

HTML

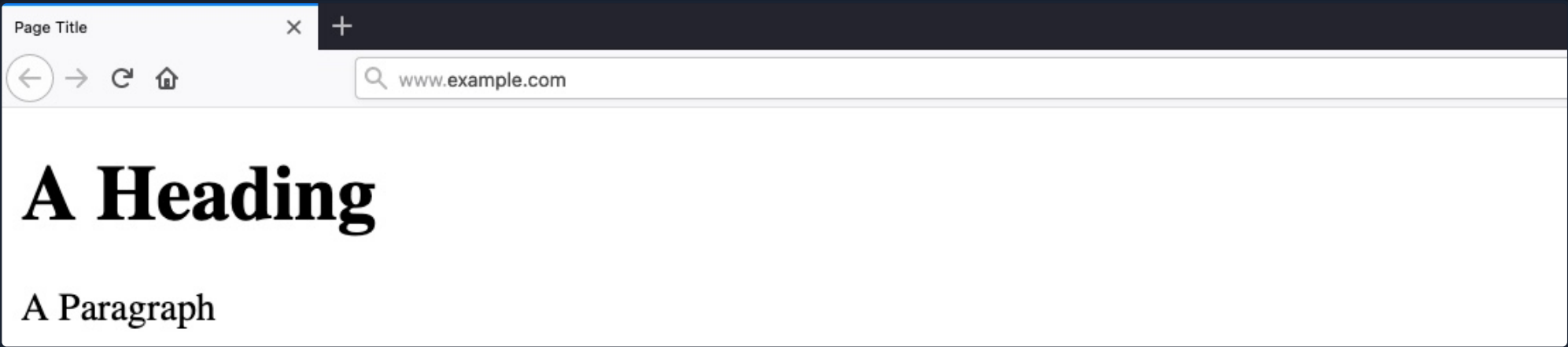
The first and most dominant component of the front end of web applications is [HTML \(HyperText Markup Language\)](#). HTML is at the very core of any web page we see on the internet. It contains each page's basic elements, including titles, forms, images, and many other elements. The web browser, in turn, interprets these elements and displays them to the end-user.

The following is a very basic example of an HTML page:

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>A Heading</h1>
    <p>A Paragraph</p>
  </body>
</html>
```

This would display the following:



As we can see, HTML elements are displayed in a tree form, similar to [XML](#) and other languages:

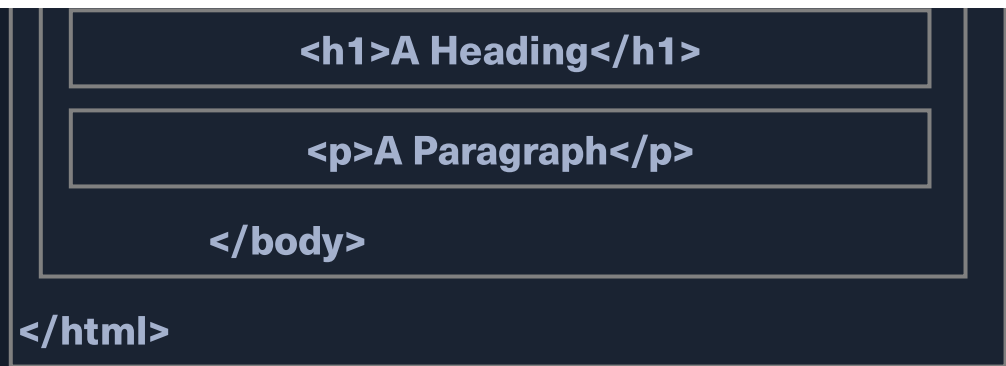
HTML Structure

```
document
- html
  -- head
    --- title
  -- body
    --- h1
    --- p
```

Each element can contain other HTML elements, while the main [HTML](#) tag should contain all other elements within the page, which falls under [document](#), distinguishing between [HTML](#) and documents written for other languages, such as [XML](#) documents.

The HTML elements of the above code can be viewed as follows:





Each HTML element is opened and closed with a tag that specifies the element's type 'e.g. `<p>` for paragraphs', where the content would be placed between these tags. Tags may also hold the element's id or class 'e.g. `<p id='para1'>` or `<p id='red-paragraphs'>`', which is needed for CSS to properly format the element. Both tags and the content comprise the entire element.

URL Encoding

An important concept to learn in HTML is [URL Encoding](#), or percent-encoding. For a browser to properly display a page's contents, it has to know the charset in use. In URLs, for example, browsers can only use [ASCII](#) encoding, which only allows alphanumerical characters and certain special characters. Therefore, all other characters outside of the ASCII character-set have to be encoded within a URL. URL encoding replaces unsafe ASCII characters with a `%` symbol followed by two hexadecimal digits.

For example, the single-quote character `'` is encoded to `'%27'`, which can be understood by browsers as a single-quote. URLs cannot have spaces in them and will replace a space with either a `+` (plus sign) or `%20`. Some common character encodings are:

Character Encoding

space	%20
!	%21
"	%22
#	%23
\$	%24
%	%25
&	%26
'	%27
(%28
)	%29

A full character encoding table can be seen [here](#).

Many online tools can be used to perform URL encoding/decoding. Furthermore, the popular web proxy [Burp Suite](#) has a decoder/encoder which can be used to convert between various types of encodings. Try encoding/decoding some characters and strings using this [online tool](#).

Usage

The `<head>` element usually contains elements that are not directly printed on the page, like the page title, while all main page elements are located under `<body>`. Other important elements include the `<style>`, which holds the page's CSS code, and the `<script>`, which holds the JS code of the page, as we will see in the next section.

Each of these elements is called a [DOM \(Document Object Model\)](#). The [World Wide Web Consortium \(W3C\)](#) defines `DOM` as:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

The DOM standard is separated into 3 parts:

- `Core DOM` - the standard model for all document types
- `XML DOM` - the standard model for XML documents
- `HTML DOM` - the standard model for HTML documents

For example, from the above tree view, we can refer to DOMs as `document.head` or `document.h1`, and so on.

Understanding the HTML DOM structure can help us understand where each element we view on the page is located, which enables us to view the source code of a specific element on the page and look for potential issues. We can locate HTML elements by their id, their tag name, or by their class name.

This is also useful when we want to utilize front-end vulnerabilities (like **XSS**) to manipulate existing elements or create new elements to serve our needs.

Questions

Answer the question(s) below to complete this Section and earn cubes!


+ 1 


What is the HTML tag used to show an image?

Submit your answer here...

 Submit

 Hint

 Previous

Next 





 [Go to Questions](#)

Table of Contents




Introduction to Web Applications

Introduction	
Web Application Layout	
Front End vs. Back End	


Front End Components

HTML
Cascading Style Sheets (CSS)
JavaScript

Front End Vulnerabilities

 Sensitive Data Exposure
 HTML Injection
 Cross-Site Scripting (XSS)
Cross-Site Request Forgery (CSRF)

Back End Components

Back End Servers
Web Servers
Databases
 Development Frameworks & APIs

Back End Vulnerabilities

Common Web Vulnerabilities

Common web vulnerabilities


Public Vulnerabilities

Next Steps

Next Steps

My Workstation

OFFLINE

 Start Instance

1 / 1 spawns left