11.4.2

Using the HTML class and id Attributes

Robin knows that she wants to extract the title and summary sentence from the first article. So, she needs a way to refer to the HTML elements that contain them. To do so, she can use their CSS selectors.

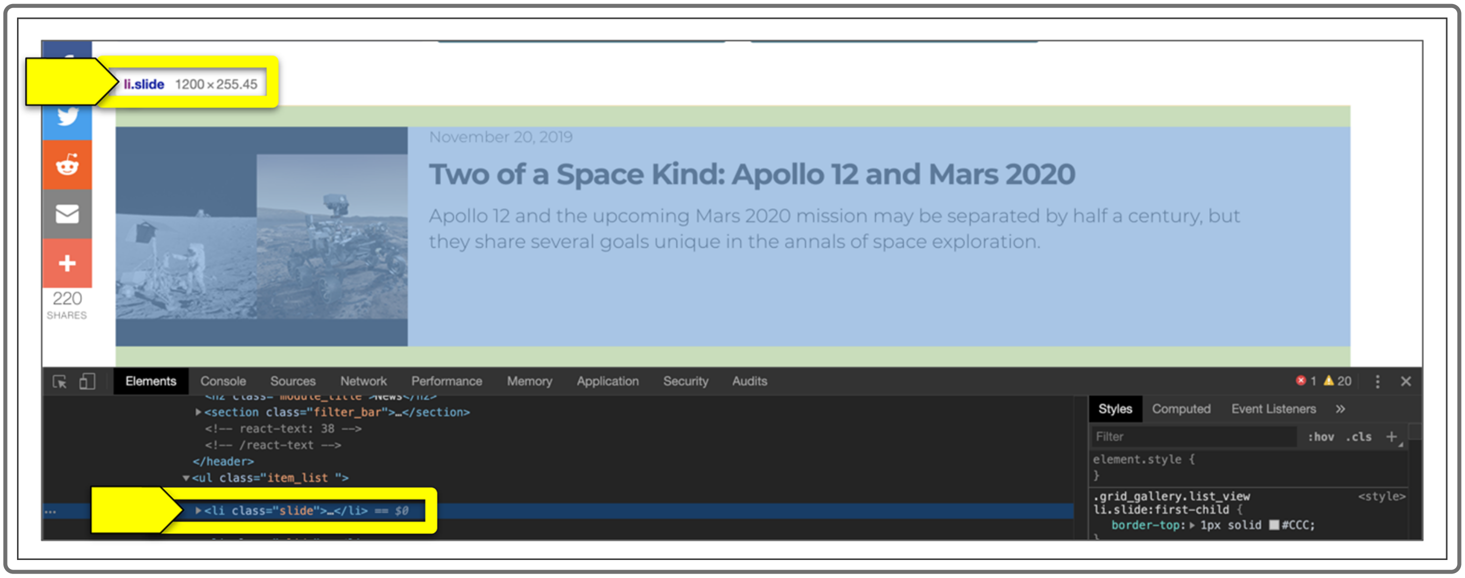
Because of the enormous number of HTML elements that a page might contain, it's important that the web developer keep the elements either unique or organized into groups. This organization can also make it easier to find what we're searching for during the web scraping process. Just as detectives might rely on certain tools to track down leads, we can rely on DevTools to help identify HTML elements when web scraping.

For example, when creating websites, developers differentiate one `div` element from another by adding a CSS selector, like `id` or `class`, as an attribute of the HTML element. So when web scraping, we can use those selectors to target the elements that we want to extract data from.

Applying Chrome DevTools

Let's revisit Robin's task: she wants to identify the elements that contain the title and summary sentence from the first article.

Let's examine DevTools in the browser again. This time, we'll more closely examine the nested elements. We want to find the element that contains only the first article on the page. This exists within the first `<li />` element that has a class of "slide". (You can verify this by examining the element that contains the `<li />` element: the `<ul />` element that has a class of "item_list". If you select it in DevTools, all the news articles on the page get highlighted.)



