TIL about revealing the password

Photo by Ion Fet on Unsplash

~~Gunnar Bittersmann~~ - ~~2024-09-15 20:08~~ browser - datenschutz - html

@@Gunnar Bittersmann

Bei meiner Recherche stieß ich auf Chris Coyiers Artikel The Options for Password Revealing Inputs (Oktober 2021).

Option 1: Use `type="password"`*, then swap it out for* `type="text"` *with JavaScript* war der Stein des Anstoßes hier im Thread; das wollen wir ja gerade vermeiden.

Option 2: Use `-webkit-text-security` *in CSS* sieht trotz Vendor-Präfix vielversprechend aus; das wird von allen gängigen Browsern unterstützt – für Texteingabefelder.

Und jetzt ratet mal, für welche Elemente das kein Browser unterstützt!
Genau! Für Passworteingabefelder! 🤔😫🤮

☞ reveal password, -webkit-text-security – `type="text"` gesetzt: die Umschaltung funktioniert. Für `type="password"`: nö, nirgends. Es ist mir auch nicht gelungen, den Browserdefault zu überschreiben – weder mit `!important` noch mit `@layer` noch durch Setzen der CSS-Eigenschaft direkt mit JavaScript.

Beim einzigen Element, wo man dieses CSS-Feature sinnvoll anwenden könnte, kann man es nicht anwenden. **W T F!**

Zur Erinnerung: `type="text"` zu setzen ist ja gerade das, was wir nicht wollen.

Option 3: `input-security` *in CSS* ist (fast) dasselbe als Versuch eines Standards; Browserunterstützung: keine. Wird wohl auch nicht kommen, denn wie im aktuellen Editor's Draft zu lesen steht: *"The CSS-WG has agreed that while be believe that providing this piece of functionality to users is important, doing it via CSS+JS is the wrong approach, and that instead it should be built into user agents."*

Ähm, sag ich doch:

> Es wird Zeit, dass die Passwort-anzeigen-Funktion endlich dort implementiert wird, wo sie hingehört: in den Browsern! Das Password muss im Klartext angezeigt werden können, ohne dass der Typ des Eingabefeldes von `password` auf `text` geändert wird.

Dort verlinkt: ein Github-Issue zum Entfernen von `input-security` aus dem Standard. Darin eine Diskussion vom Januar 2022. Was hat sich seitdem getan? `input-security` steht immer

"Websites that are hard to use frustrate customers, forfeit revenue and erode brands." —Forrester Research, 1998

Free online version of my book. Read Mobile First.

Mobile Design Details: Hide/Show Passwords

by Luke Wroblewski November 6, 2012

Passwords on the Web have long been riddled with usability issues. From overly complex security requirements to difficult to use input fields, passwords frequently result in frustrated customers and lost business. The situation is even worse on mobile where small screens and imprecise fingers are the norm. But what can we do?

Once again, the constraints and capabilities of mobile devices challenge us to to rethink long-standing design standards. In this case, the password input field. You're all intimately familiar with this interaction because the average person logs at least 15 times a day and 86% of companies in the United States use password authentication for their services. But just in case, here's how a typical password field works: you enter a character, it displays a "secure" response in the form of a •.

What's wrong with that, you may ask? Very simply put, there's no way for you to check your work by seeing what you entered. Which turns out to be very useful when you're forced to use a minimum amount of characters, some punctuation, and the birthdate of at least one French king for your password. So people often submit incorrect passwords and head into downward usability spirals.

"The fusion of art and technology that we call interface design." —Steven Johnson, 1997

Showing Passwords on Log-In Screens

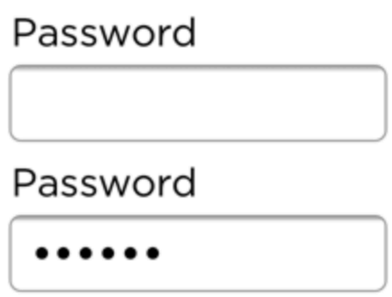
by Luke Wroblewski January 22, 2015

In 2012 I outlined why we should let people see their password when logging in to an application -especially on mobile devices. Now two years later with many large scale implementations released, here's a compendium of why and how to show passwords and what's coming next.

