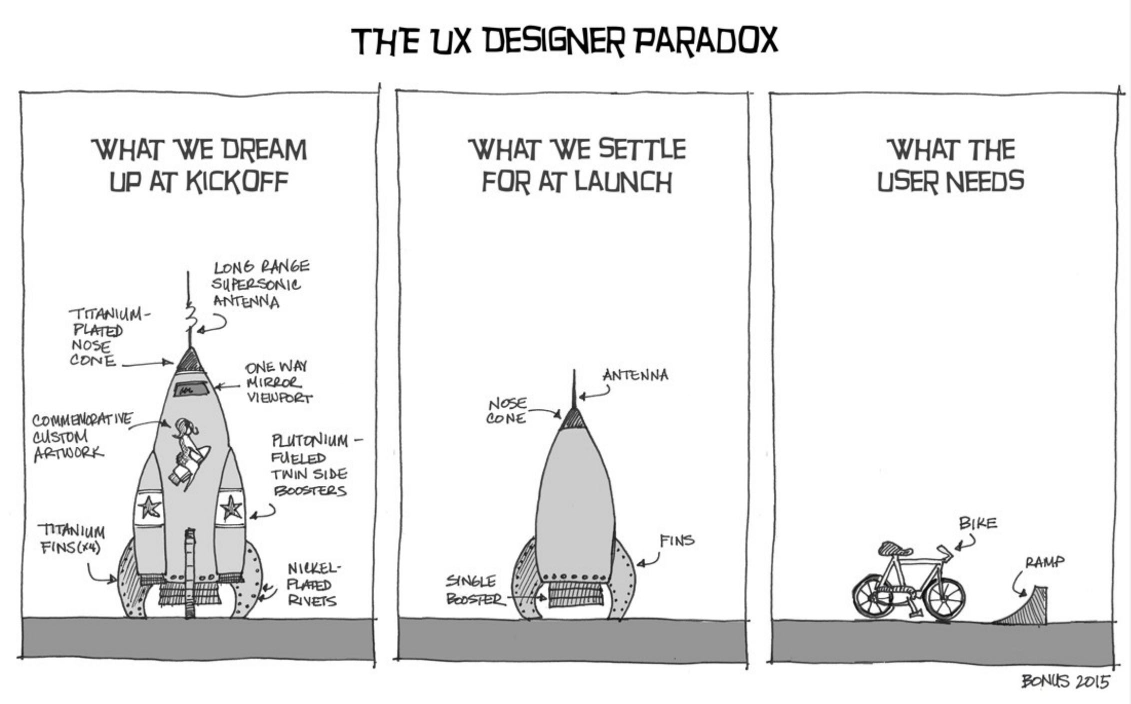
CSEN603 – Software Engineering

Lecture 5: Usability

Mervat Abuelkheir
Ammar Yasser
Mohamed Agamia
Nada Hisham

- Definition
- Design Models
- Design Patterns
- UX/UI


THE UX DESIGNER PARADOX
WHAT WE DREAM UP AT KICKOFF
LONG RANGE SUPERSONIC ANTENNA
TITANIUM-PLATED NOSE CONE
ONE WAY MIRROR VIEWPORT
COMMEMORATIVE CUSTOM ARTWORK
PLUTONIUM-FUELED TWIN SIDE BOOSTERS
TITANIUM FINS (x4)
NICKEL-PLATED RIVETS
WHAT WE SETTLE FOR AT LAUNCH
ANTENNA
NOSE CONE
FINS
SINGLE BOOSTER
WHAT THE USER NEEDS
BIKE
RAMP
BONUS 2015

Software Engineering

- Architecture
- Requirements Engineering
- Design and Design Patterns
- Implementation
- Verification and Validation
- Quality and Maintenance
- Scale and Evolution
- Economics

