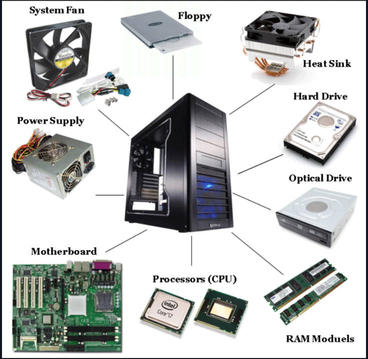
Components

Mitsiu Alejandro Carreño Sarabia



Agenda

```
├── Recap
├── Components
├── Avoid repetition
└── Putting it all together
```

Recap

Html recap

Can you help me describe the following html elements?

- `<div></div>`

It's a content division element, allows us to contain and group other elements and structure our page.

- `<h1></h1> | <h2></h2> | <h3></h3>`

Are several headers, to display, titles, subtitles, subsubtitles...

- `<p></p>`

It's the paragraph element, allows us to display text blocks

Html recap

Can you help me describe the following html elements?

- <a></a>

The anchor element, allows to hiperlink pages, emails, locations via URL's

- <ul> <li></li> </ul>

The parent element is for unordered lists (ul), to display categorical information, (all elements are equally different).
Inside we find list items (li), representing each possible category.

- <br>

Marks a point in html to produce a line break

Previously on...

Let's define a record named "Computer" with:

- ram: String
- model: String
- brand: String
- screenSize: String

And create a variable "myLaptop" of type Computer

Finally, let's make a variable "main" that reduces to:

```
<div>
  <h1>My laptop<h1>
  <div>
    <ul>
      <li>Ram: {{.ram myLaptop}}</li>
      <li>Modelo: {{.model myLaptop}}</li>
      <li>Marca: {{.brand myLaptop}}</li>
      <li>Pulgadas: {{.screenSize myLaptop}}</li>
    </ul>
  </div>
</div>
```

Previously on...

Let's define a record named "Computer" with:

- ram: String
- model: String
- brand: String
- screenSize: String

And create a variable "myLaptop" of type Computer

```elm
type alias Computer =
    { ram : String
    , model : String
    , brand : String
    , screenSize : String
    }
```

```elm
myLaptop : Computer
myLaptop =
    { ram = "32"
    , model = "Thinkpad x1"
    , brand = "Lenovo"
    , screenSize = "13.5"
    }
```

Previously on...

Finally, let's make a variable "main" that reduces to:

```
div
├── h1
└── div
    └── ul
        ├── li
        ├── li
        ├── li
        └── li
```

```
<div>
  <h1>My laptop<h1>
  <div>
    <ul>
      <li>Ram: {{.ram myLaptop}}</li>
      <li>Modelo: {{.model myLaptop}}</li>
      <li>Marca: {{.brand myLaptop}}</li>
      <li>Pulgadas: {{.screenSize myLaptop}}</li>
    </ul>
  </div>
</div>
```

Previously on...

Finally, let's make a variable "main" that reduces to:

```
div
├── h1
└── div
    └── ul
        ├── li
        ├── li
        ├── li
        └── li
```

```elm
main : Html.Html msg
main =
    Html.div
        []
        [ Html.h1 [] [ Html.text "My laptop" ]
        , Html.div
            []
            [ Html.ul
                []
                [ Html.li
                    []
                    [ Html.text "Some info" ]
                ]
            ]
        ]
```

Components

> A part that combines with other parts to form something bigger
> -> https://dictionary.cambridge.org

Components

> A React component is a JavaScript function that you can sprinkle with markup.
> -> https://react.dev/learn/your-first-component

(Hyper Text Markup Language)

Components = Html + Functions

Let's build our first component

Components = Html + Functions

Let's focus on a specific Html section:

```html
<ul>
  <li>Some content</li>
</ul>
```

We are going to begin really simple

```elm
aList : Html.Html msg
aList =
    Html.ul
        []
        [ Html.li
            []
            [ Html.text "Some content"]
        ]
```

Components = Html + Functions

Let's start by making the content more flexible i want to change the string literal "Some content" to be a parameter

```elm
aList : Html.Html msg
aList =
    Html.ul
        []
        [ Html.li
            []
            [ Html.text "Some
content"]
        ]
```

```elm
aList : String -> Html.Html msg

aList content =
    Html.ul
        []
        [ Html.li
            []
            [ Html.text content ]
        ]
```

Components = Html + Functions

Let's suppose we have three list item elements

```elm
aList : String -> String -> String -> Html.Html msg
aList content1 content2 content3 =
    Html.ul []
        [ Html.li []
            [ Html.text content1 ]
        , Html.li []
            [ Html.text content2 ]
        , Html.li []
            [ Html.text content3 ]
        ]
```

🟩 Avoid repetition!

Avoid repetition!

It sucks to write code like this!

```
aList : Html.Html msg
aList =
    Html.ul
        []
        [ Html.li [][]
        , Html.li [][]
        , Html.li [][]
        ]
```

- It's redundant
- I can make typos if i write each
- I can copy/paste but what if we want to change something? (I would have to do it three times!)

