Section 10

Collapsible & Accordion

Collapsible and **accordion** elements, widgets, that expand when clicked on. They allow you to hide content that is not immediately relevant to the user. Widgets are basically self-contained mini programs with information that live and run on the smartphone's screen. There are different ways to implement collapsible and accordions widgets and most of them are done using scripting language, JavaScript. In this section, we can create collapsible and accordion widgets using checkbox and radio input elements, label elements, and the checked pseudo-selector.

Collapsible widgets allow the user to view more than one content at a time while accordion allows the user to view one content at a time. To create collapsible and accordion in CSS, we need to understand some form element such as **`<input>`** and **`<label>,`** and **`CSS selector.`**

Creating input boxes

Most of the control elements in which users are asked to type input or choose a value are marked as input element. The general syntax of this element is:

```
<input id="id-name" type="type_input" name="name_input">
```

Where **`type`** specifies the type of input field, and the **`name`** and **`id`** attributes provide the field's name and id.

There are different types of input but the one that we are going to use for collapsible **`type="checkbox"`** and **`type="radio". checkbox`** type is used to open and close the collapsible tab-content and **`radio`** is used to open one tab-content at the time in an accordion.

Label in the an input element

The **`<label>`** tag defines a label for a `<button>`, `<input>`, `<meter>`, `<output>`, `<progress>`, `<select>`, or `<textarea>` element.

The **<label>** element does not render as anything special for the user. However, it provides a usability improvement for mouse users, because if the user clicks on the text within the **<label>** element, it toggles the control.

CSS selector

In CSS, selectors are patterns used to select the element(s) you want to style. Some of the selectors used to create collapsible and accordion are:

Selector	Example	Description
*	*	Select all elements
element > element	div > p	Selects all **<p>** elements where the parent is ** ** element
element + element	div + p	Selects all **<p>** elements that are placed immediately after ** ** elements
element ~ element	p ~ ul	Selects every ** ** element that are preceded by a **<p>** element
[attribute]	[target]	Selects all elements with a target attribute
[*attribute=value*]	[target=_blank]	Selects all elements with **target="_blank"**
:active	a:active	Selects the **active** link
::after	p::after	Insert something after the content of each **<p>** element
::before	p::before	Insert something before the content of each **<p>** element