Process, Models, Methods

Project Hints – How to Build a Good Use Case

1. **Identify actors and their goals**
2. **Write success scenario or “happy storyline”**
   - Identify when use case begins (trigger) and when it ends (goal achieved)
   - Actor’s intent (goal), actor’s responsibility, path from trigger to goal
3. **List the failure extensions**
   - And describe failure handling
4. List alternative scenarios if applicable

Project Hints – Personas, User Stories/Cases

- PERSONAS → Different Users, Attitudes, and Expected Behaviors
- USER STORIES → capture a user’s needs or what they do to complete a job/task. Describe who it is for, what the desired functionality is, and why it is useful

“As a [User Role], I want [Function/Feature],

so that [benefit from implementing feature]”

- INDEPENDENT → Each user story should be as independent as possible
- SMALL → Keep it short and concise
- VALUABLE → Valuable to the user/owner of the solution. Should be features, not tasks

Project Hints – Qualities of a Good User Story and its Scenario

- starts with a request from an actor to the system
- ends with the production of all the answers to the request
- defines the interactions (between system and actors) related to the function
- takes into account the actor's point of view, not the system's
- focuses on interaction, not internal system activities
- defines if data is needed from system or if data needs to be stored to system
- doesn't describe the GUI in detail
- has 3-9 steps in the main success scenario (success means goal of use case is achieved)
  - e.g. how to get list of MSc students
- is easy to read
- summary fits on a page

Prelude to Usability

The tragic life of Clippy, the world's most hated virtual assistant

"It looks like you're writing a letter. Would you like help?"

- Born in Office 97, Clippy, with its "Groucho eyebrows", politely offered hints for using Microsoft's Office software
- The program was widely reviled among users as intrusive and annoying
- Smithsonian Magazine called Clippy: "one of the worst software design blunders in the annals of computing"
- Time magazine included Clippy in a 2010 article listing the fifty worst inventions

http://content.time.com/time/specials/packages/article/0,28804,1991915_1991909_1991902,00.html

AirBnB vs. CouchSurfing

- Airbnb provides a marketplace for people to rent their homes
- On the main interface of the Airbnb listing, users can see prices of homes
- Much of the space on interface is used to describe the features of the homes (e.g. the space, availability, safety features)
- Airbnb encourages a host to upload lots of pictures of their homes
- Users can see the neighborhood of the place
- Interface reflects/reinfoces the idea that Airbnb is a platform for people to find places to stay over

- CouchSurfing targets building a community of travelers
- CouchSurfing has an interface for hosts to describe their homes
- Much of space on interface is used to describe host's characteristics (e.g. age, gender, languages, degree, education, birthplace)
- CouchSurfing allows the host to provide more details about themselves (e.g. interests, one amazing thing host has done)
- Users can see city level location information, not the whole address or neighborhood of the place (makes users pay less attention to location and more to hosts)
- Interface reflects/reinforces the idea that CouchSurfing is about people, not places

Good interface design can influence how people interact with each other on these systems

The User Interface is Important

- User interface **strongly affects perception of software**
  - Usable software sells better
  - Unusable web sites are abandoned
- Perception is sometimes superficial
  - Users blame themselves for UI failings
  - People who make buying decisions are not always end-users

But UIs are Hard to Design

- You are not the user
  - Most software engineering is about communicating with other programmers
  - UI is about communicating with users
- **The user is always right**
  - Consistent problems are the system's fault
- ...but the user is not always right
  - Users aren't designers

User Interfaces are Hard to Build takes a lot of software development effort

UI accounts for ~50% of design time, implementation time maintenance time, and code size

Usability Defined

Usability → How well can users use software's functionality?

