Bing Bar 7 Human Interface Guide

This document describes design considerations and standards for Bing Bar apps. Use this document to understand how users benefit from Bing Bar apps, what's possible within a Bing Bar app, and which design patterns are most appropriate within Bing Bar apps. For technical information about implementing Bing Bar apps, see *Bing Bar App Developer's Guide*.

Bing Bar Mission Statement

"We want the toolbar to be useful, unobtrusive, and anticipative. It serves your needs, gets out of the way when you don't need it, and gives you little magical moments when it presents you with information."

Bing Bar Apps

Bing Bar apps provide quick, at-a-glance access to information, entertainment, or assistance while the user also does other activities on the web. The Bing Bar app platform lets partners extend Internet Explorer with rich, interactive apps and tools. Unlike a web page, the Bing Bar travels with the user. This means apps can be aware of the user's activities and intentions. A Bing Bar app has new tools to determine a user's goal, including access to data such as the browser URL, DOM, and search history that it can use to determine the user's intent, help accomplish the user's goal, or send the user to a web page. This new app platform resides between traditional, static links found on traditional browser toolbars, and the full presentation of a web site visit. Bing Bar apps can help users stay connected to personal web services, and stay informed about topics of interest.

User Experience Guidelines

This section describes how Bing Bar apps offer value at every step of user interaction.

Arc of disclosure

Bing Bar apps guide the user through a path (or arc) of growing information disclosure. Starting with the app button and ending with the app's parent web site (if applicable), each step of the Bing Bar app experience reveals more information to the user. Design your app to provide value at every level of interaction.

The arc of disclosure of a Bing Bar app begins with a dynamic button image which might convey information such as the state of a user's account on a web site, including an overlay of text or graphics that provide visual cues when new information is available. For example, the weather app image changes to reflect current weather conditions.

Figure: Button image showing current weather conditions

A single glance might satisfy the user, or the user might hover over the button to reveal a descriptive infotip that provides more details. The weather app uses the infotip to describe current weather conditions in more detail.

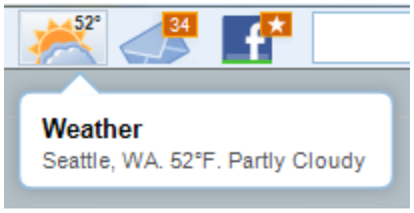


Figure: Infotip describing weather conditions in more detail

The infotip can work in conjunction with the button's image and indicator. In this example, the indicator signals a weather alert, while the infotip describes how to interpret the indicator and get more information.

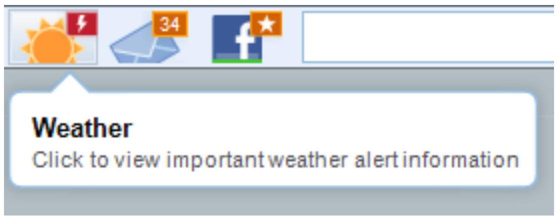


Figure: Infotip describing button indicator and outcome of a button click

Clicking the button opens the app wing, without navigating away from the current web page. After interacting with the app wing, the user might click an element in the wing that launches a detailed web page in the browser window. Throughout this process, from the button image to the web page, each interaction reveals more information, with minimal user interaction.

This table describes each UI element of a Bing Bar app across the arc of disclosure.

UI Element	**Use frequency**	**Remarks**
Button image and state indicator	20+/day	Can contain visual cues about dynamic content, so the user can determine useful information just by glancing at a button.
Infotip	5+/day	Can provide more details about the meaning of a dynamic button's visual state.
Alert	Varies	Can alert the user to app state or provide contextual information. Can include buttons, and can be clicked for more information.
Wing UI	1-3/day	Can engage user for a minute or two with a productive, useful, or entertaining set of options that may induce the user to click through to a web site.
Web site	2-3/wk	Can be a specific page the user has sought using the Bing Bar app.

App design principles

These principles guide the design of every successful Bing Bar app.

Inductive, at-a-glance app interfaces

Bing Bar apps present an inductive user interface to provide at-a-glance information and usability. Each view in an app should perform a single, easy-to-understand task. The user should not have to deduce the purpose of an app UI that is overloaded with text and overlapping tasks. App designs should present the app purpose and function with simplicity and clarity, employ see-and-point interfaces and immediate visual feedback, show only essential information at the right time, use visual metaphors instead of text, and present users with a manageable number of sensible options. See more information about inductive user interfaces at this web address:

http://msdn.microsoft.com/en-us/library/ms997506.aspx

Information hierarchy

Because the features of a Bing Bar app differ from those of a full browser, approaches to complexity for Bing Bar apps must differ from those employed on full web sites. Avoid full hierarchies of navigable categories. Strive for concise or even image-based navigation flows. Navigation tools such as bread crumbs and nested pages that rely on a home button are not appropriate for Bing Bar apps.

App state persistence

Bing Bar apps do not persist like a browser window. If the user clicks another application, even accidentally, the Bing Bar app closes. Expect users to unintentionally close your app wing when the clicking away to use the web browser or another app on their computer. Most Bing Bar apps should store app state so the user can proceed where they left off when they return to the wing. For example, a game app should save the game automatically when the wing closes. Don't lose the user's place in the game, or force the user to click a pause button.

Privacy

Bing Bar apps can monitor user activity, including web addresses and web page contents. Apps must take user concerns about privacy seriously, and avoid being perceived as a monitoring tool. Use helpful language like "handy tip" rather than creepy phrases like "we observe you've visited this site 4 times." Balance the potential value of messages based on observations against the risk of upsetting the user.

App archetypes

This section describes some app styles, based on visual and behavioral characteristics. These varieties are named and described to help illustrate the different kind of apps possible within Bing Bar. This list does not imply there is a rigid classification scheme that all Bing Bar apps must follow. Review these descriptions to see how different approaches can be suitable for different functionality and goals.

Utility / Radioman

Utility apps are typically read-only apps that function like a radio announcer. You turn on the radio to get organized, easy-to-understand information delivered to you. These apps present the kind of information that you get at the top of the hour from your local radio station, such as weather, sports, or news. Getting a quick summary of information becomes part of a morning routine for audiences. A utility app performs a simple task that requires minimal user input. Utility apps are visually attractive in a way that enhances the information being conveyed without overshadowing it. The user opens the app and digests the information in less than a minute with minimal interaction, and no searching or filtering is required.

The weather app is a good example of a utility app. This app summarizes current and near-future weather in the user's area. Users can click the toolbar button to view more details in the app's wing. From the wing they can pursue more details by clicking on links into Bing.com.

Productivity / Task Finisher

A Bing Bar app can help a user finish a task, saving time. The maps app is a good example of a productivity app, because it provides a map of an address without opening a new browser. Another productivity app, the language translator, automatically translates text on a web page into another language. The Hootlet extension for the Chrome browser is a great example of task finisher. When the user clicks the Hootlet button, the app combines the URL of the current web page with metadata to create a tweet for publication on Twitter.com. For more information about Hootlet, see this web address:

https://chrome.google.com/extensions/detail/bjgfdlplhmndoonmofmflcbiohgbkifn

Key attributes of a productivity app:

- Simplifies complicated tasks.
- Usually reduces the number of steps users take to accomplish a task.
- Can use the content of a web page to prepopulate data in anticipation of what the user wants to do.
- Uses simple inductive controls to reduce cognitive load. Users employ simple controls to finish the task. Avoid comprehensive designs that overwhelm the user by trying to handle every possible scenario.

Immersive

An immersive Bing Bar app provides an experience that usually meets user needs without navigating to web pages. Games and audio/video players are examples of immersive apps. An immersive app might show movie trailers or provide complete games within the app wing. More advanced tasks, such as managing a playlist or buying music, might require a visit to the parent website. Immersive apps encourage longer engagements of one or even ten minutes at a time. An immersive app can store the state of a video or game and remind the user to return later for more. These apps give a compelling reason why the user would use the app in the wing rather than visiting the web page.

