**1.ARIA –**

the Accessible Rich Internet Applications Suite, defines a way to make Web content and Web applications more accessible to people with disabilities. It especially helps with dynamic content and advanced user interface controls developed with Ajax, HTML, JavaScript, and related technologies.

Means It’s a way of taking dynamic interactions in the browser such a way you do with JS and HTML and making it accessible, making information available to assistive technology such as screen readers. Its semantic meta language that can be communicated to any assistive technology.

ARIA is a group of properties that you can attach to an HTML element to give it more meaning.

<https://www.w3.org/WAI/intro/aria.php>

***2.LABEL - label and form associate using for & id attribute?***

#### **<label> tag**

The label tag is used to “name” input elements.

<label for="firstname">First name</label>

<input id="firstname" name="firstname" type="text">

The “for” attribute corresponds to the id of the input element. When the input receives focus, the label is read to the user as the name for the input element.

Some elements, not all of them [form-associated](https://www.w3.org/TR/html5/forms.html#form-associated-element), are categorized as **labelable elements**. These are elements that can be associated with a [label](https://www.w3.org/TR/html5/forms.html#the-label-element) element.

* [button](https://www.w3.org/TR/html5/forms.html#the-button-element)
* [input](https://www.w3.org/TR/html5/forms.html#the-input-element) (if the [type](https://www.w3.org/TR/html5/forms.html#attr-input-type) attribute is *not* in the [hidden](https://www.w3.org/TR/html5/forms.html#hidden-state-(type=hidden)) state)
* [keygen](https://www.w3.org/TR/html5/forms.html#the-keygen-element)
* [meter](https://www.w3.org/TR/html5/forms.html#the-meter-element)
* [output](https://www.w3.org/TR/html5/forms.html#the-output-element)
* [progress](https://www.w3.org/TR/html5/forms.html#the-progress-element)
* [select](https://www.w3.org/TR/html5/forms.html#the-select-element)
* textarea

**3.html table with row header & colmun header tag?**

Using aria-labelledby, we can specify the header or headers that should label each input.

To make complex HTML tables more easily understandable, for accessibility, we must **ensure that the headers, column groups, row groups, etc. are clearly defined**.

tables, use the <th> element to identify the header cells and the scope attribute to declare the direction of each header. The scope attribute can be set to row or col to denote that a header applies to the entire row or column, respectively.

For eg see <https://www.mediacurrent.com/blog/accessible-names-label-all-things-part-2>

<tr>

<td></td>

<th>Monday</th>

<th>Tuesday</th>

<th>Wednesday</th>

<th>Thursday</th>

<th>Friday</th>

</tr>

<tr>

<th>09:00 - 11:00</th>

<td>Closed</td>

<td>Open</td>

<td>Open</td>

<td>Closed</td>

<td>Closed</td>

</tr>

<tr>

<th>11:00 - 13:00</th>

<td>Open</td>

<td>Open</td>

<td>Closed</td>

<td>Closed</td>

<td>Closed</td>

</tr>

<tr>

<th>13:00 - 15:00</th>

<td>Open</td>

<td>Open</td>

<td>Open</td>

<td>Closed</td>

<td>Closed</td>

</tr>

<tr>

<th>15:00 - 17:00</th>

<td>Closed</td>

<td>Closed</td>

<td>Closed</td>

<td>Open</td>

<td>Open</td>

</tr>

</table>

EXAMPLE:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Delivery slots: | | | | | |
|  | **Monday** | **Tuesday** | **Wednesday** | **Thursday** | **Friday** |
| **09:00 - 11:00** | Closed | Open | Open | Closed | Closed |
| **11:00 - 13:00** | Open | Open | Closed | Closed | Closed |
| **13:00 - 15:00** | Open | Open | Open | Closed | Closed |
| **15:00 - 17:00** | Closed | Closed | Closed | Open | Open |

Complex Tables - In such tables, each table header is represented by a (document-wide) unique id. Data cells refer to those ids by listing one or more in their headers attribute, separated by spaces. Tables with multiple headers may also need to have a caption to identify it and a summary to describe the layout of the table

The table head and table foot should contain information about the table's columns. The table body should contain rows of table data.

