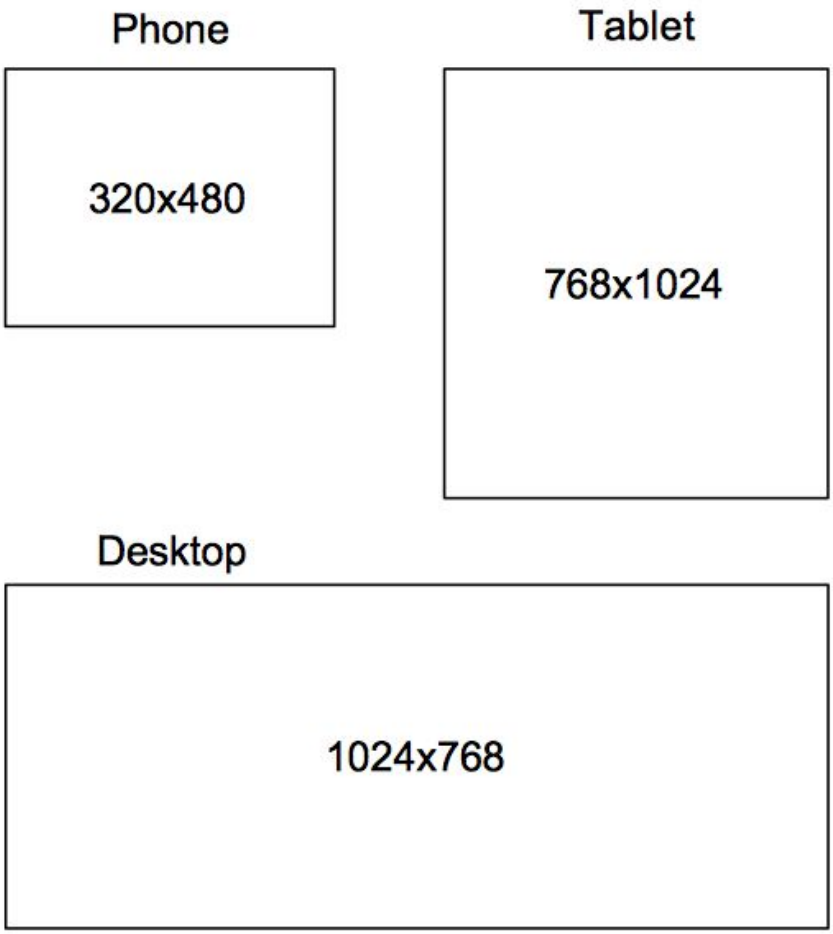


*Figure 1.1 – Common device screen resolutions*

A responsive design will change the user interface layout and elements based upon the specific user interface size. As an example, consider the elements shown below in the standard browser view of the SPA application Command Grok:

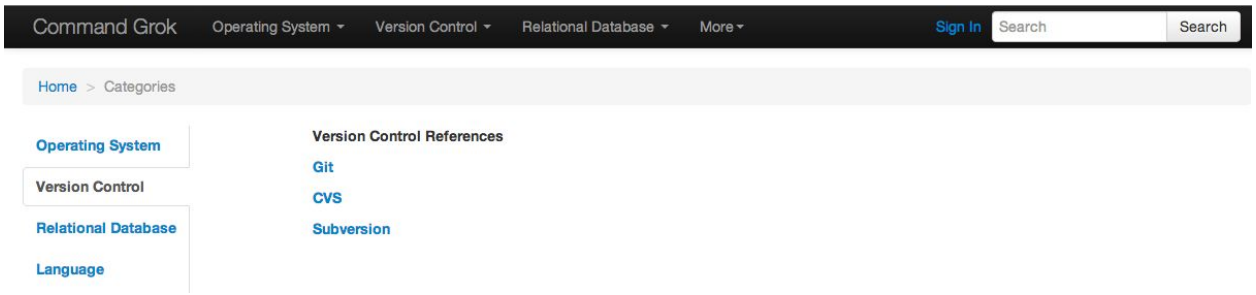


*Figure 1.2 – UI with navigation elements across the top*

Notice how in figure 1.2, the navigation options are displayed vertically across the top of the screen. When the screen dimensions change, the navigation collapses. By contrast, see how the application is viewed via a tablet device as shown in figure 1.3:

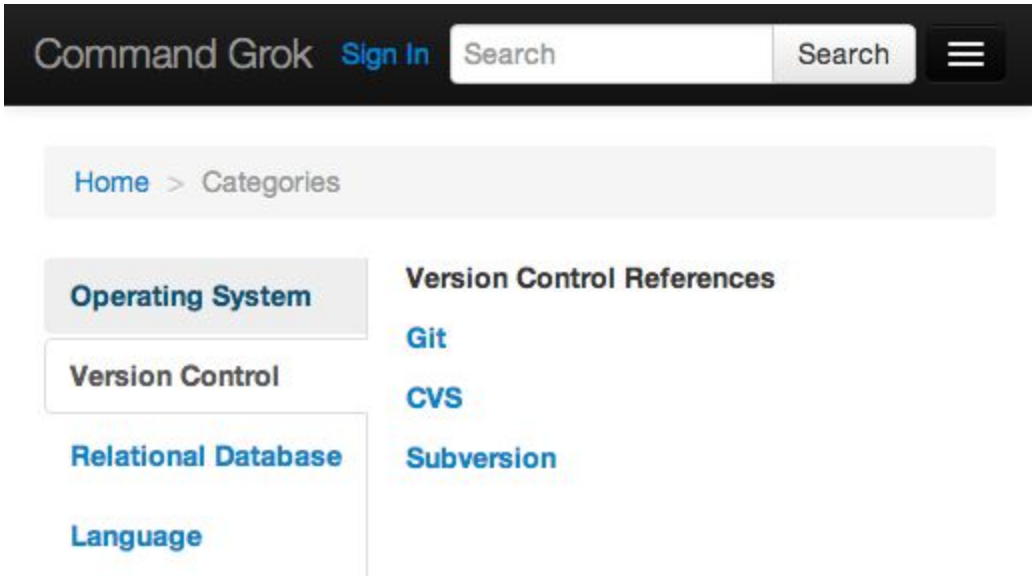


*Figure 1.3 – Navigation options collapse when a smaller resolution is detected.*

This specific responsive feature is accomplished by using a responsive layout. However, more is required that just using a layout in order to make a web-based application responsive to other devices.

# Implementing Responsive Design

Responsive design is applied with a combination of HTML/CSS magic and user interface design that applies interface layouts, widgets, and input patterns that are device-neutral and take advantage of touch capabilities. HTML and CSS provide a nice platform for laying out and partitioning elements to allow a limitless creative user interface experience. Experienced HTML/CSS user interface designers usually apply their craft to more marketing-focused web sites or applications. They usually don't find themselves in the ranks of corporate IT projects. That will likely change.

## Mobile First

One approach to pervasive design is to take a “mobile first” design approach. This means that the user interface is designed towards a targeted mobile device screen resolution and mobile-type widgets first, as then a desktop browser will present the UI without resolution issues. But, let us be realistic. A mobile web interface will display and run on a tablet or desktop, but it will look odd and not take advantage of the extra real estate. Since we are speaking about enterprise applications, it is safe to assume that targeted devices could include tablets, desktops, laptops, and some kind of soon-to-be-invented device.

Starting with a mobile phone user interface is not realistic, but starting with a tablet type device is. Look around your office and take an inventory of who is using some kind of tablet, or even taking notes on a tablet in your next meeting. I believe there is going to be some kind of convergence on surface-based, touch gesture-based devices. Look no further than Windows 8 tiles. So, why not leverage your application user interface now?

## CSS Media Queries

Version three of CSS added media queries. This allows CSS designers to “query” a device's width, height, and orientation parameters. CSS can be applied using conditional media query expressions that will allow CSS properties to be set based upon the devices browser display properties.

Listing 2.1 shows CSS expressions that use the media query capabilities to change the background color depending upon the device screen size.

