CSC420

Page Layout

Basics

- Visual Hierarchy
  - The most important content should stand out the most
  - Whitespace, bolding (text), color (text, image)...
- Visual Flow
  - What should I look at next?
  - Focal points
  - Perceived meaning changes flow (I'm here to read the tiny text, not stare at the huge flash ad)
- Grouping and Alignment
  - Put things close (and align them) to indicate they are related
  - Humans like order (is symmetry beauty?...)

(some of the) Gestalt Principles

- Proximity
  - Humans associate things that are close
- Similarity
  - Humans associate things that are similar (shape, orientation, color...)
- Continuity
  - Human eyes tend to follow continuous “curves” built of smaller elements
- Closure
  - Humans like seeing simple closed forms (rectangles, circles...) and associate smaller elements whose alignment resembles those

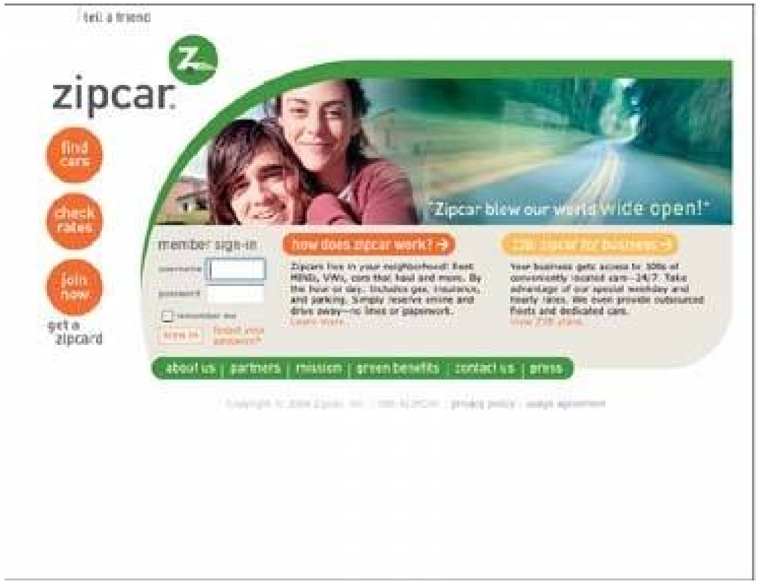
Dynamic Displays

- Gestalt is well-known and well-used for *static* content (billboards, posters, magazines, etc)
- Very little innovation in *dynamic* displays, despite the huge opportunities
- Dynamic is good:
  - Space usage
    - Scrollbars
    - Stacks, panes, etc
  - Provides another dimension (esp. with interactivity)
- Dynamic is bad:
  - Too little space (really a technology issue)
    - dpi
    - tradition of small screens
  - overused in a bad way (flashing red “CLICK ME” ads)

Page Layout Patterns

- Visual framework
- Center stage
- Titled sections
- Card stack
- Closable panels
- Movable panels
- Right/left alignment
- Diagonal balance
- Property sheet
- Responsive disclosure
- Responsive enabling
- Liquid layout

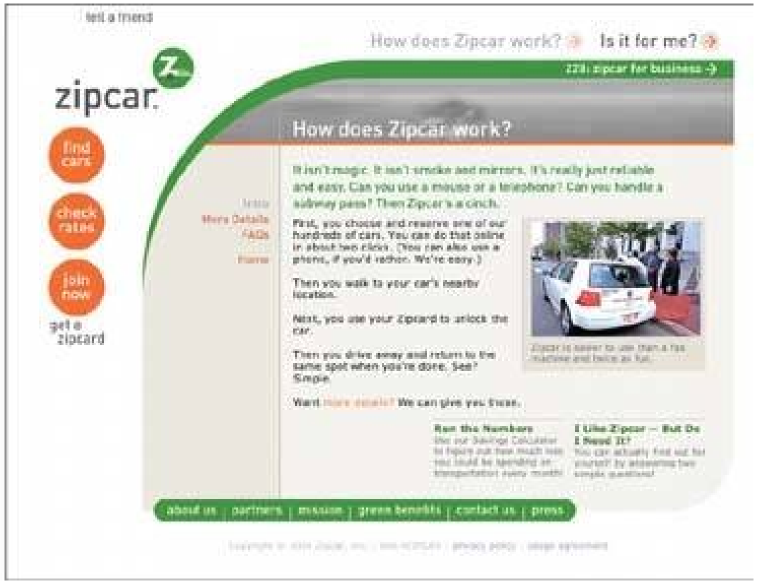
Visual Framework



Zipcars live in your neighborhood! Rent MINIs, VWs, cars that haul and more. By the hour or day. Includes gas, insurance, and parking. Simply reserve online and drive away—no lines or paperwork. Learn More

Your business gets access to 300s of conveniently located cars—24/7. Take advantage of our special weekday and hourly rates. We even provide outsourced fleets and dedicated cars. View Z2B plans

Copyright © 2006 Zipcar, Inc. | 1-866-4ZIPCAR | privacy policy | usage agreement



How does Zipcar work?