Why Show Passwords?

Passwords have long been riddled with usability issues. Because of overly complex security requirements (a minimum number of characters, some punctuation, the birthdate of at least one French king) and difficult to use input fields, password entry often results in frustrated customers and lost business.

Around 82% of people have forgotten their passwords. Password recovery is the number one request to Intranet help desks and if people attempt to recover a password while checking out on a e-commerce site, 75% won't complete their purchase. In other words, passwords are broken. And the situation is worse on mobile where small screens and imprecise fingers are the norm.



Masking passwords makes complex input even harder while doing little to improve security -especially on mobile where there's a visible touch keyboard directly below every input field that highlights each key as you press it. These bits of feedback show the characters in your password at a larger size than most input fields. So in reality,

HTML

```html
<label for="password">Password</label>
<input type="password" id="password" />
```

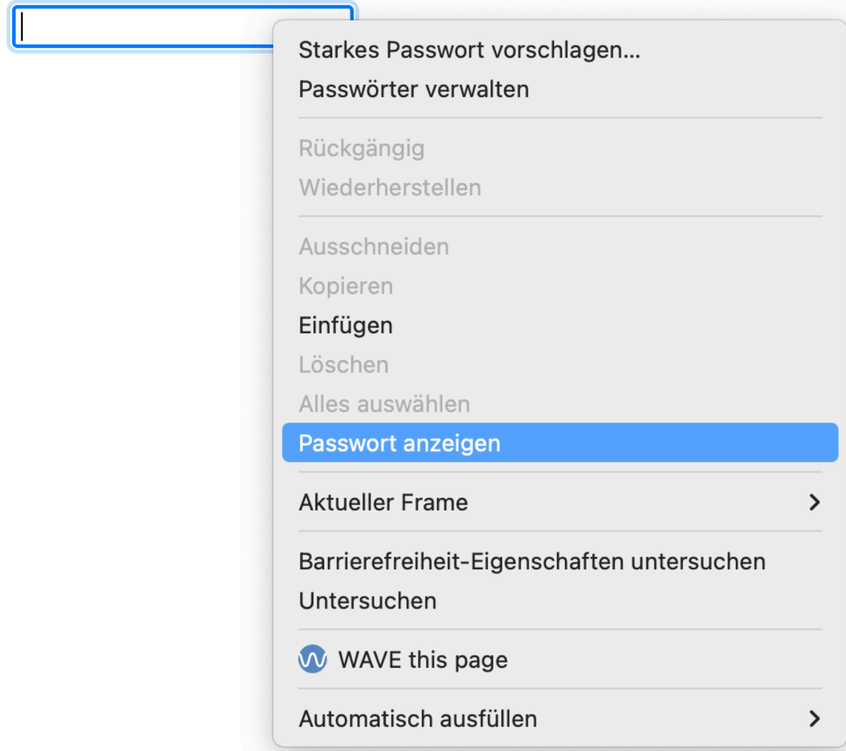
CSS

```css
label {
  display: block;
  width: fit-content;
  font: 1em/1.2 Lato, sans-serif;
  margin-block-end: .2em;
}

input#password {
  -webkit-text-security: none; /* doesn't work for <input type="password"> */
}
```

JS

Password



HTML

```html
<label for="password">Password</label>
<input type="password" id="password" />

```

CSS

```css
label {
  display: block;
  width: fit-content;
  font: 1em/1.2 Lato, sans-serif;
  margin-block-end: .2em;
}

input#password {
  -webkit-text-security: none; /* doesn't work for <input type="password"> */
}
```

JS

Password

HTML

```html
<label for="password">Password</label>
<div class="password-widget">
  <input
    type="password"
    id="password"
    spellcheck="false"
    textprediction="false"
    writingsuggestions="false"
  />
  <button aria-pressed="false">
    <span aria-hidden="true">👁</span>
    <span class="visually-hidden">reveal password</span>
  </button>
</div>
```