**Listing 2.1 – CSS expressions with media query**

```
@media screen and (max-width: 600px) { ← Background to purple if width < 600px
        .one {
                background: #F9C;
        }
        span.lt600 {
                display: inline-block;
        }
}
@media screen and (min-width: 900px) { ← Background to orange if > 900px
```

```
        .two {
                background: #F90;
        }
        span.gt900 {
                display: inline-block;
        }
}
@media screen and (min-width: 600px) and (max-width:900px) { ← Background to blue if > 600px and less
        .three {                                                 900px
                background: #9CF;
        }
        span.bt600-900 {
                display: inline-block;
        }
}

@media screen and (max-device-width: 480px) { ← Background to xx if < 480px
        .iphone {
                background: #ccc;
        }
}
```

As you can see, media query expressions provide a way for CSS designers to apply different CSS styling based upon device size with a single CSS implementation. Most current browsers support the media query. Older browsers that do not support media query have to implement and maintain separate CSS files for each device type. Also, there are some frameworks that will read and parse CSS on the fly and transform the CSS attribute based upon a device size. The device size can be determined by the HTTP user agent property.

## Responsive Layout

The media query CSS element itself does not give you a responsive user interface, as it's just a mechanism used to implement CSS that applies a responsive design framework. Most enterprises don't have the wherewithal to write custom responsive CSS for their enterprise applications. Since SPA-based enterprise applications are constructed with JavaScript, HTML, and CSS, some of the responsive design is going to depend upon the UI framework you decide to use.

### Use a CSS Framework

Writing your own responsive UI framework would be a time consuming endeavor. So one decision in selecting your SPA UI framework needs to be how responsive it is. CSS separates both the look and feel and responsive logic from HTML, and therefore markup. However, it is useful to have an introduction to some of the responsive layout frameworks that are available. Some common frameworks will be introduced in a later section.

### <DIV> not <TABLE>

User interfaces can be laid out using HTML <table> and  elements, with  being the

preferred mechanism for performance and flexibility. The reasoning behind that is that more HTML has to be processed for <table> and  has more layout control properties that CSS can manipulate.

### Avoid HTML Pos, Width, Height Type Attributes

No matter what UI framework you select, or how you define your SPA's HTML user interface elements, you should avoid defining position, width, height, and type attributes. Unless you are using a UI framework with responsive design APIs that require usage of these attributes.

## Responsive UI Layout Frameworks

A common approach for CSS frameworks is to provide a responsive layout framework that implements a grid-based CSS theme. A set number of columns are defined for a grid, and then UI HTML elements are placed within column grids. Using the media query and some math, the number of columns that can appear based upon a screen width is determined. They are displayed horizontally, and remaining columns wrap horizontally, as figure 2.4 displays:

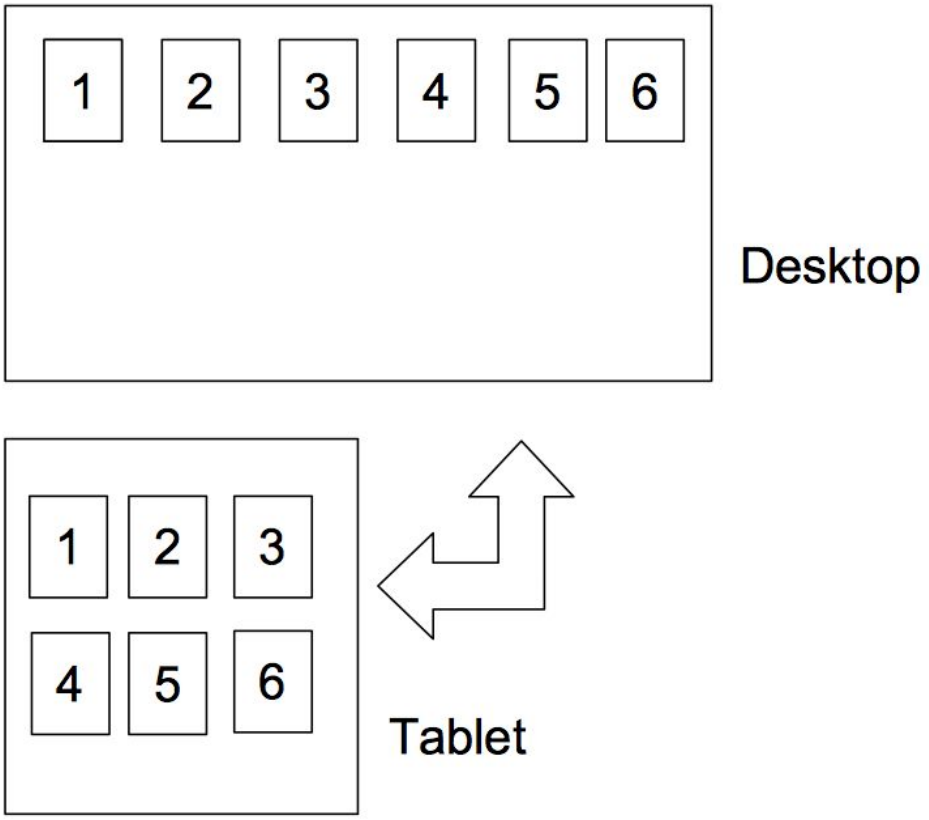


*Figure 2.4 – Grid Layout columns wrap with device width size*

There are a number of frameworks and approaches that will generate grid-based CSS. You can specify the number of columns you want and offsets between columns, and the framework

will do the math and generate responsive CSS for your specified grid CSS for you to apply. Some even allow number of columns to specified.

Listed below are a some popular grid based frameworks, as published by Mashable:

- Gridset – gridsetapp.com
- Frameless – framelessgrid.com
- Tiny Grid – tinyfluidgrid.com
- Simple Grid – gridpak.com
- Responsify – simplegrid.com
- Responsive.gs – http://responsify.it/
- Golden Grid System – goldengridsystem.com

These grid frameworks are more useful for web page designers. We are focusing on enterprise web applications where we will be using user interface widgets rendered with CSS, HTML, and Javascript to handle user interaction and events. So, just having CSS grid mechanism by itself is not enough. Other elements like forms, tables, and navigation need to work in concert with the responsive grid layout.

### Bootstrap.js - Turning Mere Developers Into UI Geniuses

One very popular open source framework, Bootstrap.js, provides a responsive grid based upon the less framework. It also provides many other features such as typography, buttons, and user interface themes such as pills, navigations, alerts, dialog components, and forms, among others. All of these elements are styled consistently, under the same responsive framework. This popular frameworks can make a developer look like a UI design genius.

Enough of the plug. The power of this framework, besides a clean look and feel and a responsive fluid-layout, is that is provides forms, buttons, modal dialogs, navigation, and other UI elements and components that enterprise-type applications can use to implement a responsive user interface. Developers can put together a clean user interface without knowing CSS magic. Actually, most of the framework is implemented with CSS and behind the scenes JavaScript. So developers apply simple HTML with CSS class attributes.

Bootstrap employs a 12 column fluid grid layout. Using <DIV> tags, class attributes are used to define a containment area that has rows and columns, or spans in bootstrap lingo.

A common UI layout with a heading area for a navigation element, then a side bar and content area, is shown in figure 2.5.