This table shows good and suboptimal ideas for features of an immersive app.

Good immersive app feature	**Suboptimal immersive app feature**
Watch a movie trailer or short viral video.	Watch a full length film. Manage a personal queue of films.
Play a simple side-scrolling game that is easy to learn and uses one or two simple controls.	Play an immersive game with 3D graphics and a complicated control set with keyboard shortcuts.
Listen to a streaming station, rate music, or see artist information.	Build, play, and share custom playlists.

Wingless app

A wingless app provides assistance to the user, and does not provide content. A wingless app can be low-key, and might not have a button or a wing UI. Since an app can examine the DOM of every web page, it could alert the user to features based on context clues or patterns. The translator app converts a web page into a different language. A wingless app could detect an event microformat in the HTML of a Facebook page and show an alert that offers to add the event to the user's calendar.

Butler

Real-life butlers, or domestic assistants, observe clients and provide expert assistance without being intrusive. They anticipate needs and offer useful, reliable services. Butler apps serve users through the Bing Bar alert platform. Alerts from a butler app are subtle, anticipative, unobtrusive, contextual, and perhaps personalized to the individual user. A great butler app also knows when to stop contacting the user. The goal is to provide subtle and intelligent communication. A great butler app shows that users don't necessarily have to come to an app; an app can bring aide to the user at just the right moment. Butler apps could have an app button, but might just make smart use of Bing Bar alerts.

Some examples of butler apps:

- An app that discreetly notifies the user when the product or travel options appearing for sale in the browser are available more cheaply through another site.
- An app that detects when the user is reading an article and discretely offers to strip extraneous layout and show the article in an easy-to-read format.

Bing Bar App Components

The Bing Bar uses the same horizontal band as a traditional toolbar. However, with the centered layout, the toolbar looks different, and the buttons and functionality launch rich web apps instead of pages on remote web sites. The goal of the Bing Bar is to be an engaging app bar, where the buttons launch an app wing at the center of the user's attention.

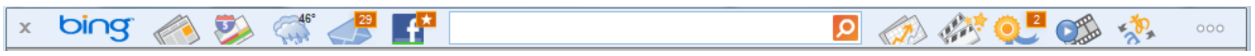


Figure: Bing Bar

App buttons

A Bing Bar app button can show dynamic images that inform the user. Button imagery should change to reflect app state. The weather app is a classic example, showing a sun behind clouds to illustrate conditions in a glance. Button images are highly visible and should help the user identify the app and its function. Your Microsoft product team partner can advise you on designing icons that illustrate your app or app states.

Button state indicators

An app can also overlay a smaller graphic or a very small amount of text in the upper-right corner of the button to show more information about app state. A text overlay can include transparent, non-urgent (orange), or urgent (red) background coloring. Most apps should include a non-urgent text overlay that says "new" on the first few app runs. The infotip and wing should show more information about the meaning of state indicators that appear on the app button. For example, the weather app uses this indicator when there is an important weather advisory in effect for the user's location.

Remove indicators promptly

An indicator shows that the app has a message or new content for the user. The indicator should disappear when the user presses the app button. The indicator should also disappear if the user views the message or information elsewhere, such as on the web site. Be sure to think holistically about what the user has already seen when determining whether an indicator should appear.

Show green lock during authentication

When your app shows a sign-in screen, overlay a green lock indicator on the button. The green lock overlay provides a small measure of anti-phishing mitigation because it connects the toolbar chrome to the app. A web page pretending to be an app cannot modify button states. Note the green lock on the Facebook app button that appears while the authentication UI is visible.

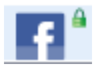
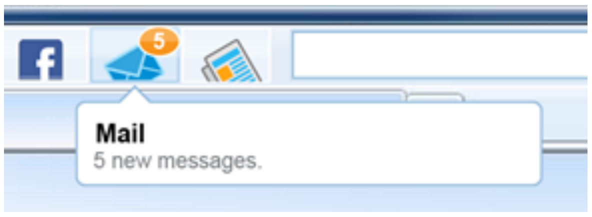


Figure: Green lock during authentication in Facebook app

Infotips

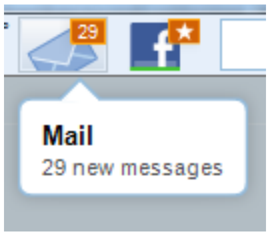
App buttons can communicate more detail about app state through a rich tooltip called an *information tip*, or *infotip*. An infotip appears when the mouse hovers over the app button.



Mail
5 new messages.

Figure: Mail button with state indicator and infotip

An infotip provides a short but meaningful description of what is waiting for the user in the app. The default infotip before the first run of an app should contain a general statement of the app value. The first infotip line shows the app name. Below the app name up to two brief lines of descriptive information about the current app state should appear. If you're using an indicator, the second line of the infotip should clarify what the indicator means. For example, the mail app indicator shows a number and its infotip explains what the number means.



Mail
29 new messages

Figure: Button state indicator with infotip

Alerts

An alert is a popup notification that appears below the Bing Bar search box. Alerts can keep the user aware of topics they care about in real time, educate the user about features related to an app or the current browser session, and help users stay connected to their personal services.

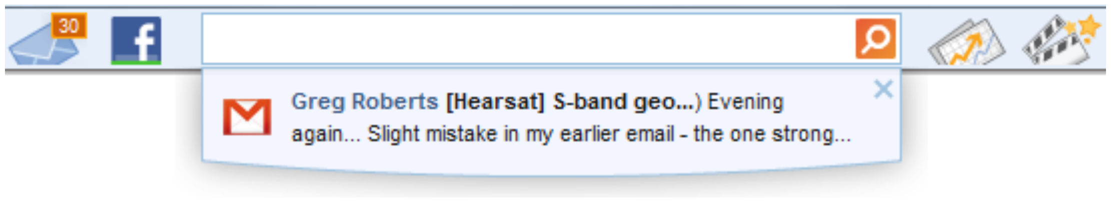

30
Greg Roberts [Hearsat] S-band geo...) Evening
again... Slight mistake in my earlier email - the one strong...

Figure: Alert from mail app

Alerts contain a 24x24 pixel icon that is probably similar to the button icon. The app button should reflect the state or event that triggered the alert as long as it still applies. Recent alerts should also appear within the wing content to help the user catch up on activity. All alerts open directly below the search box.

A click on an alert should always trigger an associated action. If the user clicks the alert in the figure above, the full message on the Gmail website appears. Use buttons when there are multiple options.

Alerts should be relevant. Avoid showing alerts for items the user has already seen. Set an appropriate time-to-live value, and remove stale notifications. If the user does not act on an alert, represent it in the button indicator. If possible, batch multiple events into a single alert.

There are four types of successful alerts:

1. Alerts containing information from the user's personal services help them stay connected. For example, an alert might inform the user of a new friend request within Facebook.
2. Alerts containing information the user would like to stay on top of. For example, breaking news from the news app, or fresh sports scores, perhaps based on the user's history of sports inquiries.
3. Alerts containing information that is relevant to the current web page or current browsing session. For example, an alert can offer deep links into a web site.
4. Alerts containing information that promotes a feature that may interest the user based on observed web activity. For example, an alert can direct the user to the mail app when the user navigates to gmail.com.

Alerts can text contain buttons, or the entire alert window can click through to a wing or web page. Alerts should only be clickable when a click can show the user **more** information, not the same information that already appeared in the alert. Alerts can be tethered to only fire while the user is active in specific browser sessions, or at a specific web address. Alerts are subject to throttles. Because you don't know if or when an alert might appear, alerts cannot be the foundation of a responsive user interface. For details, see the bingclient.ui.notifications namespace in *Bing Bar 7 API Reference*.

Wings

The wing is the main app canvas.

