Chp 2.Heuristic Evaluation

Usability Heuristics

- Jakob Nielsen described the 10 general principles for interaction design.

- These principles were developed based on years of experience in the field of usability engineering and they've become rules of thumb for human-computer interaction.

- They can help to save development teams considerable amounts of time during early usability testing, so that they can direct their attention to more complex design challenges.

- It's also worth it to use them as a checklist when designing a new product or a feature.

Heuristic principles

10 Usability Heuristics

Visibility
Show system status, tell what's happening

Mapping
Use familiar metaphors & language

Freedom
Provide good defaults & undo

Consistency
Use same interface and language throughout

Error Prevention
Help users avoid making mistakes

Recognition
Make information easy to discover

Flexibility
Make advanced tasks fluid and efficient

Minimalism
Provide only necessary information in an elegant way

Error Recovery
Help users recognize, diagonize and recover from errors

Help
Use proactive and in-place hints to guide users

#1: Visibility of system status

- The design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time.

- When users know the current system status, they learn the outcome of their prior interactions and determine next steps.

- Predictable interactions create trust in the product as well as the brand.

- The sense of control can be evoked by providing information about the system status and feedback after every interaction.

- One example is twitter making a swoosh sound when a tweet is being posted. Another example is Google Drive showing the status of a document upload. etc etc etc…..

Take a look at smartphone. Right after the screen lights up, it informs you about its battery, a wifi connection, received messages, missed calls, and much more. Imagine how insecure you would feel if this information were missing. By utilizing signs, icons, and indicators, the system communicates its status and helps user make better, more informed decisions.

#2: Match between system and the real world

- The design should speak the users' language.
- Use words, phrases, and concepts familiar to the user, rather than internal jargon.
- Follow real-world conventions, making information appear in a natural and logical order.
- The way you should design depends very much on your specific users. Terms, concepts, icons, and images that seem perfectly clear to you and your colleagues may be unfamiliar or confusing to your users.
- When a design's controls follow real-world conventions and correspond to desired outcomes, it's easier for users to learn and remember how the interface works.

An extreme example is a smart phone design, which transfers all details of real world objects into the software. At the beginning of smartphone adoption, it helped people to learn how to use their new companions through the aesthetics and processes they were familiar with before.

Great examples of real-world matching icons

Neil Patel could very well say "Sign Up" on his landing page. Instead, he chose to say ambitiously — "Yes, I want Neil to teach me how to grow my Business!". It sets the context and speaks the everyday language.

#3: User control and freedom

- Users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process.

- When it's easy for people to back out of a process or undo an action, it fosters a sense of freedom and confidence. Exits allow users to remain in control of the system and avoid getting stuck and feeling frustrated.

- Support *Undo* and *Redo*. Show a clear way to exit the current interaction, like a *Cancel* button. Make sure the exit is clearly labeled and discoverable.


If you attached a large file in Gmail by mistakenly, you can cancel it before its fully uploaded.

An appropriate emergency exit can be something as simple as an arrow back (e.g. in a browser), a trash bin, which protects us from accidental deletion, or the "undo" button, which lets the user to revert the last action. All of these examples demonstrate systems which don't let users down when they make a mistake, and instead, they allow the user to fix it.

