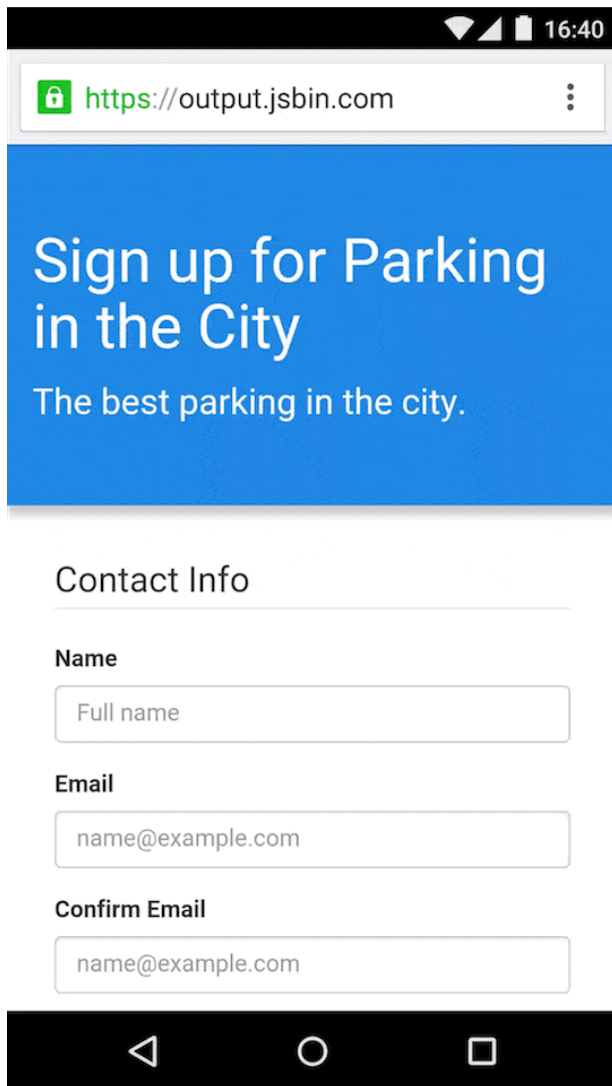


# Help users checkout faster with Autofill



By Ido Green

Ido is a Developer Advocate and a runner



The screenshot shows a mobile browser interface. At the top, the status bar displays signal strength, Wi-Fi, battery, and the time 16:40. The address bar shows a secure connection to <https://output.jsbin.com>. The main content area has a blue background with the text "Sign up for Parking in the City" and "The best parking in the city." Below this is a form titled "Contact Info" with three input fields: "Name" (containing "Full name"), "Email" (containing "name@example.com"), and "Confirm Email" (containing "name@example.com"). The bottom of the screen shows the Android navigation bar with back, home, and recent apps buttons.

People hate filling out web forms, especially on mobile devices. They can be slow and frustrating to complete and often contain multi-page steps and validation issues. This leads to high user drop-off and frustration. To help make things easier for users, browsers have long been able to autocomplete fields on behalf of the user. Chrome took this a step further in 2011 by introducing Autofill, which fills in entire forms based on a user's Autofill profile.

Starting in the next major version of Chrome (M43), we're taking yet another step to help users fill out forms faster by expanding our support for credit cards and addresses in

Google. This means that the same information users use to purchase things inside of the Google Play store are now available to them on websites. By using the standard *autocomplete* attributes, you can ensure your users' happiness by helping Chrome autofill your checkout forms with 100% accuracy.

Autocomplete attributes are a way for you, the developer, to control how the browser should populate a given form field. For example, if you are expecting a street address you can hint to the browser that you are expecting it by using `autocomplete="address-line1"`. This prevents the browser from incorrectly guessing form fields on your website which can result in a poor user experience.

We've found that by correctly using autocomplete attributes on your forms, users complete them up to 30% faster. And since *autocomplete* is part of the [WHATWG HTML](#) standard, we hope that other browsers will support it in the near future.

In the past, many developers would add `autocomplete="off"` to their form fields to prevent the browser from performing any kind of autocomplete functionality. While Chrome will still respect this tag for autocomplete data, it will not respect it for autofill data. So when should you use `autocomplete="off"`? One example is when you've implemented your own version of autocomplete for search. Another example is any form field where users will input and submit different kinds of information where it would not be useful to have the browser remember what was submitted previously.

The most common *autocomplete* attributes are shown in the table below and are documented in [Web Fundamentals](#).

