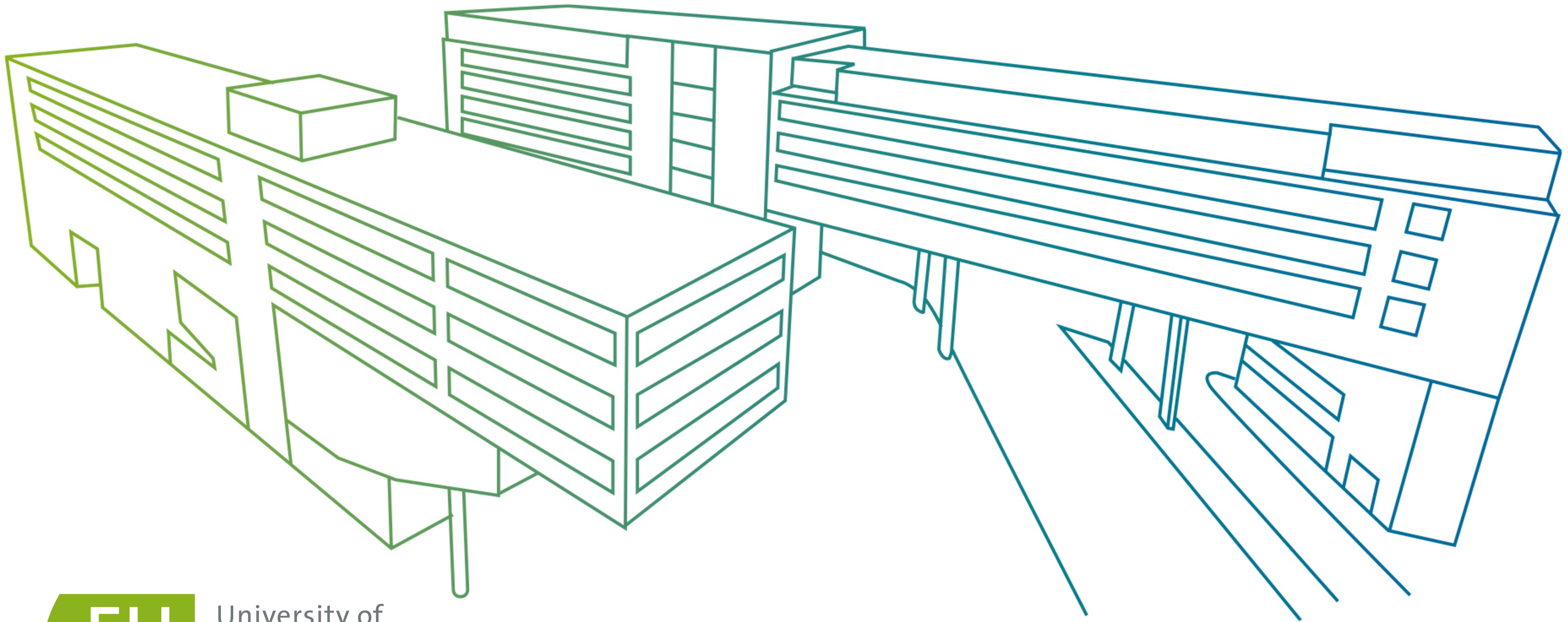


FH TECHNIKUM WIEN
University of Applied Sciences

Web Technologies

HTML

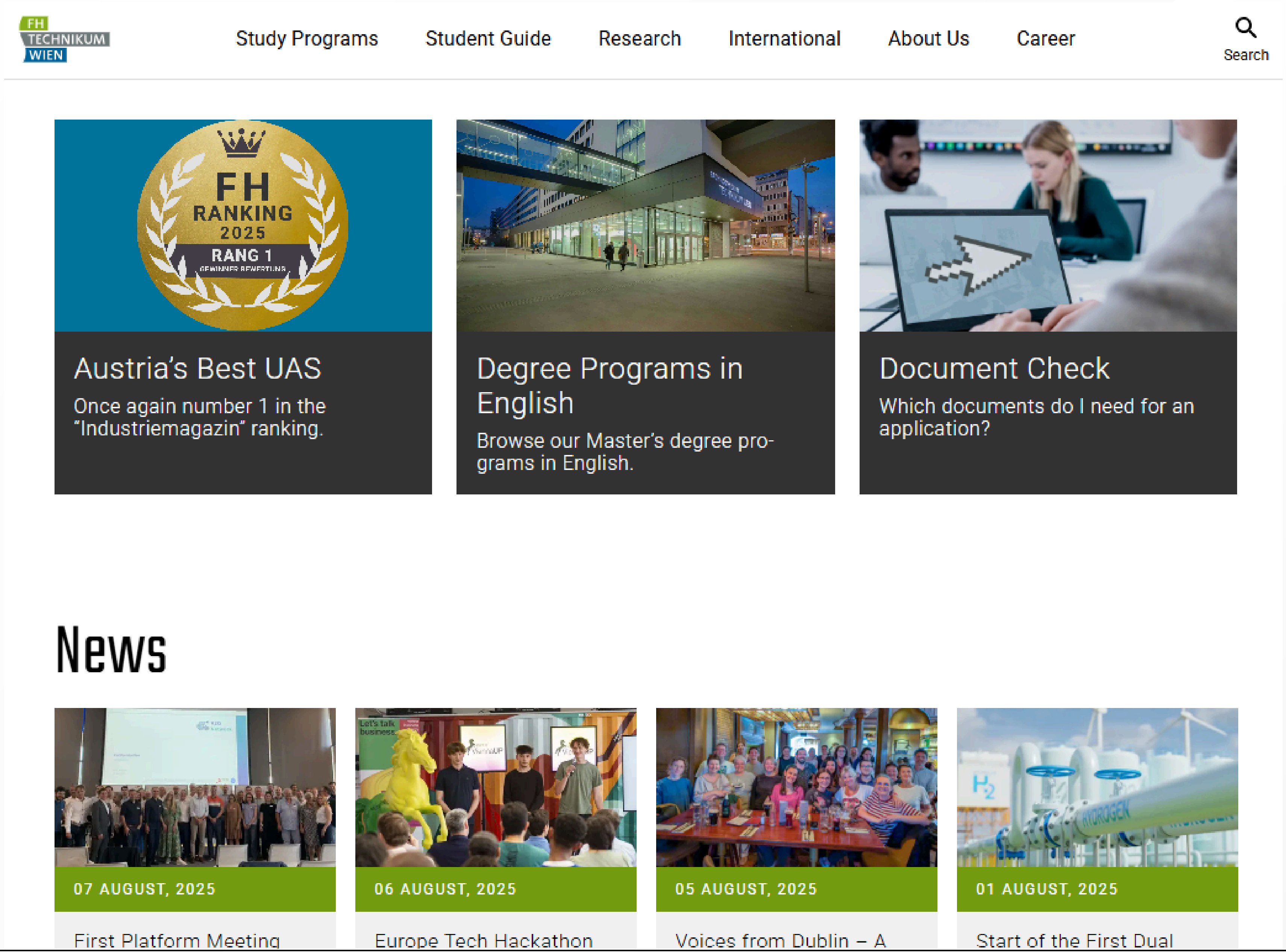
In the beginning

- For a moment, let's imagine HTML, CSS and all the other web technologies do not exist
- We have total creative freedom in creating a language that can display information to other people
- We want to display text and images and maybe even video and sound
- The information should be interactive
- The only limitation: This language should be text-based and human-readable

The vision

- A site for the UAS Technikum Wien
- A list of general topics
- Important information for all people
- Accompanying images
- Current news

The graphical mockup of those requirements is presented on the next slide:



Study Programs Student Guide Research International About Us Career

Search

Austria’s Best UAS

Once again number 1 in the “Industriemagazin” ranking.

Degree Programs in English

Browse our Master’s degree programs in English.

Document Check

Which documents do I need for an application?

News

07 AUGUST, 2025

First Platform Meeting

06 AUGUST, 2025

Europe Tech Hackathon

05 AUGUST, 2025

Voices from Dublin – A

01 AUGUST, 2025

Start of the First Dual

First steps

We know that our language should be text based and we can see text in the mockup, so let's start there:

```
Study Programs  Student Guide  Research  International  About Us  Career

Austria's Best UAS
Once again number 1 in the "Industriemagazin" ranking.

Degree Programs in English
Browse our Master's degree programs in English

Document Check
Which document do I need for an application?
```

Now the language

- If we only wanted to transmit text over the Internet, text alone would suffice
- But our text is more complicated, not all text is the same

- What is the difference between `Document Check` and `Which document do I need for an application?`?
- Is `Document Check` more similar to `Degree Programs in English` or to `Student Guide`?

The difference is meaning

- `Document Check` and `Degree Programs in English` are both headings for a short following sentence
- `Student Guide` is a broad category in a list of further categories

- Not all text is the same, because some text is more important or has another meaning based on the context of the text

- So how do we define meaning in our language?