And below is Facebook checking on me if I tapped "Cancel" by mistake. Gmail's flash message with undo action when we accidentally delete an email.
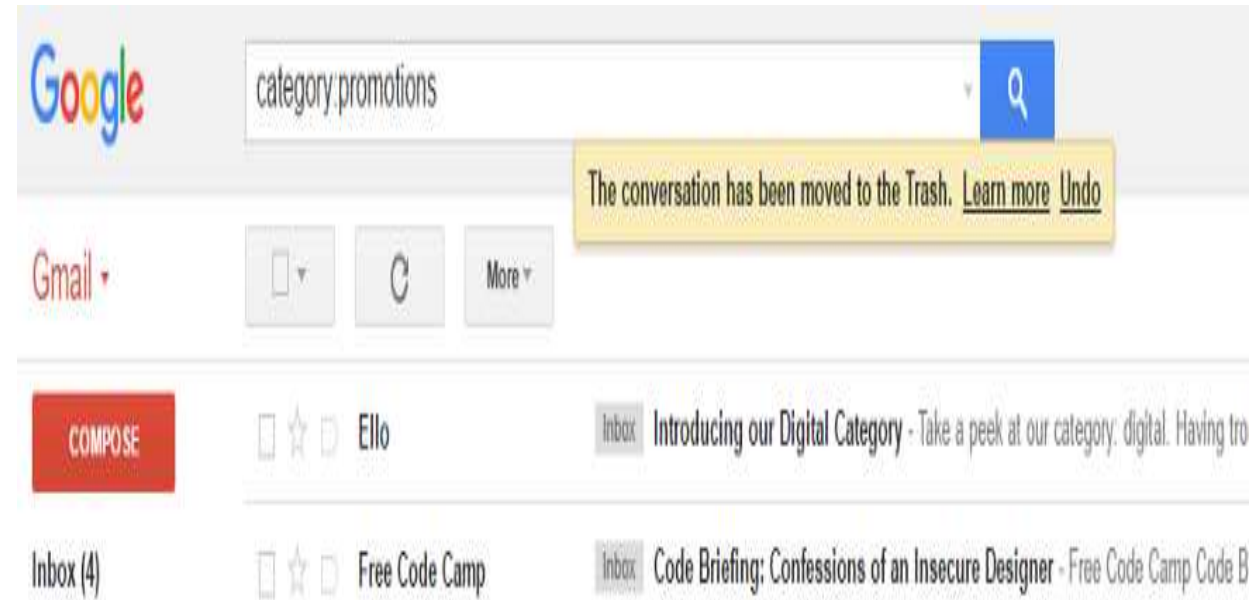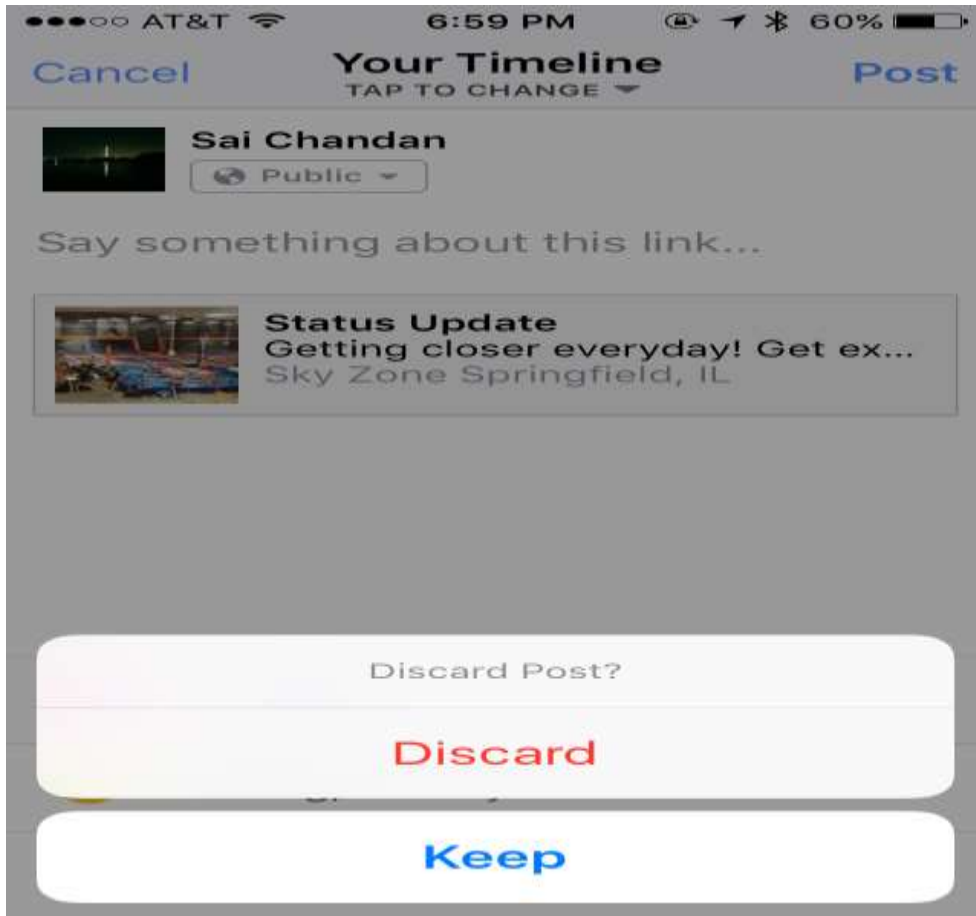
#4: Consistency and standards

- Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions.

- A comprehensible system should never confuse users by using different words, visuals, or actions for the same concepts.

- Improve learnability by maintaining of consistency
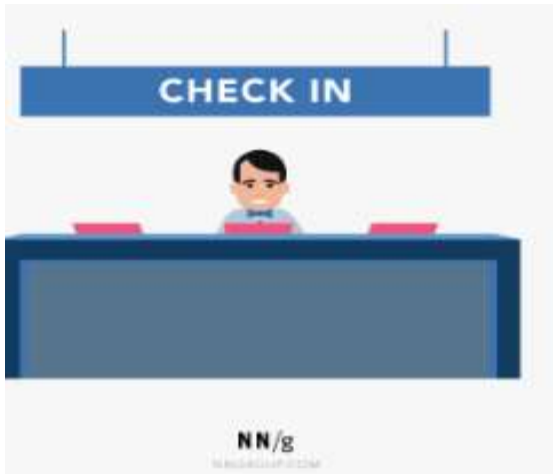
Check-in counters are usually located at the front of hotels. This consistency meets customers' expectations.

How the same button can transform across different pages of the same site. Note that this is not a change of state.
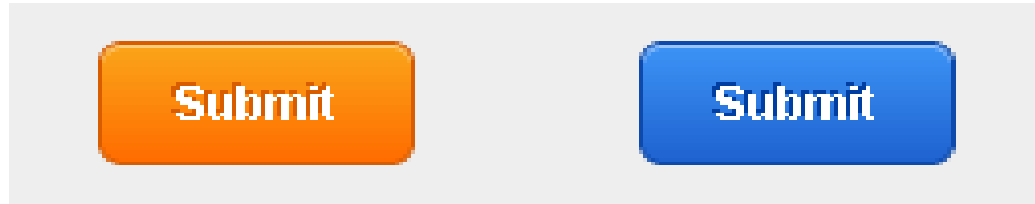
Google Plus ambitiously launched "+1" to counter Facebook's "Like" without much success. Facebook's "Like" already became a standard and sites like LinkedIn adopted it without contesting

#5: Error prevention

➢Good error messages are important, but the best designs carefully prevent problems from occurring in the first place.

