Best Practices for Evergage-Friendly CSS and URL Structures

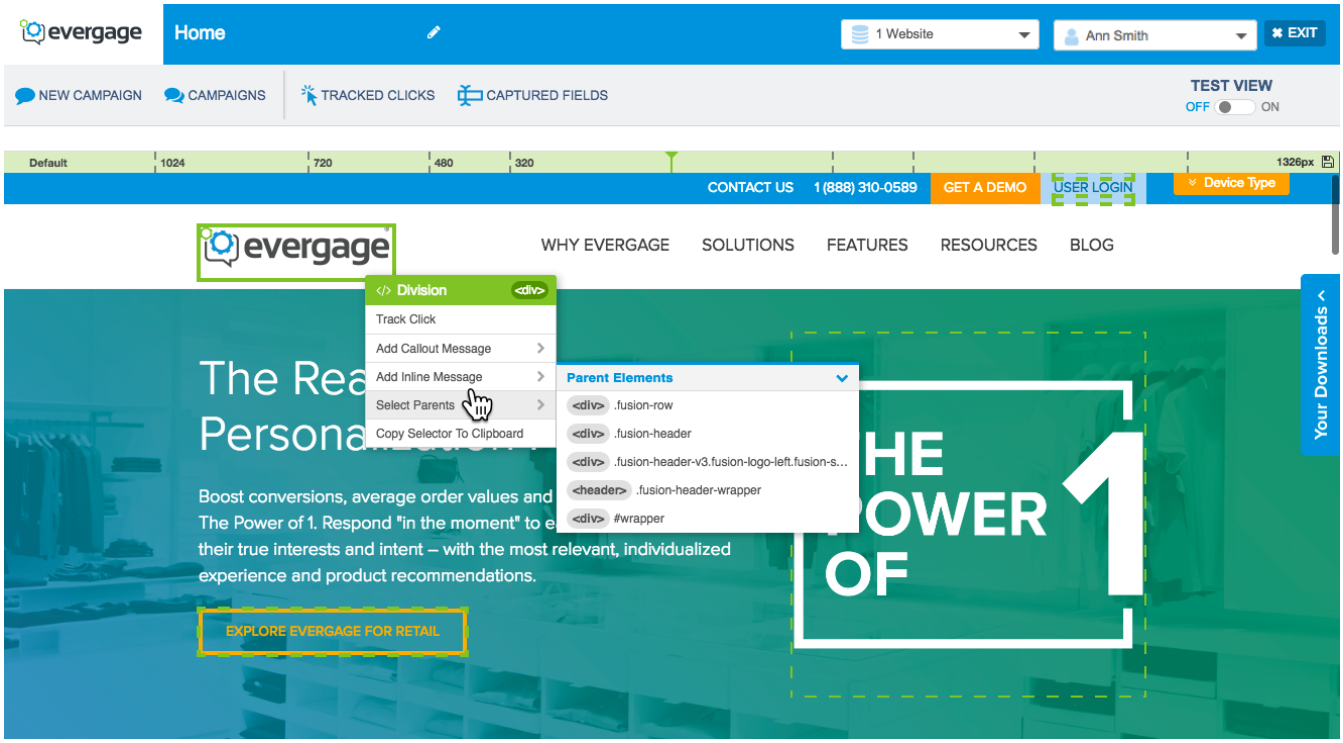
This guide covers the technical aspects of how Evergage uses the structure of your website to track behavior and respond with messages. In particular, we outline best practices for expanding or altering the structure of your site to avoid unexpected or unreliable behavior. While we strive to support the full variety of site structures, there are a few simple ways to guarantee total compatibility with Evergage.

This Article Explains
This section details how Evergage uses the structure of your website to track behavior and respond with messages.

Sections in this Article

CSS Selectors

- We use CSS selectors any time we need to target an element on your page. If you're not experienced with CSS selectors, see this introduction: https://css-tricks.com/how-css-selectors-work/
- To see the CSS selectors that we automatically determine for a given element, just have the Visual Editor open on your page, right click an element on the page, and go to Match Expression. There will be 2-3 suggested CSS selectors, with the recommended choice pre-selected:



- Some instances when a CSS selector is used are: tracking clicks, tracking fields, inline messages, and callouts. In these cases, the element being tracked, replaced, inserted next to or called out from needs to be able to have a **reliable** CSS selector. The ways we attempt to find a reliable CSS selector for an element are as follows, executed in this order:

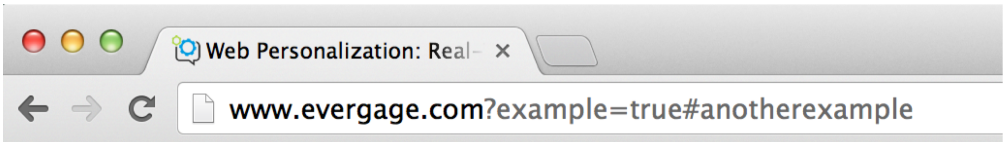
    1. We check whether the element or any parent has a "name" tag, which are most often used with input elements.
    2. We check whether the element or any parent has an ID (e.g. #example). (**We recommend** giving elements IDs if they will need a CSS selector.)
    3. We check whether the element or any parent has a unique class. That is, a class that is not shared with any other element on the page (e.g. .uniqueClass).
    4. If none of the previous apply, we use the body element as a unique parent, and assume the element will retain its relationship to the body element (for lack of a better option).

Some Warnings/Advice about using the above list:

- If you are relying on a parent with an ID, name, or a unique class to be the reference that all of its children will use for their CSS selectors, you must make sure that the relationship between that parent and its children remains consistent.
    - Example: A drop down menu has menu items that don't have IDs, names, or unique classes themselves, but the menu header has an ID. If you want to track clicks on the menu items, you'll need to make sure they appear in the same order on every page that the drop down appears, so that the relationship between the items and the header remains consistent.
    - Example: In the earlier screenshot, the match expression is "#block-block-203>div>div>h2". This means the giant orange title element found a parent with the id "#block-block-203", while the orange title itself is an h2 element, and the description of the relationship between the two is that the orange title is the child of a "div", which is a child of another "div", which is the child of "#block-block-203". If we use this match expression as our CSS selector, we will be relying on that relationship between the orange title and "#block-block-203" remaining the same.
- Be careful and sparing with depending on unique classes: sometimes a class will be unique on one page, but not on another. Relying on unique classes should be avoided if possible, and IDs should be used whenever possible.
    - Example: An item in a navigation bar that appears on every page may have a unique class on the page where you first add a click tracker to it, but that does not guarantee the class will remain unique on every page where the navigation bar appears. **If the class turns out to not be unique, we cannot guarantee the right element will be targeted, or that it will be the only element targeted.**
- Make sure that the IDs do not change what element they are associated between different pages or users. Doing so will lead to mismatched behavior, such as click trackers tracking different elements depending on the page or the user.
- If you believe the default CSS selector for an element may not be what you want, consider choosing a different selector from the Match Expression options. The alternate choices include using the text of the element itself (helpful if the text is something page-specific and brief, i.e. not a paragraph), the destination URL (helpful if it's a unique link), and other less common options. If no good option seems available, you can either try to create your own CSS selector (e.g. choose 'track click' if you want to track a click and it will allow you to input your own selector), or else you are welcome to contact support@evergage.com and we will help you solve the problem.

URLs

- We use page URLs to map pages to actions. This process is called action mapping. For example, if you want a message to appear on your home page, you must map your home page to the action 'Homepage' and then set the message to only appear on pages with the action 'Homepage'.
- Action mapping a single page can be more involved than it seems. A single page can have multiple URLs that lead to it. For instance, the URL for your home page probably has variations where the normal URL is appended by a '?' or '#' followed by extra parameters which are used by something on the page. These variations can easily be handled by our system, but often require you to map a couple different URLs of the same page to recognize that the variations exist.



- URLs should have a consistent structure to them, such that pages that will be mapped together (e.g. viewing a broom and viewing a vacuum could both map to View Product) can easily be generalized with the use of Regular Expressions. If you aren't familiar with Regular Expressions (RegEx), all you need to know is that it's a powerful tool for grouping together strings of text that are similar yet different ('bite' and 'bide' can be grouped with the single RegEx 'bi.e'). Similarly for URLs, '…/product/broom' and '…/product/vacuum' can be put in the same group with a thousand other products very easily, but it would be more tricky to group them with a product page with the URL ending in '…/extraproducts/specials/blender', since this URL differs from the pattern established by the other two.
- The rule of thumb is that pages you want to be mapped together should have the same number of forward slashes ('/') and have no more than two or three different possible patterns. In the product example above, the blender has one more forward slash--which is bad--but as long as all products follow one of the two patterns of '…/product/example' or '…/extraproducts/specials /example', then action mapping for View Product shouldn't be too troublesome.
- There is a corollary to this rule of thumb: pages that you do NOT want to be mapped together should have sufficiently different URLs. In the above example, '…/product/search' should NOT be the URL for the main product search page because it would then the same pattern as View Product pages. Rather, the search page should be '…/search', or '…/search/products' to distinguish it as a different kind of page.