Accessible Web Applications

Best Practices & Common Mistakes

Rabab Gomaa

@RubysDo

gomaarabab@gmail.com

1

Hello/Bonjour!

@RubysDo

Rabab Gomaa

IAAP CERTIFIED WAS **Web Accessibility Specialist** International Association *of* Accessibility Professionals

- Digital accessibility advocate
- Web accessibility tester & instructor
- Front-end development and web design experience

@RubysDo - gomaarabab@gmail.com

2

Recap: Web Accessibility (a11y)

- ❑ What web accessibility means
- ❑ What motivate a company to make their digital platform accessible

@RubysDo - gomaarabab@gmail.com

3

What web accessibility means

"The Web is fundamentally designed to work for **all people**, whatever their hardware, software, language, culture, location, or physical or mental ability. When the Web meets this goal, it is accessible to people with a diverse range of hearing, movement, sight, and cognitive ability." (*W3C - Accessibility*)

Accessible = usable for all

@RubysDo - gomaarabab@gmail.com

4

What motivate a company to make their digital platform accessible

It is the right thing to do!

POST COVID-19 REVENUE
Massive shift to online shopping and digital services

BUSINESS OPPORTUNITIES
Providing VPAT* is a requirement prior to procurement

LEGAL OBLIGATIONS
Risk of getting sued for inaccessible web sites

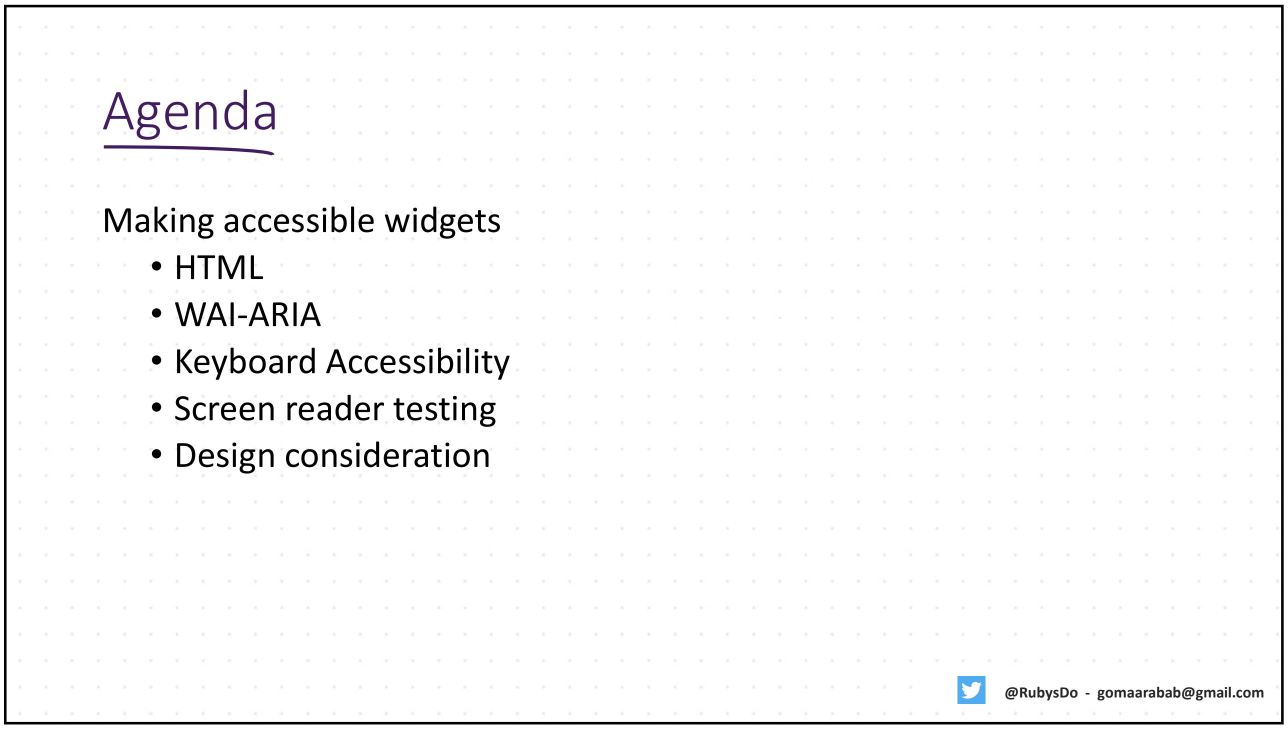
OPERATIONAL COST REDUCTION
Less load on other customer service channels i.e., call centers

*VPAT (Voluntary Product Accessibility Template)

5

The ability to produce accessible code is a selling point for developers and IT professionals services and skills!

6



Agenda

Making accessible widgets
- HTML
- WAI-ARIA
- Keyboard Accessibility
- Screen reader testing
- Design consideration

@RubysDo - gomaarabab@gmail.com

7

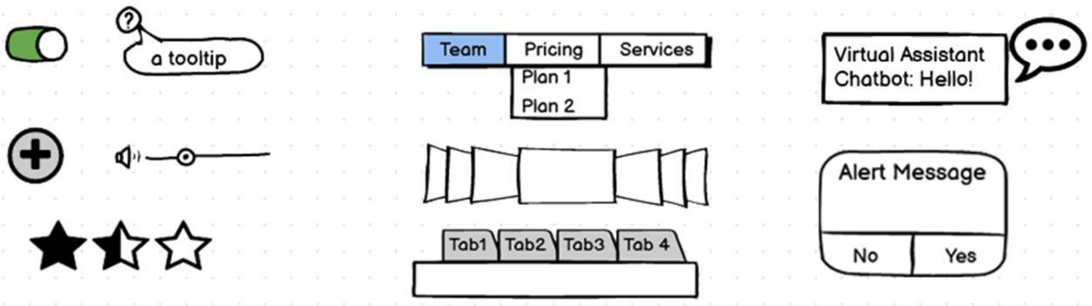
Making accessible widgets

@RubysDo - gomaarabab@gmail.com

8

Widgets

Widgets are these custom building blocs created by the developers.
They are developed using HTML, JavaScript & CSS.



Controls

UI Components

Complex
UI Components

@RubysDo - gomaarabab@gmail.com

9

Widgets: Accessibility Design Patterns

The patterns are building instruction for frontend developers to follow when creating accessible widgets.

Patterns & guides:

- W3C WAI-ARIA Authoring Practices
  https://www.w3.org/TR/wai-aria-practices-1.1/
- eBay MIND Patterns
  https://ebay.gitbook.io/mindpatterns/
- Mozilla MDN Web Docs
  https://developer.mozilla.org/en-US/docs/Web/Accessibility

@RubysDo - gomaarabab@gmail.com

10

Accessible Widget Components

HTML: Semantic foundation

@RubysDo - gomaarabab@gmail.com

11

Accessible Widget Components

HTML: Semantic foundation

WAI-ARIA: Features needed for assistive technologies (AT)

12

Accessible Widget Components

KEYBOARD
ARIA
HTML

HTML: Semantic foundation

WAI-ARIA: Features needed for assistive technologies (AT)

Keyboard : Focus management & interaction (HTML, CSS & JS)

13

Accessible Widget Components

yummy
KEYBOARD
ARIA
HTML

HTML: Semantic foundation

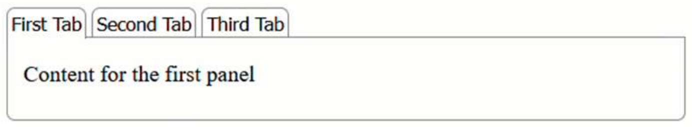
WAI-ARIA: Features needed for assistive technologies (AT)

Keyboard : Focus management & interaction (HTML, CSS & JS)

Screen reader: Testing for blind users

14

Tabbed Interface (Tabs) Pattern



Content for the first panel

Terminologies

- **Tab List:** A set of tab elements contained in a tablist element.
- **Tab:** An element in the tab list that serves as a label for one of the tab panels and can be activated to display that panel.
- **Tabpanel:** The element that contains the content associated with a tab.