➢ Either eliminate error-prone conditions, or check for them and present users with a confirmation option before they commit to the action.

❖ Google Search trying to correct my spelling:

➢There are two types of errors: <u>slips and mistakes</u>.

➢**Slips** are unconscious errors caused by inattention.

➢**Mistakes** are conscious errors based on a mismatch between the user's mental model and the design.

➢**Slips** happen when the user tends to do an action, but due to low attention, performs another one (e.g. when performing well known task).

➢The strategy to prevent users from experiencing a slip is to minimize the chance of it occuring by guiding them only through the safe areas.

It seems like you have forgotten to attach a file.

You wrote "are attached" in your message, but there are no files attached. Send anyway?

Cancel   OK

Smart slip prevention in the gmail web app. Unfortunately, the mobile app lacks this feature.

➢**Mistakes** are often caused by a user's incorrect mental model of how the system works.

➢The user misunderstands the communication and consciously performs an action which leads to a different result than they intended.

➢Use clear communication and a consistent design system to prevent mistakes.

Confirmation dialog before a potentially dangerous operation.

The input doesn't let you write an invalide value.

Subtle information about reaching the character limit.

#6: Recognition rather than recall

➢ Minimize the user's memory load by making elements, actions, and options visible.

➢ The user should not have to remember information from one part of the interface to another.

➢ Information required to use the design (e.g. field labels or menu items) should be visible or easily retrievable when needed.

➢ The recognition happens when you easily recognize a person or an object that you're familiar with.

➢ The recall happens when you have to find rarely used information in your memory (names, years, details, etc.)

➢ Reduce the information that users have to remember.

➢Quora suggesting possible questions based on what I am trying to type.
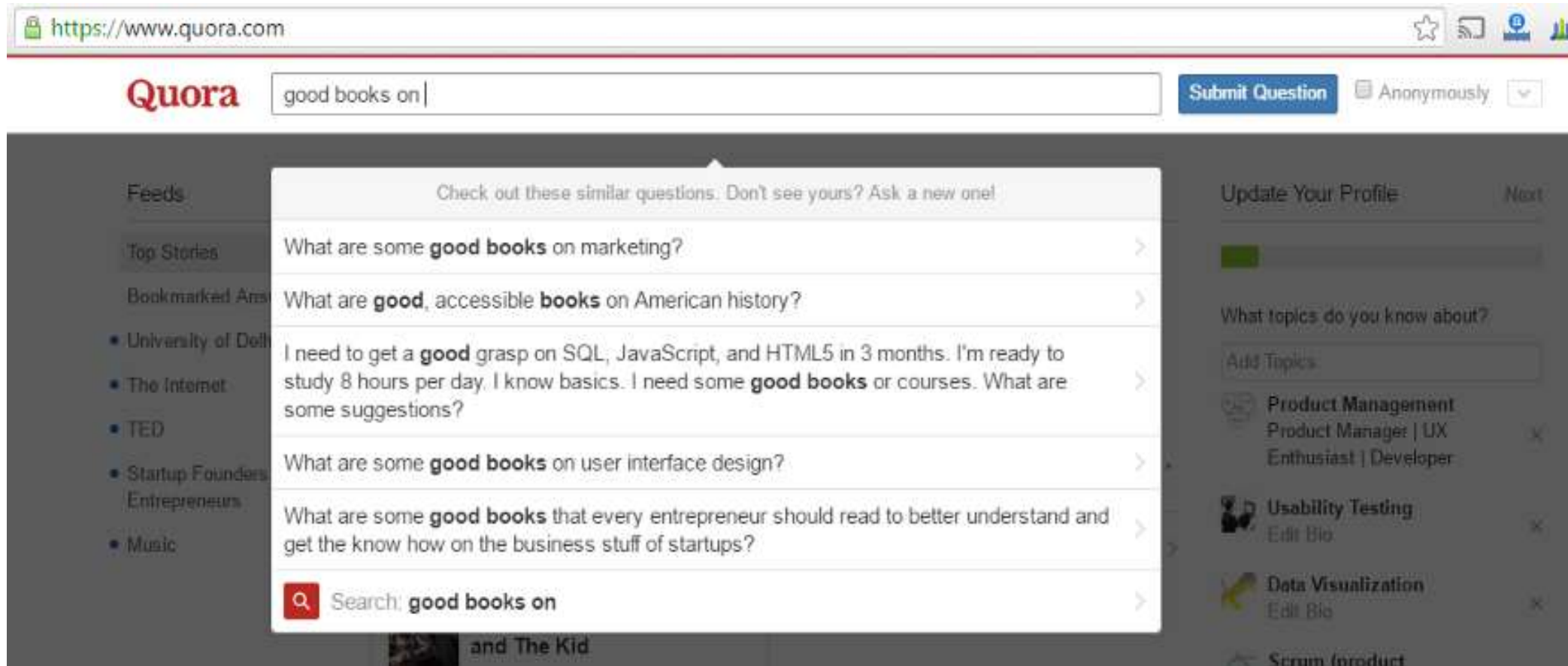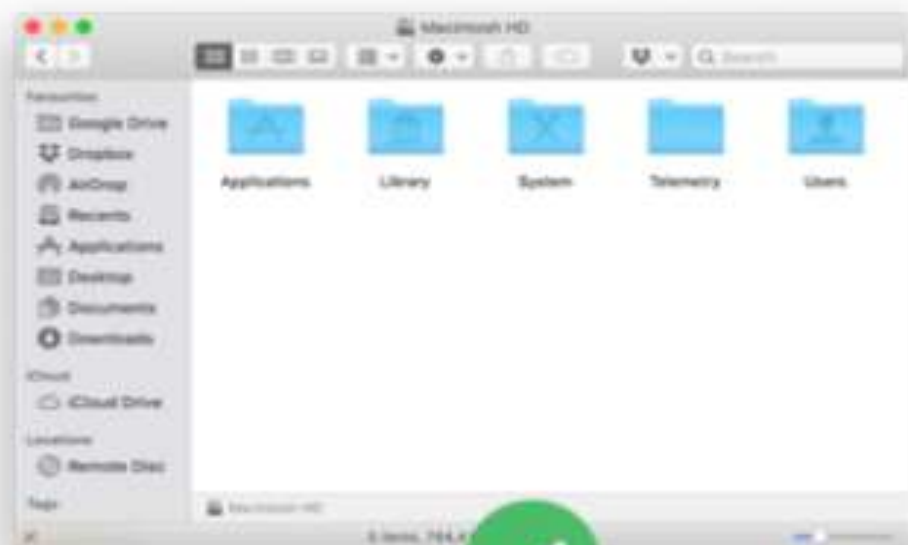
Users who are not familiar with the syntax of terminal commands can't perform as easy operation as opening or deleting the file.

