**Q) What does href=”#” mean?**

**A)**

**About hyperlinks:**

The main use of anchor tags - <a></a> - is as [**hyperlinks**](http://www.w3.org/MarkUp/html-spec/html-spec_7.html). That basically means that they take you somewhere. Hyperlinks require the href property, because it specifies a location.

**Hashtag:**

A hashtag - # within a hyperlink specifies an html element id to which the window should be scrolled.

href="#some-id" would scroll to an element on the current page such as <div id="some-id">.

href="//site.com/#some-id" would go to site.com and scroll to the id on that page.

**Scroll to Top:**

href="#" doesn't specify an id name, but does have a corresponding location - the top of the page. Clicking an anchor with href="#" will move the scroll position to the top.

[**See this demo.**](http://jsfiddle.net/yrmnmd1q/)

This is the expected behaviour according to the [**w3 documentation.**](http://www.w3.org/TR/html5/browsers.html#scroll-to-fragid)

## Hyperlink placeholders:

An example where a hyperlink placeholder makes sense is within template previews. On single page demos for templates, I have often seen <a href="#"> so that the anchor tag is a hyperlink, but doesn't go anywhere. Why not leave the href property blank? A blank href property is actually a hyperlink to the current page. In other words, it will cause a page refresh. As I discussed, href="#" is also a hyperlink, and causes scrolling. Therefore, the best solution for hyperlink placeholders is actually href="#!" The idea here is that there hopefully isn't an element on the page with id="!" (who does that!?) and the hyperlink therefore refers to nothing - so nothing happens.

Template previews:

For example, see this page and the links on it: [ironsummitmedia.github.io/startbootstrap-creative](http://ironsummitmedia.github.io/startbootstrap-creative/) Just html pages that are meant to demonstrate a set of CSS/HTML, WordPress themes, etc, but aren't real pages, so the links don't need to go anywhere.

## About anchor tags:

Another question that you may be wondering is, "Why not just leave the href property off??". A common response I've heard is that the href property is required, so it "should" be present on anchors. This is FALSE! The href property is required only for an anchor to actually be a hyperlink! Read [**this from w3**](http://www.w3.org/html/wg/drafts/html/master/links.html#attr-hyperlink-href). So, why not just leave it off for placeholders? Browsers render default styles for elements and will change the default style of an anchor tag that doesn't have the href property. Instead, it will be considered like regular text. It even changes the browsers behavior in regards to the element. The status bar (bottom of the screen) will not be displayed when hovering an anchor without the href property. It is most optimal, then, to use a placeholder href value on an anchor to ensure it is treated as a hyperlink.

[**See this demo**](http://jsfiddle.net/zpg537b3/) demonstrating style and behavior differences.

**Q) What does href=”javascript:void(0)” mean?**

A)

The void operator evaluates the given expression and then returns undefined.

The void operator is often used merely to obtain the undefined primitive value, usually using “void(0)” (which is equivalent to “void 0”). In these cases, the global variable [undefined](https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Global_Objects/undefined) can be used instead (assuming it has not been assigned to a non-default value).

An explanation is provided here: [void operator](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/void).

The reason you’d want to do this with the href of a link is that normally, a javascript: URL will redirect the browser to a plain text version of the result of evaluating that JavaScript. But if the result is undefined, then the browser stays on the same page. void(0) is just the **smallest script possible** that evaluates as undefined.

**Q) JavaScript onclick return functionality**

what is the difference in terms of these 2 statements?

<input type="checkbox" onclick="doAlert()" />

vs

<input type="checkbox" onclick="return doAlert();" />

Essentially both returned the same results and called the function, but is there more to it?

A)

onclick with return syntax will abort the effect of the checking. Without the word "return" will just run the function, and lose its returning value, as if you've wrote:

function doAlert() {

//alert("#");

var fail\_this\_check = true;

if(fail\_this\_check)

return false;

else

return true;

}

Function some\_local\_function won't return any value, although doAlert returns.

When you write "return", it's like you wrote the second function like this:

function some\_local\_function() {

return doAlert();

}

which preserves the returning value of doAlert, whatever value it would be. If it's true - the action will perform (the checkbox will be checked) - otherwise - it will cancel.

You can see live example here: <http://jsfiddle.net/RaBfM/1/>

Some html elements have JS events that behave differently when true/false is retur- ned. For instance:

<input type='submit' value='Click Me' onSubmit='ValidateForm();'>

...vs...

<input type='submit' value='Click Me' onSubmit='return ValidateForm();'>

In the second instance, if the ValidateForm function returned false the form will not submit, in the first even if the function returns false the form will still submit.

I think this scenario, you can see the different between using the return keyword and not.

# **Q)** **What do <form action=”#> and <form method=”post” action=”#”> do?**

# **A)**

action="" will resolve to the page's address. action="#" will resolve to the page's address + #, which will mean an empty fragment identifier.

Doing the latter might prevent a navigation (new load) to the same page and instead try to jump to the element with the id in the fragment identifier. But, since it's empty, it won't jump anywhere.

Usually, authors just put # in href-like attributes when they're not going to use the attribute where they're using scripting instead. In these cases, they could just use action="" (or omit it if validation allows).

# 

Action normally specifies the file/page that the form is submitted to (using the method described in the method paramater (post, get etc.))