CSS Selectors

This page corresponds to the textbook section on CSS selectors.

The default look of webpages made with the most basic HTML is quite unattractive. The aesthetically pleasing pages we see today are made using CSS. CSS is used to add style to webpages. The fact that all pages for a company have the same style is usually a result that they all use the same CSS file. The general way these CSS files work is by defining how each of the elements of a webpage will look. The title, headings, itemized lists, tables, and links, for example, each receive their own style including font, color, size, and distance from the margin, among others.

To do this CSS leverages patterns used to define these elements, referred to as *selectors*. An example of pattern we used in a previous video is `table` but there are many many more. If we want to grab data from a webpage and we happen to know a selector that is unique to the part of the page, we can use the `html_nodes()` function.

However, knowing which selector to use can be quite complicated. To demonstrate this we will try to extract the recipe name, total preparation time, and list of ingredients from this guacamole recipe. Looking at the code for this page, it seems that the task is impossibly complex. However, selector gadgets actually make this possible. SelectorGadget is piece of software that allows you to interactively determine what CSS selector you need to extract specific components from the webpage. If you plan on scraping data other than tables, we highly recommend you install it. A Chrome extension is available which permits you to turn on the gadget highlighting parts of the page as you click through, showing the necessary selector to extract those segments.

For the guacamole recipe page, we already have done this and determined that we need the following selectors:

```
h <- read_html("http://www.foodnetwork.com/recipes/alton-brown/guacamole
recipe <- h %>% html_node(".o-AssetTitle__a-HeadlineText") %>% html_text
prep_time <- h %>% html_node(".m-RecipeInfo__a-Description--Total") %>%
ingredients <- h %>% html_nodes(".o-Ingredients__a-Ingredient") %>% html
```

You can see how complex the selectors are. In any case we are now ready to extract what we want and create a list:

```
guacamole <- list(recipe, prep_time, ingredients)
guacamole
```

Since recipe pages from this website follow this general layout, we can use this code to create a function that extracts this information:

```
get_recipe <- function(url){
    h <- read_html(url)
    recipe <- h %>% html_node(".o-AssetTitle__a-HeadlineText") %>% html_
    prep_time <- h %>% html_node(".m-RecipeInfo__a-Description--Total")
    ingredients <- h %>% html_nodes(".o-Ingredients__a-Ingredient") %>%
    return(list(recipe = recipe, prep_time = prep_time, ingredients = ir
}
```

and then use it on any of their webpages:

```
get_recipe("http://www.foodnetwork.com/recipes/food-network-kitchen/panc
```

There are several other powerful tools provided by **rvest**. For example, the functions `html_form()`, `set_values()`, and `submit_form()` permit you to query a webpage from R. This is a more advanced topic not covered here.