

1	2
---	---

Mitsiu Alejandro
Carreño Sarabia

```

appendTo: "parent", axis: false, connectWith: false, containerCache: {}, this.element.addClass("ui-sortable");
false, handle: false, helper: "original", items: ">
*", opacity: false, placeholder: false, revert: false, scroll: true, scrollSensitivity: 20, scrollSpeed: 10,
(var d=this.options;this.containerCache={};this.element.addClass("ui-sortable");
this.refresh();this.floating=this.items.length?d.axis=="x"||/left/right/.test(this.helper.css("float")):false;this.offset=this.element.offset();this._mouseInit();
elt/.test(this.items[0].item.css("display")):false;this.offset=this.element.offset();this._mouseInit();
sortable-disabled").removeData("sortable").unbind("sortable");this._mouseInit();
item");return this},_setOption: function(d,c){if(d=="
disabled")(this.options[d]=c;this.widget().c?addClass:"removeClass")("ui-sortable-disabled"
a.widget.prototype._setOption.apply(this,arguments)),_mouseCapture: function(i){if(this._mouseMoveCapture
false;if(this.options.disabled||this.options.type=="static")return false;this._refreshPositions
(if(a.data(this,"sortable-item")==h){e=a(this);return false});if(a.data(this,"sortable-item")
le)(var g=false;a(this.options.handle,e).find(".").andSelf().each(function(){if(this.className
this.currentItem=this;this.refreshPositions();this.helper=this._mouseInit();this.offset=this.offset
this.offset=this.offset;this.helper.css("position","absolute");this.helper.css("position","absolute");

```

Agenda

- Recap
- Html elements
- Attributes
- Elm
- elm reactor
- Challenge
- Elm



Recap



Code recap

What can we infer from the following type annotation?

```
something : (Bool -> Int) -> Bool  
-> Int
```

What does this function application produce?

```
double : Int -> Int  
double x =  
    2 * x
```

```
List.map double [1,2,3,4,5]
```

```
mystery : List Int -> Int -> Int  
mystery l acc =  
    case l of  
        [] ->  
            acc  
        x :: xs ->  
            mystery xs  
                (if x > acc then  
                    x  
                else  
                    acc  
                )  
mystery [1,2,3,4,5,4,3,2,1] 0
```

What does this expression reduce to?
What does mystery calculate?

Commands recap

Create an elm project?

- `elm init`

Track changes in git?

- `git add <file>`

Commit changes in git?

- `git commit -m'description'`

Push to remote repo in git?

- `git push origin main`

Enforce format rules on our code?

- `elm-format src/`



Htm1

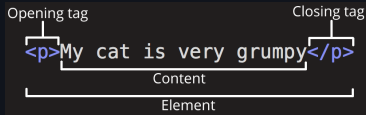
Html

HTML (HyperText Markup Language) is a markup language that tells web browsers how to structure the web pages you visit.

HTML lives inside text files called HTML documents, or just documents, with a `.html` file extension. An HTML document contains the web page's `content` and specifies its `structure`.

Html elements

HTML consists of a series of elements:



- Opening tag -> Name of the element (eg p for paragraph) in angle brackets, marks the element starts.
- Content
- Closing tag -> Slash & name of the element, marks the element end.

Html demo

1. Create a folder "Ex2-html"
2. Inside "Ex2-html" create a file named `main.html`
3. Inside write "<p>Web applications</p>" and save changes
4. Open the file in your browser.
5. Right click on the page and choose "Inspect" or "Inspect element"
6. In the new sub-window, select "Elements" or "Inspector"

How many html element's are on our page?

Is there any hierarchical relationship between this elements?

Elements can be placed within other elements. This is called nesting.

Html demo



Let's update our main.html

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <p>Web application</p>
  </body>
</html>
```

1. Notice the "<!DOCTYPE html>" at the first line, it defines that this document is an html5 document.
2. It's easier writing both opening and closing tags first
3. It's easier aligning opening and close tags indentation

Html elements

- `<p>` -> Paragraph
- `<h1>` -> Header 1
- `` -> Bold text
- `<div>` -> Divider (Structural) We can place several elements inside a container (div)

Let's update our file with:

```
<div>
  <h1>This is an header 1</h1>
  <p>This is a paragraph with a <strong>strong</strong> word</p>
</div>
```

Void elements

Not all elements follow the pattern of an opening tag, content, and a closing tag.

Some elements consist of a single tag, which is typically used to insert/embed something in the document.

Such elements are called void elements.

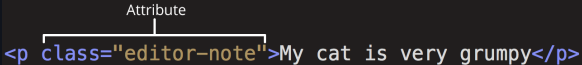
- `
` -> Jump to a new line

```
<p>Hello<br>World</p>
```


Html attributes

Elements can also have attributes.

Attributes contain extra information about the element that won't appear in the content.



```
<p class="editor-note">My cat is very grumpy</p>
```

- A space between the attribute and the element name
- The attribute name followed by an equal sign.
- An attribute value, wrapped with opening and closing quotes
- Attributes are always specified in the start tag
- Attributes usually come in name/value pairs like: name="value"

Attributes demo

Let's add the attribute class into any of our html elements and see what happens on the browser.

Elements and attributes

Let's dissect the following html code:

```
<a href="https://upa.edu.mx">My school</a>
```

- What's the element name?
 - a
- Does it has attributes?
 - Yes
- What are the attribute key and value?
 - key = href
 - Value = https://upa.edu.mx
- What is the element content?
 - My school
- Is it avoid element?
 - No (it has closing tag)

Elm

1. Let's create a new elm project in our "Ex2-html" folder (`elm init`)
2. Create a file "Main.elm"

```
module Main exposing (main)

import Html
import Html.Attributes

main : Html msg
main =
```

Elm html

In elm we can focus only on the content:

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <p>Web application</p>
  </body>
</html>
```

```
module Main exposing (main)

import Html
import Html.Attributes

main : Html msg
main =
  Html.p
```

```
Html.p : List (Html.Attribute msg) -> List (Html.Html msg) -> Html.
Html msg
```

Elm html

```
Html.p : List (Html.Attribute msg) -> List (Html.Html msg) -> Html.  
Html msg
```

```
<!DOCTYPE html>  
<html>  
  <head></head>  
  <body>  
    <p>Web application</p>  
  </body>  
</html>
```

Do we have any attribute?

```
module Main exposing (main)  
  
import Html  
import Html.Attributes  
  
main : Html msg  
main =  
  Html.p  
    []
```

Elm html

```
Html.p : List (Html.Attribute msg) -> List (Html.Html msg) -> Html.  
Html msg  
--  
Html.text : String -> Html.Html msg
```

```
<!DOCTYPE html>  
<html>  
  <head></head>  
  <body>  
    <p>Web application</p>  
  </body>  
</html>
```

```
main : Html msg  
main =  
  Html.p  
    []  
    [ Html.text "Web  
applications"]
```

Elm html

Final complete code:

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <p>Web application</p>
  </body>
</html>
```

```
module Main exposing (main)

import Html
import Html.Attributes

main : Html msg
main =
  Html.p
    []
    [ Html.text "Web
applications" ]
```



```
elm reactor
```

Once our code is complete we can test it the regular way with:

```
elm repl  
import Main
```

We should get no errors! Then we can proceed with:

```
elm reactor
```

And open a browser tab at `localhost:8000`
Navigate the file explorer to `src/Main.elm`



Challenge

Challenge

Are you able to produce the following html code in elm?

```
<div>  
  <h1>Nested h1</h1>  
</div>
```