HTML5

INTRODUCTION

What is HTML 5?

HTML 5.2 is the **latest version of the HTML markup language**. It includes all valid elements from HTML 4 and XHTML 1.0.

Simpler markup

Simpler doctype, simpler character encoding, simplified `<link>` and `<script>` elements and more.

Extended elements

More flexible roles for elements such as `<dl>` and `<a>` elements.

New semantic markup

New structural elements such as `<header>`, `<section>`, `<article>` and more.

Native functionality

Native form functionality designed to replace JavaScript.

Specifications

There are two groups working on **slightly different versions** of the HTML 5 specification.

The World Wide Web Consortium (W3C) produces the following specification:

W3C HTML 5.2 specification

https://www.w3.org/TR/html52/

The Web Hypertext Application Technology Working Group (WHATWG) produces the following specification:

WhatWG HTML 5 specification

https://html.spec.whatwg.org/multipage/

Sometimes the **two specifications differ considerably** - such as in the case of the `<hgroup>` element.

The `<hgroup>` element **has been removed** from the HTML5 (W3C) specification, but **it's still in** the WHATWG version of HTML.

Simpler markup

The following changes make it **easier for authors to write HTML documents**, as well as making our markup cleaner.

Doctype

With HTML5, we can use a **simpler doctype** than before.

```
<!-- HTML 4.01 -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
```

```
<!-- HTML 5 -->
<!DOCTYPE html>
```

```
<!-- HTML 5 in lower case -->
<!doctype html>
```

Character encoding

With HTML5, we can use a **simpler character encoding** than before.

```
<!-- HTML 4.01 -->
<meta http-equiv="content-type"
content="text/html; charset=utf-8">
```

```
<!-- HTML 5 -->
<meta charset="utf-8">
```

Character encoding should be placed within the **first 512 characters of your document** - before any content-based elements (like the `<title>` element).

The type attribute

The `type` attribute is **no longer required** for CSS or JavaScript links.

```
<link rel="stylesheet" href="a.css"
type="text/css">

<link rel="stylesheet" href="a.css">
```

```
<style type="text/css"></style>

<style></style>
```

```
<script src="a.js" type="text/javascript">
</script>

<script src="a.js">
</script>
```

Void elements

The trailing slash is **no longer required** for empty or void elements.

```
<br />      => <br>
<hr />      => <hr>
<img />     => <img>
<input />   => <input>
<link />    => <link>
<meta />    => <meta>
```

The decision to use or not use slashes is **up to you and your team**. The key is to be consistent!

Attribute minimisation

With HTML5, boolean attributes can now be **mimimised**.

```
checked="checked"     => checked
compact="compact"     => compact
declare="declare"     => declare
disabled="disabled"   => disabled
multiple="multiple"   => multiple
selected="selected"   => selected
```

Exercise 1: simplify a document

Open **exercise01-simlify/exercise1-start.htm**

Review your work against **exercise01-simlify/exercise1-finished.htm**

1. Simpler doctype
2. Simpler character encoding
3. Remove type (CSS and JS)
4. Remove trailing slashes
5. Remove boolean values

Changes to existing elements

The following changes **clarify the purpose of some elements**, as well as changing how we use others.

The <address> element

The `<address>` element represents contact information for **nearest article** or body element.

```
<!-- HTML markup -->
<address>
  Written by <a href="#">Jen Smith</a>
</address>
```

The `<address>` element is **not to be used for physical address information** (such as street or postal address details).

The <b> element

The <b> element now allows us to **draw attention to text**, without conveying any extra importance.

```
<!-- HTML markup -->
<p>
  Some text including <b>bold</b> text.
</p>
```

The <cite> element

The `<cite>` element now solely represents a **title of a work** (e.g. a book, paper, essay, poem, song, painting etc).

```
<!-- HTML markup -->
<p>
  The <cite>The Chronicles of Narnia</cite>
</p>
```

The `<cite>` element is **not to be used** for names or for quotations.

