

Getting Started

```
require 'hpricot'  
require 'open-uri'  
doc = Hpricot(open('http://www.somesite.com'))
```

doc is and Hpricot::Document

Both Hpricot Documents and Elements can have .search and .at called on them

doc.search('tag')	Returns ALL HTML tags of type "tag" specified within '' as an Array of Hpricot::Elements
doc.at('tag')	Returns FIRST HTML tag of type "tag" specified within '' as an Hpricot::Element

HTML tags you might look for: p, a, table, tr, td, span, div

Searching Help from CSS

If you are looking for a tag that has a class or id attribute, you can search how you would write CSS:

". " looks for a class

" # " looks for an id

doc.search('tag.value')	"value" is the value of the class attribute because of the "."
doc.search('tag#value')	"value" is the value of the id attribute because of the "#"
doc.search('.value')	does not look at any specific tags, returns any tag that has class="value"
doc.search('#value')	does not look at any specific tags, returns any tag that has id="value"

Example Searching:

```
<p class="find_me" id="one"> paragraph 1 </p>  
<p class="find_me too" id="two"> paragraph 2 </p>  
<span class="find_me" id="three"> I am not a paragraph </span>
```

doc.at('p.find_me')	returns paragraph 1 only
doc.search('too')	returns paragraph 2 only
doc.search('p.find_me')	returns paragraphs 1 AND 2 because it matches find_me in both classes
doc.search('.find_me')	returns both paragraphs AND the span because no tag was specified

Searching by Other Tag Attributes

You can also find a tag with another attribute, such as href or width or height or align, etc.

doc.search('tag[@attr]')	notice the "@" before the name of the attribute
--------------------------	---

Example Searching:

```
<a href="page.html"> relative link anchor tag </a>  
<a href="http://www.sagebit.com"> absolute link anchor tag </a>
```

doc.search('a[@href]')	returns both the anchor tags because they both have an href
doc.search('a[@href="page.html"]')	returns only the relative link anchor tag
doc.search('a[@href*="http"]')	returns only the absolute link; "*" is like string.match()

Searching by inner_text

You can also search by the inner_text of an element:

doc.search('tag[text()="some text"]')	notice instead of "@text" it is "text()"
---------------------------------------	--

Example Searching:

```
<p> Please find me! </p>  
<p> Please leave me alone! </p>
```

doc.search('p[text*="find"]')	returns only the top paragraph
doc.search('p[text()="leave me alone"]')	returns nothing, "=" is looks for an exact match

Once You Have What You're Looking For...

Hpricot::Elements have some fun methods!

elem.innerHTML	returns a string of the text AND HTML that are inside the element
elem.inner_text	returns ALL text inside the element including the text inside HTML tags
elem.search('tag')	returns ALL the HTML 'tag' elements inside of the element
elem.at('tag')	returns the FIRST 'tag' inside of the element
elem.parent	returns the Hpricot::Element that elem is immediately inside of
elem.next_sibling	returns the Hpricot::Element that is immediately after it in the Document

More Search Tips

When using @attribute= or @attribute*= make sure the quotes match up correctly:

('tag[@attribute= "hey "] ') or ("tag[@attribute= 'hey '] ")

Example where you construct the string to match ahead of time:

```
match_string = Date.today.strftime("%m-%d")  
doc.search("a[@href*=#{match_string}]")
```

Notice the lack of '' quotes and the "" quotes being on the outside

" = " matches exactly, "*" = " matches partially

Remember, .at gives you the FIRST element, .search gives you an ARRAY

If you want to get rid of an element that keeps you from getting what you want, (clean inner_text for example) Remove it!

doc.search('tag').remove	PERMANENTLY takes results of .search out of the Document
--------------------------	--

You must use .search for .remove to work. You have get a new Document if you mess up.