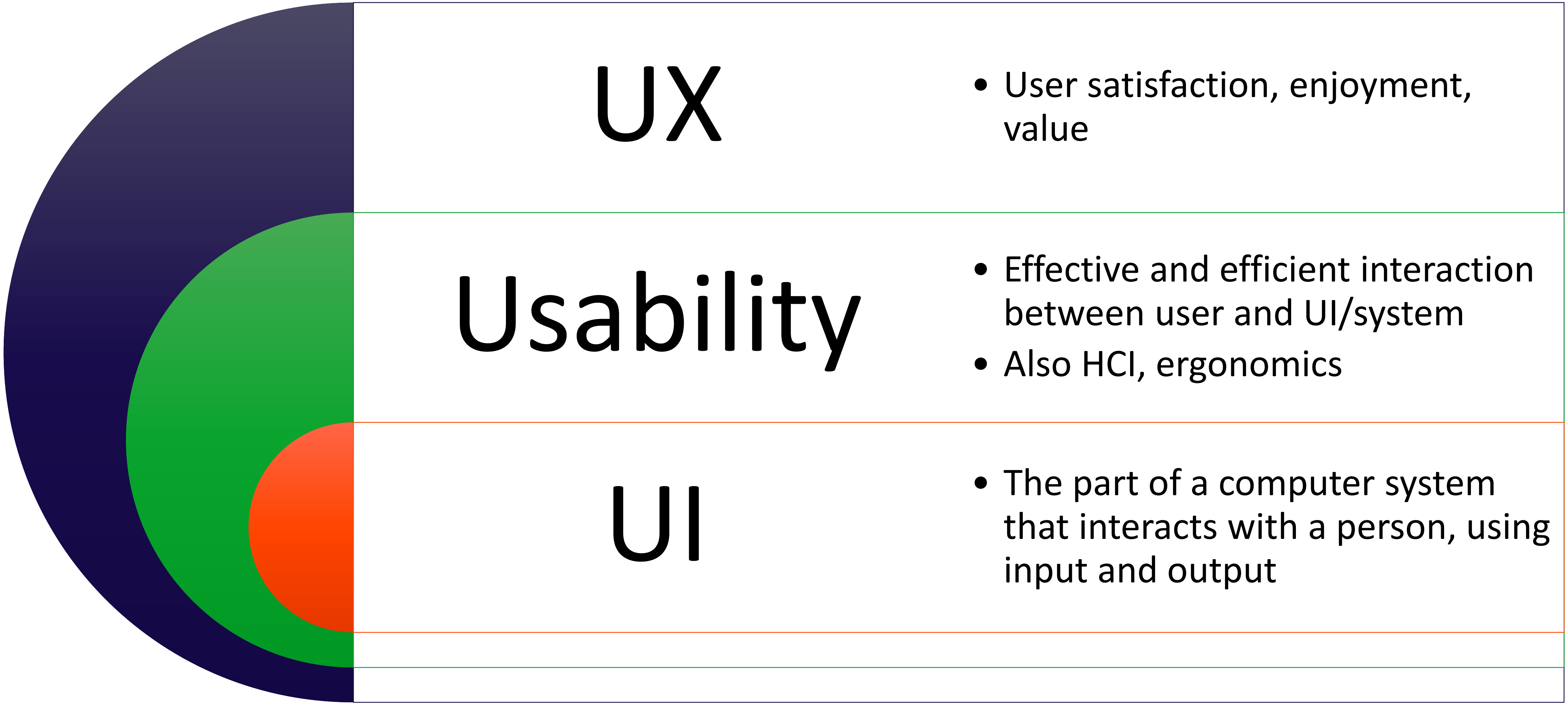
Dimensions of Usability

- **Learnability** → is it easy to learn?
- **Visibility** → is system state visible?
- **Efficiency** → is it fast to use?
- **Errors** → are errors few and recoverable?
- **Satisfaction** → is it enjoyable to use?