Start and End

- The meaning (e.g., a heading) starts before the actual text and ends after the text

```
heading_start Document Check heading_end
```

- But how do we know that `heading_start` isn't just normal text that we want to display

- What can we do to clearly differentiate normal text and the start and end of meaning?

The tag

- We can use symbols that are rarely used in normal text, e.g., `<` and `>`

```
<heading_start>Document Check<heading_end>
```

- This is called a **tag**
- And since `_start` and `_end` would have to be repeated hundreds of times, we can omit `_start` and replace `_end` with a slash `/`

```
<heading>Document Check</heading>
```

The element

Let's define what an element is:

- It has meaning (heading, content, category, ...)
- It starts with a tag `<tag>`
- It ends with a tag `</tag`
- It has content
    - Text
    - Other elements
    - Media?

An update for our document

How can we represent our information now?

```
<information>
    <heading>Degree Programs in English</heading>
    <summary>
        Browse our Master's degree programs in English
    </summary>
</information>
```

Notice the formatting? What is the advantage of formatting the text like that?

Element relations

- The formatting suggests, that elements are inside other elements
  - Each indentation represents another level of depth
- We could also use the terms *parent*, *child* and *sibling* to describe element relations
- This data type is more commonly known as a tree
- A tree represents a hierarchical structure with a set of connected nodes (elements)

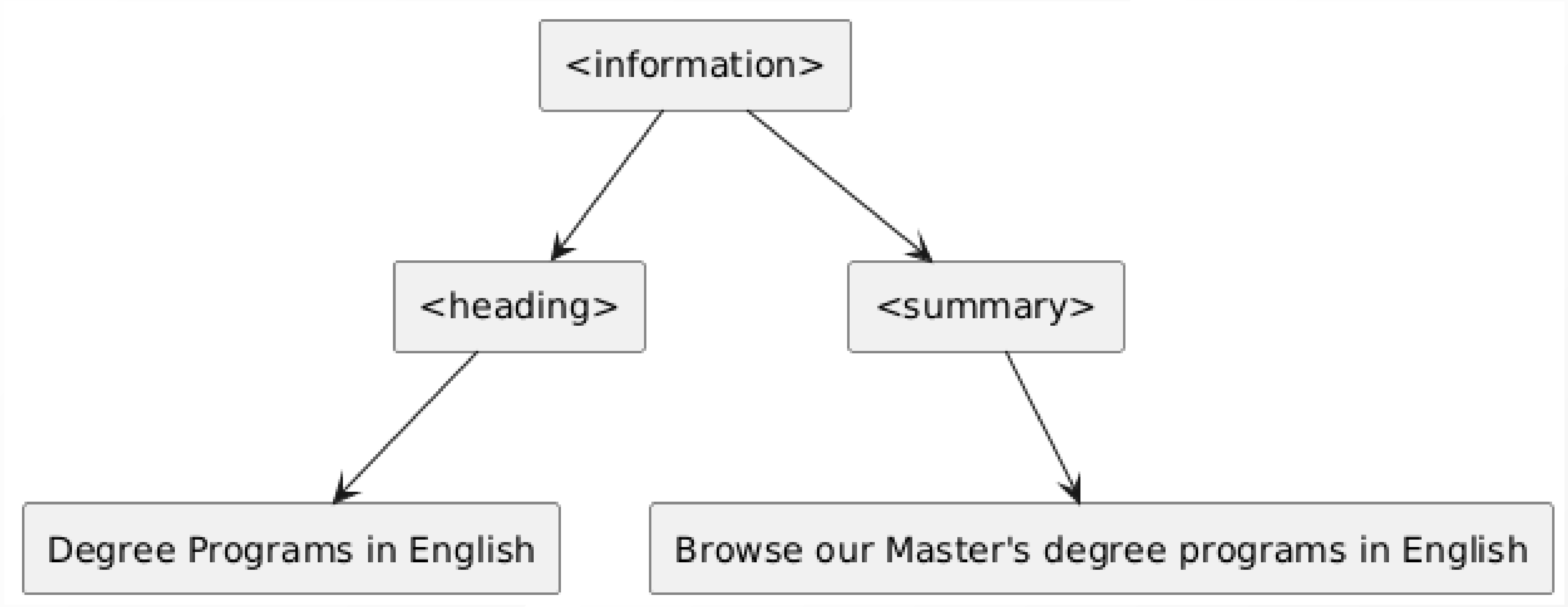
Element relations

In this snipped, what element is the *parent*, which elements are *children* and which elements are *siblings*?

```
<information>
    <heading>Degree Programs in English</heading>
    <summary>
        Browse our Master's degree programs in English
    </summary>
</information>
```

Is there another way to visualize these relations?