It isn't magic. It isn't smoke and mirrors. It's really just reliable and easy. Can you use a mouse or a telephone? Can you handle a subway pass? Then Zipcar's a cinch.

First, you choose and reserve one of our hundreds of cars. You can do that online in about two clicks. (You can also use a phone, if you'd rather. We're easy.)

Then you walk to your car's nearby location.

Next, you use your Zipcard to unlock the car.

Then you drive away and return to the same spot when you're done. See? Simple.

Want more details? We can give you those.

Zipcar is easier to use than a fax machine and twice as fun.

Run the Numbers
Use our Savings Calculator to figure out how much less you could be spending on transportation every month!

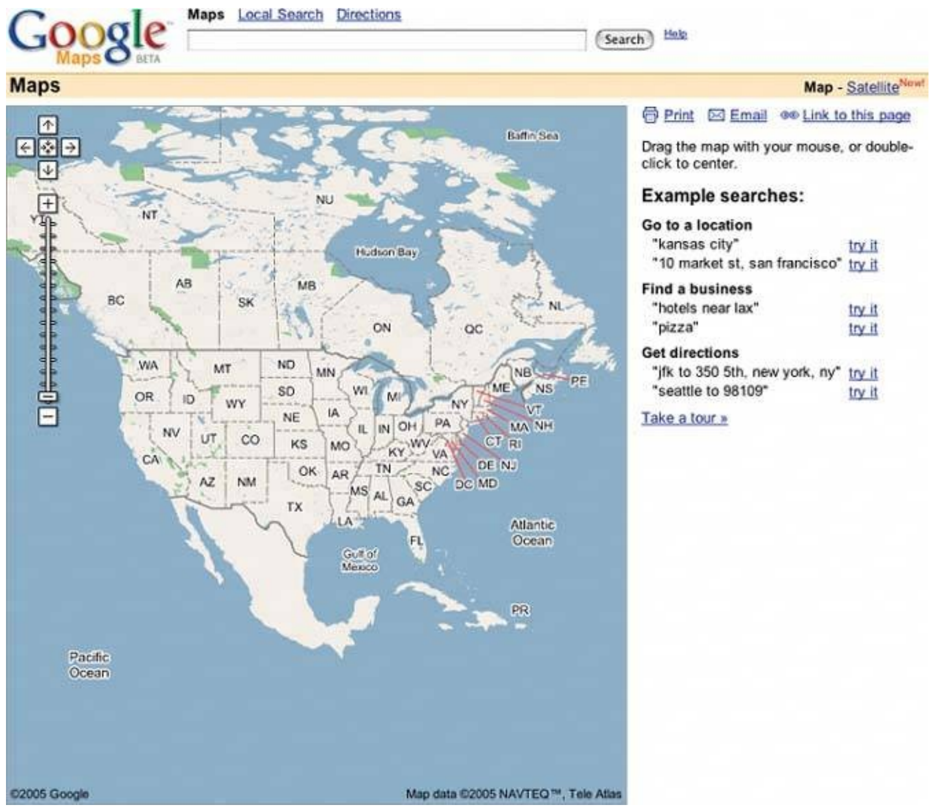
I Like Zipcar — But Do I Need It?
You can actually find out for yourself by answering two simple questions!

Copyright © 2006 Zipcar, Inc. | 1-866-4ZIPCAR | privacy policy | usage agreement

- Design each page with the same basic layout (colors, style, etc) but allow for varying page content
- Consistency matters for navigation (and is generally good: consistency == order)

Center Stage

- Put the most important part of the UI into the largest subsection of the page or window; cluster secondary tools and content around it in smaller panels
- Leads to Clear Entry Points
- Establishes the purpose of the UI


Google Maps BETA
Maps Local Search Directions
Search
Help
Maps
Map - Satellite New!
Print Email Link to this page
Take a tour »
©2005 Google
Map data ©2005 NAVTEQ™, Tele Atlas

Drag the map with your mouse, or double-click to center.

Example searches:

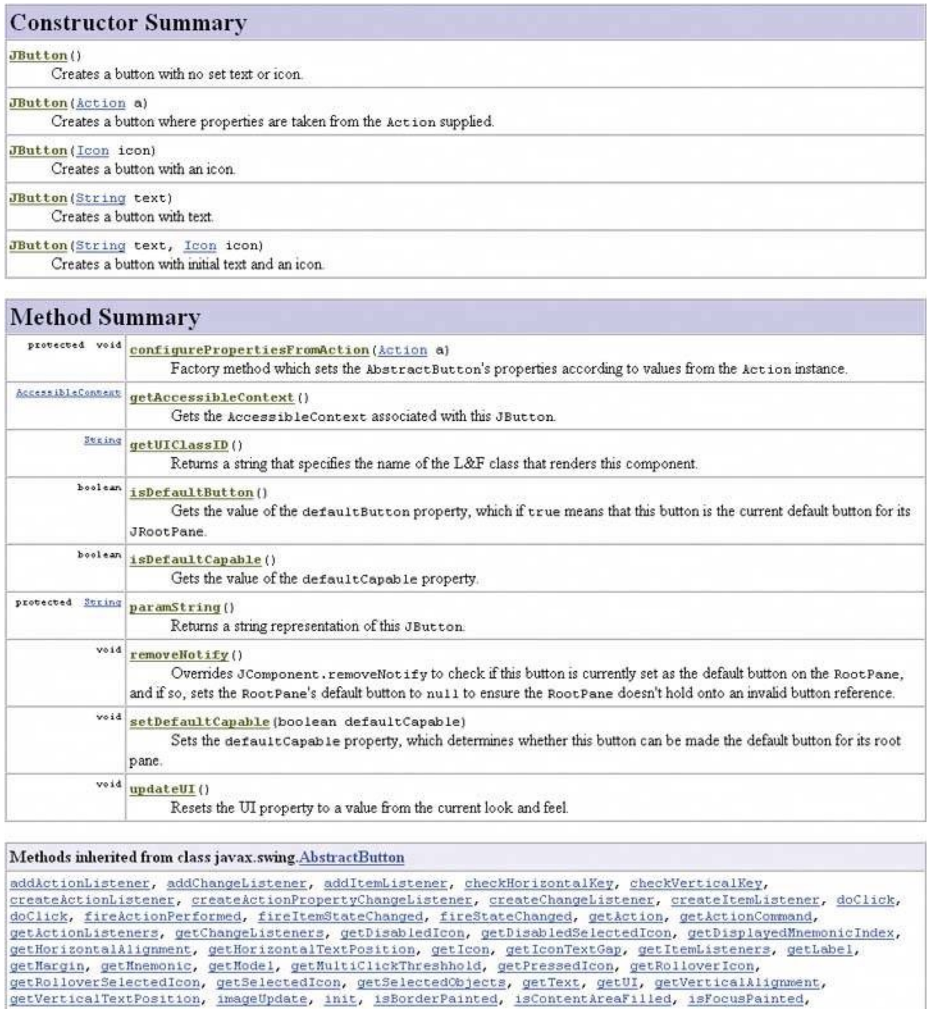
Go to a location
"kansas city" try it
"10 market st, san francisco" try it

Find a business
"hotels near lax" try it
"pizza" try it

Get directions
"jfk to 350 5th, new york, ny" try it
"seattle to 98109" try it

Titled Sections

- Define separate sections of content by giving each one a visually strong title, and then laying them all out on the page together
- Makes info architecture obvious
- Guides the eye (which naturally looks for bigger patterns)



Constructor Summary
`JButton()` Creates a button with no set text or icon.
`JButton(Action a)` Creates a button where properties are taken from the `Action` supplied.
`JButton(Icon icon)` Creates a button with an icon.
`JButton(String text)` Creates a button with text.
`JButton(String text, Icon icon)` Creates a button with initial text and an icon.

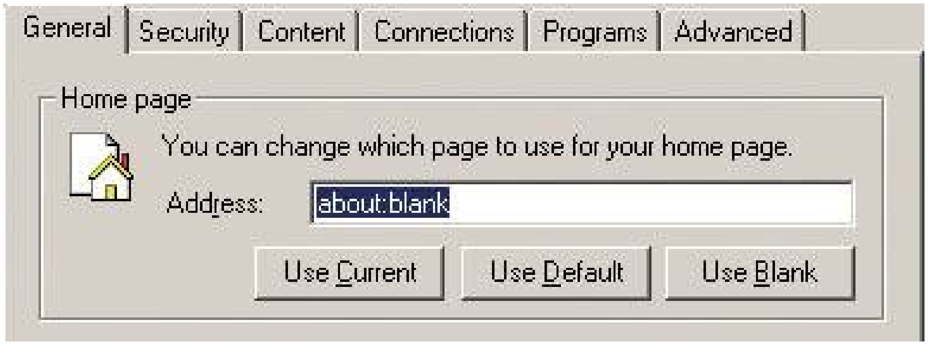
Method Summary	
`protected void`	`configurePropertiesFromAction(Action a)` Factory method which sets the `AbstractButton`'s properties according to values from the `Action` instance.
`AccessibleContext`	`getAccessibleContext()` Gets the `AccessibleContext` associated with this `JButton`.
`String`	`getUIClassID()` Returns a string that specifies the name of the L&F class that renders this component.
`boolean`	`isDefaultButton()` Gets the value of the `defaultButton` property, which if `true` means that this button is the current default button for its `JRootPane`.
`boolean`	`isDefaultCapable()` Gets the value of the `defaultCapable` property.
`protected String`	`paramString()` Returns a string representation of this `JButton`.
`void`	`removeNotify()` Overrides `JComponent.removeNotify` to check if this button is currently set as the default button on the `RootPane`, and if so, sets the `RootPane`'s default button to `null` to ensure the `RootPane` doesn't hold onto an invalid button reference.
`void`	`setDefaultCapable(boolean defaultCapable)` Sets the `defaultCapable` property, which determines whether this button can be made the default button for its root pane.
`void`	`updateUI()` Resets the UI property to a value from the current look and feel.