1. Header cells must be scoped – in effect, they must tell the screen reader whether the data they refer to is in a column or in a row.
2. Nested headers – headers and subheaders that are their direct children, must each have unique IDs. As in webpages, IDs must be unique within the document.
3. Data cells that are children of the nested headers must have code that identifies which headers are associated with them, so that the screen reader will read the nested headers in descending order and then read the data cell contents every time a new cell is focused.

**SCOPE ATTRIBUTE** – This attribute specifies the set of data cells for which the current header cell provides header information.

In other words it helps us associate the data cells with their corresponding header cells.

The scope attribute can have any one of these four values; col, row, rowgroup, colgroup to refer to a column’s header, a row’s header, a group of columns’ header and a group of rows’ header respectively.

## Caption & Summary

Each table may have an associated caption (see the [CAPTION](https://www.w3.org/TR/html401/struct/tables.html#edef-CAPTION) element) that provides a short description of the table's purpose. A longer description may also be provided (via the [summary](https://www.w3.org/TR/html401/struct/tables.html#adef-summary) attribute) for the benefit of people using speech or Braille-based user agents.

## Column Group –

Just adding colspan="2" does not create the column groups, it only indicates that the particular cell is to occupy two cells’ worth of space.

**Row Groups** *-* Row groups can not be created using a tag like colgroup because there is no rowgroup element.

HTML rows are generally grouped using <thead>, <tbody> and <tfooter>elements. A single HTML <table> element can have one <thead>, one <tfoot>and multiple <tbody>. We can use multiple tbody in our table to create the row groups, and add the respective scope to header cells.

<http://www.hongkiat.com/blog/html-table-accessibility/>

#### <https://www.w3.org/TR/html401/struct/tables.html#h-11.2.4>

**4.getelementbyid to retrieve element?**

The most common way to access an HTML element is to use the id of the element.

The most common way for you to reference an element is by accessing an HTML element by its id. The id is a *unique* name you give an element in your HTML. Note the emphasis on unique. Your document can only have one unique id value in it. The getElementByID function is always called on your document, and it takes one argument - the id of the element you are trying to reference. You can declare a variable to store a pointer to the returned element:

In the example above the getElementById method used id="demo" to find the element.

**5.title attribute( screen reader picks title attribute) ?**

The title attribute specifies extra information about an element.

The information is most often shown as a tooltip text when the mouse moves over the element.

The **title** [global attribute](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes) contains text representing advisory information, related to the element it belongs to.  Some typical uses:

Link: title or description of the linked document

Media element like an image: description or associated credits

Paragraph: footnote or a commentary about it

Quotation: information about the author, and so on.

WHY NOT USE THE TITLE ATTRIBUTE?

Someone who uses a keyboard to navigate websites, such as a person who lacks the fine motor control required by a mouse, will never see your title attributes because they have no way to hover over elements.

A person who uses a screen reader because they are blind or have low vision will likely never have the title attribute read to them. Some screen readers don’t support the title attribute and those that do, don’t read the title attribute by default. It requires changing a setting for the title to be read, which not all users do.

Touchscreen devices such as phones and tablets don’t support title attributes which creates an accessibility issue for everyone and not just those with a disability.

WHAT NOT TO DO?

First of all, don’t rely on the title attribute to convey important information. For instance, if you need to alert a user to something that will happen when clicking on a link, that information should go in the link text, not a title attribute.

Next, don’t use it to duplicate text from elsewhere. If an image’s alt attribute already describes what needs to be described, don’t put it in the title attribute too. For those people that do have the title enabled in their assistive technology, it’s an annoyance to have to listen to the same thing twice.

WHAT TO DO?

In general, use of the title attribute is discouraged...with one exception. You should ALWAYS provide a title attribute for frame elements. This is supported in screen readers.

<frame title="Advertisement">

If you find yourself in a situation where you do need to provide complementary text, because of design constraints for instance, it’s better to either revise your link text when possible or use CSS to hide a portion of the link text.

For example:

<a href=”road-less-traveled.html”>Read more <span class=”element-invisible”> about The Road Not Taken, a poem by Robert Frost</span></a>

And if you really, really, really [insert temper tantrum here] need the title attribute for some reason, consider putting the text in a [data-\* attribute](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using_data_attributes) instead (i.e. data-title).

