

Design Guide example

I contribute to use cases, accessibility analyses, usability analyses and standards documentation for the company as the UI SME for the CTO office. If requested, I can provide lengthier excerpts from articles, coursework and a book chapter I've published on design elements. Below are some writing examples from a corporate style guide I co-authored.

From the *User Interface Design Guide for Extranet Applications*

Understanding Web Technology Principles

This section helps to build the framework of understanding regarding Web technology by providing a comprehensive discussion on the types of Web site designs available.

What is a User Interface?

Interface refers to a surface lying between two portions of matter or space, and forming their common boundary (Oxford English Dictionary); interface implies a connection or relationship between different elements. Interface is also explained as a contact surface which reflects the physical properties of the interactors, the functions performed, and the balance of power and control. In computer terms, this contact surface can refer to either the visible toolbars or menus of a software program or, the underlying operating system itself. Transparency of this boundary or connection is crucial for a good interface design. If the interface is well designed, it is no longer "visible" in the sense that the participant or user is conscious of it. Interface guru Theo Mandel notes that giving users work objects rather than system objects to manipulate contributes to the transparency of an interface. The transparent user focuses on the tasks that serve work objects rather than the underlying system. Creating an interface that is "in sync with the user's mental model" is one of the primary goals of interface design (Mandel, Theo).

Characteristics of Web Sites and Web Applications

A Web site's primary purpose is to convey information, sometimes dynamically. A Web application is designed for a specific purpose and with greater functionality and extensibility than a Web site. In other words, a Web site is not an application and Web applications are not Web sites. Indeed, typical design standards for a Web site may not be appropriate for a Web-based application. The goals of a Web application may differ from that of a Web site; and the same design rules for each may not apply, since they are not intended to achieve the same ends. Web-based applications (or Web applications) are dynamic and specific in terms of context. The goals of a Web application are to go beyond the display of information and to the interpretation, management, distribution, analysis, and transformation of information.

The Focus of the Internet Browser

The Internet browser engine (such as MS Explorer, Mozilla Firefox, AOL Netscape, and Apple's Safari) is focused primarily on the display of static information that is based on a typical Microsoft Windows, icons, menus, and pointers (WIMP) interface, leaving the developer who completes the final stages of coding, to implement the interactive components.

User-Driven vs. Data-Driven Web Design

Web applications are functional, interactive experiences and should be designed according to the user needs for the specific requirements of the business. Form should follow function. In general, Web applications are more than just a collection of data on a browser; they are dynamic applications with potentially unique controls and specific

purposes. The amounts of data displayed as well as its manipulation are both factors in the architecture and infrastructure of an application. By contrast, a Web site is data-driven, not application-oriented or functional in design. Establishing and maintaining user interface design standards specifically for Web applications is essential to clearly differentiate the action and activity of an application from the information of a Web site.

Static View vs. Dynamic View

Static view refers to an unchanging visual display of information on a Web site. In some cases, it also refers to the (largely) static display of information within designated content areas. The user cannot alter static information.

EXAMPLE

A news Web site may contain scrolling information in a specified area. Although the actual content changes over time, the layout of the content area remains unchanged.

Dynamic view refers to data that is generated ad hoc (and kept current) when the user (or application) requests it. It can represent data that is extracted from outside of the system.

EXAMPLE

A Web site that provides current weather conditions may retrieve data from a third party, such as the National Digital Forecast Database.

The data the user sees may take different forms, such as a graphical representation. Dynamic views are active views that the user sees on the screen, for example, a series of tables containing information that is visually indicated or, a motion-oriented chart that displays the system monitoring status.

Web-Based Designs: Portal Versus Service-Oriented

Emerging industry best practices, designs, and architectural patterns are important considerations for user interface design. The use of portal technologies and service-oriented architectures directly influence the functionality and design of Web applications.

Portal technologies and Service-Oriented applications (SOAs) offer the ability to cater to applications that interoperate and collaborate. The leveraging of common standards for Web applications, with regards to patterns, designs, and infrastructure across a network, affords users enhanced functionality and usability.

Portal-Oriented Designs

Web applications expressed in portals have specific usability patterns that apply to and impact usability designs. In the long-term, the use of portals also promises to deliver a platform that can provide consistency with branding and co-branding, as well as accessing a variety of shared application services.

Service-Oriented Designs

Service-Oriented Architectures (SOAs) and Web services promise to restructure Web applications and provide greater re-use of business functionality and services. The use of Web services can help unify various multi-tier architectures, enabling greater re-use of application infrastructure.

[...]

Layout Consistency and Visibility

A clear conceptual model refers to consistency in the presentation of operations and a coherent, consistent system image (that is, the model is comprehensible). Consistency is key at any point when the interface interacts with a user. To ensure there is visual consistency, developers should:

- Collaborate with other internal groups regarding the layout.
- Verify that the data enclosure does not detract from the data.
- Avoid creating a space that brackets data—visual clutter can make the wrong feature stand out and create a vibrating space that diverts from more relevant data.

Design layouts to display on no larger than a 1024 X 768 screen. The user's browser typically has a toolbar, a scroll bar, and a status bar that also consumes some of the screen area. Since the visibility of relevant data is of the highest importance, an implementation of the Web application can be a "chromeless" design. The "chrome" refers to the buttons and toolbars at the top of a Web page as well as the status bar at the bottom of the page. Also, take into consideration that part of the power of Web applications is the ability to leverage the known browser interface.

Types of Adaptable Designs

The forms of adaptable design available include:

- Designing For Multiple Platforms
- Portal Design
- Wizard Designs

Designing For Multiple Platforms

It is important that Web applications not lock out the disabled or users of earlier browsers and older operating systems, as the goal should be to reach the largest possible audience. All products should reliably function across all of the target platforms. To ensure cross browser functionality developers can:

- Be informed as to each target browser's (and browser version) limitations; never rely solely on the implementation of any advanced features. Make it a practice to test the Web pages to ensure the displayed functionality is operating reliably in each brand and version of the browser.
- Ensure all code is backwards compatible, within reason.
- Rigorously test all code on supported browsers and versions as identified by the business owners.
- In developing the user interface for a business application, consider providing alternate page content as "light" (or abbreviated) versions. This ensures that the page content is always accessible, regardless of the client version in use.

NOTE: *When considering alternate page content, keep in mind that application business owners need to be informed, as this may impact the project in terms of development and testing costs.*