CSS

JS

```js
for (const passwordWidget of document.querySelectorAll('.password-widget')) {
  passwordWidget.passwordInput =
passwordWidget.querySelector('input[type="password"]');

  passwordWidget.addEventListener('click', event => {
    const buttonElement = event.target.closest('button');
    if (buttonElement) {
      const isRevealed = buttonElement.getAttribute('aria-pressed') ==
```

Password

SCHLIESSEN

Anmelden | Registrieren

E-Mail

Passwort

••••••••••

Anmelden

Passwort vergessen

oder

Über Google anmelden

T+ Spekulation … ion beginnt das Wer-wird-was-Spiel

INPUTS PASSWORDS SECURITY

The Options for Password Revealing Inputs

Chris Coyier on Oct 8, 2021

TOC

- Option 1: Use type="password", then swap it out for type="text" with JavaScript
- Option 2: Use -webkit-text-security in CSS
- Option 3: input-security in CSS
- Demos, all together

In HTML, there is a very clear input type for dealing with passwords:

```HTML
<input type="password">
```

If you use that, you get the obfuscated bullet-points when you type into it, like:

••••••••

That's the web trying to help with security. If it didn't do that, the classic problem is that someone could be peering over your shoulder, spying on what you're entering. Easier than looking at what keys your fingers press, I suppose.

TABLE OF CONTENTS

1 Introduction
1.1 Purpose

2 Module Interactions
2.1 Value Definitions

3 Outline properties
3.1 Outlines Shorthand: the ‘outline’ property
3.2 Outline Thickness: the ‘outline-width’ property
3.3 Outline Patterns: the ‘outline-style’ property
3.4 Outline Colors: the ‘outline-color’ property
3.5 Offsetting the Outline: the ‘outline-offset’ property

4 Resizing
4.1 Resizing Boxes: the ‘resize’ property

5 Pointing Devices and Keyboards
5.1 Pointer interaction
5.1.1 Styling the Cursor: the ‘cursor’ property
5.1.1.1 Cursor of the canvas
5.2 Insertion caret
5.2.1 Coloring the Insertion Caret: the ‘caret-color’ property
5.2.2 Animation of the insertion caret: ‘caret-animation’
5.2.3 Shape of the insertion caret: ‘caret-shape’
5.2.4 Insertion caret shorthand: ‘caret’
5.3 Keyboard control
5.3.1 Directional Focus Navigation: the ‘nav-up’, ‘nav-right’, ‘nav-down’, ‘nav-left’ properties

§ 7.4. Obscuring sensitive input: the ‘input-security’ property

ISSUE 13 The CSS-WG has agreed that while be believe that providing this piece of functionality to users is important, doing it via CSS+JS is the wrong approach, and that instead it should be built into user agents: this needs to work consistently from site to site for it to be discoverable and understandable by users, this needs to work even when JS is turned off, this needs to have consistently solid accessibility… We therefore intend to remove this from the specification and instead, we would like to see this behavior specified in the HTML specification as part of the interaction model of password fields. Holding off deleting until the situation with HTML is clarified. See https://github.com/w3c/csswg-drafts/issues/6788 and https://github.com/whatwg/html/issues/7293.

Name:	***‘input-security’***
Value:	auto | none
Initial:	‘auto’
Applies to:	sensitive text inputs
Inherited:	no
Percentages:	N/A
Computed value:	as specified
Canonical order:	per grammar
Animation type:	by computed value type

For the purpose of this specification, a ***sensitive text input*** is a text input whose purpose is to accept sensitive input, as defined by the host language. For example, in HTML `<input type=password>` is a sensitive text input.