Methods inherited from class javax.swing.AbstractButton

`addActionListener, addChangeListener, addItemListener, checkHorizontalKey, checkVerticalKey, createActionListener, createActionPropertyChangeListener, createChangeListener, createItemListener, doClick, doClick, fireActionPerformed, fireItemStateChanged, fireStateChanged, getAction, getActionCommand, getActionListeners, getChangeListeners, getDisabledIcon, getDisabledSelectedIcon, getDisplayedMnemonicIndex, getHorizontalAlignment, getHorizontalTextPosition, getIcon, getIconTextGap, getItemListeners, getLabel, getMargin, getMnemonic, getModel, getMultiClickThreshhold, getPressedIcon, getRolloverIcon, getRolloverSelectedIcon, getSelectedIcon, getSelectedObjects, getText, getUI, getVerticalAlignment, getVerticalTextPosition, imageUpdate, init, isBorderPainted, isContentAreaFilled, isFocusPainted,`

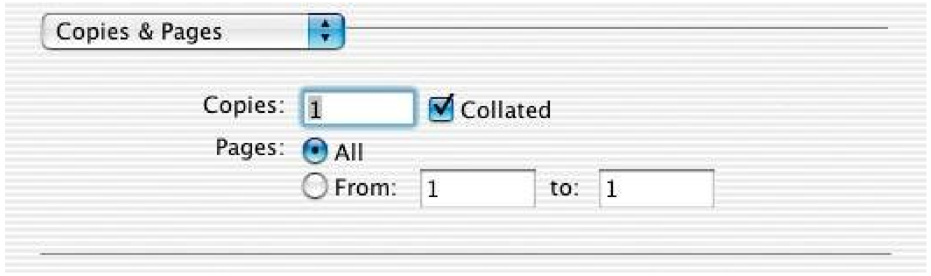
Card Stack

- Put sections of content onto separate panels or "cards," and stack them up so only one is visible at a time; use tabs or other devices to give users access to them
- Tabs are very familiar (== “user-intuitive”)
- Structures content into easily digestible chunks



You can change which page to use for your home page.

Address: about:blank

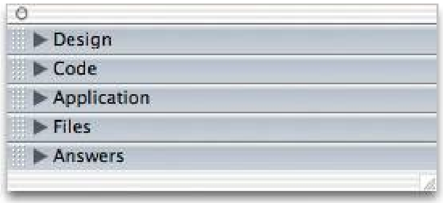


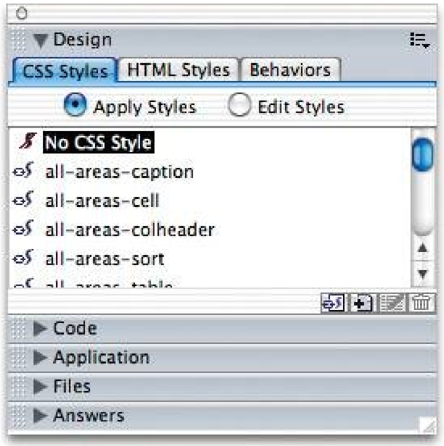
Copies: 1 [✓ Collated]

Pages: [✓ All]

From: 1 to: 1

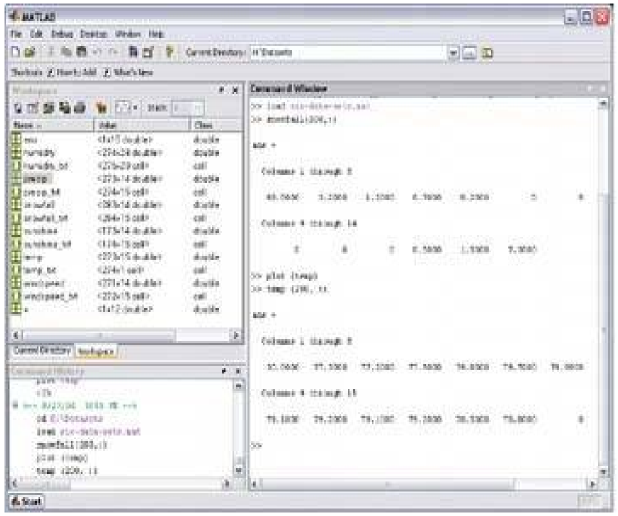
Closable Panels

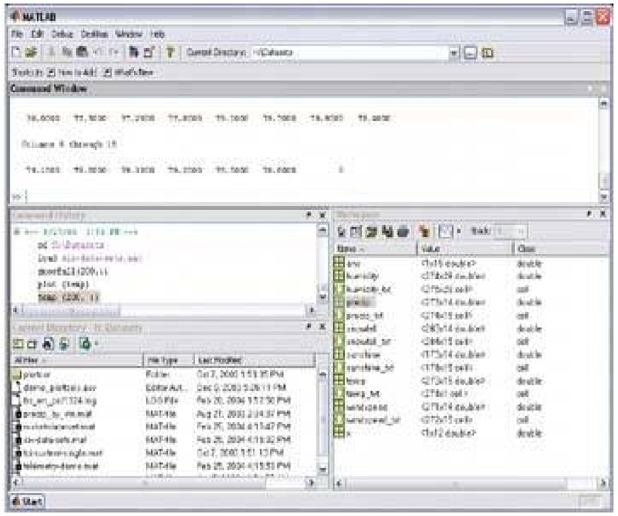




- Put sections of content onto separate panels, and let the user open and close each of them separately from the others
- Too much stuff; users might want more than one at a time
- Extras on Demand
- Cost: not very familiar