Usability Dimensions Vary in Importance

- Depends on the user
  - Novice users need learnability
  - Infrequent users need memorability
  - Experts need efficiency
- But no user is uniformly novice or expert
  - Domain experience
  - Application experience
  - Feature experience

Terminology



UX

- User satisfaction, enjoyment, value

Usability

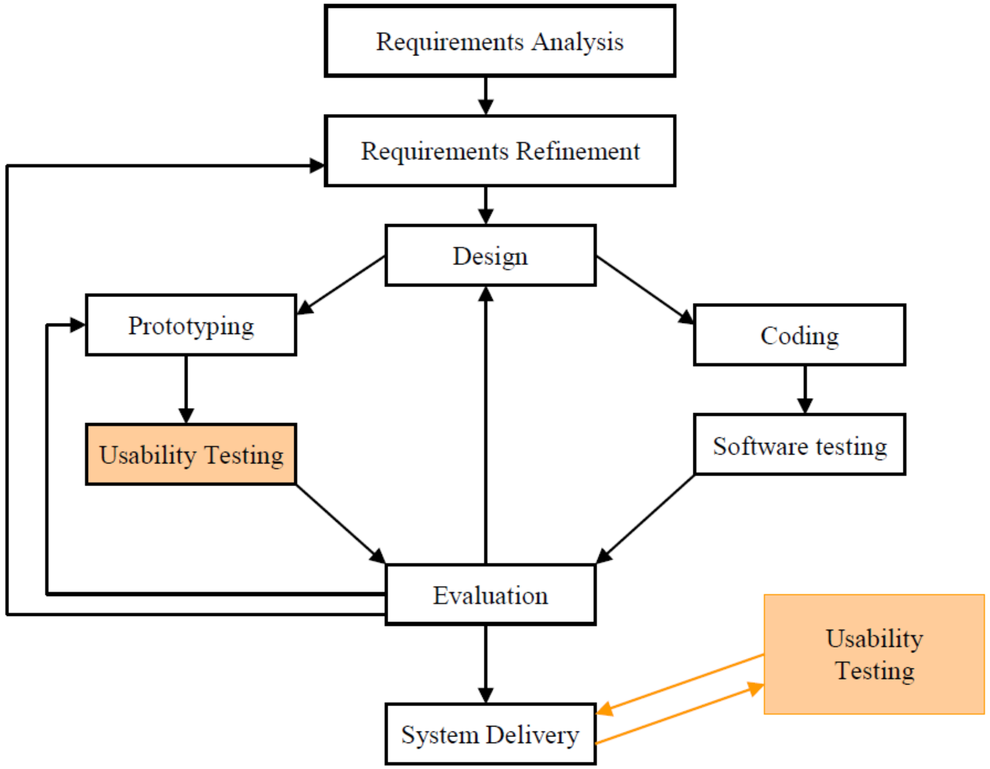
- Effective and efficient interaction between user and UI/system
- Also HCI, ergonomics

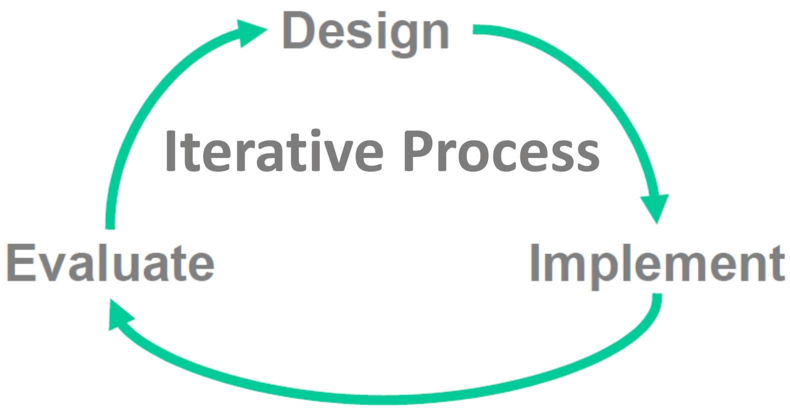
UI

- The part of a computer system that interacts with a person, using input and output