@RubysDo - gomaarabab@gmail.com

15

HTML: Accessible Widget Components

HTML: Semantic foundation

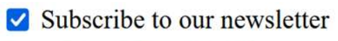
@RubysDo - gomaarabab@gmail.com

16

Native HTML

Use native HTML semantic elements when structuring your widgets to benefit from their built-in accessibility i.e. <input> <button>, <ul>

Native Checkbox


☑ Subscribe to our newsletter

```
<!--native HTML checkbox -->
  <input type="checkbox" id="nletter">
  <label for="nletter" id="nletter-lb">
  Subscribe to our newsletter
  </label>
```

Accessibility Tree
Accessible name: “Subscribe to our newsletter“
Role: checkbox
State: checked

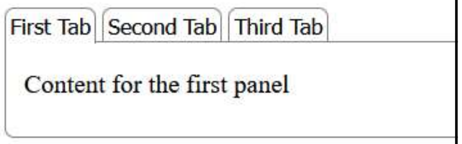
Screen reader announcing
“Subscribe to our newsletter” ***checkbox checked***

@RubysDo - gomaarabab@gmail.com

17

Tabbed Interface: HTML

```
<div class="tabs">
  <div role="tablist" aria-label="Sample Tabs">
    <button role="tab" aria-selected="true" aria-controls="panel-1" id="tab-1" tabindex="0">
      First Tab
    </button>
    <button role="tab" aria-selected="false" aria-controls="panel-2" id="tab-2" tabindex="-1">
      Second Tab
    </button>
    <button role="tab" aria-selected="false" aria-controls="panel-3" id="tab-3" tabindex="-1">
      Third Tab
    </button>
  </div>
  <div id="panel-1" role="tabpanel" tabindex="0" aria-labelledby="tab-1">
    <p>Content for the first panel</p>
  </div>
  <div id="panel-2" role="tabpanel" tabindex="0" aria-labelledby="tab-2" hidden>
    <p>Content for the second panel</p>
  </div>
  <div id="panel-3" role="tabpanel" tabindex="0" aria-labelledby="tab-3" hidden>
    <p>Content for the third panel</p>
  </div>
</div>
```



Content for the first panel

HTML

- <button>
- <div>

JS and CSS for functionality and presentation

18

WAI-ARIA: Accessible Widget Components

WAI-ARIA: Features required for assistive technologies (AT)

@RubysDo - gomaarabab@gmail.com

19

WAI-ARIA: Roles, States, and Properties

ARIA are HTML attributes necessary to make custom widgets accessible for people who use assistive technologies.

ARIA Roles
Type of user interface element

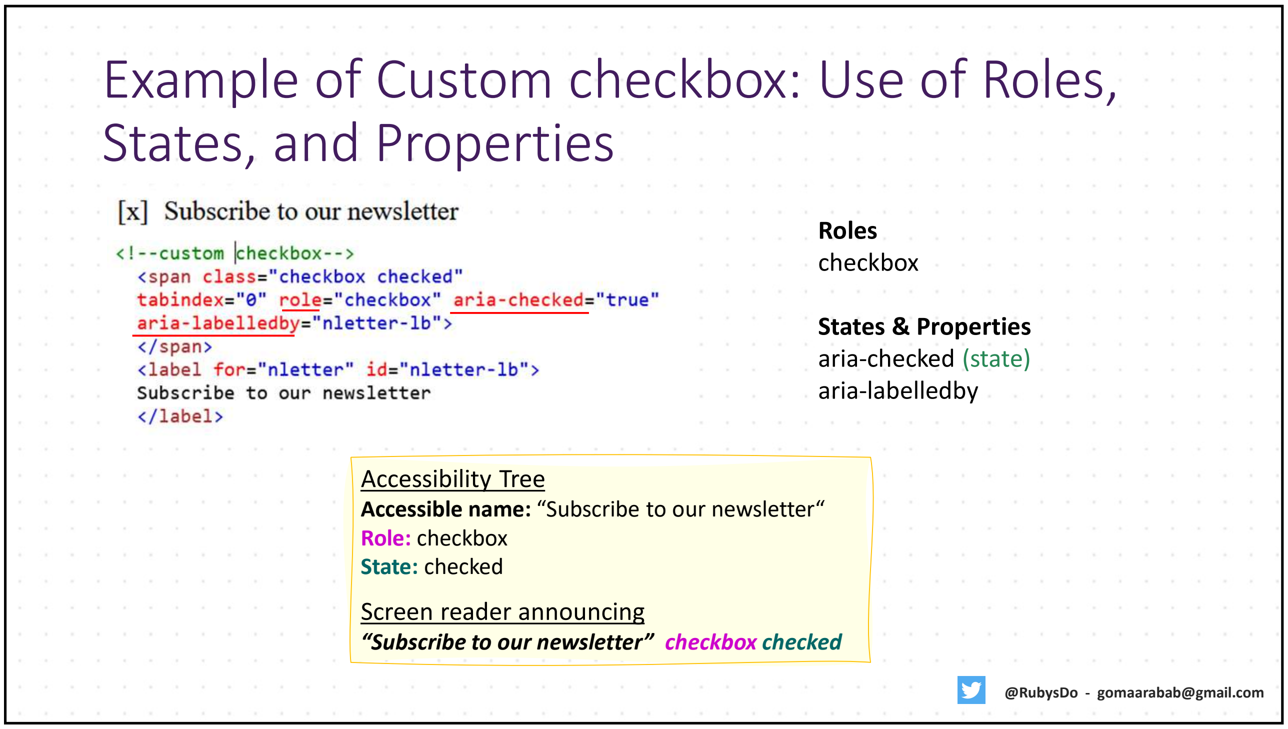
- checkbox
- tablist / tab / tabpanel
- slider
- tree / treeitem
- menubar / menuitem

ARIA States and ARIA Properties
States & properties supported by a role

- aria-current (state)
- aria-describedby
- aria-disabled (state)
- aria-hidden (state)
- aria-label / aria-labelledby
- aria-checked (state) *(i.e. checkboxes, radio buttons)*

@RubysDo - gomaarabab@gmail.com

20



Example of Custom checkbox: Use of Roles, States, and Properties

[x] Subscribe to our newsletter

```
<!--custom checkbox-->
  <span class="checkbox checked"
  tabindex="0" role="checkbox" aria-checked="true"
  aria-labelledby="nletter-lb">
  </span>
  <label for="nletter" id="nletter-lb">
  Subscribe to our newsletter
  </label>
```