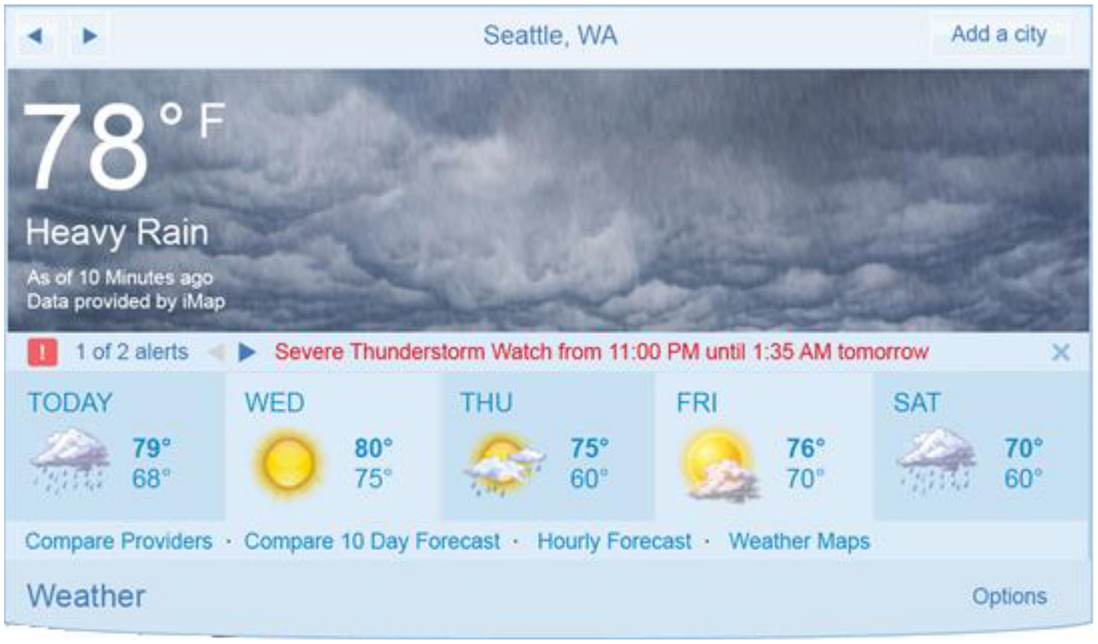


Figure: App wing for the weather app

Wings provide a lightweight way to access simple information without having to load a full web page. A wing is smaller than a full browser window. Its smaller size forces apps to implement simpler designs. When the task warrants a full browser, the app can close its wing and transfer the user to a full web page. A wing opens when the user presses the app button, and closes when the user concludes the wing session or when focus leaves Internet Explorer. An app can contain many views within a wing.

Wing size

All app wings are 550 pixels wide, including the wing frame. The suggested height for a wing is 308 pixels. A wing can grow temporarily to accommodate larger UI scenarios, but should not exceed 440 pixels to remain visible on low-resolution displays.

Common wing components

Wings contain optional and required components.

Wing header

Your app can provide a header in its wing. You are not required to include a header, and this SDK does not provide sample code for headers. Only add a header if it makes sense for your app.

Wing footer

A wing must include a footer, provided to apps by source code in shared\footer.js. The footer includes a title, and can include an Options button. The title should match the app title in the infotip, and can be a hyperlink to a parent web site. Icons or logos in the footer are not allowed.

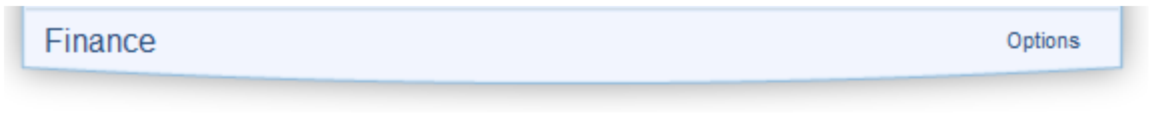


Figure: Footer with Options button

When pressed, the Options button provides a visual transition to your app settings user interface, and the Options button becomes a Close button. The user interface for app settings is modeless, and changes commit immediately. If the user leaves an app settings page, the Close button is pressed implicitly, so the user will not return to the app settings page when returning to the app.

Cached content and load delays in the wing

Apps should avoid showing a blank wing. Cached content can prevent a blank wing, and a timestamp can make users aware when the wing is showing cached content. The weather app shows a load delay indicator when there is no good cached content and the app needs to fetch data to generate the UI.

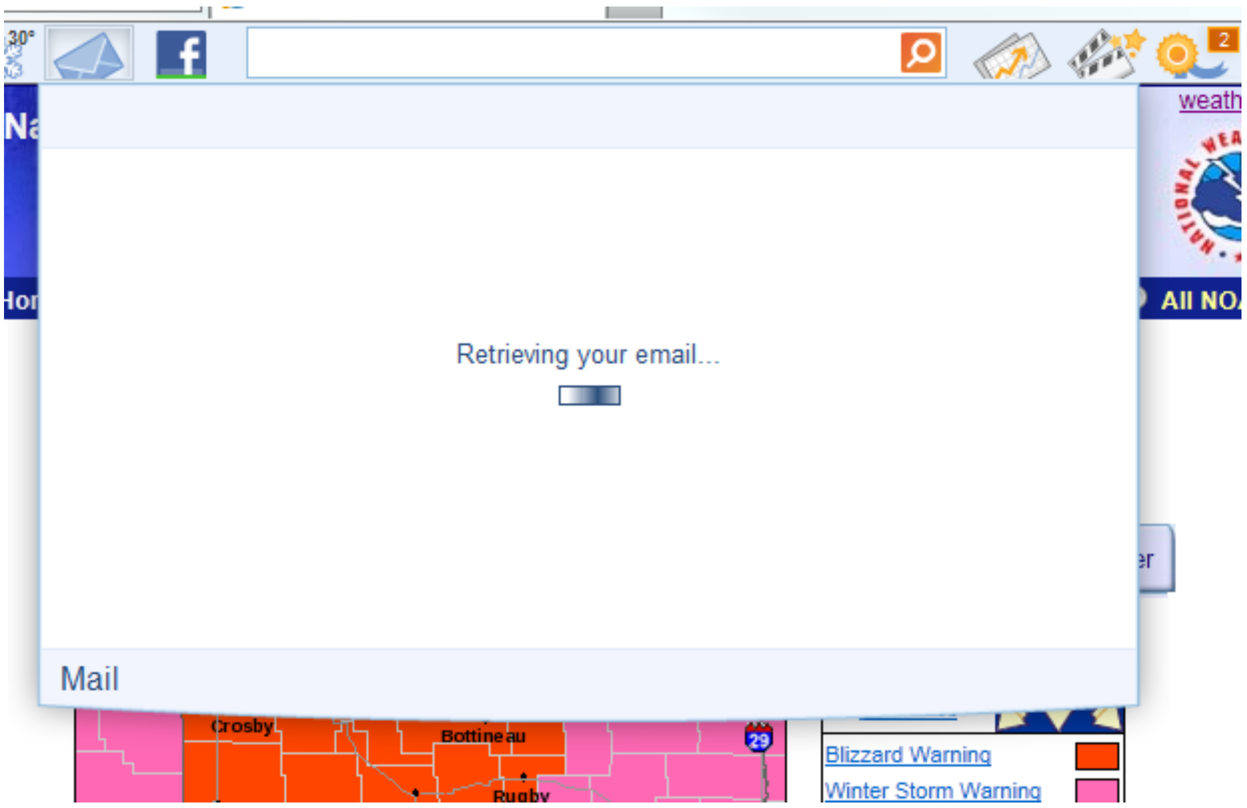


Figure: A load delay indicator during a global delay

When an app cannot populate its user interface because it is waiting on remote data or calculations, it should show a load delay indicator in the center of the wing. When an app can show content or user interface elements while waiting for a remote data or calculations, it should show a load delay indicator in the footer of the wing, where the Options or Close link typically appears.