<input type=password> should provide UI to show/hide its value #7293

New issue

Open **MatsPalmgren** opened this issue on Nov 3, 2021 · 16 comments

MatsPalmgren commented on Nov 3, 2021

`<input type=password>` obscures its value by default, however users may wish to temporarily disable this obscuring in order to confirm that they've typed their sensitive information correctly. I propose that the password control should provide built-in UI for this function. For example:

(There is a proposed CSS `input-security` property to enable web authors to provide this function, but I think that would be harmful to users and thus I strongly believe `input-security` should **not** be added to the web-platform, as I've explained in w3c/csswg-drafts#6788.)

The right solution to fulfill this user need is for password controls to have built-in UI for it. I think the HTML spec should at least recommend that UAs implement it as a best practice.

I hope other UA vendors will join me in adding this feature for all users on all platforms. It would be great if we could converge on similar behavior and iconography for it.
(Microsoft Edge already provides this UI by default. The work to add this feature to Gecko is done in bug 502258 and I'm making it look/behave roughly as in Edge.)

24 1 2

annevk added the topic: forms label on Nov 3, 2021

Assignees

No one assigned

Labels

topic: forms

Milestone

No milestone

Development

No branches or pull requests

Notifications Customize

Subscribe

You're not receiving notifications from this thread.

11 participants

Jared Wein [:jaws] (please needinfo? me)
Comment 29 • 9 years ago

(In reply to Ryan Feeley [:rfeeley] from ~~comment #28~~)

> Created ~~attachment 8680842 [details]~~
> Hide/hide no divider

I worry that the context menu will have low discoverability for the feature, but it will be a good v1. The 'no divider' option appears best to me, as the mockups with the dividers have a divider after almost every menuitem and it looks very zebra-stripey to me.

Alejandro Moreno Calvo
Comment 30 • 9 years ago

A good article about this topic by Luke Wroblewski.
http://www.lukew.com/ff/entry.asp?1941

Ryan Feeley [:rfeeley]
Comment 31 • 9 years ago

I think the contextual menu is the right place for this. Putting browser UI on web content is problematic in so many cases (even though Apple now does it in Safari for passwords). I'm deferring to Stephen on this one, unless there are some big UX security issues I'm missing. The idea is that you can enable this per field and it is not saved for previous visits. Stephen, which above the above 4 options do you think works? (if any)

Flags: needinfo?(shorlander)

Ryan Feeley [:rfeeley]
Comment 32 • 9 years ago

(In reply to Alejandro Moreno Calvo from ~~comment #30~~)

> A good article about this topic by Luke Wroblewski.
> http://www.lukew.com/ff/entry.asp?1941

Love this article and referred to it heavily for password manager work, and even Firefox Accounts.

Andrew Sutherland [:asuth] (he/him)
Comment 33 • 9 years ago

Rolf B: [Deprecation]The 'textprediction' attribute will be removed in the future. Use the 'writingsuggestions' attribute instead.

Beitrag lesen

SELF-Forum ~~[Deprecation]The 'textprediction' attribute will be removed in the future. Use the 'writingsuggestions' attribute instead.~~ +2

~~Rolf B~~ ~~2024-09-06 10:29~~ datenschutz - html - ki

Hallo alle,

diese Meldung bringt mir Edge gerade in den Developer Tools, in der Quelltextansicht, für irgendeinen blöden `if`. Offenbar wird diese Quelltextansicht per HTML erzeugt und offenbar möchte Edge nicht, dass der Copilot da mitquatscht.

Aber...

textprediction? writingsuggestions? Schon mal gehört?

Das ist offenbar ein neues HTML Attribut in Chromia 124+, mit denen sich ein eventuell vorhandener KI-Copilot auf dem Gerät des Benutzers **deaktivieren** lässt. Als 'writingsuggestions' hat es sogar seinen Weg in die Spec geschafft.

```
<label>Ungeholfene Eingabe:
<input type="text" writingsuggestions="false"></label>
```

Das Böse daran ist, dass der Defaultwert dieses Attributs **true** ist. Ich habe keine Ahnung von diesen KI Copiloten und weiß darum nicht, ob die rein lokal agieren oder ihre Dünntelligenz von einem Server bekommen. Woraus folgt: Privatsphärenalarm!

Weiß jemand mehr darüber?

Rolf