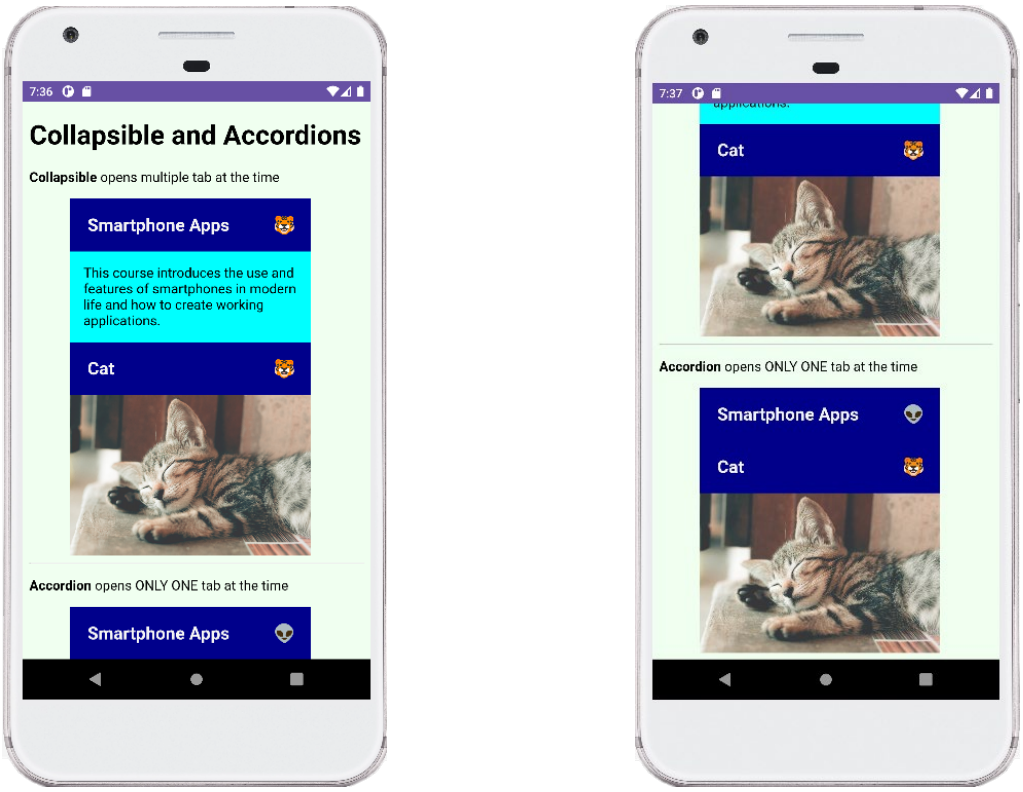
Unicode UTF-8

UTF-8 (**8**-bit **Unicode** Transformation Format) is a variable width character encoding capable of encoding all 1,112,064 valid code points in **Unicode** using one to four **8**-bit bytes.

We can use UTF-8 characters to open and close symbols by using its decimal or hexadecimal reference. Some of those characters values are:

Character	Hexadecimal	Name
☀	2600	Black sun with rays
☁	2601	Cloud
☆	2606	White star
☎	260E	Black telephone
☽	263D	First quarter moon
╳	26CC	Crossing lanes
⛔	26D4	No entry
⚽	26BD	Soccer ball
☰	2630	Trigram for lake

Example*)* Create a collapsible and accordion element for two tabs: one with a paragraph and another one with an image.



The first step is to create the HTML structure with a main container, which is called **`half-container`:**

HTML

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link href="index.css" rel="stylesheet" type="text/css"/>
    <title>Collapsible</title>
  </head>
  <body>
    <h1>Collapsible and Accordion</h1>
     <section class="half-container">

     </section><!-- end of half-container -->
  </body>
</html>
```

Collapsible

Once we have the **half-container,** in CSS we can add a **border** in the **half-container** as reference, remember that the **border** is removed once the container is set:

CSS

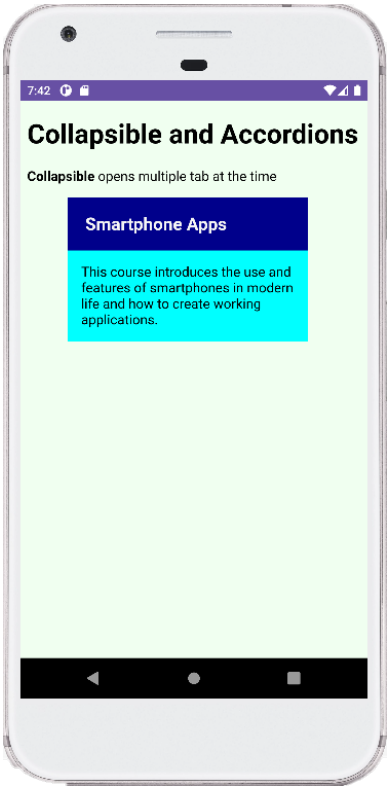
```css
*{box-sizing: border-box;}
html{scroll-behavior: smooth;}
body{ background-color: honeydew;}
.half-container{
    width: 80%;
    padding: 0em 1em;
    margin: 0em 2em;
    border: solid red;
}
```

Now, within the **half-container** we can add another tab container named **tab** with **<input>** element to set the open and close symbol, **<label>** as the tab title, and a ** ** for the tab content:

HTML