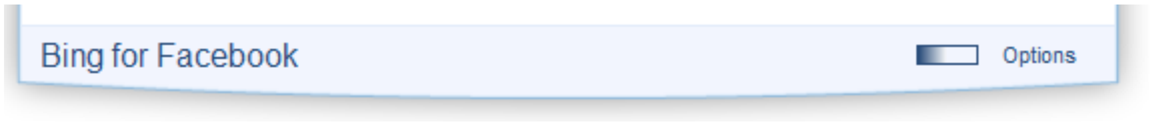


Figure: Load delay indicator when interface or content remains available during delay

Design guidelines

Follow these guidelines when designing a Bing Bar app.

Show cached content

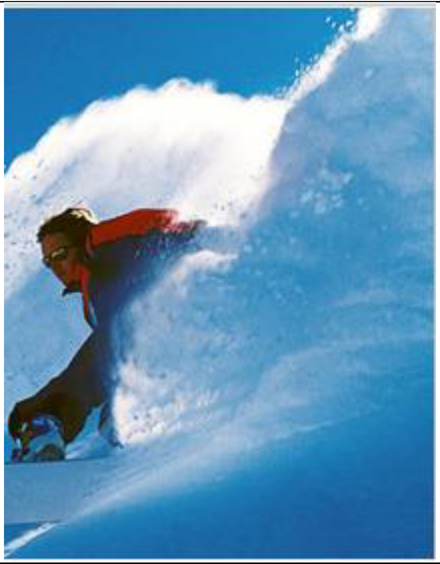
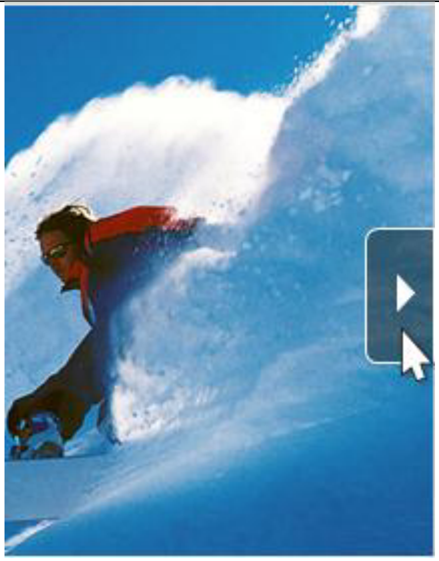
Design dataflow that supports cached content, and show cached content when network services are slow or unavailable. Stale content is always better than an error message. The user might not care that some content is stale. If stale content could cause confusion, include a timestamp. The goal is to provide an app that never stops working, even if back-end services are occasionally unavailable. The Bing Bar API includes file storage features that can store content between sessions.

Pre-fetch content

Content loaded in the background enables a more responsive experience. For example, if your UI uses pages, when the user clicks the page control, load the next page and then pre-fetch a few pages ahead. Where ever possible, avoid making the user wait for a remote resource.

Hide controls until needed

Users understand interfaces faster when their initial view is a simplified presentation of interface features. Apps can hide controls until the user enters the interface area using the mouse or keyboard. For example, if you're app uses carousel navigation to page through views, you can hide the navigation controls until the user hovers near the area.

Default	**On mouse hover**
	

Provide value before consistency

Bing Bar wings only require a footer, and some Bing Bar apps use a header and other common elements. Beyond these simple elements, it's up to you to design the best app that meets your user's needs, your requirements and your branding guidelines. Don't use the standard color palette if it doesn't make sense for your app. Provide value to users and represent your brand before providing consistency with other Bing Bar apps.

Implement modal dialogs in slide-down panes

If you need to implement a sign-in screen or a UI that is modal to the app, use a slide-down pane which animates up from the footer area. The footer options pane and mail app account pane use this technique.

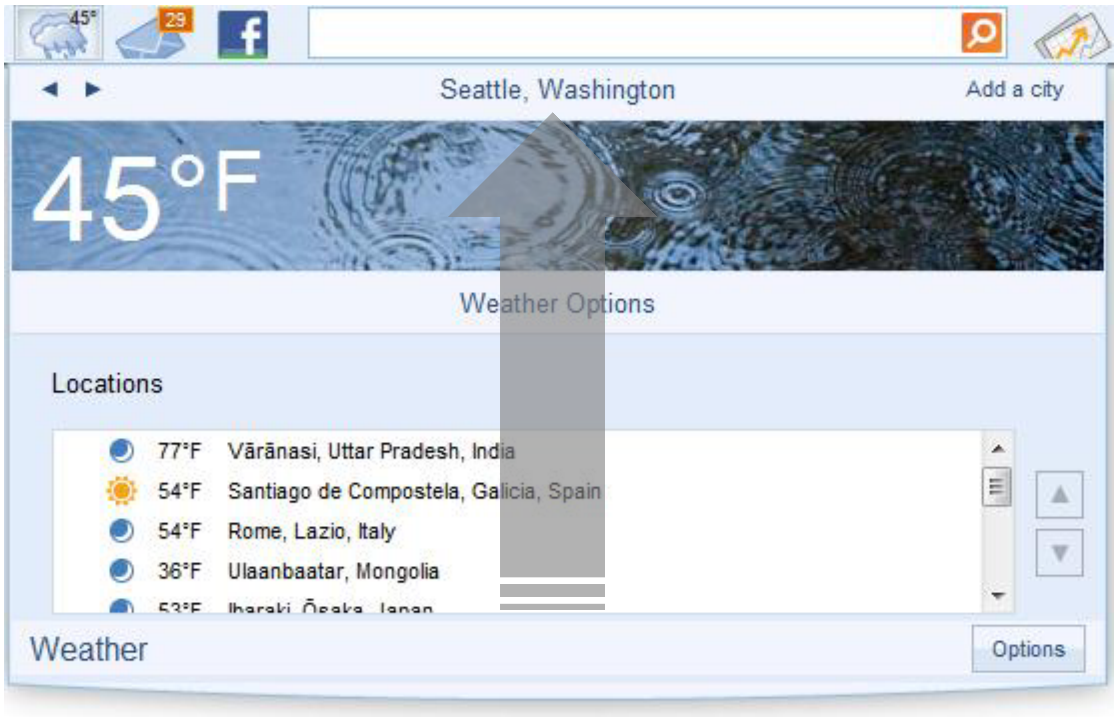

45°
29
Seattle, Washington
Add a city
45°F
Weather Options
Locations
77°F Vārānasi, Uttar Pradesh, India
54°F Santiago de Compostela, Galicia, Spain
54°F Rome, Lazio, Italy
36°F Ulaanbaatar, Mongolia
Weather
Options

Figure: Wing showing side direction when user presses Options button

If not used, stop polling

Design your app to notice when the user has not opened the app for an extended period. To save user and server resources in this case, stop periodic data polling. Periodic polling can resume when the user re-opens the app.

Avoid these design mistakes

When designing a Bing Bar app user interface, remember that the Bing Bar app wing is not a full-featured web browser. Many concepts that work within full browsers do not translate into useful user interfaces within Bing Bar apps. Try to avoid these design mistakes.

Don't add back buttons

Apps are not mini-web sites. Avoid navigation schemes that require a back button to return to a previous view in your app's wing.

Don't use breadcrumbs

Bread crumb navigation works on the web but is not appropriate for Bing Bar apps. Bing Bar apps should not have complex information architectures. A typical app consists of a single hub view supported by a few secondary views. In this example, the user might use a close button in the wing to return from a satellite view to the main hub view.

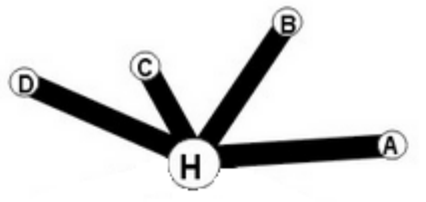


Figure: Hub View

These diagrams represent inappropriate wing UI structures.