We should aim to write less code because it means directly less posible bugs.

Avoiding repetition

Ok I want to tell how all <li><li/> elements in my list are going to be but only once!

```
aList : String -> String -> String -> Html.Html msg
aList content1 content2 content3 =
    Html.ul []
        [ Html.li []
            [ Html.text content1 ]
        , Html.li []
            [ Html.text content2 ]
        , Html.li []
            [ Html.text content3 ]
        ]
```

Avoiding repetition

I want to write something like this:

```
aList : String -> String -> String -> Html.Html msg
aList content1 content2 content3 =
    Html.ul []
        [ anItem content1
        , anItem content2
        , anItem content3
        ]
```

Which typeAnnotation does anItem has to have?

```
anItem : String -> Html.Html msg
```

Avoiding repetition

Which function body (definition) does anItem could have?

```elm
anItem : String -> Html.Html msg
```

```elm
anItem content =
    Html.li [] [ Html.text content]


aList : String -> String -> String -> Html.Html msg
aList content1 content2 content3 =
    Html.ul []
        [ anItem content1
        , anItem content2
        , anItem content3
        ]
```

Avoiding repetition

```
 1 anItem : String -> Html.Html msg
 2 anItem content =
 3     Html.li [] [ Html.text content]
 4
 5
 6 aList : String -> String -> String ->
   Html.Html msg
 7 aList content1 content2 content3 =
 8     Html.ul []
 9         [ anItem content1
10         , anItem content2
11         , anItem content3
12         ]
```

Now if my li element must change I only have to modify it in a single place (Line 3)

Hardcoded logic

This code is a good refactor but what if I want 4 items? Or 10? Or 1?

```
anItem : String -> Html.Html msg
anItem content =
    Html.li [] [ Html.text content]


aList : String -> String -> String -> Html.Html msg
aList content1 content2 content3 =
    Html.ul []
        [ anItem content1
        , anItem content2
        , anItem content3
        ]
```

Hardcoded logic

We can change our aList inputs to a list of strings but something would break

```
anItem : String -> Html.Html msg
anItem content =
    Html.li [] [ Html.text content]


aList : List String -> Html.Html msg
aList contents =
    Html.ul []
        [ anItem content1 -- Now I cant access content1
        , anItem content2 -- or content2
        , anItem content3 -- or content3
        ]
```

Hardcoded logic

What a problem, let's try to see it in context:

```
1 anItem : String -> Html.Html msg
2 anItem content =
3     Html.li [] [ Html.text content]
4
5 aList : List String -> Html.Html msg
6 aList contents =
7     Html.ul []
8         -- Generate a list of Html.Html msg with the <li> from anItem
```

- On L:9 I want to transform my List String (contents) into a List Html.Html msg (<li>'s)

Hardcoded logic

Do we know anything that can help us transform from a List String -> List Html.Html msg?

```
List.map : (a -> b) -> List a -> List b
```

We know that List a is contents (List String) and List b is our <li>'s (List Html.Html msg)

- a = String
- b = Html.Html msg

```
List.map : (String -> Html.Html msg) -> List String -> (List Html.
Html msg)
```

Hardcoded logic

Do we know anything that can help us transform from a List String -> List Html.Html msg?

```
anItem : String -> Html.Html msg
anItem content =
    Html.li [] [ Html.text content]

aList : List String -> Html.Html msg
aList contents =
    Html.ul []
        List.map __________ contents
     -- List.map : (String -> Html.Html msg) -> List String -> (List Html.Html msg)
```

Hardcoded logic

Isn't this just beautiful!

```elm
anItem : String -> Html.Html msg
anItem content =
    Html.li [] [ Html.text content]

aList : List String -> Html.Html msg
aList contents =
    Html.ul []
        List.map anItem contents
```

Putting it all together

Putting it all together

```
main : Html.Html msg
main =
    Html.div
        []
        [ Html.h1
            []
            [ Html.text "My laptop"]
        , Html.div
            []
            [
                aList
                    ["Some text"
                    , "Other text"
                    , "Final text"
                    ]
            ]
        ]
```

```
anItem : String -> Html.Html msg
anItem content =
    Html.li [] [ Html.text content]


aList : List String -> Html.Html
msg
aList contents =
    Html.ul []
        List.map anItem contents
```

- Notice I had to wrap aList on []

Homework

Homework

- Create a component "headers" that given a String parameter, generates the following html code:

```html
<div>
  <h1>{{param}}</h1>
  <h2>{{param}}</h2>
  <h3>{{param}}</h3>
  <h4>{{param}}</h4>
  <h5>{{param}}</h5>
  <h6>{{param}}</h6>
</div>
```

Homework

- Create a component "hyperlink" that receives two Strings
    - The url
    - The text That produces the following html:

```
<a href="{{url}}">{{text}}</a>
```