November 20, 2019

Two of a Space Kind: Apollo 12 and Mars 2020

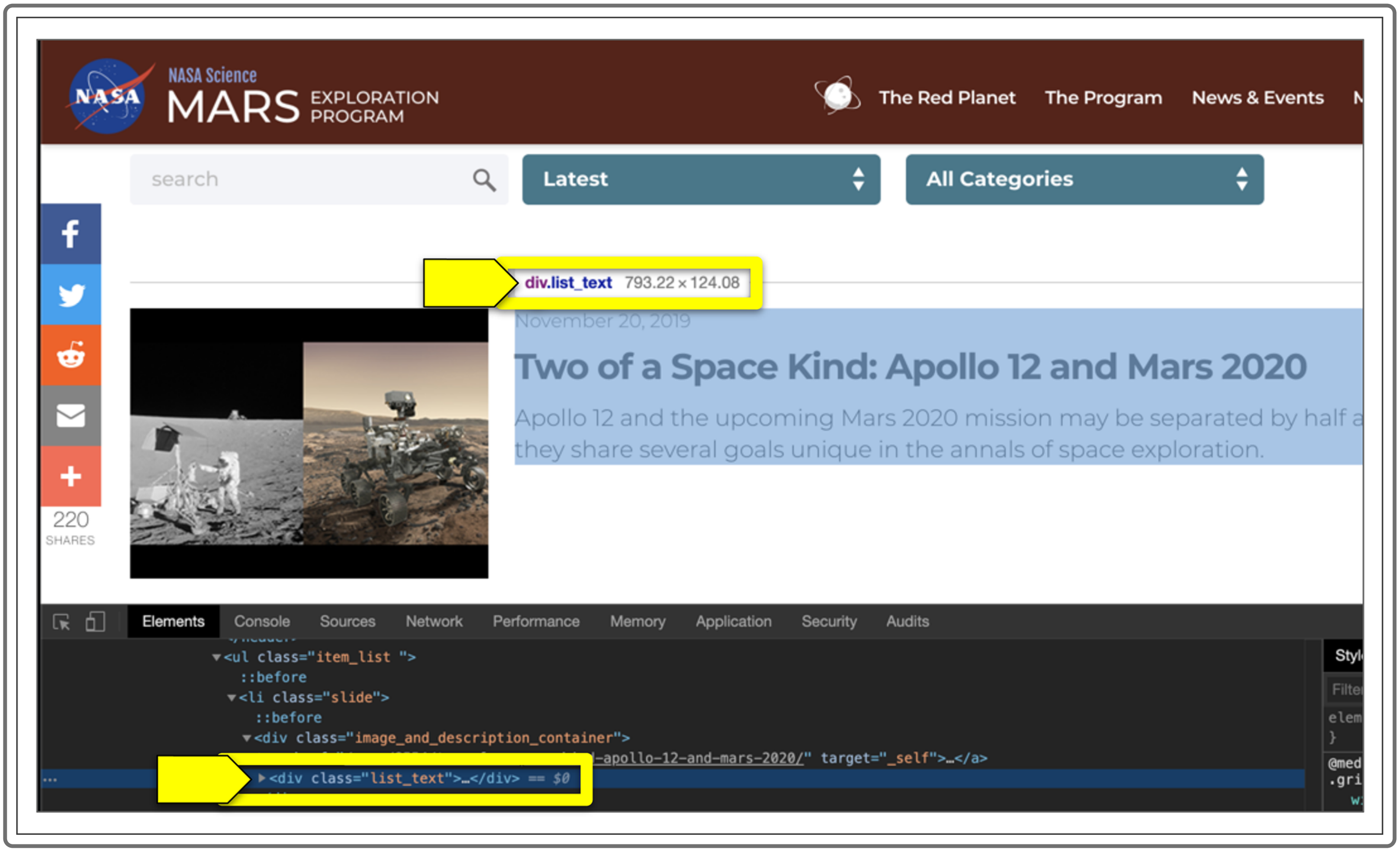
Apollo 12 and the upcoming Mars 2020 mission may be separated by half a century, but they share several goals unique in the annals of space exploration.

```
   ▸<section class="filter_bar">…</section>
    <!-- react-text: 38 -->
    <!-- /react-text -->
   </header>
  ▾<ul class="item_list ">
   ▸<li class="slide">…</li> == $0
```

```
element.style {
}
.grid_gallery.list_view        <style>
li.slide:first-child {
  border-top: ▸ 1px solid #CCC;
```

The content that we want (that is, the article title and summary sentence) is nested further in. And, we need to take numerous steps to get there.

First, expand the `<li class="slide">` element (if it isn't already expanded). In the expanded content, notice another element: a `div` element that has a class of "image_and_description_container". Expand that element, as well. Within that, notice yet another element: `<div class="list_text">`.



November 20, 2019

Two of a Space Kind: Apollo 12 and Mars 2020

Apollo 12 and the upcoming Mars 2020 mission may be separated by half a they share several goals unique in the annals of space exploration.

```
▾<ul class="item_list ">
   ::before
  ▾<li class="slide">
     ::before
    ▾<div class="image_and_description_container">
                              -apollo-12-and-mars-2020/" target="_self">…</a>
      ▸<div class="list_text">…</div> == $0
```

This final container holds the information that we want: the article title and summary sentence. With DevTools, we were able to maneuver through the nested HTML code to find the exact tags that we'll need to use in our web scraping code.

NOTE

> We call maneuvering through these nested elements **drilling down**. You'll often use this skill as you continue to work with HTML.

But, this process required lots of clicking to get to a single section of a webpage. So, let's try another approach that condenses all these steps.

Close your DevTools panel, and then consider the webpage one more time. Find the title and summary sentence of the first article, right-click either the summary sentence or the space immediately outside of it, and then click Inspect.

The DevTools panel opens. But this time, the highlighted section is closer to the element that we want. We can tell by hovering over the highlighted element. Recall that doing so simultaneously highlights the corresponding location on the webpage. So, using this method reduces the time that we need to spend drilling down into the elements on the webpage. We can then use the CSS selectors of these elements to do our web scraping. Our detective work is paying off!

Here's a final observation: The news items on the webpage share the same structure. All the article titles share the same `class`, for example. When we do our web scraping, we'll thus be able retrieve the information from all the news items all at once. We'll be able to do so by extracting the content of all the items that share the same `class` attribute on that page. That's a powerful technique that uses a `for` loop to automate the tedious work of collecting data, as you'll discover in the next lesson.

Fill in the blanks below:

What type of tag is used for the article title?

Is there a class attribute? If so, what is it? (Type Yes or No)

The article’s tag is nested within another tag; what is this tag and its attribute?

Check Answer

Finish ▶