```html
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link href="index.css" rel="stylesheet" type="text/css"/>
    <title>Collapsible</title>
  </head>
  <body>
    <h1>Collapsible and Accordion</h1>
    <section class="half-container">
      <p><strong>Collapsible: </strong>Open multiple tab at the time</p>
      <div class="tab">
        <input type="checkbox" id="tab-one"  name="tabs">
        <label for="tab-one">Smartphone Apps</label>
        <div class="tab-content">
          <p>This course introduces the use and features of smartphones in
           modern life and how to create working applications.</p>
        </div>
      </div>
    </section><!-- end of half-container -->
  </body>
</html>
```

In CSS, we can set the **max-height: 30em;** in the **.tab-content** division to see the styling of the **<tab>** and **tab-content**



CSS

```
/* Collapsible styles */
.tab {
    position: relative;
    overflow: hidden;
}
.tab input {
    position: absolute;
    opacity: 0;
}
.tab label {
    position: relative;
    display: block;
    width: 100%;
    padding-left: 1em;
    background-color: darkblue;
    color: whitesmoke;
    line-height: 3;
    cursor: pointer;
    font-size: 1.1em;
    font-weight: bolder;
}
.tab-content {
    max-height: 30em;
    overflow-y: scroll;
    background-color: aqua;
    transition: max-height 0.5s;
}
.tab-content p {margin: 1em;}
```

Once the styling of the **<tab>** and **tab-content** is set, we can change the **max-height: 0em;** as a CSS to trick to hide the **tab-content**.

Now, we have to set the CSS to open and close the **tab-content** on click, but since we created the **tab** using **<input>** element with attribute **type="checkbox"** , we have to declare that when **input** is checked, the **tab-content** will set the height to 100px;

```
.tab input:checked ~ .tab-content {
  max-height: 30em;
}
```

CSS

```css
/* --- when input element is checked ---*/
.tab input:checked ~ .tab-content{max-height: 30em;}

/*--- change icon when open and close the collapsible tab --- */
.tab label::after{
    position: absolute;
    right: 0em; top: 0em;
    display: block;
    width: 3em; height: 3em;line-height: 3;
    text-align: center;
    transition: all 350ms;
}
.tab input[type=checkbox] + label::after { content: "\26BD" ; }
.tab input[type=checkbox]:checked + label::after {
  transform: rotate(360deg); content:"\263D";}
```

Unicode of a tiger

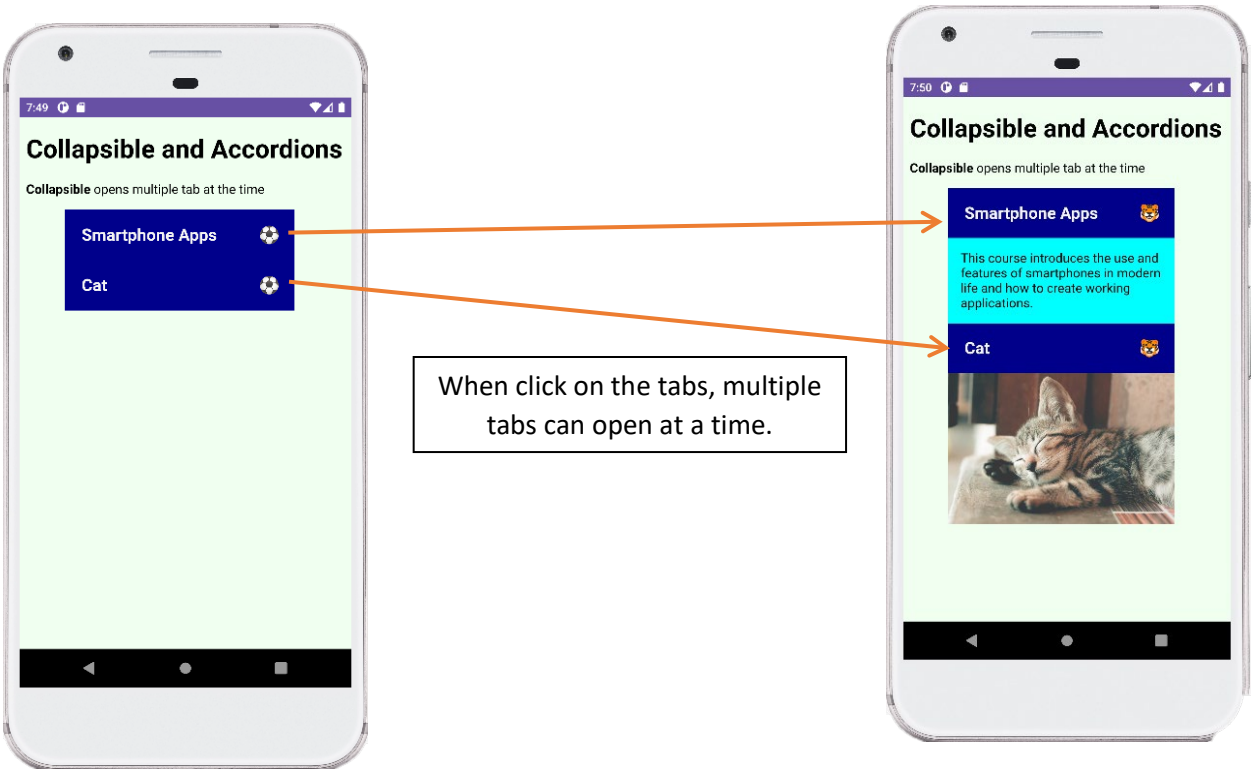
Unicode of a soccer ball

If we want to create another tab, we can change the input **id** name and reuse the CSS code by using the same class name:

HTML