A tree


<information>
<heading>
<summary>
Degree Programs in English
Browse our Master's degree programs in English

Interactivity?

- Our information should be interactive
- That can mean a lot of things...
- To keep it simple, let's just think about linking to another information document
    - This is what a Hypertext is!

- How can we update our language to incorporate the ability to link to another document?

Link Option 1

We need to add information to our elements. If we click on an element, it should open another document.

```
<information link="degrees.doc">
    <heading>Degree Programs in English</heading>
    <summary>
        Browse our Master's degree programs in English
    </summary>
</information>
```

This is called an attribute. An attribute has a name (`link`) and a value (`"degrees.doc"`).

Link Option 2

Or we can create a new element that is just designed to link to other documents.

```
<link target="degrees.doc">
    <information>
        <heading>Degree Programs in English</heading>
        <summary>
            Browse our Master's degree programs in English
        </summary>
    </information>
</link>
```

What option do you like better?

Separation of Concerns

- Allowing every element to have an `link` attribute would blur the meaning of elements, making the structure of a document less clear
- Using the `<link>` tag explicitly communicates the developer's intent to create a hyperlink
  - Remember *Hypertext*: Hyperlinks turn a text into Hypertext!

What would the category list on top of our page look like in our new language?

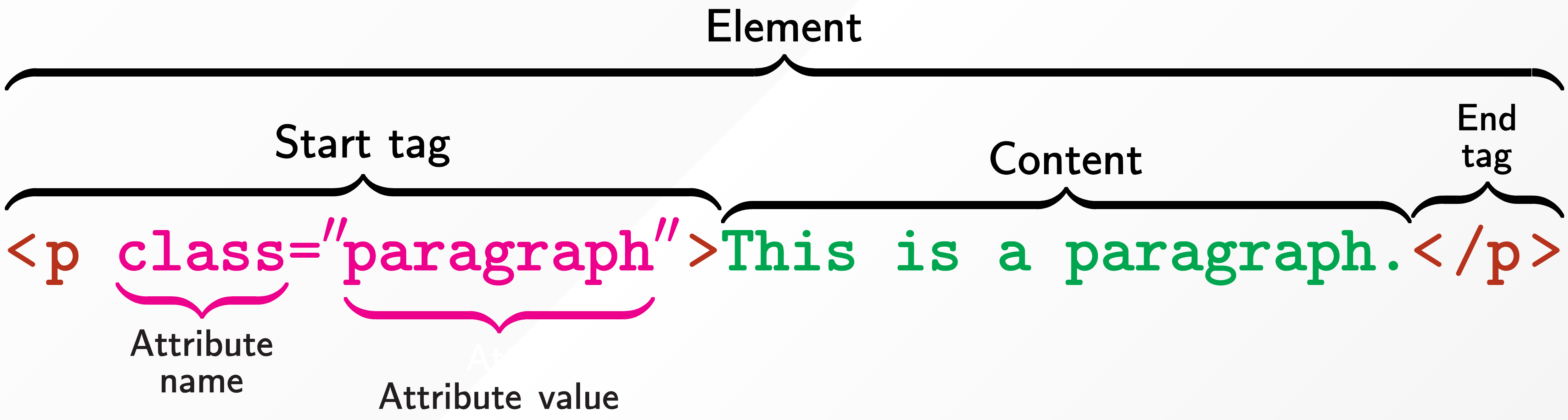
A list of links

Something like this maybe:

```
<navigation>
    <link target="programs.doc">
        <category>Study Programs</category>
    </link>
    <link target="guide.doc">
        <category>Student Guide</category>
    </link>
    <link target="research.doc">
        <category>Research</category>
    </link>
</navigation>
```

A definition of an element

- An element consists of a start tag (that might include on or more attributes), content (text or elements) and an end tag
- A attribute has a name and a value



```
<p class="paragraph">This is a paragraph.</p>
```

Images

- There is still something missing from our initial mockup
- We would like to include images, but currently there is no way to add an image to our text
- We could create an `image` element
- The problem: Images are not text based, but rather binary files
- This means we cannot *write* them into the content part of our element

- What could be a possible solution?