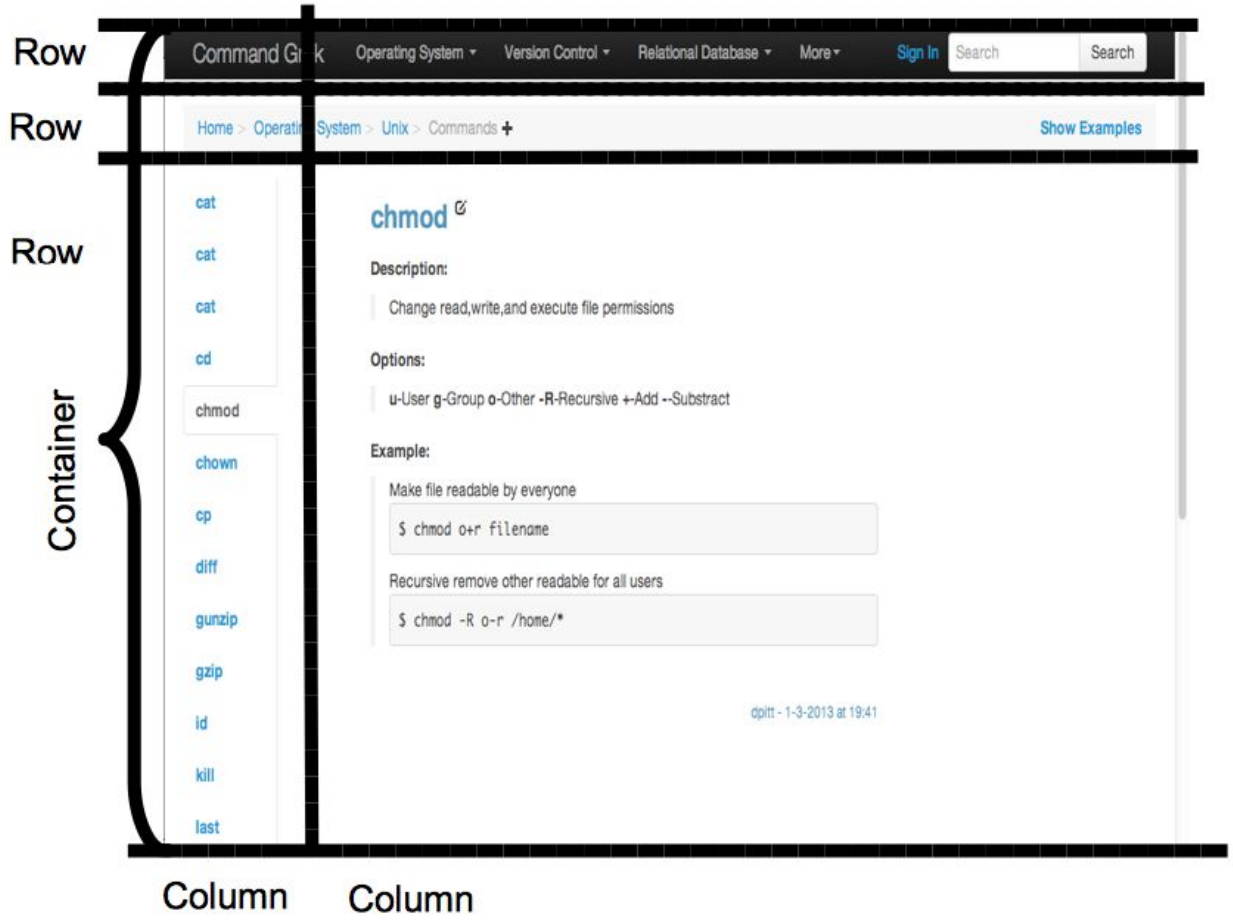


*Figure 2.5 – Bootstrap Container row/column layout*

Another valuable feature worth pointing out, especially for enterprise applications, is the form support. If you ever tried to implement an input form using plain html <FORMS> then you know that trying to get fields to line up and look cohesive can be very frustrating. But as you can see in the previous form example, the input form is lined up and looks cohesive. Bootstrap allows this to be defined using plain old HTML tags. The framework uses CSS attributes to communicate styling and layout.

The novel approach that Bootstrap takes is that it is completely class-based. It also has many other components and features. Plugins are also available so other GUI components can be added. The bottom line is Bootstrap provides most of the elements you need for a clean responsive user interface. One possible downside is the lack of table, tree, and grid type components available out of the box. But there are many Bootstrap plugin components available. With the popularity of Bootstrap, many more are sure to follow.

### GUI Libraries

There are numerous libraries available that emulate user interface controls beyond the standard HTML form controls. This includes trees, controls, grids, tables, sliders, robust drop down entry fields, date pickers, and more. The primary theme is to emulate widgets that are common to windowing-based operation systems. In the enterprise, they are most likely emulating Microsoft Windows type widget controls. Popular widget libraries include jQuery UI, DOJO, YUI, and many more are available in the open source world. These libraries provide a rich user experience, however, many of them are not responsive. To make them responsive you will have to adapt them to a responsive CSS framework.

# Responsive UI Design Decisions

More than just CSS magic is required for applications to be responsive. Developers and user interface designers need to implement user interfaces that depart from the traditional "windows" GUI metaphors we are used to. Why you may ask? These user interface designs attempt to emulate native operating systems for desktops. The desktop, or laptop for that matter, is no longer the only device from which your enterprise application will be viewed from. So let us discuss some design decisions you can make that will help make your applications more responsive across many devices.