Roles
checkbox

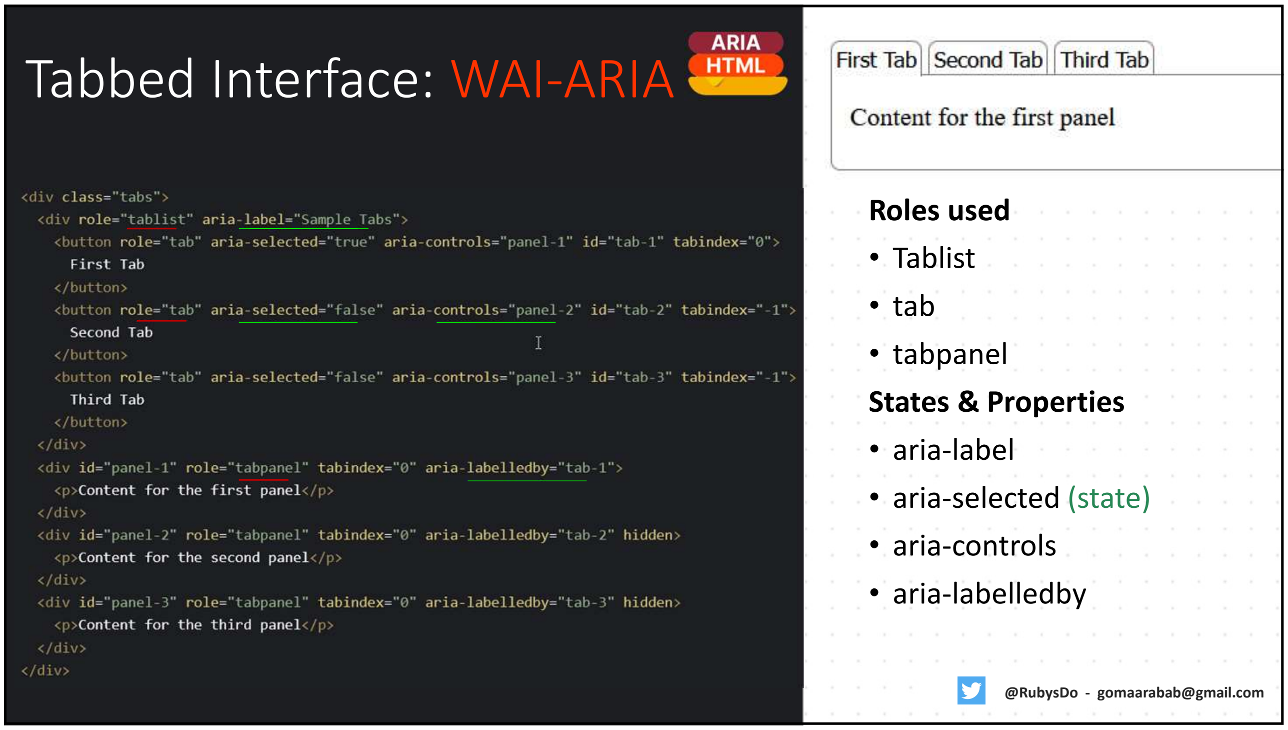
States & Properties
aria-checked (state)
aria-labelledby

Accessibility Tree
Accessible name: "Subscribe to our newsletter"
Role: checkbox
State: checked

Screen reader announcing
"Subscribe to our newsletter" ***checkbox checked***

@RubysDo - gomaarabab@gmail.com

21



Tabbed Interface: WAI-ARIA

ARIA HTML

```
<div class="tabs">
  <div role="tablist" aria-label="Sample Tabs">
    <button role="tab" aria-selected="true" aria-controls="panel-1" id="tab-1" tabindex="0">
      First Tab
    </button>
    <button role="tab" aria-selected="false" aria-controls="panel-2" id="tab-2" tabindex="-1">
      Second Tab
    </button>
    <button role="tab" aria-selected="false" aria-controls="panel-3" id="tab-3" tabindex="-1">
      Third Tab
    </button>
  </div>
  <div id="panel-1" role="tabpanel" tabindex="0" aria-labelledby="tab-1">
    <p>Content for the first panel</p>
  </div>
  <div id="panel-2" role="tabpanel" tabindex="0" aria-labelledby="tab-2" hidden>
    <p>Content for the second panel</p>
  </div>
  <div id="panel-3" role="tabpanel" tabindex="0" aria-labelledby="tab-3" hidden>
    <p>Content for the third panel</p>
  </div>
</div>
```

First Tab | Second Tab | Third Tab

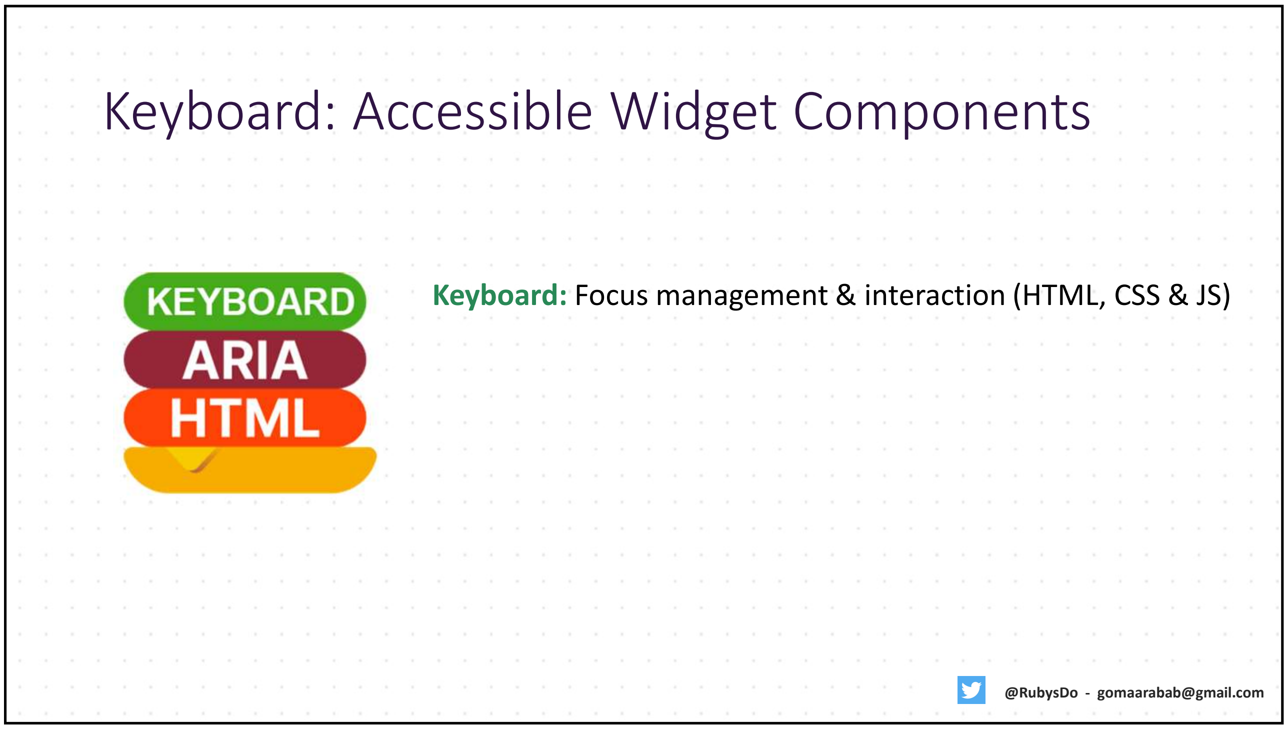
Content for the first panel

Roles used
- Tablist
- tab
- tabpanel

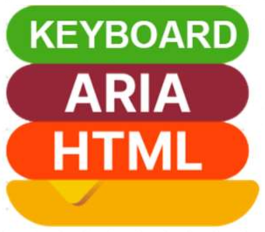
States & Properties
- aria-label
- aria-selected (state)
- aria-controls
- aria-labelledby