No Content

- Not every element needs content if the content is provided by the meaning or an attribute of the element
- The image needs to know where the image file is
  - This could be an attribute

```
<image file="/images/uastw_logo.png">
```

- There is no end tag, because there is no content

- What *types* of elements can we think of?

Element types

Normal elements: Have both a start tag and an end tag. Have some amount of content, including text and other elements.

Void elements: Only have a start tag. Do not contain any children, such as text or other elements.

Raw text elements: Have both a start tag and an end tag. Have some amount of text content, but no elements.

- What do you think might be the use case for raw text elements?

A final look at our page

```
<navigation>
    <link target="programs.doc">
        <category>Study Programs</category>
    </link>
</navigation>
<link target="degrees.doc">
    <information>
        <image file="images/uastw_building.jpeg">
        <heading>Degree Programs in English</heading>
        <summary>
            Browse our Master's degree programs in English
        </summary>
    </information>
</link>
```

The switch

- You may have already guessed that we did not create a new language, but rather tried to imagine what the development of HTML might have felt like
- HTML stands for **H**yper**T**ext **M**arkup **L**anguage
- HTML elements are the building blocks of HTML pages
- It defines the **content** and **structure** of web content
- Browsers do not display the HTML tags, but use them to interpret the content of the page

A real HTML document

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Minimal HTML</title>
    </head>
    <body>
        <h1>Heading on the page</h1>
        <p>
            This is a text on the page. To be more specific, the
            "p" element means this text is a paragraph.
        </p>
    </body>
</html>
```

HTML document tree

Can you create a tree diagram for the HTML snippet from the previous slide?

HTML compared to our language

- Real HTML looks similar, but there are some differences

- What's the reason fot the `<!DOCTYPE html>` element?
- What's the difference between `<head>` and `<body>`?
- Why `<h1>` and not `<heading>`? Does it imply `<h2>`?
- How would our page really look like?

`<!DOCTYPE html>`

- At the moment, our HTML documents are saved in a `.html` file
- The file ending tells our browser, that it has to interpret HTML
- But there is no `.html` file, when you access `https://www.technikum-wien.at/`
- The text is sent by a server *(more on that later in this course)*
- If the first line of that text is `<!DOCTYPE html>` it tells the browser that the text will be HTML
- More specific: HTML5, the final and last version of HTML

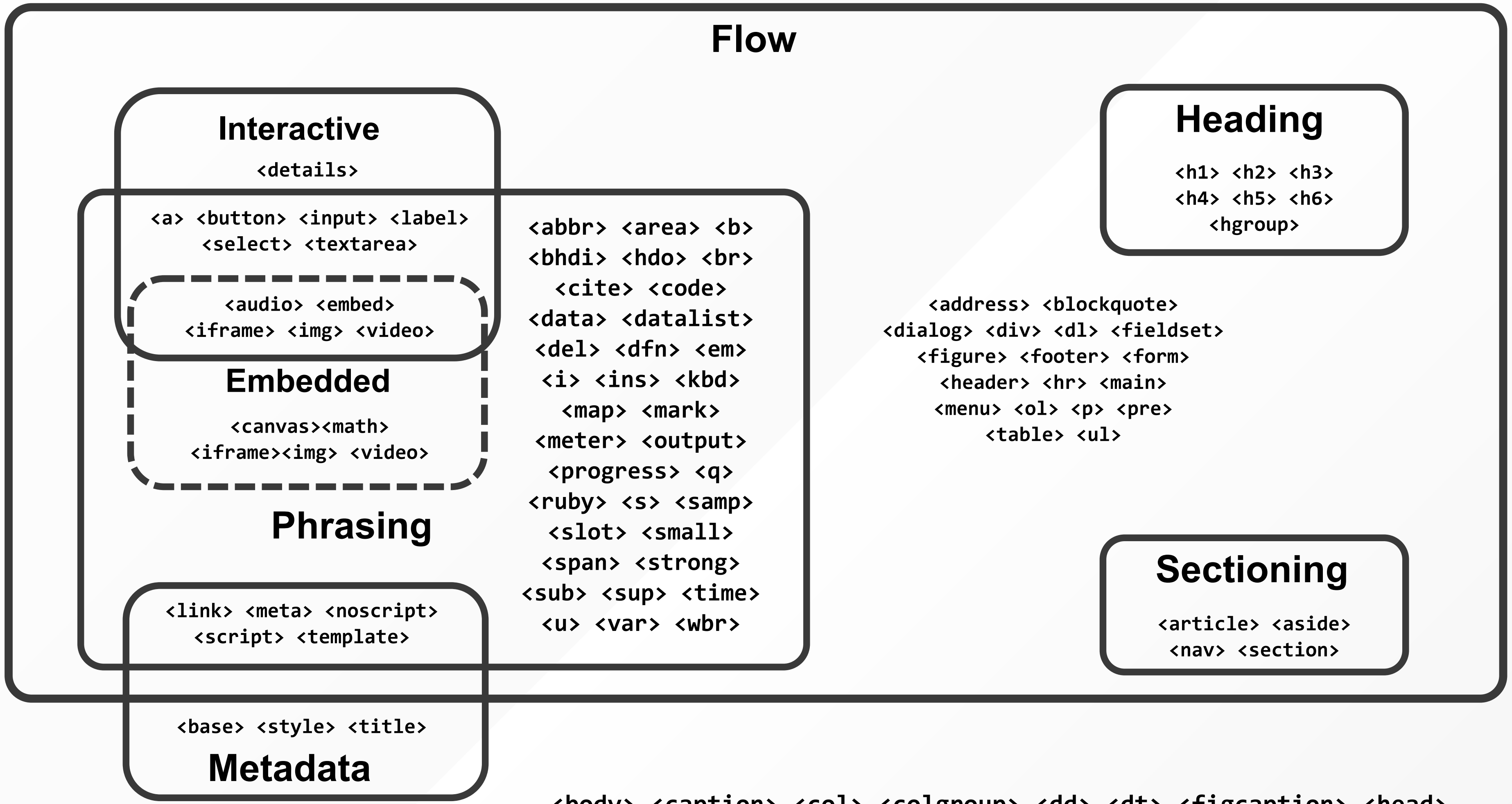
`<head>` and `<body>`

- A real website contains more information than just the part that the user sees
- The `<body>` is for the user, it represents the content of an HTML document
- The `<head>` is for the browser, it contains machine-readable information (metadata) about the document
- There can be only one `<head>` and `<body>` element in a document

Why `<h1>` and not `<heading>`?

- HTML has it's own set of HTML elements
- The number of elements have grown over the years
- Sadly we cannot add our own elements but have to use the ones provided to us by HTML
- Luckily there are enough for almost every use case
- Headings are represent by six levels of section headings: `<h1>` to `<h6>`
- Images use a `<img>`, the navigation is in `<nav>`, links are `<a>`

HTML element content categories



<body> <caption> <col> <colgroup> <dd> <dt> <figcaption> <head>
<html> <legend> <li> <optgroup> <option> <rp> <rt> <source>
<summary> <thody> <td> <tfoot> <th> <thead> <tr> <track>

How would our page really look like?

The navigation:

```
<nav>
    <ul>
        <li><a href="/programs">Study Programs</a></li>
        <li><a href="/guide">Student Guide</a></li>
        <li><a href="/research">Research</a></li>
        <li><a href="/international">International</a></li>
        <li><a href="/about">About Us</a></li>
        <li><a href="/career">Career</a></li>
    </ul>
</nav>
```

How would our page really look like?

The information:

```
<div id="must-read">
    <div class="information">
        <img src="/img/uastw_building.jpeg" alt="image of UAS Technikum Wien">
        <h3>Degree Programs in English</h3>
        <p>Browse our Master’s degree programs in English.</p>
    </div>
    <!-- ... -->
</div>
```

- What is `<div>` used for?

`<div>` and `<span>`

- `<div>` does not have any inherent semantic meaning
- It is often used to structure and organize content on a webpage
  - Grouping related elements together for layout purposes
  - Acting as a container to group elements
- The `<span>` element is an inline container used to group text or other inline elements
  - Wrapping a specific part of text within a larger block of text
- Like `<div>`, it has no inherent styling or semantic meaning.

Key Takeaways

HTML is Structure & Meaning: HTML defines the content and structure of web pages through elements that give semantic meaning to text and media.

Elements = Tags + Content: Elements consist of start tags, content, and end tags. Some void elements (like images) only need a start tag.

Semantic Elements: Use meaningful elements like `<nav>`, `<article>`, `<section>` instead of generic `<div>` when possible

Key Takeaways

Document Structure Matters: Every HTML document has a `<head>` for metadata and a `<body>` for user-visible content, wrapped in `<html>` tags.

Hyperlinks Create Hypertext: The "H" in HTML stands for Hypertext - links connect documents and make the web interactive.

Attributes Add Information: Attributes provide additional information about elements, like `href` for links or `src` for images.

“ *HTML is the foundation of the web - master the basics, and you can build anything!* ”