WAIT. ISN’T THE TITLE ATTRIBUTE GOOD FOR SEO?

By every account I’ve read, no. Not on the actual page where it resides, anyway. Since the title attribute can be used on every html element, it would be a great way to do some serious keyword stuffing so it would make sense that the search engines would ignore it. You’re much better off putting your keywords in the link text than the title attribute. (Disclaimer: I am not an SEO expert. If you have evidence to the contrary, please add a comment!)

SUMMARY

The title attribute provides little if any benefit at best and can be annoying or confusing to some users at worst. Avoid using it unless absolutely necessary...except for frame elements where it’s required.

- See more at: http://www.mediacurrent.com/blog/dont-rely-title-attribute-accessibility-seo#sthash.G9oVcZ6N.dpuf

**6.css focus selector ? -**

|  |  |  |
| --- | --- | --- |
| [:focus](https://www.w3schools.com/cssref/sel_focus.asp) | input:focus | Selects the input element which has focus |

In CSS, selectors are patterns used to select the element(s) you want to style.

:focus - The :focus selector is used to select the element that has focus. Select and style an input field when it gets focus. **Tip:** The :focus selector is allowed on elements that accept keyboard events or other user inputs.

input:focus {   
    background-color: yellow;  
}

**7.what does do javscript focus methods ?**

The focus() method is used to give focus to an element (if it can be focused). The focus event occurs when an element gets focus (when selected by a mouse click or by "tab-navigating" to it).

The focus() method triggers the focus event, or attaches a function to run when a focus event occurs.

**Tip:** Use the [blur()](https://www.w3schools.com/jsref/met_html_blur.asp) method to remove focus from an element.

An element receives a focus when the user either clicks on it or uses the Tab key on the keyboard. There’s also an autofocus HTML attribute that puts the focus into an element by default when a page loads and other means of getting a focus.

<!DOCTYPE html>

<html>

<body>

<p>In this example, the text field gets focus immediately after the document window has been loaded.</p>

<input type="text" id="myText" value="A text field">

<script>

window.onload = function() {

document.getElementById("myText").focus();

};

</script>

</body>

</html>

**9.FOCUS -** refers to selecting an element and then directing all of the keyword events to that element. To use this you need to make elements focusable. For this we use tabindex attribute.

Focus can be used as primary means of reaching everything on screen. To navigate the focus in the document is done using tab keys or shift-tab keys.

The order in which elements are inserted in the page/document are focused are named as tab order.

Some elements are implicitly included in tab order,Native controls like input, select, button these are all implicitly focusable. So we don’t have to do any additional work to make sure that the user can reach them with their keyboard.

Not all elements are focusable. focus/blur support is guaranteed for elements that a visitor can interact with: <button>, <input>, <select>, <a> and so on.

By default many elements do not support focusing. elements that exist to format something like <div>, <span>, <table> , <li>– are unfocusable by default. The method elem.focus() doesn’t work on them, and focus/blur events are never triggered.

This can be changed using HTML-attribute tabindex. Most elements do not support focus by default. Use tabindex to make anything focusable.

The current focused element is available as document.activeElement.

The purpose of this attribute is to specify the order number of the element when Tab is used to switch between them.

That is: if we have two elements, the first has tabindex="1", and the second has tabindex="2", then pressing Tab while in the first element – moves us to the second one.

There are two special values:

tabindex="0" makes the element the last one.

tabindex="-1" means that Tab should ignore that element.

**10.TABINDEX –** It is used when you built custom control and add your own keyboard support.

The **tabindex** [global attribute](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes) is an integer indicating if the element can take input focus (is *focusable*), if it should participate to sequential keyboard navigation, and if so, at what position. It can take several values:

* a negative value means that the element should be focusable, but should not be reachable via sequential keyboard navigation;
* Tabindex -1 is used where you need to manage focus, u don’t want to be interactive but move focus somewhere else in the page. Or you Want something that needs to be in DOM but maybe off screen Or Temporarily disable custom interactive control.
* 0 means that the element should be focusable and reachable via sequential keyboard navigation, but its relative order is defined by the platform convention;
* tab index ZERO means that element is programmatically focusable. So its called as focus method in Javascript , which directs focus to it.
* a positive value means should be focusable and reachable via sequential keyboard navigation; its relative order is defined by the value of the attribute: the sequential follow the increasing number of the tabindex. If several elements share the same tabindex, their relative order follows their relative position in the document.
* Tabindex greater than zero means that element will be jumped ahead of everything else in the natural tab order. Value greater than zero is considered as anti-pattern bcoz there are many components on page with different tabindex values then focus control is going to bounce like a pinball. Also, Screen Reader navigates the DOM in a linear fashion. So not necessarily users will land on higher tabindex controls before something else in the document. If u want something higher than zero as tabindex then move it earlier in the DOM.