@RubysDo - gomaarabab@gmail.com

22



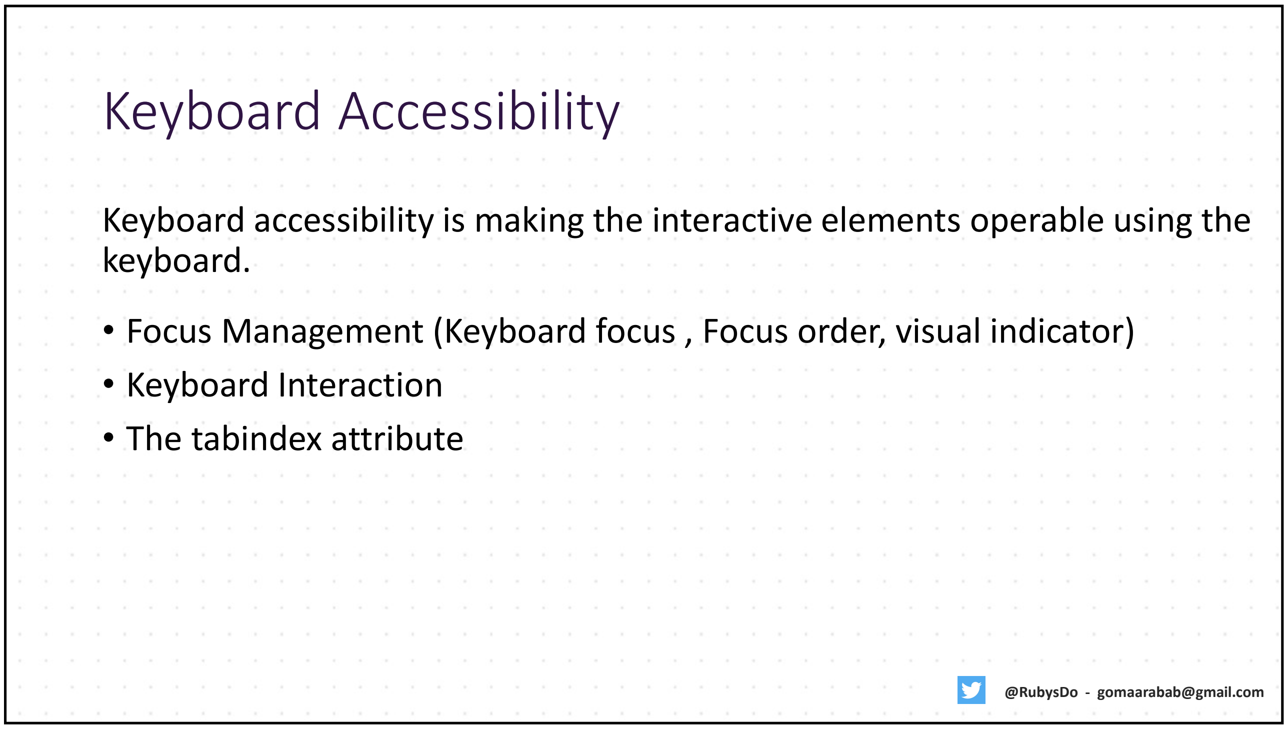
Keyboard: Accessible Widget Components



Keyboard: Focus management & interaction (HTML, CSS & JS)

@RubysDo - gomaarabab@gmail.com

23



Keyboard Accessibility

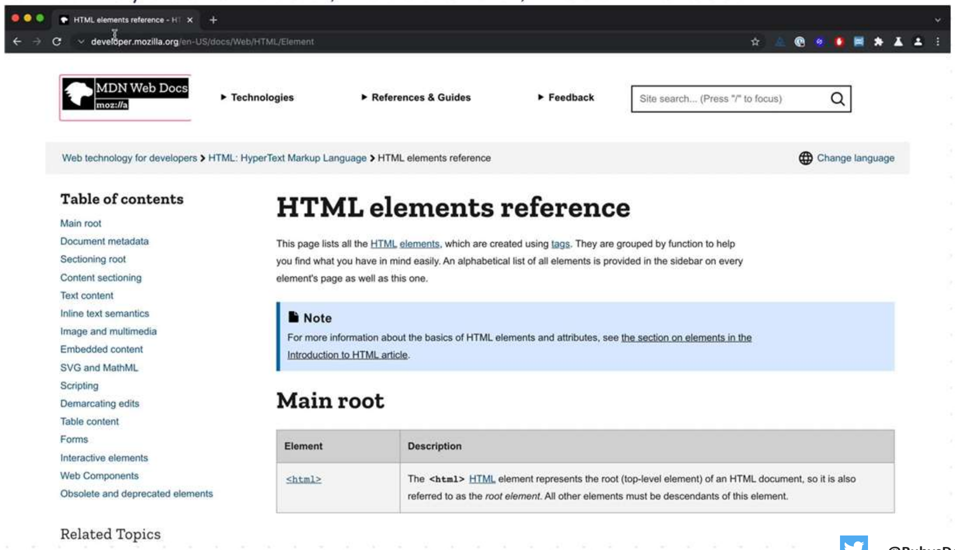
Keyboard accessibility is making the interactive elements operable using the keyboard.

- Focus Management (Keyboard focus , Focus order, visual indicator)
- Keyboard Interaction
- The tabindex attribute

@RubysDo - gomaarabab@gmail.com

24

Focus Management
Demo: Keyboard focus , Focus order, visual indicator



HTML elements reference - H ×

developer.mozilla.org/en-US/docs/Web/HTML/Element

MDN Web Docs moz://a

▸ Technologies ▸ References & Guides ▸ Feedback

Site search... (Press "/" to focus)

Web technology for developers › HTML: HyperText Markup Language › HTML elements reference

Change language

Table of contents

Main root
Document metadata
Sectioning root
Content sectioning
Text content
Inline text semantics
Image and multimedia
Embedded content
SVG and MathML
Scripting
Demarcating edits
Table content
Forms
Interactive elements
Web Components
Obsolete and deprecated elements

Related Topics

HTML elements reference

This page lists all the HTML elements, which are created using tags. They are grouped by function to help you find what you have in mind easily. An alphabetical list of all elements is provided in the sidebar on every element's page as well as this one.

Note
For more information about the basics of HTML elements and attributes, see the section on elements in the Introduction to HTML article.