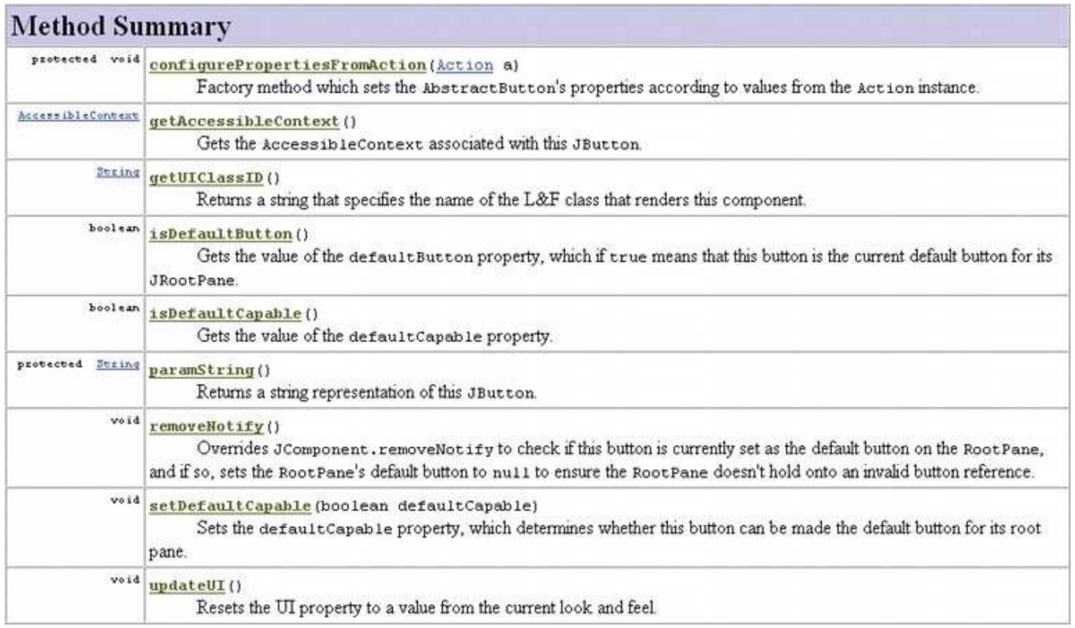
Movable Panels





- Put different tools or sections of content onto separate panels, and let the user move them around to form a custom layout
- Use when sections are self-evident and won't benefit (much) from a rigid layout
- Allows users to assert their preferences