An element with a 0 value, an invalid value, or no tabindex value should be placed after elements with a positive tabindex in the sequential keyboard navigation order

We can add tabindex from JavaScript by using the elem.tabIndex property. That has the same effect.

<https://www.w3.org/TR/html4/interact/forms.html#adef-tabindex>

**tabindex - no postive value is bad practice**

**0 = comes in tab order, doesn't fo tp text**

**-1 = element goes out of tab order**

**11.Onfocus Event –**

HTML DOM allows JavaScript to react to HTML events: Execute a JavaScript when an input field gets focus:

Enter your name: <input type="text" onfocus="myFunction(this)">

<p>When the input field gets focus, a function is triggered which changes the background-color.</p>

<script>

function myFunction(x) {

x.style.background = "yellow";

}

</script>

The onfocus event occurs when an element gets focus.

The onfocus event is most often used with <input>, <select>, and <a>.

<https://www.w3schools.com/jsref/event_onfocus.asp>

* HTML Attributes 🡪 Attributes provide **additional information** about an element

Declaring a language is important for accessibility applications (screen readers) and search engines:

<!DOCTYPE html>  
<html lang="en-US">  
<body>  
  
...  
  
</body>  
</html>

## The alt Attribute

The **alt** attribute specifies an alternative text to be used, when an image cannot be displayed.

## The class Attribute

The HTML class attribute makes it possible to define equal styles for elements with the same class name.

## HTML Screen Readers

A screen reader is a software program that reads the HTML code, converts the text, and allows the user to "listen" to the content. Screen readers are useful for people who are blind, visually impaired, or learning disabled.

## Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

Examples of block-level elements:

* <div>
* <h1> - <h6>
* <p>
* <form>

## Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

Examples of inline elements:

* <span>
* <a>
* <img>

## The <span> Element

The <span> element is often used as a container for some text.

The <span> element has no required attributes, but both **style** and **class** are common.

When used together with CSS, the <span> element can be used to style parts of the text:

HTML5 offers new semantic elements that define the different parts of a web page:

|  |  |
| --- | --- |
| HTML5 Semantic Elements | * <header> - Defines a header for a document or a section * <nav> - Defines a container for navigation links * <section> - Defines a section in a document * <article> - Defines an independent self-contained article * <aside> - Defines content aside from the content (like a sidebar) * <footer> - Defines a footer for a document or a section * <details> - Defines additional details * <summary> - Defines a heading for the <details> element |

To display a less than sign (<) we must write: **&lt;** or **&#60;**

 A common character entity used in HTML is the non-breaking space: **&nbsp;**

A non-breaking space is a space that will not break into a new line.

UTF-8 (Unicode) covers almost all of the characters and symbols in the world. To display an HTML page correctly, a web browser must know the character set used in the page.

<meta charset="UTF-8">

## 12. CSS Selectors - In CSS, selectors are patterns used to select the element(s) you want to style.

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

## The element selector selects elements based on the element name.

The selector points to the HTML element you want to style.

You can select all <p> elements on a page

**The id selector** uses the id attribute of an HTML element to select a specific element.

The id of an element should be unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

**The class selector** selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class.

**13. Things to consider for Website Accessibility**

1.Separating content from presentation is the key to (almost) everything.

2. Do not use tables for layout,

3.Do not use inline HTML styles: Use CSS. Write valid HTML code

4.Do not use popups and fancy effects.

**GENERAL NOTES**

**Testing website http://www.nextgenzone.org/**

**< a href =”#”></a> - # means**

## 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.

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.

< a href = “/”>

<i></i> tag

data-toggle

data-target