Main root

Element	Description
`<html>`	The `<html>` HTML element represents the root (top-level element) of an HTML document, so it is also referred to as the *root element*. All other elements must be descendants of this element.

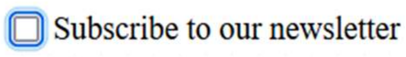
Video from: WCAG-compliant focus indicators

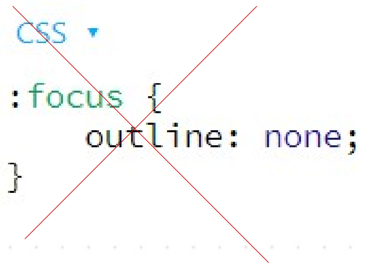
@RubysDo - gomaarabab@gmail.com

25

Focus Management

Never remove the default CSS focus indicator of the browser





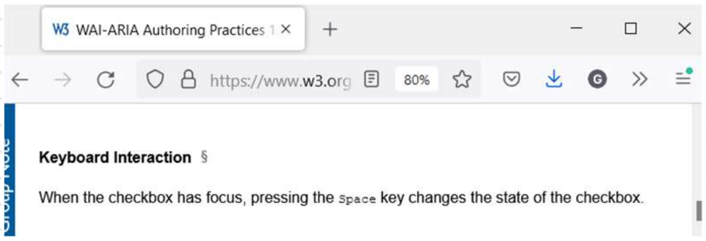
```
CSS ▾

:focus {
    outline: none;
}
```

@RubysDo - gomaarabab@gmail.com

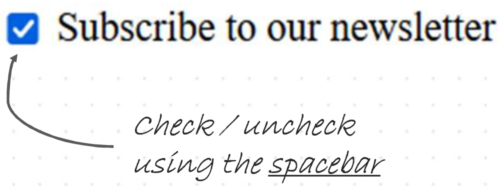
26

Keyboard Interaction



Keyboard Interaction §

When the checkbox has focus, pressing the `Space` key changes the state of the checkbox.



@RubysDo - gomaaarabab@gmail.com

27

Keyboard Interaction

Keyboard events

Mouse Event	Keyboard Event
mousedown	keydown
mouseup	keyup
click	keypress
mouseover	focus
mouseout	blur

JavaScript ▾

```
onMouseover = "doSomething();"
onfocus = "doSomething();"

function doSomething() {alert('Hello World!');}
```

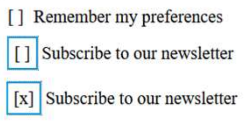
@RubysDo - gomaaarabab@gmail.com

28

Focus: Use of tabindex attribute

- **tabindex="0"** Keyboard focus that follows the logical navigation order.
- **tabindex="-1"** A "programmatic" focus, meaning focus can be set to the element through scripting.

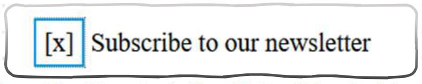
Keyboard Accessibility: Custom control



[ ] Remember my preferences

[ ] Subscribe to our newsletter

[x] Subscribe to our newsletter



[x] Subscribe to our newsletter

```
<span class="checkbox checked" tabindex="0" role="checkbox"
aria-checked="true" aria-labelledby="nletter-lb"
onclick="changeCheckbox()" onKeyDown="changeCheckbox(event.keyCode)"></span>
```

HTML

Focus + Focus order
(Add <span> in the logical focus order)
tabindex="0“

CSS

Focus indicator
*(Add focus ring via **CSS**)*
[role="checkbox"]:focus {border: 2px solid #0198E1;}
[aria-checked="true"]::before {content: "[x]";}
[aria-checked="false"]::before {content: "[ ]";}

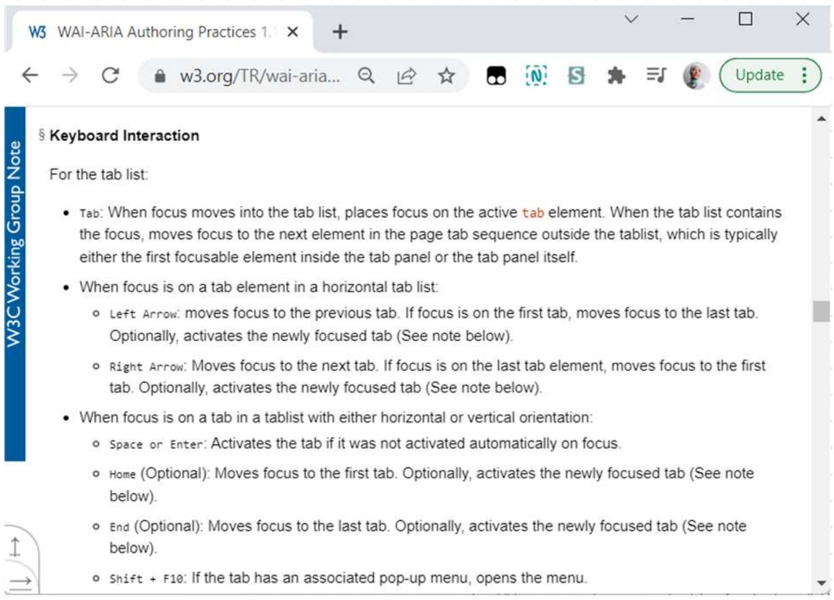
JavaScript

Event handler
(Listen to spacekey & mouse)
onKeyDown: spacebar key
onclick: mouse clicks

JavaScript Function
*(Switch state checked/unchecked via **JS**)*
e.keyCode == 32
item.setAttribute("aria-checked", "false")

Tabbed Interface

WAI-ARIA Authoring Practices 1.1



§ Keyboard Interaction

For the tab list:

- `Tab`: When focus moves into the tab list, places focus on the active `tab` element. When the tab list contains the focus, moves focus to the next element in the page tab sequence outside the tablist, which is typically either the first focusable element inside the tab panel or the tab panel itself.
- When focus is on a tab element in a horizontal tab list:
  - `Left Arrow`: moves focus to the previous tab. If focus is on the first tab, moves focus to the last tab. Optionally, activates the newly focused tab (See note below).
  - `Right Arrow`: Moves focus to the next tab. If focus is on the last tab element, moves focus to the first tab. Optionally, activates the newly focused tab (See note below).
- When focus is on a tab in a tablist with either horizontal or vertical orientation:
  - `Space or Enter`: Activates the tab if it was not activated automatically on focus.
  - `Home` (Optional): Moves focus to the first tab. Optionally, activates the newly focused tab (See note below).
  - `End` (Optional): Moves focus to the last tab. Optionally, activates the newly focused tab (See note below).
  - `Shift + F10`: If the tab has an associated pop-up menu, opens the menu.

