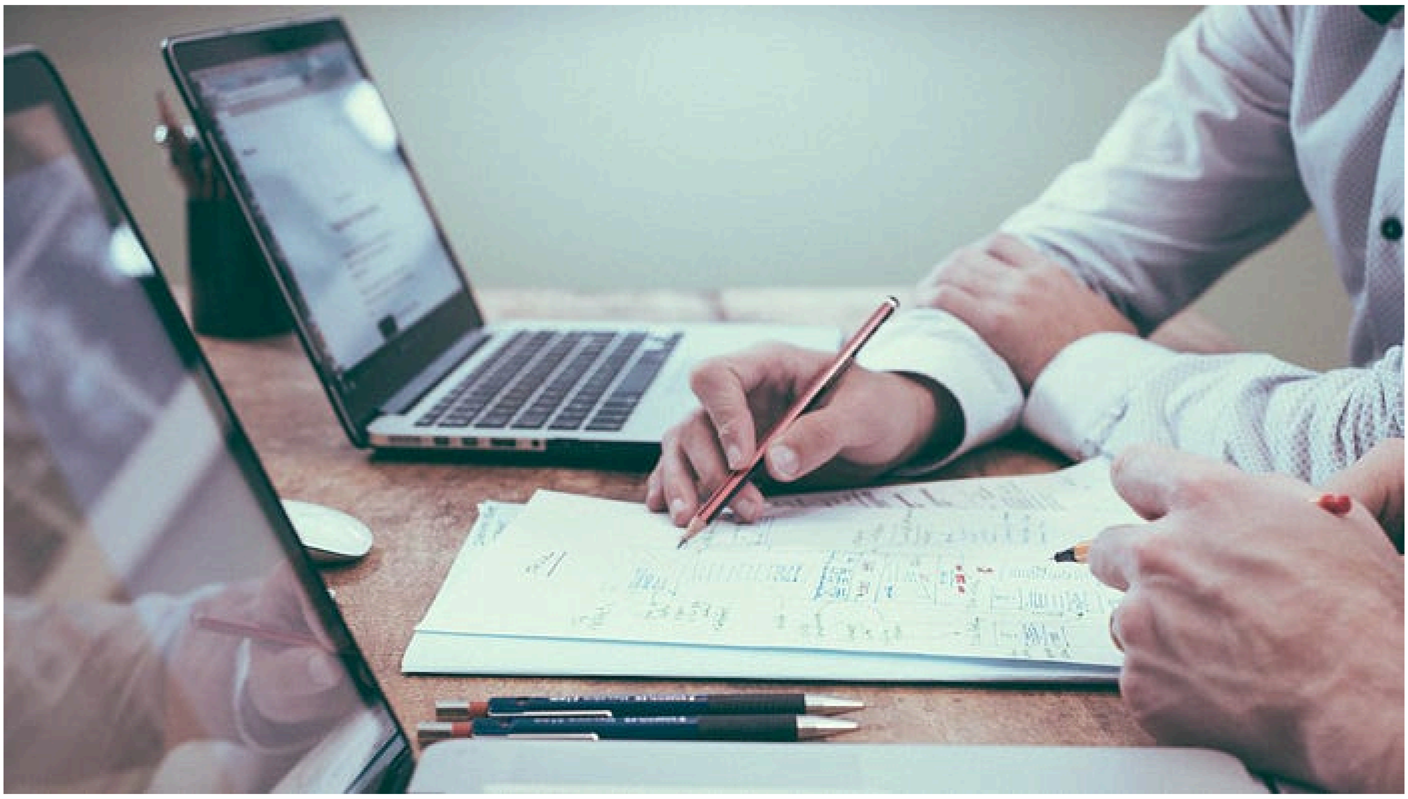
< Go to the original



Checklist for better web forms

Forms can be painless or painful for visitors. Keep in mind that visitors want to get their tasks accomplished as quickly as possible…

Mert TOL

Follow

UX Collective a11y-light ~4 min read · September 24, 2019 (Updated: March 3, 2022) · Free: Yes

Forms can be painless or painful for visitors. Keep in mind that visitors want to get their tasks accomplished as quickly as possible, and with the least amount of effort. Proper planning and design can maximize task efficiency.