## Common Attributes

### Credit Card

name attribute	autocomplete attribute
ccname	cc-name
cardnumber cvc ccmmonth ccyear exp-date card-type	cc-number cc-csc cc-exp-month cc-exp-year cc-exp cc-type

```
<label for="frmNameCC">Name on card</label>  
<input name="ccname" id="frmNameCC" required placeholder="Full Name" autocomplete="cc-name">  
  
<label for="frmCCNum">Card Number</label>  
<input name="cardnumber" id="frmCCNum" required autocomplete="cc-number">
```

```
<label for="frmCCCVc">CVC</label>
<input name="cvc" id="frmCCCVc" required autocomplete="cc-csc">

<label for="frmCCExp">Expiry</label>
<input name="cc-exp" id="frmCCExp" required placeholder="MM-YYYY" autocomplete="c
```

## Name

name attribute	autocomplete attribute
name fname mname lname	name (full name) given-name (first name) additional-name (middle name) family-name (last name)

```
<label for="frmNameA">Name</label>
<input name="name" id="frmNameA" placeholder="Full name" required autocomplete="n
```



## Email

name attribute	autocomplete attribute
email	email

```
<label for="frmEmailA">Email</label>
<input type="email" name="email" id="frmEmailA" placeholder="name@example.com" re
```



```
<label for="frmEmailC">Confirm Email</label>
<input type="email" name="emailC" id="frmEmailC" placeholder="name@example.com" r
```

## Address

name attribute	autocomplete attribute
address city region province state zip zip2 postal country	For one address input: street-address For two address inputs: address-line1 , address-line2 address-level1 (state or province) address-level2 (city) postal-code (zip code) country

```
<label for="frmAddressS">Address</label>
<input name="ship-address" required id="frmAddressS" placeholder="123 Any Street"
```



```

<label for="frmCityS">City</label>
<input name="ship-city" required id="frmCityS" placeholder="New York" autocomplete="on">

<label for="frmStateS">State</label>
<input name="ship-state" required id="frmStateS" placeholder="NY" autocomplete="on">

<label for="frmZipS">Zip</label>
<input name="ship-zip" required id="frmZipS" placeholder="10011" autocomplete="on">

<label for="frmCountryS">Country</label>
<input name="ship-country" required id="frmCountryS" placeholder="USA" autocomplete="on">

```

## Phone

name attribute	autocomplete attribute
phone mobile country-code area-code exchange suffix ext	tel

```

<label for="frmPhoneNumA">Phone</label>
<input type="tel" name="phone" id="frmPhoneNumA" placeholder="+1-650-450-1212" required>

```

The autocomplete attributes can be accompanied with a section name, such as:

- **shipping** - given-name
- **billing** - street-address

It is recommended because it will make your markup easier to parse and understand. The browser will autofill different sections separately and not as a continuous form.

## An example of a payment form

```

<label for="frmNameCC">Name on card</label>
<input name="ccname" id="frmNameCC" required placeholder="Full Name" autocomplete="on">

<label for="frmCCNum">Card Number</label>
<input name="cardnumber" id="frmCCNum" required autocomplete="cc-number">

<label for="frmCCVC">CVC</label>
<input name="cvc" id="frmCCVC" required autocomplete="cc-csc">

<label for="frmCCExp">Expiry</label>
<input name="cc-exp" id="frmCCExp" required placeholder="MM-YYYY" autocomplete="cc-exp">

```

## Forms best practices

1. **Use labels on form inputs**, and ensure they're visible when the field is in focus. The label element provides direction to the user, telling them what information is needed in a form element. Each label is associated with an input element by placing it inside the label element. Applying labels to form elements also helps to improve the touch target size: the user can touch either the label or the input in order to place focus on the input element.
2. **Use placeholder to provide guidance** about what you expect. The placeholder attribute provides a hint to the user about what's expected in the input, typically by displaying the value as light text until the user starts typing in the element. Placeholders disappear as soon as the user starts typing in an element, thus they are not a replacement for labels. They should be used as an aid to help guide users on the required format and content.

## Demo

You can see it in action over at: [greenido.github.io/Product-Site-101/form-cc-example.html](https://greenido.github.io/Product-Site-101/form-cc-example.html)

Or check the code: <https://github.com/greenido/Product-Site-101>

---

*Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see our [Site Policies](#). Java is a registered trademark of Oracle and/or its affiliates.*

*Last updated July 2, 2018.*