Usability Engineering is a Process

- **Design**
  - Task analysis – “know thy user”
  - Design guidelines – avoid bonehead mistakes
- **Implement**
  - **Prototyping**
    - Cheap, throw-away implementations
  - **GUI implementation techniques**
    - Input/output models, Toolkits, UI builders
- **Evaluate**
  - Expert evaluation – heuristics and walkthroughs
  - Predictive evaluation – against an engineering model
  - Empirical evaluation – watching users do it



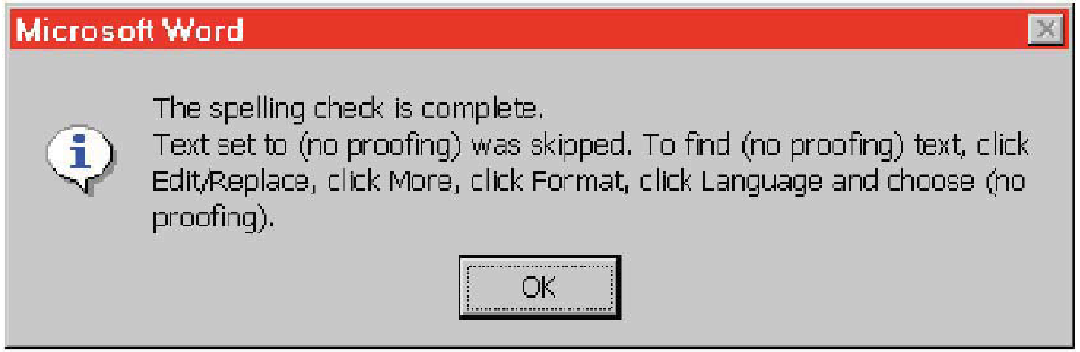


Dimensions – Learnability

Just because you've said it, doesn't mean they know it

- **Recognition** – remembering with the help of a *visual cue*
  - Knowledge from outside world
- **Recall** – remembering with no help
  - *Memory*, knowledge in the head
- Recognition is much easier
  - **Menus are more learnable than commands**

What is the problem with this dialog?



Microsoft Word

The spelling check is complete.
Text set to (no proofing) was skipped. To find (no proofing) text, click Edit/Replace, click More, click Format, click Language and choose (no proofing).

http://hallofshame.gp.co.at/shame.htm

- Overreliance on the user's memory
- User can't start following its instructions until after clicking OK
- Ok makes the instructions vanish from the screen, and the user will struggle to remember them

Interaction Styles

- **Command Language –** Commands, search queries, URLs
- **Menus and forms –** Menu bars, icons, dialog boxes
- **Direct manipulation –** Exploits perceptual and motor skills of the human machine

  - **Three principles:**

1. Continuous visual representation of system objects – file/folder icons, drawing editors, word processors, messages in email
   - Verbal, iconic, **continuously displayed and not on demand**
2. Physical actions or button presses – clicking on objects, drag and drop, resizing
   - **Interaction** with virtual objects in what **seems like a physical way**
   - Are all interaction functions doable through a physical action? e.g. bolding text?
3. Rapid, incremental, and reversible effect of action, with immediate visible outcome – scrolling speed and amount, undo (either with opposite action or with undo), **no confirmation needed**

Comparison of Interaction Styles

	Command Language	Menus and Forms	Direct Manipulation
Learnability	Significant learning needed Manuals, help, …	Easily learnable	Easily learnable
Error messages	have error messages	have error messages	rarely needs error messages
Efficiency	Experts are efficient command histories and scripting facilities	Needs good shortcuts	Dependent on task
User type	Experts	Novice/infrequent users	Novice/infrequent users
Programming difficulty	relatively easy to implement	substantial toolkit support e.g. Java Swing widgets	hardest to program draw, handle keyboard/mouse input, display feedback
Accessibility	more textual easier for users with accessibility needs	harder for users with accessibility needs	harder for users with accessibility needs