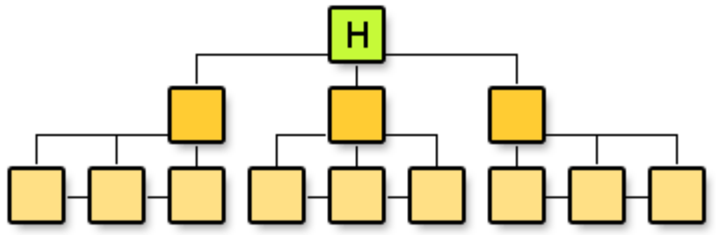


Figure: Hierarchy (avoid)

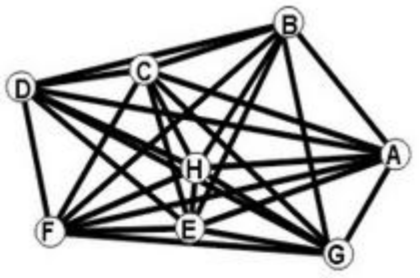


Figure: Complicated cross linking of UI views (avoid)

Limit wizards

If your app requires a wizard for initial configuration, consider these common sense guidelines:

- Avoid designing a wizard that requires more than two steps. Bing Bar apps are modeless and it's easy to a user to get distracted and click away, losing their place in the wizard.
- Try to limit each wizard panel to one logical task.
- Make the wizard a special case of your app's options panel or, if possible, part of the first-run experience so the user only encounters it once. Remember that most users don't invest a lot of time and focus into toolbars. Anything that reduces setup time and additional clicks helps ensure the success of your app.

Don't provide a multiple sets of tabs, a wall of buttons, or an "accordion" menu

The value of an app needs to be apparent the first time the user clicks the app's feature button. Avoid an interface design that presents the user with a list of features exposed via a tree control or an accordion menu control. Choose an initial view that is immediately engaging and as self-explanatory as possible. Don't present the user with a list of options or navigation choices that require interaction. Craft the initial experience and let the user find other options by necessity.

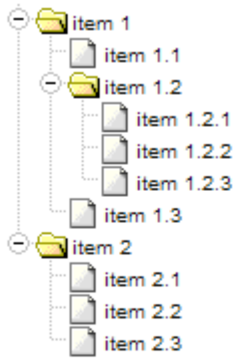


Figure: Accordion menu and tree control (avoid)

Don’t include splash screens

If possible, avoid splash screens that upsell the user on the app, especially if they require user interaction. An authentication screen may be an exception. You can show a splash or welcome screen the first time the user opens your app, but don’t show it every time. If your app fetches data when it launches, show cached data if possible, and inform the user that the app is fetching updates.

Don’t require horizontal scrollbars in app UI

Don’t show horizontal scrollbars for content whose width does not fit in the wing. If necessary, you can use vertical bars to show app content and UI, or try a carousel design where arrows at the left and right ends let the user spin through a palette of options.

Don’t use too many tabs in the header

If your app relies on multiple tabs to navigate between content (for example, a news app with many categories), avoid showing more than three tabs in the wing header. Multiple tabs might not scale well in the fixed-width wing without a horizontal overflow control, which can cause localization problems. Ideally, your app will avoid tabs altogether by determining a more engaging way to make your content navigable. You can also scale vertically by showing tabs vertically in the wing and specifying a taller wing height at design time.

Don’t use a pop-up modal dialog box

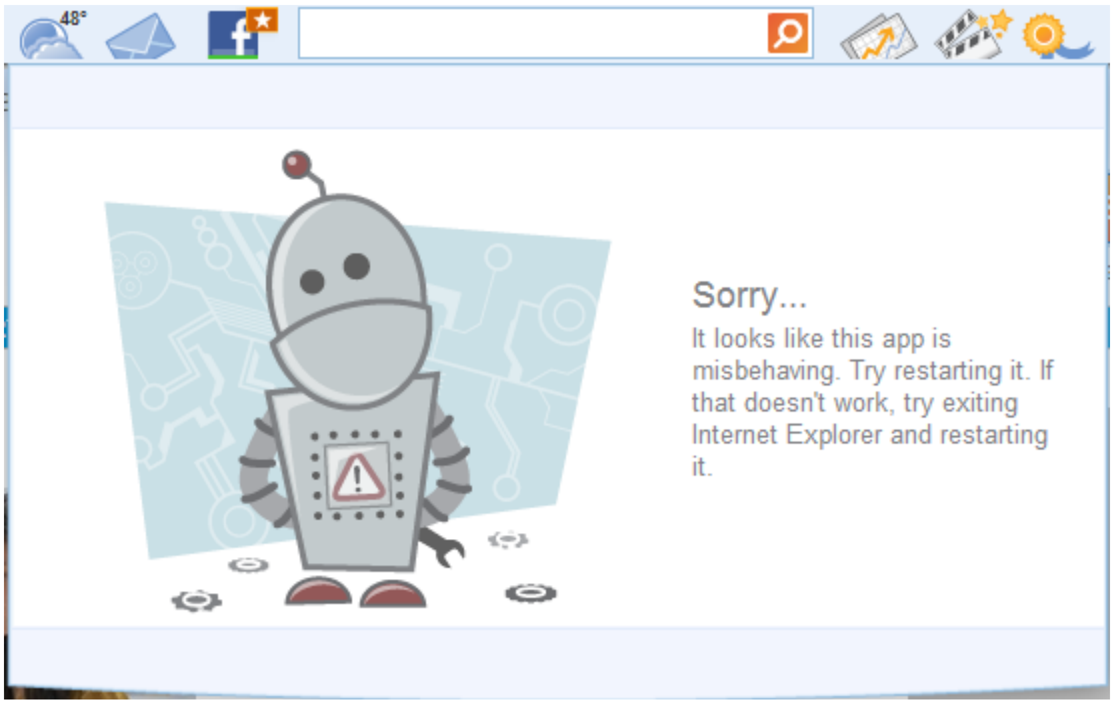
If your app requires a modal interaction, don't show a pop-up dialog box. Instead, use a slide-down pane as you see in the Options window for most apps.

Don't use an iFrame (except for local Flash)

It is possible to host an iFrame element to a remote web site in your wing, but the result won’t meet user expectations for Bing Bar apps. Bing Bar hosts highly engaging interactive apps. Pages hosted in an iFrame are less responsive, report errors inadequately, and face cross-domain calling restrictions. More importantly, Bing Bar is a client-side app platform, and should not be used as a miniature web browser for hosting web pages. You can use an iFrame to host a Flash app if all code and content for the Flash app resides on the local computer.

Don't show the sad robot if your app can handle an error

For unhandled exceptions, the Bing Bar shows a general error message called the sad robot. Your app should only show the sad robot if it cannot handle the error condition and show an appropriate message to the user within the app. Don't show the sad robot error for data retrieval failures, authentication errors, and other common error scenarios.



Sorry...

It looks like this app is misbehaving. Try restarting it. If that doesn't work, try exiting Internet Explorer and restarting it.

Figure: Sad robot error

Accessibility

Bing Bar apps attempt to comply with MATS (Microsoft Accessible Technology Standards) 1.0 Level 2. This standard meets WCAG 2.0 Level AA and all Section 508 requirements. Developers of apps that will not meet portions of the MATS 1.0 Level 2 standard must tell the Bing Bar team which standards they will not meet.

Accessibility standards require apps consider these details.

Support high contrast user interfaces

Examine user interfaces in Windows high-contrast black and high contrast white themes. If possible, avoid relying solely on images to convey vital information, as images might not be visible in high contrast modes. You don't have to provide high contrast versions of button icons and other symbol graphics, but avoid using graphics for arrow buttons and other essential navigation controls.

Accept default focus state

Because Bing Bar runs in Internet Explorer 7 compatibility mode, apps cannot use CSS pseudo-classes to override the appearance of a UI element's default focus state. Controls that have the keyboard focus include a dotted line some find unappealing. This visual cue of focus state helps impaired users to determine that a control has focus.