@RubysDo - gomaarabab@gmail.com

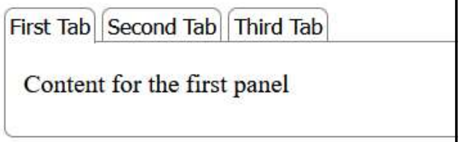
31

Tabbed Interface: Keyboard

KEYBOARD
ARIA
HTML

HTML

```
<div class="tabs">
  <div role="tablist" aria-label="Sample Tabs">
    <button role="tab" aria-selected="true" aria-controls="panel-1" id="tab-1" tabindex="0">
      First Tab
    </button>
    <button role="tab" aria-selected="false" aria-controls="panel-2" id="tab-2" tabindex="-1">
      Second Tab
    </button>
    <button role="tab" aria-selected="false" aria-controls="panel-3" id="tab-3" tabindex="-1">
      Third Tab
    </button>
  </div>
  <div id="panel-1" role="tabpanel" tabindex="0" aria-labelledby="tab-1">
    <p>Content for the first panel</p>
  </div>
  <div id="panel-2" role="tabpanel" tabindex="0" aria-labelledby="tab-2" hidden>
    <p>Content for the second panel</p>
  </div>
  <div id="panel-3" role="tabpanel" tabindex="0" aria-labelledby="tab-3" hidden>
    <p>Content for the third panel</p>
  </div>
</div>
```



Focus Management

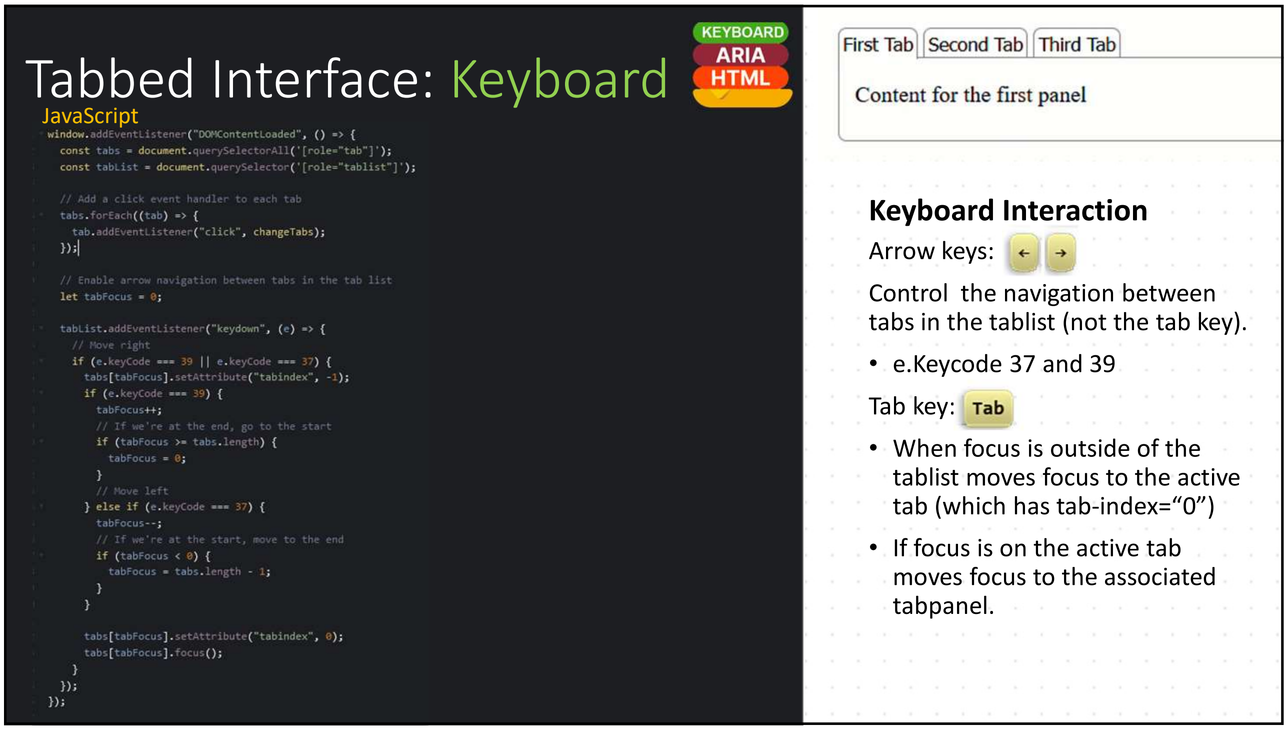
Tabs

- tabindex="0" on the active tabs
- tabindex=“-1” on remaining tabs

Tabpanel

- tabindex="0" to make them tabbable
- All but the currently active one have the hidden attribute.

32



Tabbed Interface: Keyboard

KEYBOARD
ARIA
HTML

JavaScript

```
window.addEventListener("DOMContentLoaded", () => {
  const tabs = document.querySelectorAll('[role="tab"]');
  const tabList = document.querySelector('[role="tablist"]');

  // Add a click event handler to each tab
  tabs.forEach((tab) => {
    tab.addEventListener("click", changeTabs);
  });

  // Enable arrow navigation between tabs in the tab list
  let tabFocus = 0;

  tabList.addEventListener("keydown", (e) => {
    // Move right
    if (e.keyCode === 39 || e.keyCode === 37) {
      tabs[tabFocus].setAttribute("tabindex", -1);
      if (e.keyCode === 39) {
        tabFocus++;
        // If we're at the end, go to the start
        if (tabFocus >= tabs.length) {
          tabFocus = 0;
        }
        // Move left
      } else if (e.keyCode === 37) {
        tabFocus--;
        // If we're at the start, move to the end
        if (tabFocus < 0) {
          tabFocus = tabs.length - 1;
        }
      }

      tabs[tabFocus].setAttribute("tabindex", 0);
      tabs[tabFocus].focus();
    }
  });
});
```

First Tab | Second Tab | Third Tab

Content for the first panel

Keyboard Interaction

Arrow keys: ← →

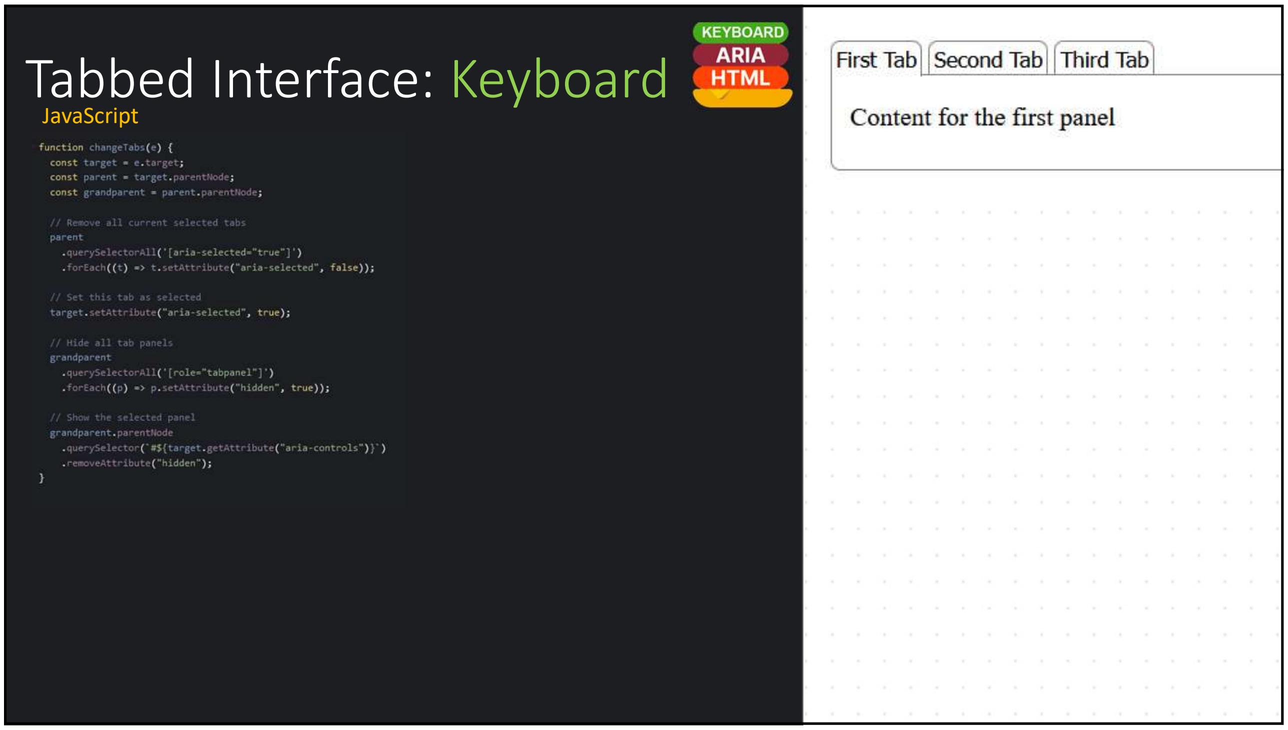
Control the navigation between tabs in the tablist (not the tab key).

