Rules for User Input Categories

In this lesson, let's choose which rules will be used to validate various user input types such as birthday, name and username.

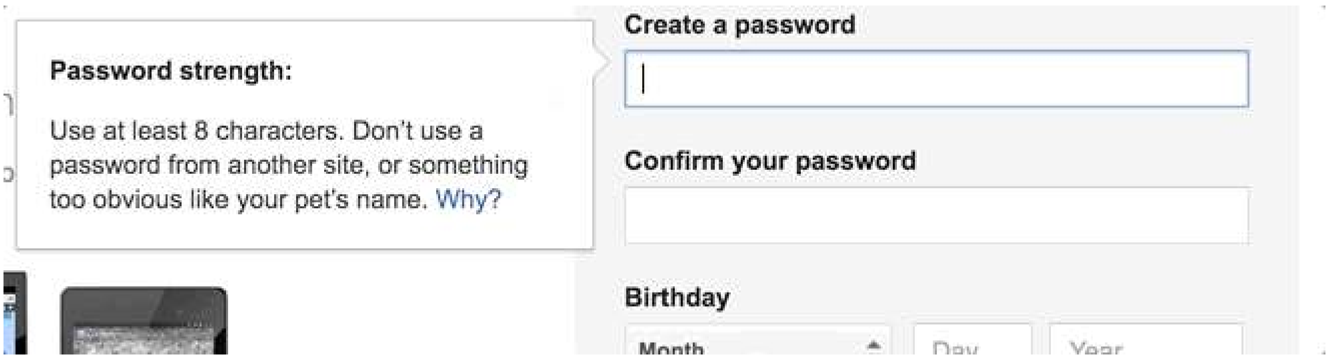
WE'LL COVER THE FOLLOWING

What is the goal of adding auto validation to forms?

When forms validate inputs for users before they hit submit, it's to prevent frustration when they later hit submit with improper inputs and are rejected. The second worst user experience you can provide is to wait until the user has filled out all the information and hits submit, only to be rejected for an unclear reason and forced to re-enter information- I'll get to the worst in a second.

Therefore, let's only validate when the user appears to be finished with that input.

There are some exceptions to this rule, and that's where the requirements are ambiguous. Everyone knows what emails are supposed to look like, but different websites can have different requirements for passwords depending on their audience, risk of security breaches (it's not the end of the world if someone gains access to your yoga membership account, but it could be disastrous if they do gain access to your bank account). It's reasonable to show the requirements of inputs with varying rules before the user enters them – or better yet, as the user enters them:


Password strength:

Use at least 8 characters. Don't use a password from another site, or something too obvious like your pet's name. Why?

Create a password

Confirm your password

Birthday

Month Day Year

So we'll validate after the user is complete for fields with universal rules. Otherwise, we want to guide them. We also want to choose the best way to do it. For example, for birthdays, giving individual fields for month, date, and year removes ambiguities for region-specific formats. You want to create as little chance as possible for errors when they hit submit.

What to avoid

Adding client-side validation can also accidentally have the opposite effect of preventing good input or adding undue confusion. The validation shouldn't be too stringent. For example, some forms will validate *as* the user is typing, showing an error before they're even finished working on the field.

The **worst** user experience, however, is when valid input can't be submitted. I've encountered sites where an email field wouldn't accept emails that didn't end with `.com`.

So now let's establish a set of rules for the form we're building (these won't necessarily be Gmail's):

Rules

- Name: A non-empty string of alpha characters
  - Validate after the field is blurred (cursor no longer in the field)
- Username: A string of characters which are either numbers, letters, periods, or underscores.
  - Validate after the field is blurred
- Password: A string of alphanumeric characters greater than length 6
  - Validate as typed
  - Rated by the following:
    - Weak: Less than 6 length

        - Fair: Greater than 6 length
        - Good: Greater than 6 length and has a mix of letters and numbers
- Birthday: Numbers
    - Validate after the field is blurred
    - Extra guidance provided by:
        - inability to continue typing past character limit (2 for day, 4 for year)
        - inability to type non-numeric characters (typing ‘a’ does nothing)
- Phone number: Numbers with hyphens and parentheses
    - Parentheses can only surround the initial set of numbers
    - Hyphens must appear between numbers
- Current email address: String of ASCII characters with `@` and `.` existing between any characters (can’t be at the beginning or end)

Let’s proceed forward with actually setting up our own autovalidating form.