Provide keyboard access and test tab order

User interfaces should be navigable without a mouse using the Tab key, the Spacebar, Enter, and arrow keys. A tab sequence should start at the top of the UI and work its way down. Users should be able to use the arrow keys to navigate and use collections of similar UI elements, such as items in a list. Test all user interfaces to assure usability with little or no mouse activity.

Implement space bar to click

In Windows, pressing the space bar clicks a UI element, and pressing the Enter key only clicks the default button of a dialog. Bing Bar toolbar buttons and wing UIs are not dialog boxes and do not define a default command button. For buttons anywhere in Bing Bar, Enter should behave like the Spacebar to cause an element click. For other UI elements, the Spacebar is a click, while Enter has no effect.

Include component names for screen readers

UI elements should have component names. Component names enable a visually impaired user with a screen reader to navigate the UI. Keep the component names short. For example, call a refresh button *refresh* rather than *refresh button*. Bing Bar apps seek to simplify and expedite common tasks for all users, including visually impaired users. At minimum, use Microsoft Narrator to examine your app's narrated user experience.

Support content at high dots-per-inch settings

Many new computers are pre-set to 120% zoom to accommodate high resolution monitors. Your app should look good at 120% and 150%, and should operate at 200% without corruption. Images and other graphics can stretch, but avoid designs that will leave gaps when zoomed, and avoid anchoring content to one side of the UI.

Animated content over five seconds must be stoppable

If a wing contains animation that runs more than 5 seconds, give users the option to turn this animation off.

Internationalization and Marketization

Microsoft localizes Bing Bar for 66 language-market groups. You can specify which of these markets your app ships to, and you can customize your app in each market. Microsoft will translate text in your en-us (English-USA) locStrings.js file and appmanifest.xml file into translated files for the markets where you ship. Send email to bxcintl@microsoft.com to coordinate handoff and completion dates for translations. A complete list of language-market groups appears in *Appendix B: Language-Market Groups*.

Marketization

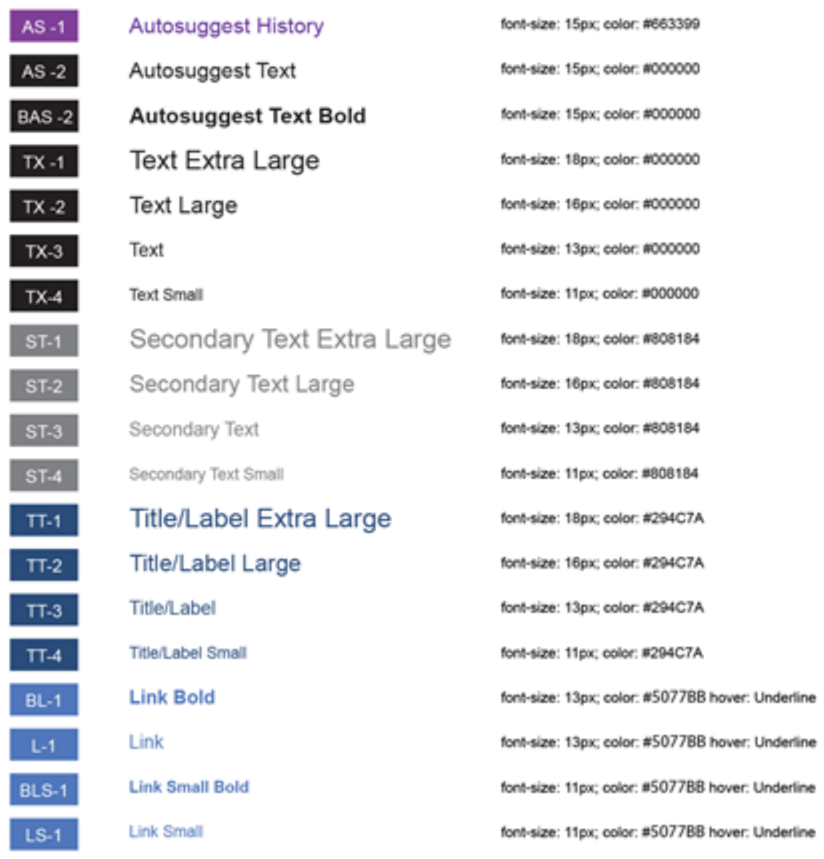
In addition to language translation, your app can localize settings for each market by providing localized settings.js files. For example, to specify a different URL for users in Japan, specify the URL in a key-value pair within settings.js, and provide a ja-jp/settings.js with a different URL in the key-value pair for Japanese users. Changes to settings.js files can be deployed more easily than app updates, so consider storing control parameters that might need tuning in your settings.js file.

Appendix A: Color Palettes

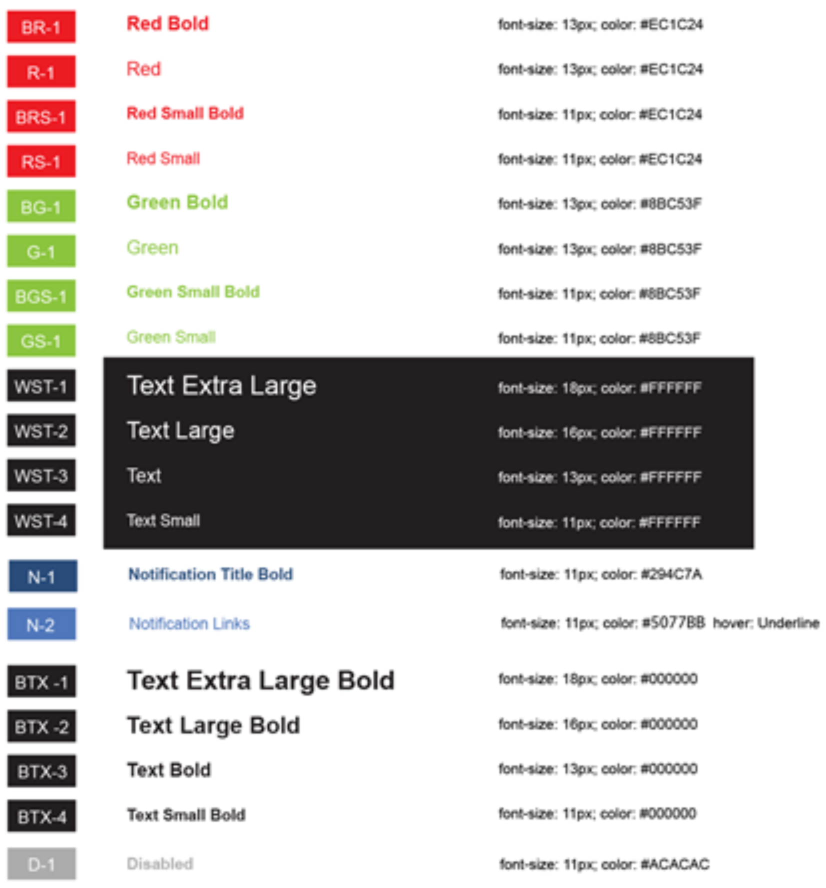
This appendix shows element, text, and button styles provided by bingclient.css.

Text styles

This figure shows CSS text styles provided by shared\css\bingclient.css.