Usability — Did you:

1. Do everything possible to reduce visitor effort — visually, cognitively, and physically?
2. Minimize visitor clicks, keystrokes, mouse movements, and scrolling?

4. Avoid asking for unnecessary or redundant data?
5. Indicate which elements are required and which are optional?
6. Present input controls in the expected order?
7. Use wizards and control panels appropriately?
8. If employing control panels, chunk the input elements by category?
9. Specify an appropriate tab order and initial focus, when appropriate?

Instructions — Did you:

1. Provide adequate instructions so that separate help is rarely needed?
2. Accurately label each form control, and place each in close proximity and visual connection to the fields it explains?
3. Make documentation apparent but unobtrusive?
4. Use a consistent vocabulary?
5. Include the title attribute on form elements to provide concise instructions on rollover?

Visitor Support — Did you:

1. Practice defensive design, so that the form is so intuitive and easy to use that little additional support is necessary?
2. Try to be brief and salient, telling visitors only what they absolutely need to know to accomplish their tasks?
3. Provide obvious documentation for probabilities, not possibilities?

Accessibility — Did you:

1. Whenever possible, place labels to the left of controls, or associate the label and its control with the `<label>` tag?
2. Surround related form elements with `<fieldset>` tags and include the `<legend>` tag for labeling?

multiple attribute)?

5. Avoid setting the default value for a single yes/no checkbox to "yes"?
6. Use the title attribute to provide tooltips on rollover?

Help — Did you:

1. Use contextual help whenever possible?
2. Provide a hot-linked table of contents or site map, and a search function for larger Help systems?

Feedback — Did you:

1. Provide feedback, particularly for long waits and errors?
2. Use modeless, rather than modal, feedback whenever possible?

Error Handling — Did you:

1. Provide error messages that are helpful, clear, non-technical, concise, polite, tactful, non-accusatory, apologetic, and visible?
2. When a server-side error is encountered, either redisplay the entire form or just the subset of controls that are in error?
3. Display an error message on the same page as the field that needs to be corrected?
4. Remember that a successful resolution of a problem builds stronger customer loyalty than delivering a product that performs flawlessly?

Form Submission — Did you:

1. Use the appropriate control to submit a form?
2. Avoid telling visitors not to click a Submit button twice?
3. Avoid using Reset buttons?

5. Position buttons in close proximity and visually connected to their form elements?
6. Position the safer or more positive action button on the left, and the riskier or more negative action button on the right, making sure that default focus lands on the safer of the two actions?
7. Confirm risky actions?
8. After submission, reassure a visitor with a response page that re-displays the entered information?
9. Verify completeness and format of all entries as much as possible before submitting the form to the server?

Text Input Controls — Did you:

1. Use `<input type="text" />` for one-line text input, or `<input type="password" />` for a one-line text input with the entered characters hidden by asterisks?
2. Specify size and max length that roughly correspond?
3. Display relevant field formatting and punctuation, such as dashes within dates?
4. Use the value attribute to specify a default value if 80% or more of the audience can use the default?
5. Avoid using the value attribute to display instructions for the field?
6. Use the `<textarea>` tag for multiple-line text input?
7. Use wrap="soft", "hard", or "off" to determine whether carriage returns are used and whether they're saved with the `<textarea>` field?
8. Use `<textarea>` to display informational text that needs to scroll?

Selection Controls — Did you:

1. Use `<input type="radio" />`, `<input type="checkbox" />`, or `<select>` for a limited number of choices?
2. Filter out those options that become irrelevant because of earlier choices?

4. Use `<input type="checkbox" />` for a short list that allows multiple choices?
5. Align check boxes and radio buttons vertically?
6. Visually chunk related radio buttons and checkboxes?
7. Use `<select>` for a longer list that would cover too much screen if all the choices were displayed at once?
8. Decide whether or not to place the most common choices at the top of the list?
9. Use `<size="1">` for a drop-down list, and size with a value greater than 1 for a scrollable list?
10. Use `<optgroup>` to chunk related options within a select box?

I hope this checklist will help in your development process. Did I miss anything? If so, please comment below. Thank you for reading.

> *Write me for work inquiries or just to say hello.* ***LinkedIn*** *—* ***Instagram*** *—* ***Twitter***

#web-design #forms #ui #user-experience #usability