## Limit Data Entry Dialogs

The "windowing" metaphor that we are used to across major operating system environments allows users to have many windows open at the same time. A common way to accept input from users has been to open a modal or dialog window, which essentially locked the input window until the user entered and applied data input or canceled the operation. An approach that supports a responsive layout and small screen sizes, is to eliminate dialogs and use a direct edit approach. Consider an edit option, as seen in figure 3.1, that shows how a user interface allows a timesheet to be edited.

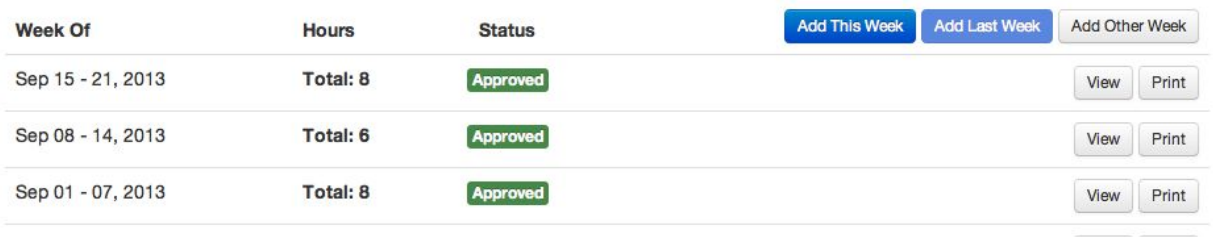


*Figure 3.1 – Timesheet form*

When the view/edit button is selected, instead of opening a dialog to view and edit a timesheet record, an edit form is exposed inline using a UI transition. When done, the edit form transitions away. This is shown in figure 3.2:

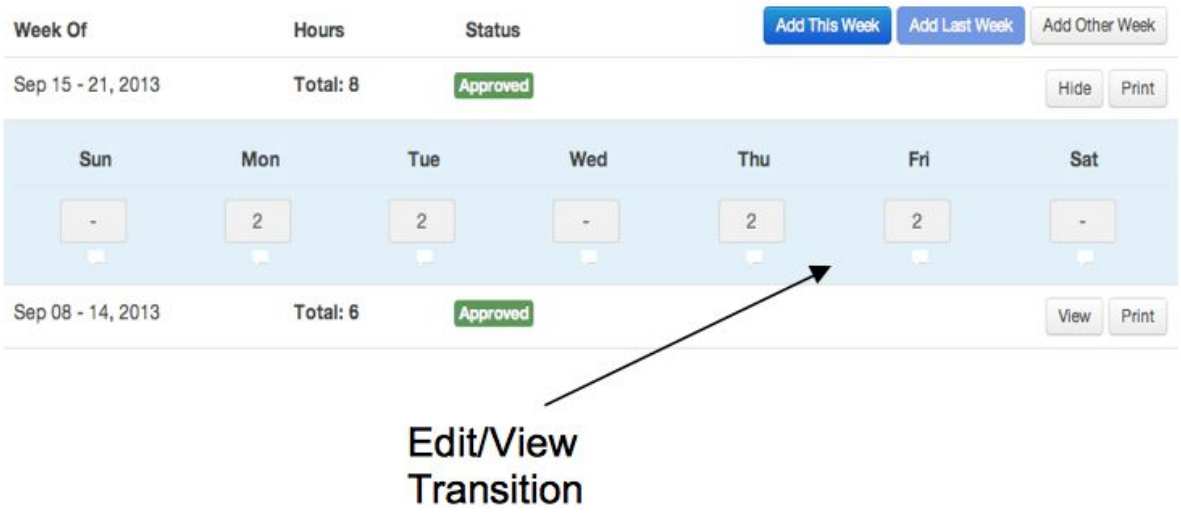


*Figure 3.2 – An example of the View/Edit Transition.*

By hitting the "View" button on the Sept 08 – 21 row, a form is exposed inline providing more details and editing capabilities.

Dialogs, alerts, and short messages serve a purpose when in situations that require user attention. Otherwise, try to minimize their usage as they can act wonky on difference device screen sizes. If they are too big, they might not even work on some devices.