Learnability Principles

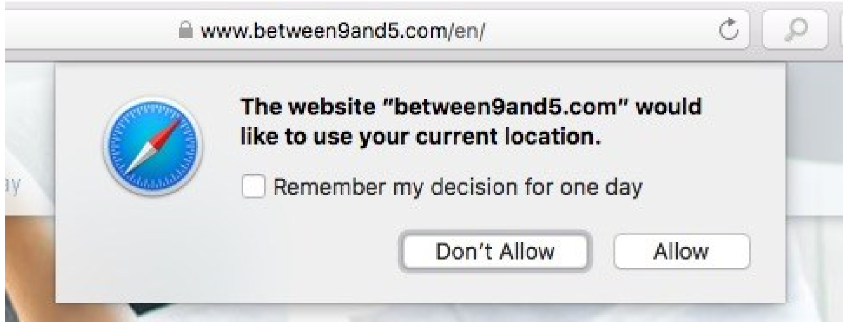
Cues that communicate the system model to users

- **Affordances**
  - Affordances are how an interface communicates nonverbally
  - e.g. scrollbar, textbox/picker for date
- **Natural mapping**
  - Physical arrangement of controls should match arrangement of function
  - e.g. audio mixer, smart home control application (think about the lighting controls – individual? Groups? How to group? How to turn on and off?)
- **Visibility**
  - Relevant parts of the system should be visible
  - e.g. **drag & drop -** little visibility; many users simply don’t realize when drag & drop is possible
  - Do you know you can rearrange tabs in a browser by dragging them?
  - Do you know you can drag website’s icon out of address bar to make a bookmark?
- **Feedback**
  - Actions should have immediate visible effect – visual, audio, haptic
  - e.g. buttons depressed, clicks heard, vibrations

Learnability Principles

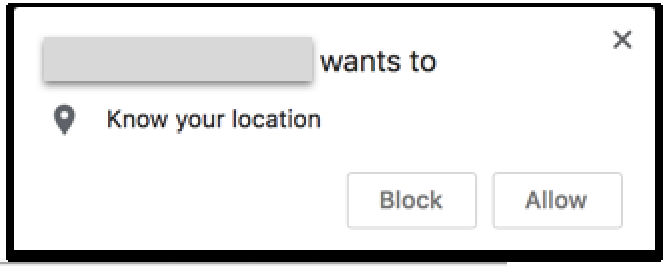
Consistency (principle of last surprise)

- **Similar things should look and act similar, and different things should act different** (e.g. arrow keys)
  - Internal (within app), external (apps with similar functionality or on same platform), metaphorical (in relation to physical world)
- **Consistency is wording/naming of controls**
  - Don't get creative when you're writing text for a user interface!
- **Interfaces are easier to learn if they're already familiar**
- **Interfaces are easier to learn if they have fewer special cases, exceptions, or internal contradictions**
- **Interfaces are easier to learn if they speak the user's language**
  - Use common words, not jargon
  - Use domain terms for domain-specific applications
- **Follow platform standards** (e.g. Apple guidelines, Android guidelines, Java Look and Feel guidelines)
- **Use metaphors** (bring the real world into your interface)
  - e.g. recycle bin (Apple filed a lawsuit for ownership of its icon!), Samsung note

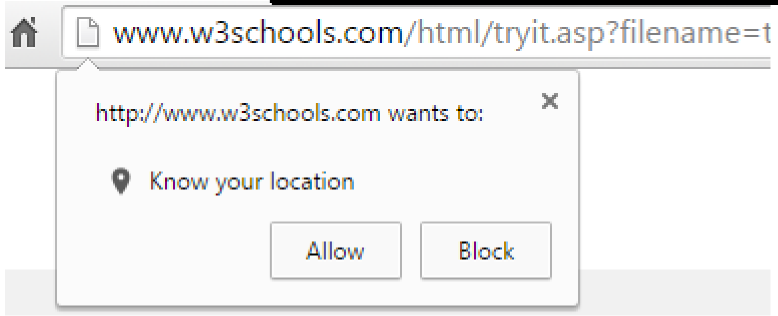

www.between9and5.com/en/

The website "between9and5.com" would like to use your current location.