#7: Flexibility and efficiency of use

➢ Every user is unique; each have their own different needs and skills. Equally, every task is unique and requires different controllers.
➢ Shortcuts — hidden from novice users — may speed up the interaction for the expert user such that the design can cater to both inexperienced and experienced users.
➢ Flexible processes can be carried out in different ways, so that people can pick whichever method works for them.
➢ A good user interface should offer appropriate functionality to both inexperienced and experienced users.

➢ An example of setting up Exchange on Android which hides the complex features under Advanced.

Example of Usability Heuristic #7:
Regular routes are listed on maps, but locals with more knowledge of the area can take shortcuts.

#8: Aesthetic and minimalist design

➢Minimalism aims to reduce the description of a subject just to its necessary elements.

➢Minimalism helps users to quickly access important information and come to the result quickly.

➢A minimal design uses only the necessary colours/fonts to support the visual hierarchy.

➢Keep the <u>content</u> and <u>visual design</u> of UI focused on the essentials.

➢Don't let unnecessary elements distract users from the information they really need.

➢ Google has been resisting the temptation to show more information on their search page for years. This is could be shown as the example of the best possible minimalist design.
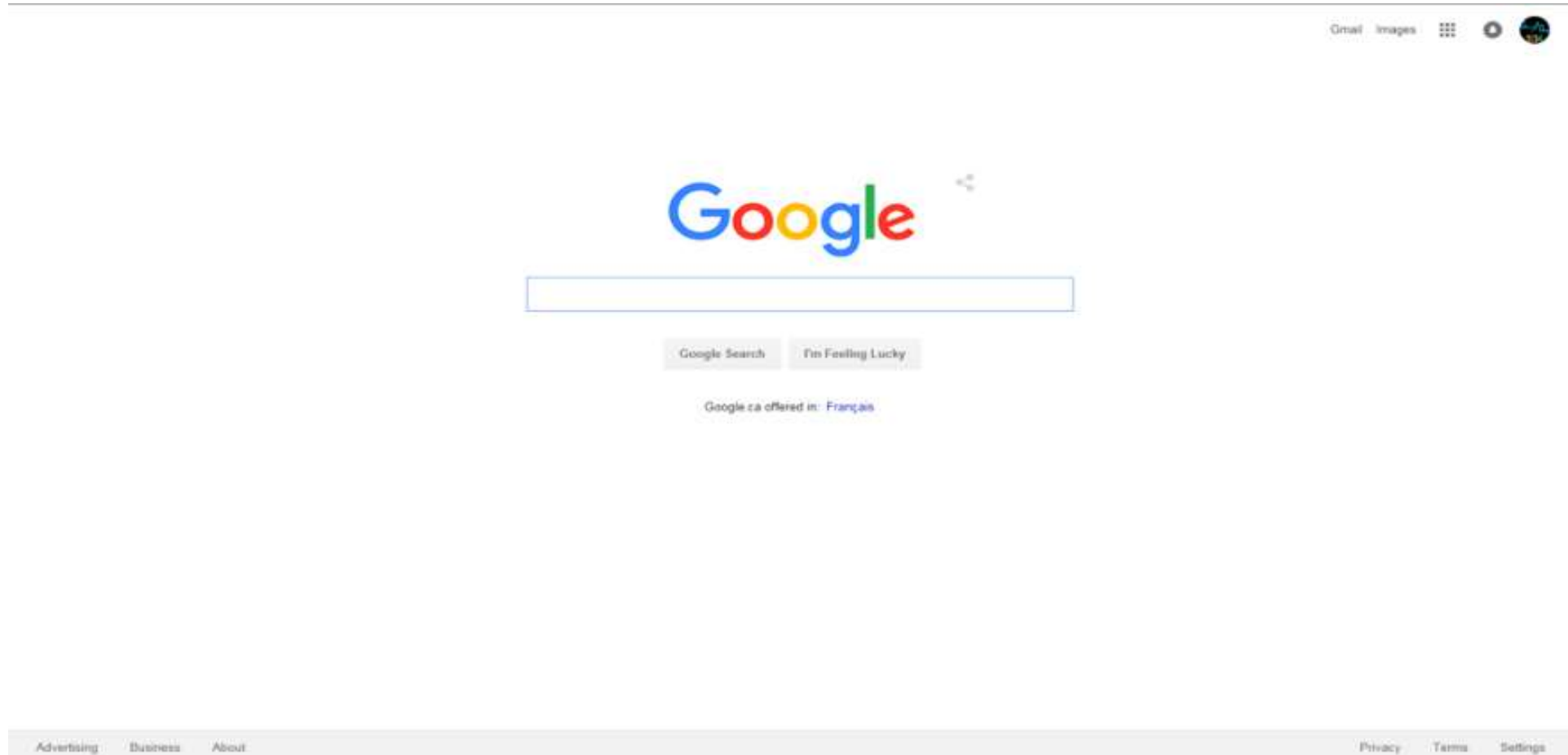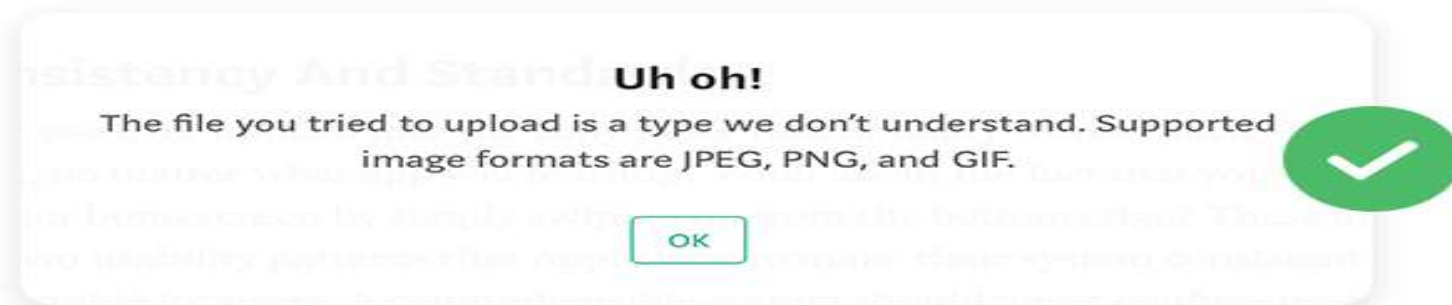
➢ Apple provides only the basic information of feature hiding additional information under "Learn More".
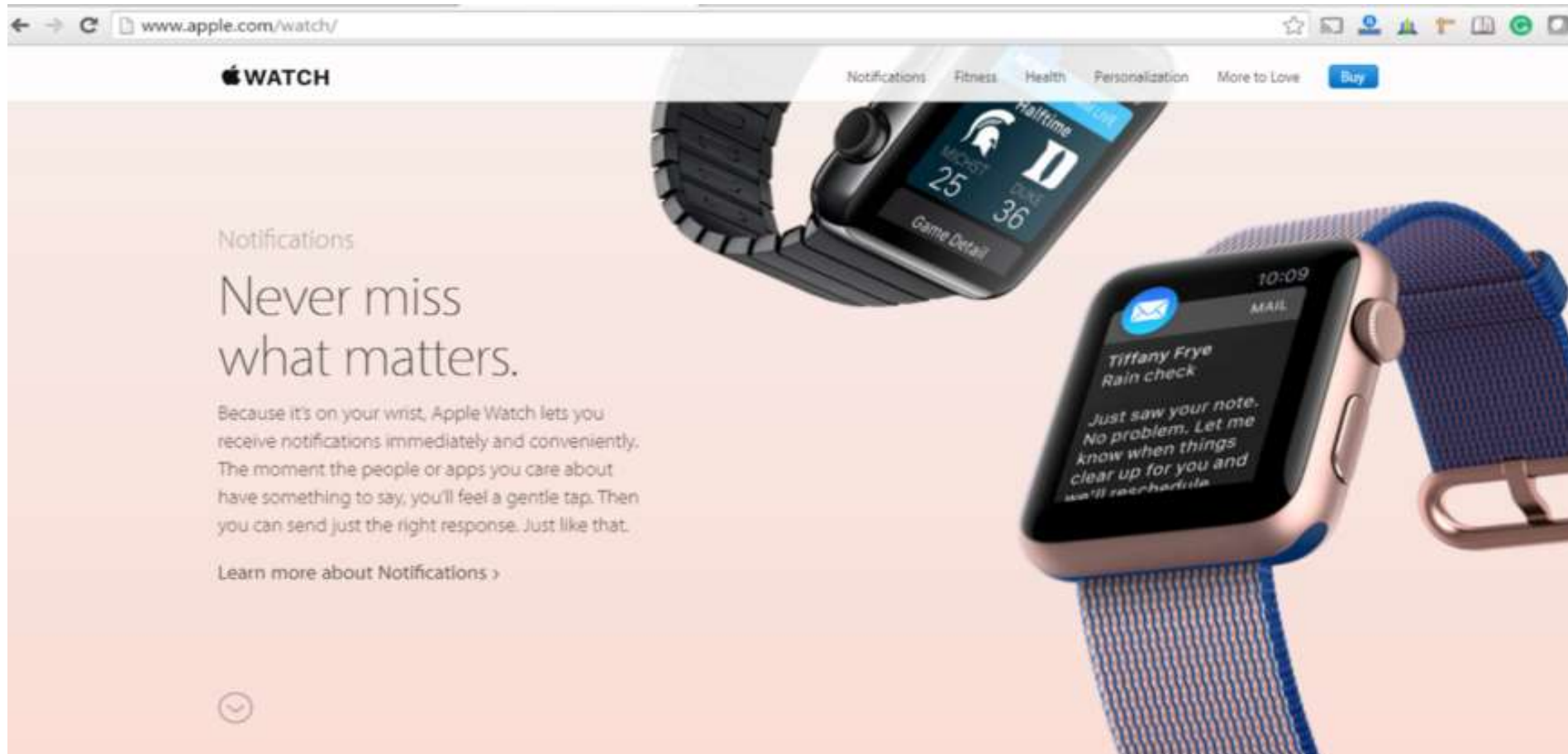
#9: Help users recognize, diagnose, and recover from errors