- e.Keycode 37 and 39

Tab key: Tab

- When focus is outside of the tablist moves focus to the active tab (which has tab-index="0")
- If focus is on the active tab moves focus to the associated tabpanel.

33



Tabbed Interface: Keyboard

KEYBOARD
ARIA
HTML

JavaScript

```
function changeTabs(e) {
  const target = e.target;
  const parent = target.parentNode;
  const grandparent = parent.parentNode;

  // Remove all current selected tabs
  parent
    .querySelectorAll('[aria-selected="true"]')
    .forEach((t) => t.setAttribute("aria-selected", false));

  // Set this tab as selected
  target.setAttribute("aria-selected", true);

  // Hide all tab panels
  grandparent
    .querySelectorAll('[role="tabpanel"]')
    .forEach((p) => p.setAttribute("hidden", true));

  // Show the selected panel
  grandparent.parentNode
    .querySelector(`#${target.getAttribute("aria-controls")}`)
    .removeAttribute("hidden");
}
```

First Tab | Second Tab | Third Tab

Content for the first panel

34

Screen reader: Accessible Widget Components

HTML: Semantic foundation

WAI-ARIA: Features needed for assistive technologies (AT)

Keyboard : Focus management & interaction (HTML, CSS & JS)

Screen reader: Testing for blind users

@RubysDo - gomaarabab@gmail.com

35

Screen reader: Accessible Widget Components

[Accessible tabbed interface (NVDA screen reader demo)]

@RubysDo - gomaarabab@gmail.com

36

Design Considerations

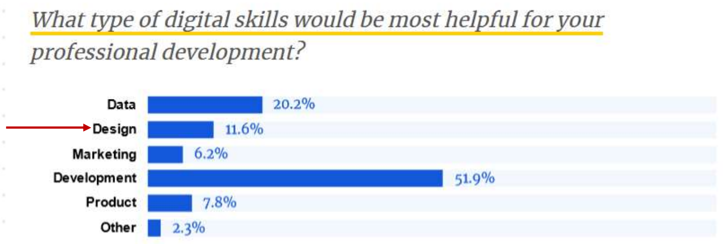
Web Developers are motivated by solving problems for their clients while UX Designers solve problems for their users.

@RubysDo - gomaarabab@gmail.com

37

Design Considerations

Knowledge of UX design is an asset for developers!

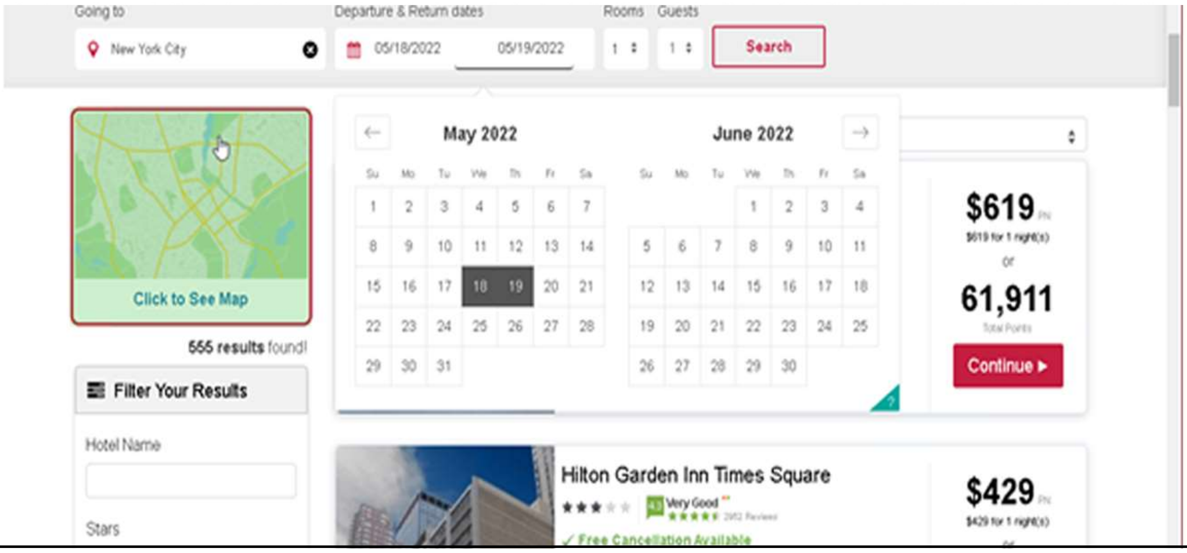


2020 Digital Skills Survey - Development Survey Results

@RubysDo - gomaarabab@gmail.com

38

Design 1


Going to: New York City

Departure & Return dates: 05/18/2022 05/19/2022

Rooms: 1

Guests: 1

555 results found!