Other ways you can eliminate dialogs is to use UI form controls that integrate validation and selections mechanism in the form control entry field. An example would be date/time pickers and range-based entry fields. Don's use or implement a dialog to assist the user in selecting a date. Make the date appear as an inline transition within an input form.

## Apply Touchable Controls

There are many mobile touch specific UI libraries. However, many of these controls work equally well in desktop applications and work just fine with a keyboard and mouse. So, apply them in enterprise desktop applications. Examples of where a touch control instead of traditional input form are show in figure 3.3.

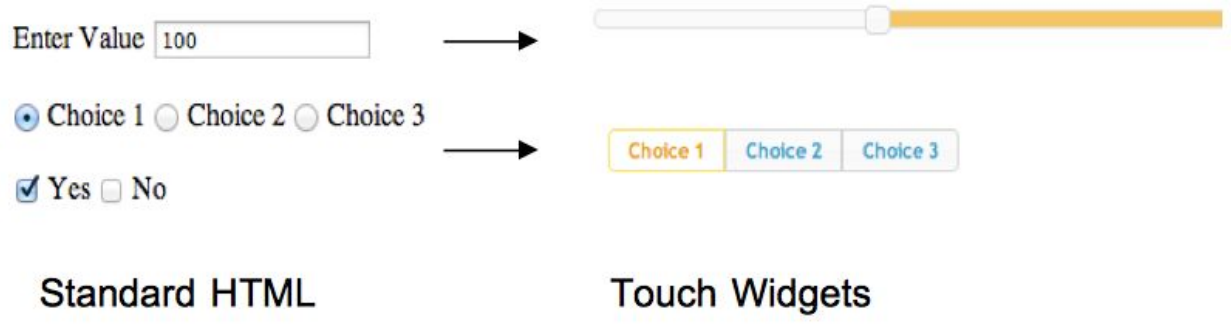


*Figure 3.3 – Use tough widgets instead of standard HTML widgets*

Slider controls can be used for numeric input. Toggle buttons can be used in the place of radio buttons and check boxes. Touch controls provide the additional benefit of a modern UI look to your application, which can help with user satisfaction.

## Design Vertically

Moving or scrolling up and down, vertically, is more natural to the eye than horizontally, or left and right. Avoid designing UI screens with a lot of information going across the screen. If you do, partition into panels, so they move naturally vertically when the responsive layout stacks them vertically as device screen size changes.

## Think About Navigation

Enterprise applications can offer much functionality with a variety of user interface screens that have lots of information to display and maintain. Implementing a simple way to access and navigate features or functions of the application makes applications easy to use. The right decisions on navigation can help with responsive design.

For instance, a common navigation scheme is to put tabs across the top of the user interface. They can be thought of as menus in traditional windows applications, as shown in figure 3.4.