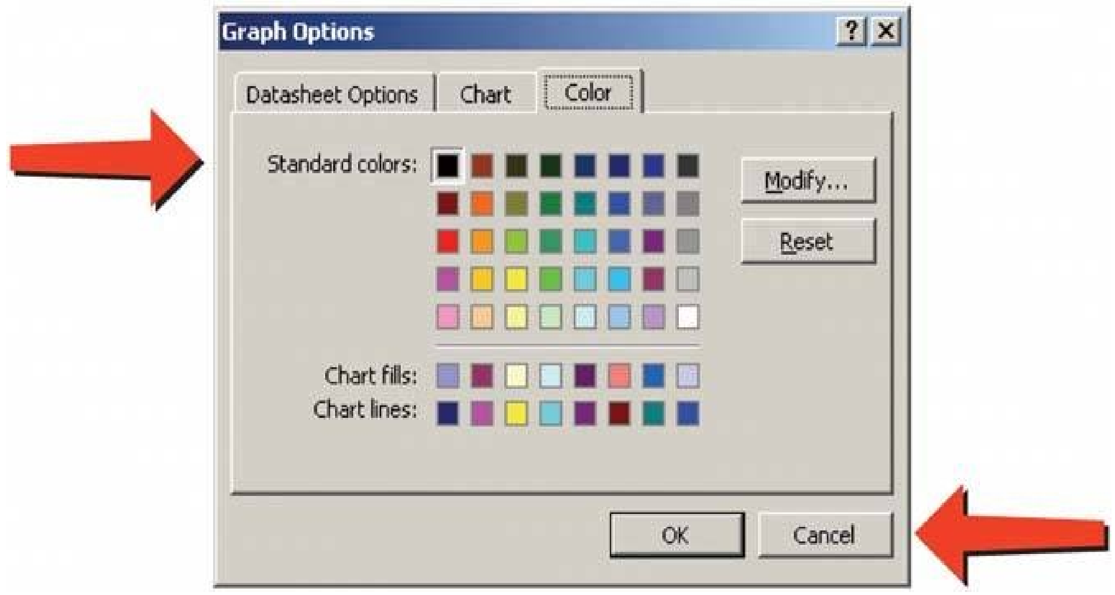
Right/Left Alignment



Method Summary	
protected void	configurePropertiesFromAction(Action a) Factory method which sets the AbstractButton's properties according to values from the Action instance.
AccessibleContext	getAccessibleContext() Gets the AccessibleContext associated with this JButton.
String	getUIClassID() Returns a string that specifies the name of the L&F class that renders this component.
boolean	isDefaultButton() Gets the value of the defaultButton property, which if true means that this button is the current default button for its JRootPane.
boolean	isDefaultCapable() Gets the value of the defaultCapable property.
protected String	paramString() Returns a string representation of this JButton.
void	removeNotify() Overrides JComponent.removeNotify to check if this button is currently set as the default button on the RootPane, and if so, sets the RootPane's default button to null to ensure the RootPane doesn't hold onto an invalid button reference.
void	setDefaultCapable(boolean defaultCapable) Sets the defaultCapable property, which determines whether this button can be made the default button for its root pane.
void	updateUI() Resets the UI property to a value from the current look and feel.

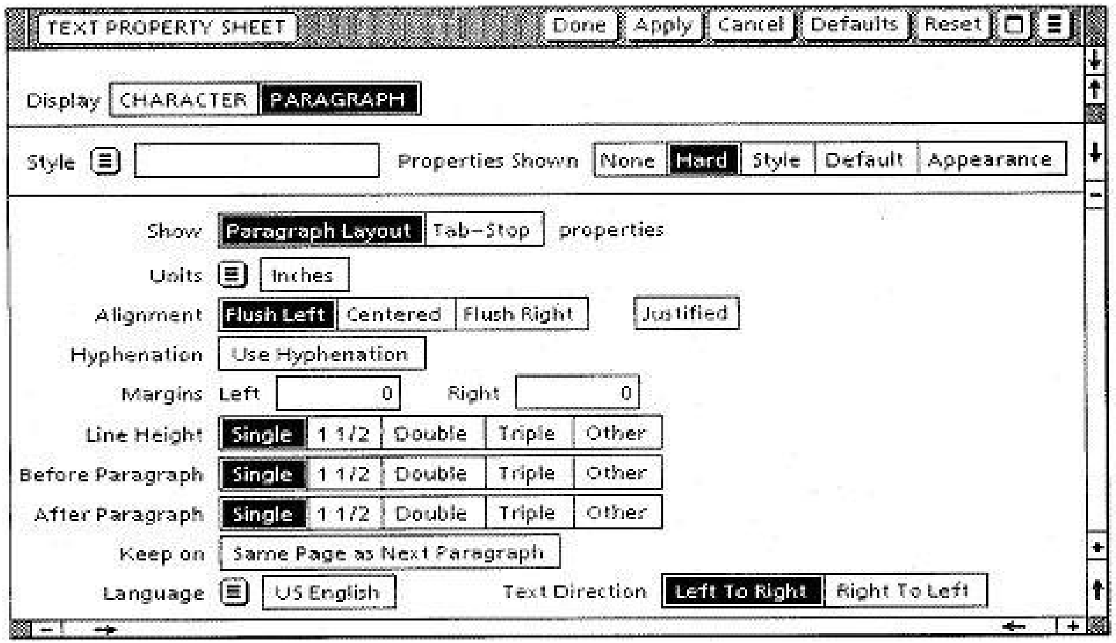
- When designing a two-column form or table, right-align the labels on the left, and left-align the items on the right
- Proximity (labels with controls); continuity (line down the middle)

Diagonal Balance



- Arrange page elements in an asymmetric fashion, but balance it by putting visual weight into both the upper-left and lower-right corners
- Beauty (symmetry); easy movement of the eyes (for left-to-right languages!)

Property Sheet



TEXT PROPERTY SHEET — Done | Apply | Cancel | Defaults | Reset

Display: CHARACTER [✓ PARAGRAPH]

Style: | Properties Shown: None [✓ Hard] Style Default Appearance

Show: [✓ Paragraph Layout] Tab-Stop properties

Units: Inches

Alignment: [✓ Flush Left] Centered Flush Right Justified

Hyphenation: Use Hyphenation

Margins: Left 0 Right 0

Line Height: [✓ Single] 1 1/2 Double Triple Other

Before Paragraph: [✓ Single] 1 1/2 Double Triple Other

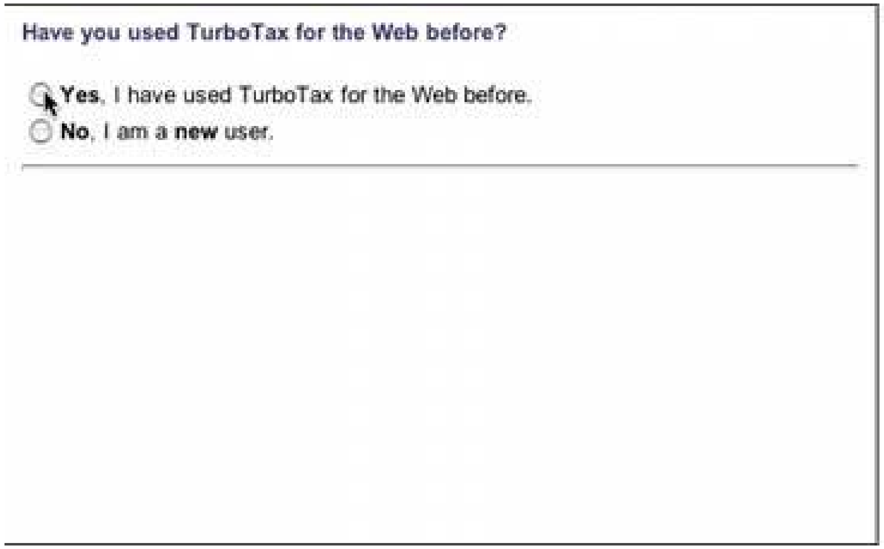
After Paragraph: [✓ Single] 1 1/2 Double Triple Other