Hotel Name

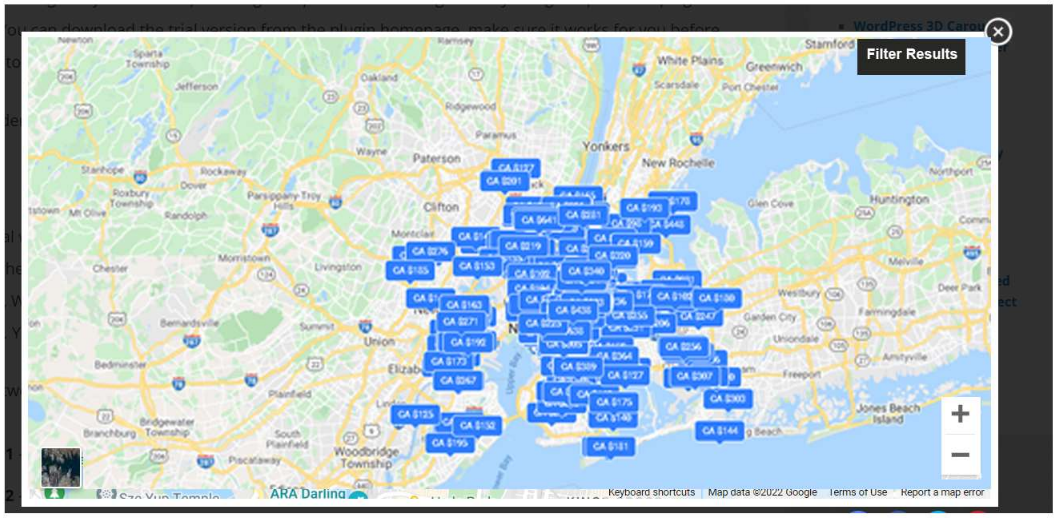
Stars

$619 PN
$619 for 1 night(s)
or
61,911
Total Points

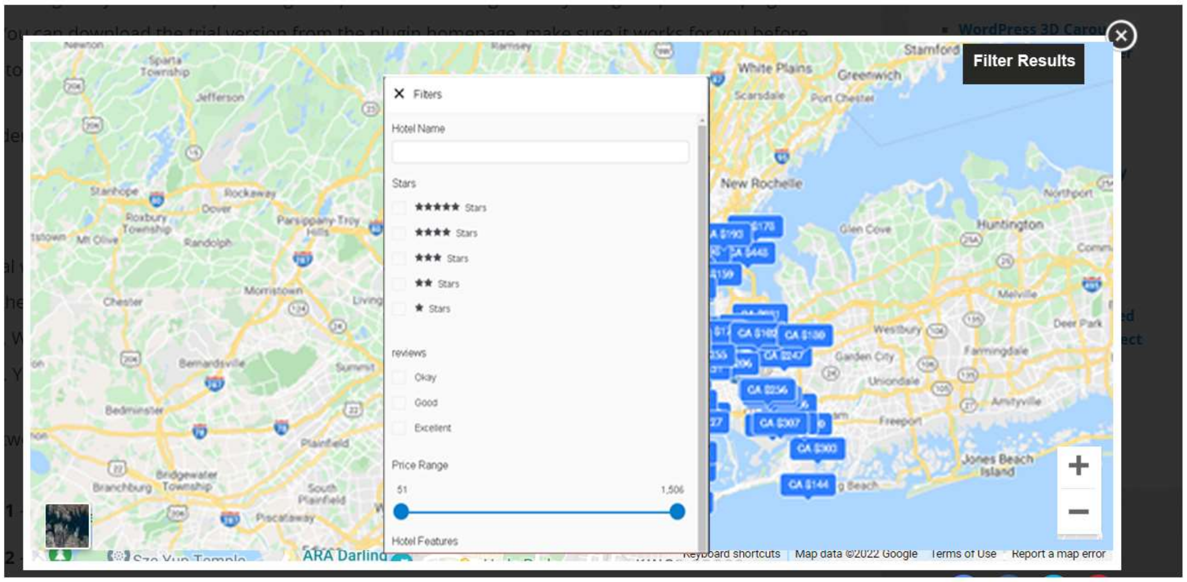
Hilton Garden Inn Times Square
Very Good
2952 Reviews
✓ Free Cancellation Available

$429 PN
$429 for 1 night(s)

39



40



Filters

Hotel Name

Stars

★★★★★ Stars

★★★★ Stars

★★★ Stars

★★ Stars

★ Stars

reviews

Okay

Good

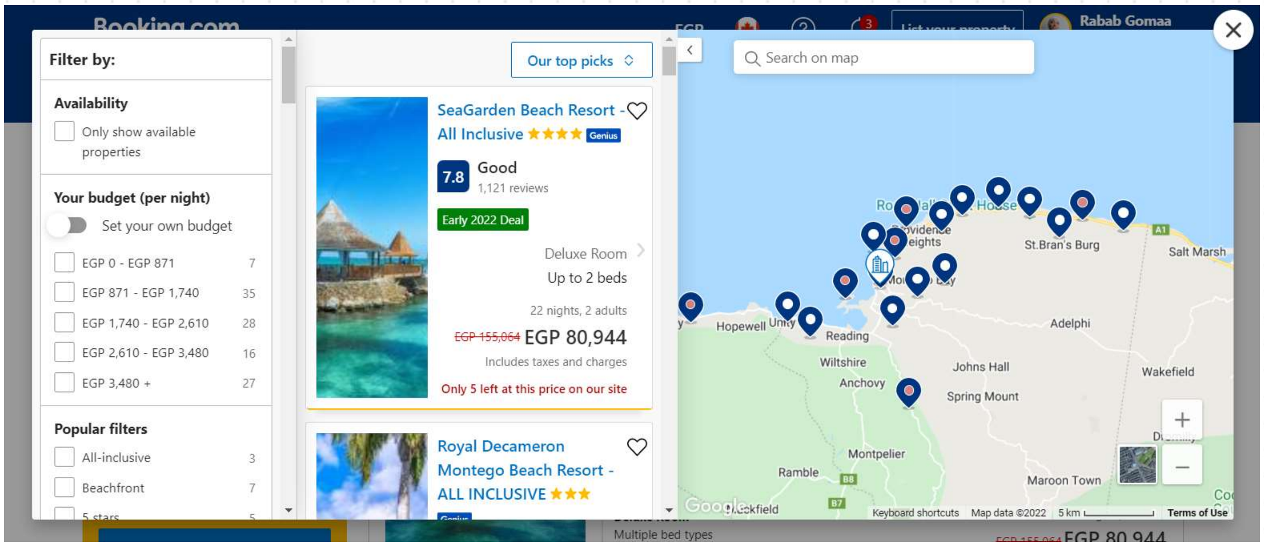
Excelent

Price Range

51 1,506

Hotel Features

41



Filter by:

Availability

Only show available properties

Your budget (per night)

Set your own budget

EGP 0 - EGP 871	7
EGP 871 - EGP 1,740	35
EGP 1,740 - EGP 2,610	28
EGP 2,610 - EGP 3,480	16
EGP 3,480 +	27

Popular filters

All-inclusive	3
Beachfront	7
5 stars	5

SeaGarden Beach Resort - All Inclusive ★★★★ Genius

7.8 Good
1,121 reviews

Early 2022 Deal

Deluxe Room
Up to 2 beds

22 nights, 2 adults

~~EGP 155,064~~ EGP 80,944

Includes taxes and charges

Only 5 left at this price on our site

Royal Decameron Montego Beach Resort - ALL INCLUSIVE ★★★

42

Accessible Widget Checklist

To make a widget accessible,

❑The widget follows an accessible pattern
❑ The interaction is presented in an intuitive and predictable way
❑ JavaScript event handlers work with a **keyboard** and a mouse.
❑ The focus indicator is visible on focus
❑WAI-ARIA elements are used properly
❑The widget is tested using adaptive technology
❑Use ARIA live regions and roles to announce changes in content

@RubysDo - gomaarabab@gmail.com

43

Accessibility = usable for all

Accessibility = usable for all

44

Questions Period

Thank you for your attention!

Rabab Gomaa
@RubysDo
gomaarabab@gmail.com

45