```html
<body>
  <h1>Collapsible and Accordion</h1>
  <section class="half-container">
     <p><strong>Collapsible: </strong>Open multiple tab at the time</p>
     <div class="tab">
       <input id="tab-one" type="checkbox" name="tabs">
       <label for="tab-one">Smartphone Apps</label>
       <div class="tab-content">
         <p>This course introduces the use and features of smartphones in
          modern life and how to create working applications.</p>
       </div>
     </div>
     <div class="tab">
       <input type="checkbox" id="tab-two" name="tabs">
       <label for="tab-two">Cat</label>
       <div class="tab-content">
           <img src="cat.jpg" >
       </div>
     </div>
  </section><!-- end of half-container -->
</body>
```

Running the app will look as the following:



Accordion

To create the accordion, we can use the division for all the tabs of the accordion. In this case, we can use the same CSS as **`half-container.`** Also, we can create a division for each tab using the same CSS as **`tab`**. We can change the tab's color by adding other class name, **`green`**, to the **`tab`** division.

HTML

```
<section class="half-container">
  <p><strong>Accordion: </strong>Open one tab at the time</p>
  <div class="tab">
    <input id="tab-four" type="radio" name="tabs2">
    <label for="tab-four">Smartphone Apps</label>
    <div class="tab-content">
      <p>This course introduces the use and features of smartphones in modern
        life and how to create working applications.</p>
    </div>
  </div>
</section><!-- end of half-container -->
```

After it, we can also add an **`<input>`** element to set the open and close symbol, a **`<label>`** as the tab title, and a **`<div>`** for the tab content. In this case, the **`<input>`** element will have **`type="radio`**" because **`radio`** will allow the user to select one input at the time.

CSS

```
/* tab input is not checked */
.tab input[type=radio] + label::after{content: "\1F47D";}
/* tab input is checked*/
.tab input[type=radio]:checked + label::after{content: "\1F42F";transform: rotate(360deg);}
```

Now, we can add another **tab** to see the accordion effect:

HTMl

```
<section class="half-container">
  <p><strong>Accordion: </strong>Open one tab at the time</p>
  <div class="tab">
    <input id="tab-three" type="radio" name="tabs2">
    <label for="tab-three">Smartphone Apps</label>
    <div class="tab-content">
      <p>This course introduces the use and features of smartphones in modern
        life and how to create working applications.</p>
    </div>
  </div>
  <div class="tab">
    <input id="tab-four" type="radio" name="tabs2">
    <label for="tab-four">Cat</label>
    <div class="tab-content">
       <img src="cat.jpg" >
    </div>
  </div>
</section><!-- end of half-container -->
```