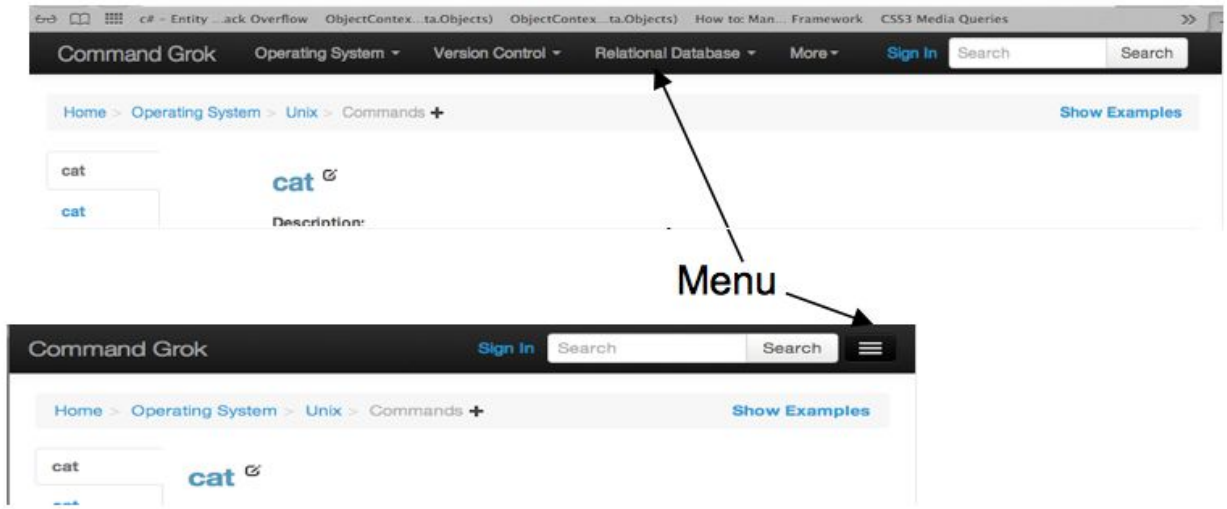


*Figure 3.4 – An example of collapsing menu options*

When the screen is resized, menu options are collapsed to an icon. Options and sub options are displayed horizontally when selected.

Using small meaningful icons are a great way to conserve real estate and keep your interfaces clean. Use icons that commonly infer the operation that you are trying to communicate, as you can see in figure 3.5.


*Figure 3.5 – An example of meaningful icons to replace menu options*

This may be obvious, but don't underestimate the power of icons that can be used for commands and points that allows the use to transition between application screens. You don't need a graphic artist to obtain icons, many libraries provide them out of the box.

Also, tabs are your friend. Use tabs instead of popping up windows upon new windows to make it an easier experience for your user.

# Summary

This chapter introduced the concepts of responsive design for enterprise web applications. The goal is so that applications function well on devices other than desktop browsers. Here is a recap of what we covered in this chapter:

- Responsive design allows a single application to work across many devices.
- A common responsive design technique is to divide UI elements into a grid.
- CSS3's media query is key technology element in implementing a responsive design.
- Frameworks are available with ready-to-use responsive layouts.
- Bootstrap.js is a library that provides a responsive layout mechanism, as well as UI controls and a standard look and feel that can make mere developers look like UI geniuses.
- Touch controls can be applied not only to make them touch compatible, but to conserve screen UI and add some pizazz to applications.
- Responsive design is not just applying a responsive CSS, but making smart UI element choices.

## References

http://webdesignerwall.com/tutorials/css3-media-queries

http://www.smashingmagazine.com/responsive-web-design-guidelines-tutorials/

## About The Author

David Pitt is a Sr. Solutions Architect and Managing Partner of Keyhole Software with nearly 25 years IT experience. Since 1999, he has been leading and mentoring development teams with software development utilizing Java (JEE) and .NET (C#) based technologies. Most recently, David has been helping organizations to make the architecture shift to JavaScript/HTML5 and use best practices to create rich client and single page applications.

## About Keyhole Software

Keyhole Software is a Midwest-based software development and consulting firm with a team that loves technology. Our expert employee consultants excel as “change agents,” helping our clients to be successful with software technologies that bring competitive advantage. See some of our recent projects here.

We frequently assist clients with custom application design, development, and modernization

initiatives with Java, JavaScript/SPA, and .NET technologies.

Keyhole was founded on the principle of delivering quality solutions through a talented technical team, and as such, knowledge transfer is important to us. To our clients, we offer various techniques to provide the most value: one-on-one or group mentoring, lab/lecture educational courses, and access to our knowledge transfer engine GrokOla.

### Related Services Snapshot

- Technology Consulting – Expert Keyhole Consultants work with organizations to analyze the current state of applications, recommend technical directions, and develop plans for modernization.
- Application Development – Individual members or entire teams of Keyhole Software Consultants perform expert development services: application analysis, user interface design, development, testing, and enhancement of custom applications.
- Education – Instruction of a wide range of custom courses and technical mentoring programs for knowledge transfer to groups or individuals, including Introduction to Microservices, SPA Development With Full Stack JavaScript Using AngularJS and Node.js, JavaScript UI Development Using AngularJS or Backbone.js, and others.

# For More Information

***Keyhole Corporate Kansas City***
8900 State Line Road, Suite 455
Leawood, KS 66206
Tel: (877) 521-7769

***Keyhole St. Louis***
Tel: (314) 477-7962