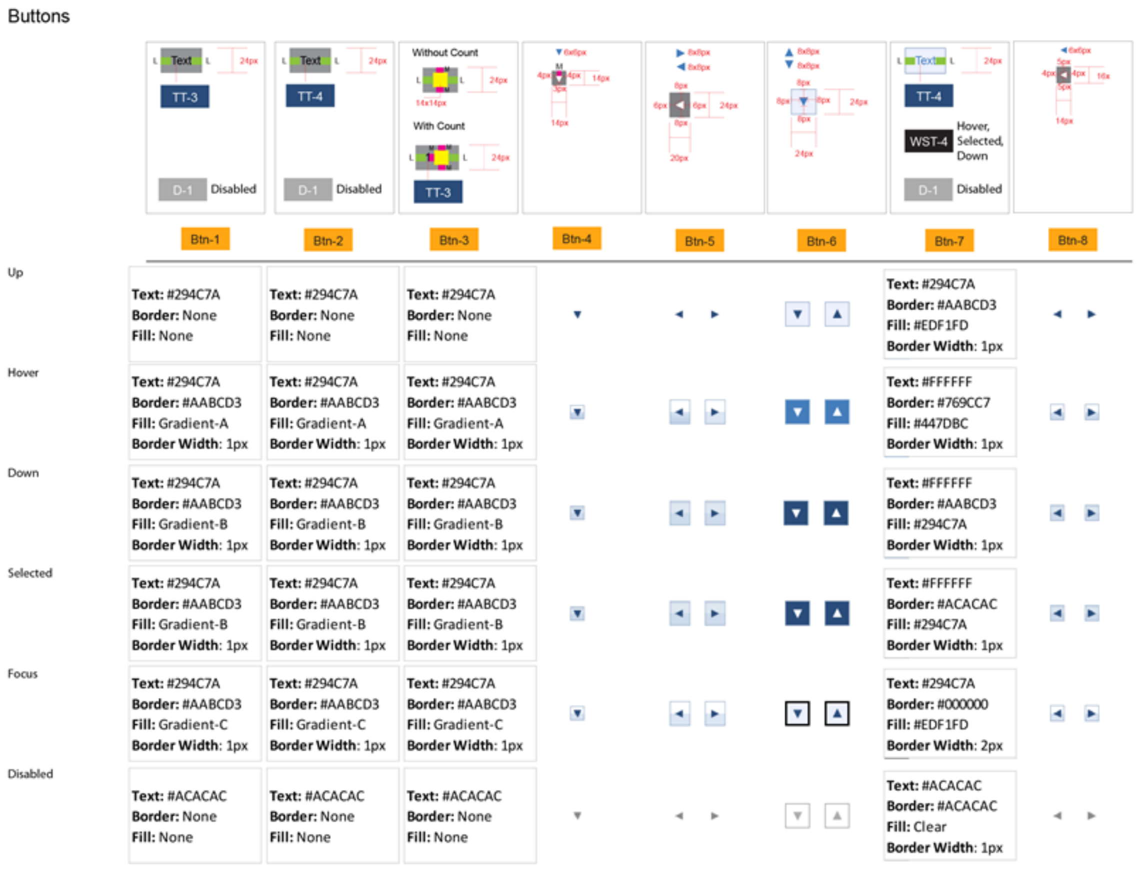
AS -1	Autosuggest History	font-size: 15px; color: #663399
AS -2	Autosuggest Text	font-size: 15px; color: #000000
BAS -2	**Autosuggest Text Bold**	font-size: 15px; color: #000000
TX -1	Text Extra Large	font-size: 18px; color: #000000
TX -2	Text Large	font-size: 16px; color: #000000
TX-3	Text	font-size: 13px; color: #000000
TX-4	Text Small	font-size: 11px; color: #000000
ST-1	Secondary Text Extra Large	font-size: 18px; color: #808184
ST-2	Secondary Text Large	font-size: 16px; color: #808184
ST-3	Secondary Text	font-size: 13px; color: #808184
ST-4	Secondary Text Small	font-size: 11px; color: #808184
TT-1	Title/Label Extra Large	font-size: 18px; color: #294C7A
TT-2	Title/Label Large	font-size: 16px; color: #294C7A
TT-3	Title/Label	font-size: 13px; color: #294C7A
TT-4	Title/Label Small	font-size: 11px; color: #294C7A
BL-1	**Link Bold**	font-size: 13px; color: #5077BB hover: Underline
L-1	Link	font-size: 13px; color: #5077BB hover: Underline
BLS-1	**Link Small Bold**	font-size: 11px; color: #5077BB hover: Underline
LS-1	Link Small	font-size: 11px; color: #5077BB hover: Underline



BR-1	Red Bold	font-size: 13px; color: #EC1C24
R-1	Red	font-size: 13px; color: #EC1C24
BRS-1	Red Small Bold	font-size: 11px; color: #EC1C24
RS-1	Red Small	font-size: 11px; color: #EC1C24
BG-1	Green Bold	font-size: 13px; color: #8BC53F
G-1	Green	font-size: 13px; color: #8BC53F
BGS-1	Green Small Bold	font-size: 11px; color: #8BC53F
GS-1	Green Small	font-size: 11px; color: #8BC53F
WST-1	Text Extra Large	font-size: 18px; color: #FFFFFF
WST-2	Text Large	font-size: 16px; color: #FFFFFF
WST-3	Text	font-size: 13px; color: #FFFFFF
WST-4	Text Small	font-size: 11px; color: #FFFFFF
N-1	Notification Title Bold	font-size: 11px; color: #294C7A
N-2	Notification Links	font-size: 11px; color: #5077BB hover: Underline
BTX -1	Text Extra Large Bold	font-size: 18px; color: #000000
BTX -2	Text Large Bold	font-size: 16px; color: #000000
BTX-3	Text Bold	font-size: 13px; color: #000000
BTX-4	Text Small Bold	font-size: 11px; color: #000000
D-1	Disabled	font-size: 11px; color: #ACACAC

Button styles

This figure shows CSS button styles provided by shared\css\bingclient.css.


Buttons
Text
TT-3
TT-4
D-1
Disabled
Without Count
With Count
WST-4
Hover, Selected, Down
24px
14px
16x

	Btn-1	Btn-2	Btn-3	Btn-4	Btn-5	Btn-6	Btn-7	Btn-8
Up	**Text:** #294C7A **Border:** None **Fill:** None	**Text:** #294C7A **Border:** None **Fill:** None	**Text:** #294C7A **Border:** None **Fill:** None	▼	◀ ▶	▼ ▲	**Text:** #294C7A **Border:** #AABCD3 **Fill:** #EDF1FD **Border Width**: 1px	◀ ▶
Hover	**Text:** #294C7A **Border:** #AABCD3 **Fill:** Gradient-A **Border Width**: 1px	**Text:** #294C7A **Border:** #AABCD3 **Fill:** Gradient-A **Border Width**: 1px	**Text:** #294C7A **Border:** #AABCD3 **Fill:** Gradient-A **Border Width**: 1px	▼	◀ ▶	▼ ▲	**Text:** #FFFFFF **Border:** #769CC7 **Fill:** #447DBC **Border Width**: 1px	◀ ▶
Down	**Text:** #294C7A **Border:** #AABCD3 **Fill:** Gradient-B **Border Width**: 1px	**Text:** #294C7A **Border:** #AABCD3 **Fill:** Gradient-B **Border Width**: 1px	**Text:** #294C7A **Border:** #AABCD3 **Fill:** Gradient-B **Border Width**: 1px	▼	◀ ▶	▼ ▲	**Text:** #FFFFFF **Border:** #AABCD3 **Fill:** #294C7A **Border Width**: 1px	◀ ▶
Selected	**Text:** #294C7A **Border:** #AABCD3 **Fill:** Gradient-B **Border Width**: 1px	**Text:** #294C7A **Border:** #AABCD3 **Fill:** Gradient-B **Border Width**: 1px	**Text:** #294C7A **Border:** #AABCD3 **Fill:** Gradient-B **Border Width**: 1px	▼	◀ ▶	▼ ▲	**Text:** #FFFFFF **Border:** #ACACAC **Fill:** #294C7A **Border Width**: 1px	◀ ▶
Focus	**Text:** #294C7A **Border:** #AABCD3 **Fill:** Gradient-C **Border Width**: 1px	**Text:** #294C7A **Border:** #AABCD3 **Fill:** Gradient-C **Border Width**: 1px	**Text:** #294C7A **Border:** #AABCD3 **Fill:** Gradient-C **Border Width**: 1px	▼	◀ ▶	▼ ▲	**Text:** #294C7A **Border:** #000000 **Fill:** #EDF1FD **Border Width**: 2px	◀ ▶
Disabled	**Text:** #ACACAC **Border:** None **Fill:** None	**Text:** #ACACAC **Border:** None **Fill:** None	**Text:** #ACACAC **Border:** None **Fill:** None	▼	◀ ▶	▼ ▲	**Text:** #ACACAC **Border:** #ACACAC **Fill:** Clear **Border Width**: 1px	◀ ▶