Keep on: Same Page as Next Paragraph

Language: US English | Text Direction: [✓ Left To Right] Right To Left

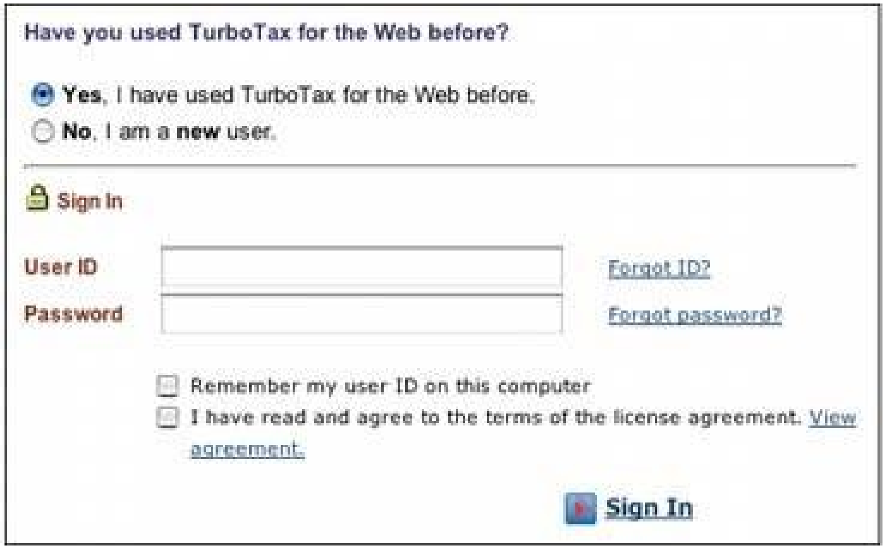
- Use a two-column or form-style layout to show the user that an object's properties are edited on this page
- Familiar; useful for mixing WYSIWYG with programming; helps users build a mental model of an object

Responsive Disclosure



Have you used TurboTax for the Web before?

- [ ] **Yes**, I have used TurboTax for the Web before.
- [ ] **No**, I am a **new** user.



Have you used TurboTax for the Web before?

- [✓] **Yes**, I have used TurboTax for the Web before.
- [ ] **No**, I am a **new** user.

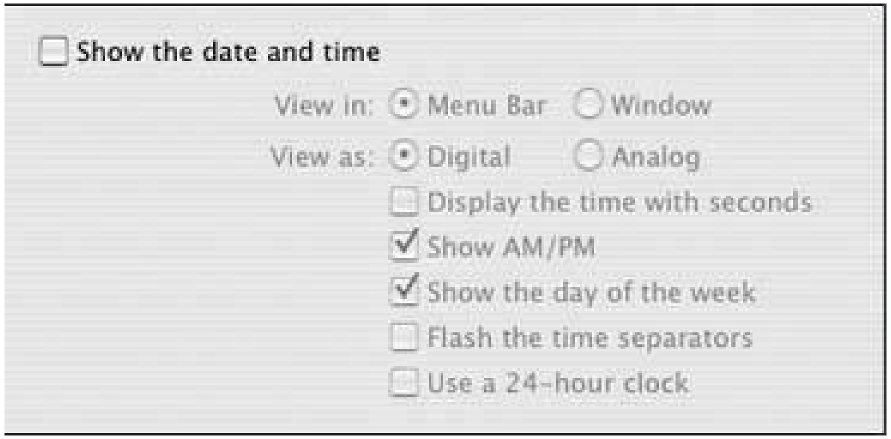
Sign In

User ID		Forgot ID?
Password		Forgot password?

- [ ] Remember my user ID on this computer
- [ ] I have read and agree to the terms of the license agreement. View agreement.