➢Error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.

➢These error messages should also be presented with visual treatments that will help users notice and recognize them.

Uh oh!

The file you tried to upload is a type we don't understand. Supported image formats are JPEG, PNG, and GIF.

OK

Great example of the error message which immediately suggests the next steps.

➢ Here we are not informing the user if the username is invalid or if the password is wrong.
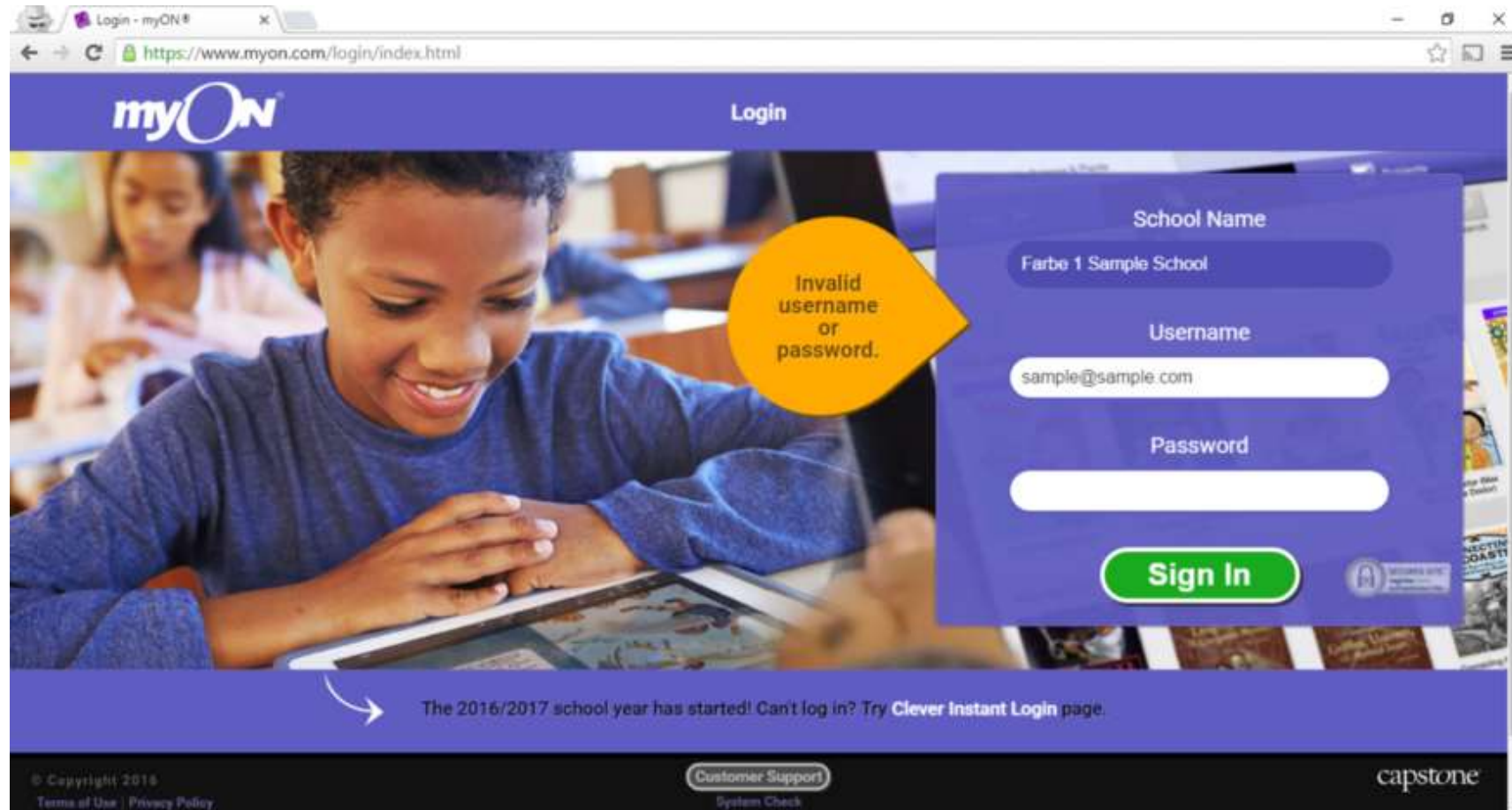
➤ Instead an example of how MailChimp is handling this scenario:

#10: Help and documentation

➢It's best if the system doesn't need any additional explanation. However, it may be necessary to provide documentation to help users understand how to complete their tasks.

➢Help and documentation content should be easy to search and focused on the user's task.

➢Keep it concise, and list concrete steps that need to be carried out.

➢ An example of GoDaddy's Help page. While there is a search field, there are main categories and frequently asked queries on the same page.
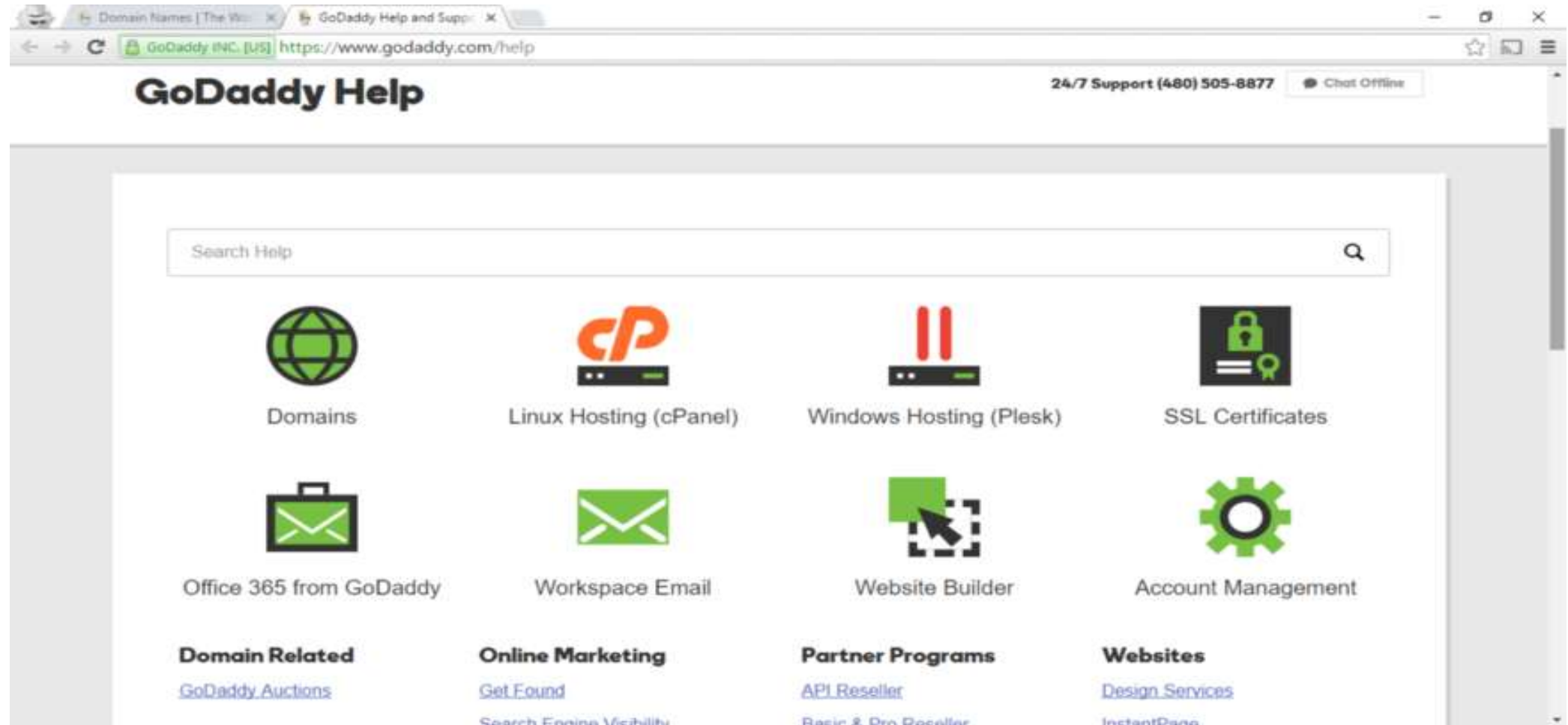
➢These guidelines are general rules of thumb and will mostly be applicable to any web & mobile applications.

➢ Always use your judgment to implement these principles or any other UX practices by keeping yourself in end user's shoes.

Heuristic Evaluation

- A Heuristic Evaluation is a usability inspection technique where one or a number of usability experts evaluate the user interface of a product .

- Independent walkthroughs are conducted and issues are reported.