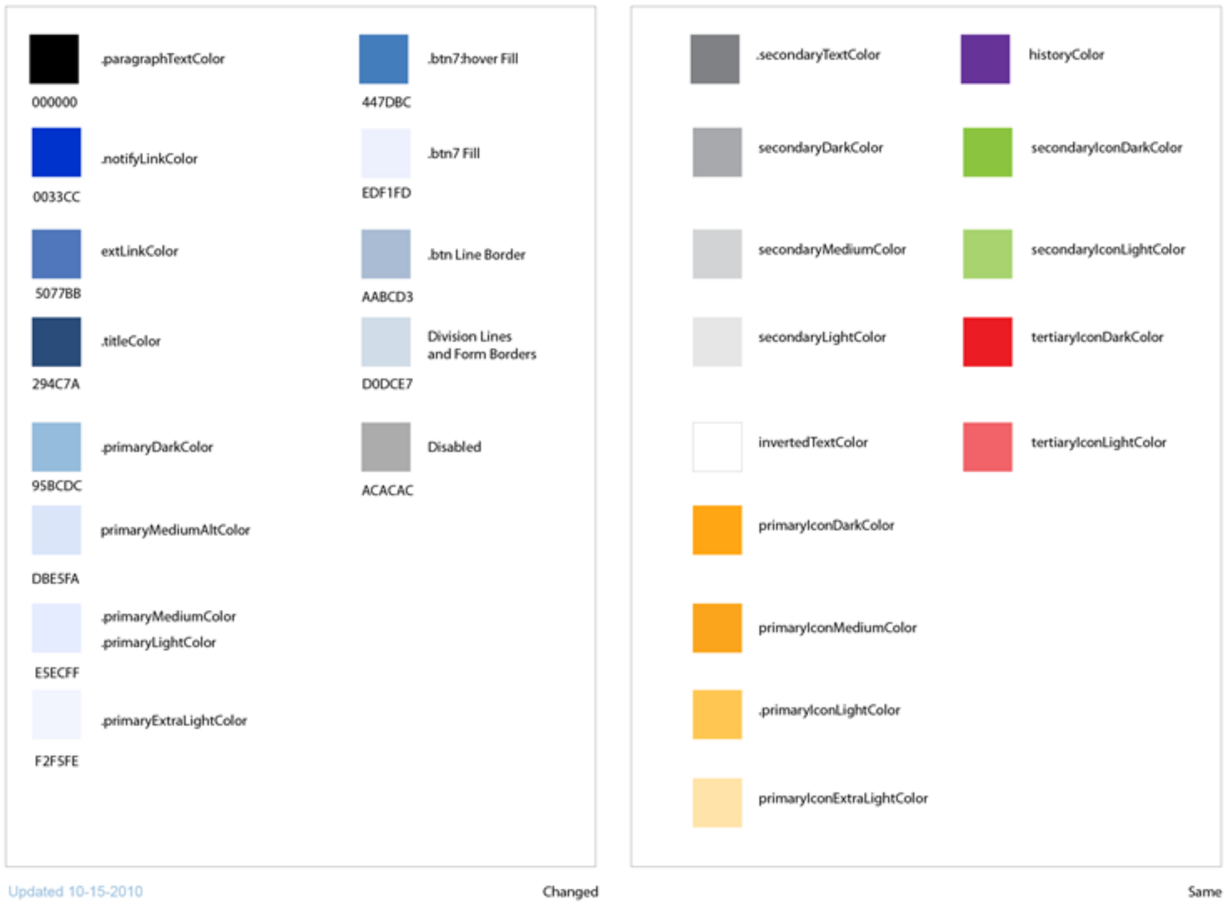
Color class name roots

This section lists valid class names for colors provided by shared\css\bingclient.css. Add a scope suffix—Color, Background, or Border—to form a color class name. See *Bing Bar 7 App Developer's Guide* for more information.

ExtLink, History, InvertedText, ParagraphText, PrimaryDark, PrimaryExtraLight, PrimaryIconDark, PrimaryIconExtraLight, PrimaryIconLight, PrimaryIconMedium, PrimaryLight, PrimaryMedium, SecondaryDark, SecondaryIconDark, SecondaryIconLight, SecondaryLight, SecondaryMedium, SecondaryText, TertiaryIconDark, TertiaryIconLight, Title

Colors assigned to class name roots in this release

This figure shows colors assigned to color class name roots in this release of Bing Bar.



Appendix B: Language-Market Groups

This appendix lists language-market groups where Bing Bar apps can designate availability and localize.

Language	**Country**	**Language-Market Code**
French	Arabia	fr-145
English	Arabia	en-145
Arabic (Complex Script/BiDi)	Arabia	ar-145
Spanish	Argentina	es-ar
English	Australia	en-au
German	Austria	de-at
Basque	Spain	eu-es
Dutch	Belgium	nl-be
French	Belgium	fr-be
Portuguese (Brazil)	Brazil	pt-br
Bulgarian	Bulgaria	bg-bg
English	Canada	en-ca
French	Canada	fr-ca
Catalan	Spain	ca-es
Spanish	Chile	es-cl
Chinese (Sim) (IME)	China	zh-cn
Croatian	Croatia	hr-hr
Czech	Czech	cs-cz

Danish	Denmark	da-dk
Estonian	Estonia	et-ee
Finnish	Finland	fi-fi
French	France	fr-fr
German	Germany	de-de
Greek	Greece	el-gr
Chinese (Trad) (IME)	Hong Kong	zh-hk
Hungarian	Hungary	hu-hu
English	India	en-in
Gujarati	India	gu-in
Hindi	India	hi-in
Kannada	India	kn-in
Malayalam	India	ml-in
Marathi	India	mr-in
Tamil	India	ta-in
Telugu	India	te-in
English	Indonesia	en-id
Indonesian	Indonesia	id-id
English	Ireland	en-ie
Hebrew (Comp Script/BiDi)	Israel	he-il
Italian	Italy	it-it
Japanese	Japan	ja-jp
Korean	Korea	ko-kr
Spanish	LATAM ROR	es-419
Latvian	Latvia	lv-lv
Lithuanian	Lithuanian	lt-lt
English	Malaysia	en-my
Malay (Malaysia - Brunei Darussalam)	Malaysia	ms-my
Spanish	Mexico	es-mx
Dutch	Netherlands	nl-nl
English	New Zealand	en-nz
Norwegian - Bokmål	Norway	nb-no
English	Philippines	en-ph
Polish	Poland	pl-pl
Portuguese (Portugal)	Portugal	pt-pt
Romanian	Romania	ro-ro
Russian	Russia	ru-ru
Serbian (Latin)	Serbia	sr-latn-cs
Serbian (Cyrillic)	Serbia	sr-cyrl-cs
English	Singapore	en-sg
Slovak	Slovakia	sk-sk
Slovenian	Slovenia	sl-si
English	South Africa	en-za
Spanish	Spain	es-es
Swahili	Kenya	sw-ke
Swedish	Sweden	sv-se
French	Switzerland	fr-ch
German	Switzerland	de-ch
Chinese (Trad) (IME)	Taiwan	zh-tw
Thai	Thailand	th-th

Turkish	Turkey	tr-tr
English	UK	en-gb
Ukranian	Ukraine	uk-ua
English	US	en-us
Spanish	US	es-us
Vietnamese	Vietnamese	vi-vn
English	World-wide - English	en-001
Spanish	World-wide -Spanish	es-001