- Starting with a very minimal UI, guide a user through a series of steps by showing more of the UI as he completes each step
- No context switches necessary (as in e.g., wizards) – everything unfolds on a single page

Responsive Enabling (Disabling)


Show the date and time

View in: [✓ Menu Bar] Window

View as: [✓ Digital] Analog

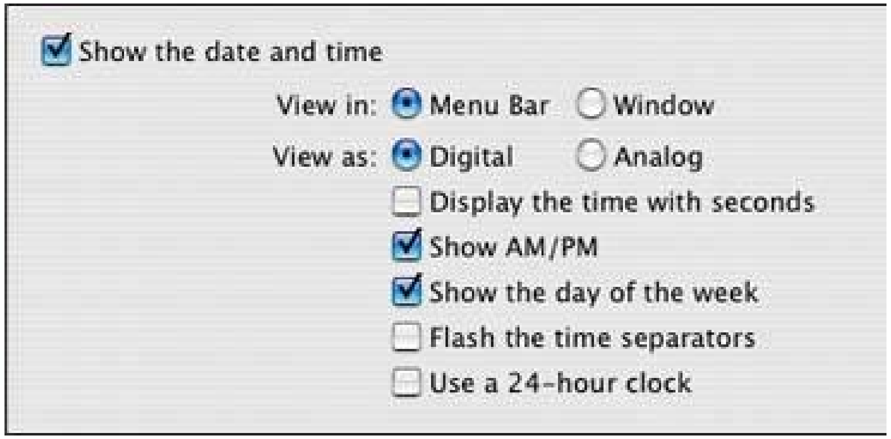
Display the time with seconds

[✓ Show AM/PM]

[✓ Show the day of the week]

Flash the time separators

Use a 24-hour clock


[✓ Show the date and time]

View in: [✓ Menu Bar] Window

View as: [✓ Digital] Analog

Display the time with seconds

[✓ Show AM/PM]

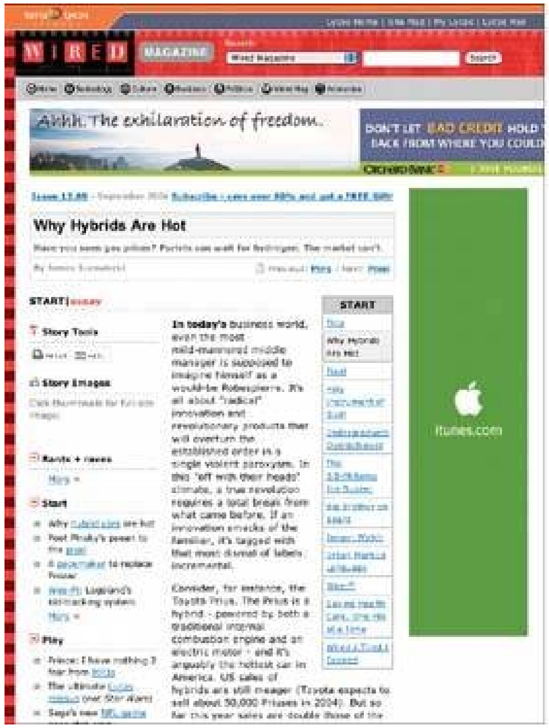
[✓ Show the day of the week]

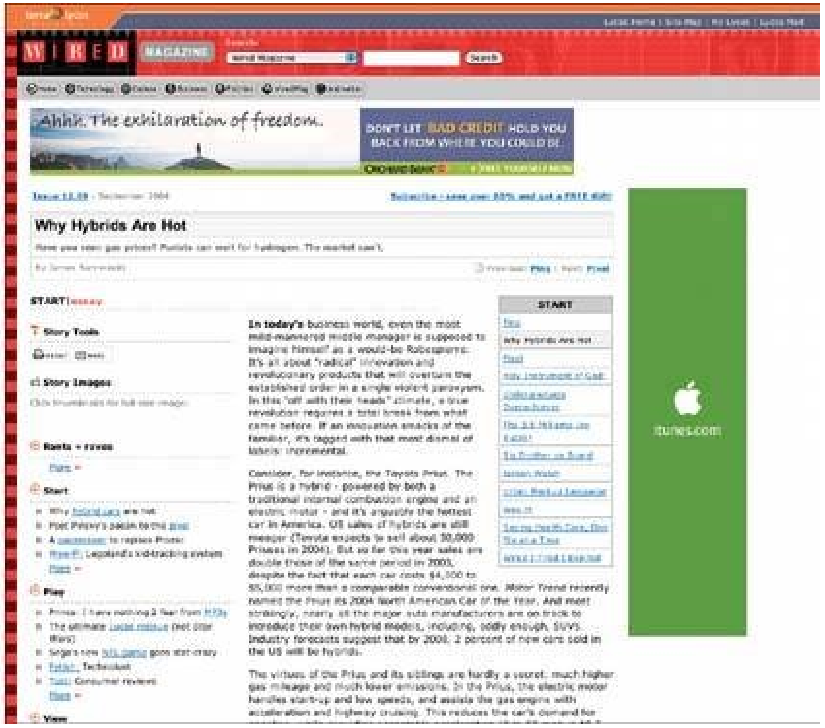
Flash the time separators

Use a 24-hour clock

- Starting with a UI that's mostly disabled, guide a user through a series of steps by enabling more of the UI as each step is done
- Allows the user to form a cause-and-effect model of the interface
- Unnecessary error messages are avoided (by locking out options)

Liquid Layout





- As the user resizes the window, resize the page contents along with it so the page is constantly filled
- Allows users to exert their preferences on the layout (page size, fonts, etc) without changing its flow