- Evaluators use established heuristics principles and reveal insights that can help design teams to enhance product usability from early in development.

- Heuristic evaluation is a usability engineering method for finding usability problems in a user interface design, thereby making them addressable and solvable as part of an iterative design process.

- Such processes help prevent product failure post-release.

- It is usually conducted by a group of experts because it is very likely that one person will not be able to find all usability problems , analyze an interface from different angles and as a result are more likely to identify a wider set of areas for improvement.

To conduct a heuristic evaluation following steps are conducted:

➢**Know what to test and how** – Whether it's the entire product or one procedure, clearly define the parameters of what to test and the objective.

➢**Know your users and have clear definitions of the target audience's goals, contexts, etc**. <u>User personas</u> can help evaluators see things from the users' perspectives.

➢**Select 3–5 evaluators**, ensuring their expertise in usability *and* the relevant industry.

➢**Define the heuristics** (around 5–10) – This will depend on the nature of the system/product design.

➢**Brief evaluators on what to cover in a selection of tasks**, suggesting a scale of severity codes (e.g., critical) to flag issues.

➢**1st Walkthrough** – Have evaluators use the product freely so they can **identify** elements to analyze.

➢**2nd Walkthrough** – Evaluators **scrutinize** individual elements according to the heuristics. They also examine how these fit into the overall design, clearly **recording** all issues encountered.

➢**Debrief evaluators** in a session so they can collate results for analysis and suggest fixes.

1) Know what to test and how –

Whether it's the entire product or one procedure, clearly define the parameters of what to test and the objective.

➢ **Define the scope of your evaluation.**
  ❑ The first thing you should define the scope of your evaluation in keeping with your budget and deadline.
  ❑ Do you need to test every aspect of your product or should you concentrate on particular user-flows?
  ❑ Do you need to identify major issues in a small time frame?

➢ **Specific usability parameters(scope) that you want to test for your product, such as:**
  ❑ Registration
  ❑ Login/out
  ❑ Email signup
  ❑ Navigation
  ❑ Shopping cart
  ❑ Checkout

➢ Having a limited scope is easier to control and assess.

2) **Know your users:**

Understanding who your end-user is and what their goals are will aid the usability evaluation.

➢ This is an important part of mapping out the user flow, as different user groups have distinct expectations and user behaviors.

➢ *For instance:*

❑ Some users might not have issues registering for a product while others may see it as an unnecessary step and abandon a product.

❑ User motivation depends on various factors, including:

▪ Demographics

▪ Personal preferences

▪ Skillsets

▪ and more

➢ Know your end-user, and create advanced user personas to assist evaluators during the evaluation process.

3) Select 3–5 evaluators

➢Ensuring their expertise in usability *and* the relevant industry.

➢It is very likely that one person will not be able to find all usability problems. On the other hand, a group of different people tend to analyze an interface from different angles and as a result are more likely to identify a wider set of areas for improvement.

4) Choose your set of heuristics.

➤ Select which heuristics the evaluators are going to use.

➤ This will ensure that they are all using the same guidelines throughout the evaluation. We mentioned a few of the most popular heuristics earlier, so check them out.

➤ Without heuristics, the usability evaluation will produce unreliable, inconsistent, and ultimately, useless results. Essentially, all your efforts will be for nothing.

5) Set up an evaluation system and identify issues

➢Decide how evaluators will evaluate and report the usability of your product.

➢Try setting up a simple evaluation system using a severity rating:
  - ➢**Critical issue**
  - ➢**Normal issue**
  - ➢**Minor issue**
  - ➢**Good practice**

➢Whichever evaluation system you choose, discuss it with evaluators beforehand to make sure everyone is on the same track.

➢Evaluators should keep track of issues by making detailed notes of where they encountered the issue and how serious it is. This will help organize the design team's backlog later on.

6) Analyse and summarize findings

➢ As the evaluation draws to a close, it's time to gather, compare, and summarize the findings.

➢ One of the key benefits of using multiple evaluators is that they will each find issues that their counterparts have missed.

➢ Start by removing duplicates and organizing the data consistent with the severity rating of each issue. This will facilitate the design team in prioritizing their workflows.

➢ The findings will become the launch pad for improved UX design and an all-around better product.

Severity Ratings

➢ Severity ratings can be used to allocate the resources to fix the most serious problems and can also provide a rough estimate of the need for additional usability efforts.

➢ If the severity ratings indicate that several disastrous usability problems remain in an interface, it will probably be unadvisable to release it. But one might decide to go ahead with the release of a system with several usability problems if they are all judged as being cosmetic in nature

➢ The severity of a usability problem is a combination of three factors:

❑ The **frequency** with which the problem occurs: Is it common or rare?

❑ The **impact** of the problem if it occurs: Will it be easy or difficult for the users to overcome?

❑ The **persistence** of the problem: Is it a one-time problem that users can overcome once they know about it or will users repeatedly be bothered by the problem?

The following 0 to 4 rating scale can be used to rate the severity of usability problems:

0 = I don't agree that this is a usability problem at all

1 = Cosmetic problem only: need not be fixed unless extra time is available on project

2 = Minor usability problem: fixing this should be given low priority

3 = Major usability problem: important to fix, so should be given high priority

4 = Usability catastrophe: imperative to fix this before product can be released