□ Remember my decision for one day

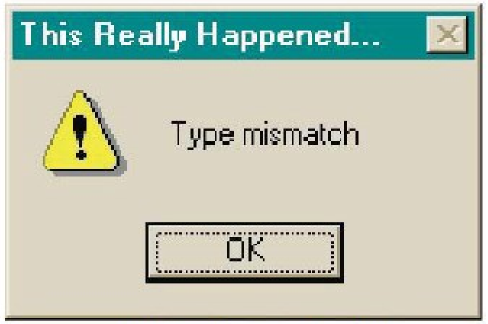

wants to

Know your location


www.w3schools.com/html/tryit.asp?filename=t

http://www.w3schools.com wants to:

Know your location


This Really Happened...

Type mismatch

© Interface Hall of Shame. All rights reserved.

Dimensions – Visibility

- Visibility conveys information and supports learnability
- Aesthetic appeal does not automatically confer usability
- If the user can't *see* an important control (no clues and no affordances), they would have to:
  - guess that it exists
  - guess where it is
- Visible **actions**
- Visible **State**
- Visible **Feedback**

Visible Actions

- **Actions** – the **things the user can do in the interface**
  - Information “scent” – cues on a link/navigation tool indicating how profitable it will be to follow link to destination
    - e.g. link content preview on Google Search, icon of “Printers and Other Devices in Windows Control Panel
- **Affordances** – indicators of what can the user do and where can the user go
  - A clickable arrow does two things: visibly indicate more options, and click action to make choices available
  - A textbox that shows selection but does not allow editing has poor perceivable affordance
- Example affordances: buttons and links, drop-down arrows, texture, mouse cursor, highlight on mouse over

Visible State

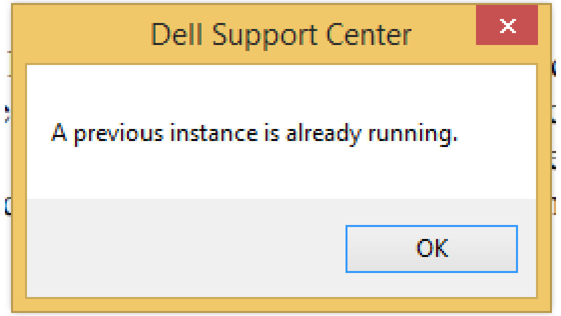
- **State** – the **current configuration of the interface and its backend**
- Make state visible
- **Spotlight of attention** – where the user's attention will likely be?
  - Helps you decide where to make an important state visible
  - e.g. Caps Lock light on keyboard, mouse cursor blinking in Acrobat

- **What states to make visible?** Show more of system model or show less? **Visibility or simplicity?**
  - e.g. word count in Word is always visible, why not use a menu command to show count?
- **How to make state visible?**
  - Where am I now? Where else could I go?
    - Navigation options – pagination, highlighted tabs, hierarchies
  - How can I interact with state?
    - Interaction options – selection highlight, drag and drop, selection handles

Visible Feedback

- **Feedback** – the result of a user's action
- Actions should have immediately visible effects
  - Low-level (e.g. button press) or high-level (e.g. new page starts loading)
- Feedback is dependent on **perceptual fusion**
  - Human perceptual cycle is $T_p \sim 100ms$
  - Upper bound on response time
    - < 0.1 s → **instantaneous**
    - 0.1 – 1 s → **user notices delay**
    - 1 -5 s → **display busy indicator**
    - 1 – 5 → **display progress bar or give ability to cancel**

- Sometimes you sacrifice visibility for security (login screens)
- Audio feedback, haptic feedback are good for low-level feedback
- Feedback should be important, with no superfluous action needed (e.g. Dell's "Another instance is already running" dialog box)



A previous instance is already running.

Disclaimer

Content is adapted from MIT’s User Interface Design and Implementation course

Thank You

mervat.abuelkheir@guc.edu.eg