The <dl> element

The `<dl>` element now represents an **association list of name-value groups** such as terms and definitions, Q&A information etc.

```
<!-- HTML markup -->
<dl>
  <dt>Allegory</dt>
  <dd>a form of extended metaphor</dd>
  <dt>Amplification</dt>
  <dd>the use of bare expressions</dd>
</dl>
```

The `<dl>` element is **not to be used** for dialogue.

The <hr> element

The <hr> element now represents a **paragraph-level thematic break**.

```
<!-- HTML markup -->
<p>A paragraph of text.</p>
<hr>
<p>Another paragraph of text.</p>
```

The <i> element

The `<i>` element now represents a

span of text in an alternate voice

such as a taxonomic designation, a technical term, an idiomatic phrase etc.

```
<!-- HTML markup -->
<p>
  Eastern Grey Kangaroo, <i>Macropus
  giganteus</i>, is a marsupial mammal
</p>
```

The <s> element

The <s> element has resurfaced and again represents **information that is no longer accurate** or relevant.

```
<!-- HTML markup -->
<p>
  Buy now:<br>
  <s>Only $2.95 per unit</s>
  Only $2.50 per unit
</p>
```

The <small> element

The `<small>` element now represents **side comments** such as small print.

```
<!-- HTML markup -->
<p>
  Only $2.95 <small>(inc GST)</small>
</p>
```

The `<small>` element **should not be used** to visually represent smaller content.

Exercise 2a: change elements

Open **exercise02-change/exercise2-start.htm**

Review your work against **exercise02-change/exercise2-finished.htm**

1. Add `<cite>`
2. Add `<address>`
3. Replace `<em>` with `<i>`
4. Replace `<strong>` with `<b>`

Changes to existing attributes

The ID attribute

Before HTML 5, values for the `ID` **attribute** had to begin with a letter.

```
<!-- HTML markup -->
<div id="wide480"></div>
```

The `ID` attribute is **now allowed to have any value**, as long as it is unique, is not the empty string, and does not contain space characters.

```
<!-- HTML markup -->
<div id="480wide"></div>
```

However, if `ID` values start with a number, they **must be escaped** when writing CSS selectors.

```
<!-- HTML markup -->
<div id="480wide"></div>

/* escaped number */
#\34 80wide { }
```

The name attribute

Before HTML 5, the `name` attribute was **used to name elements**.

```
<!-- HTML markup -->
<h2 name="source">Source</h2>
```

HTML 5 requires that the `name` attribute be **replaced by the `ID` instead**.

```
<!-- HTML markup -->
<h2 id="source">Source</h2>
```

The summary attribute

The `summary` attribute was used to **describe `<table>` elements** to assistive technologies.

```
<!-- HTML markup -->
<table summary="table summary here">
...
</table>
```

The `summary` attribute has been dropped in the HTML 5 specification. There are **several alternative solutions** available:

The `<caption>` element can be used to **describe the purpose of the `<table>`**.

```
<!-- HTML markup -->
<table>
  <caption>Caption here</caption>
</table>
```

The `aria-describedby` attribute can be used to **describe the purpose** of the `<table>`.

```
<!-- HTML markup -->
<table aria-describedby="table-dates">
...
</table>
<p id="table-dates">
  The table below is a three column table
  with city, workshop, and dates.
</p>
```

The media attribute

Before HTML 5, the `media` attribute only allowed **one or more comma-separated media values**.

```
<!-- HTML markup -->
<link media="screen, print">
```

The `media` attribute **now accepts** media queries.

```
<!-- HTML markup -->
<link media="screen and (min-width:700px)">
```

The tabindex attribute

Before HTML 5, the `tabindex` attribute **only allowed positive numbers** between 0 and 32,767.

```
<!-- HTML markup -->
<input type="text" tabindex="1">
```

The `tabindex` attribute **now allows negative values** which indicate that the element cannot receive focus.

```
<!-- HTML markup -->
<input type="text" tabindex="-1">
```

Developers can use JavaScript to change the `tabindex` value **from a negative to a positive number** if they require an element to become focusable based on specific events.

```
<!-- HTML markup -->
<input type="text" tabindex="0">
```

The language attribute

The `language` attribute was used to **define the language** of `<script>` elements.

```
<!-- HTML markup -->
<script language="javascript"></script>
```

The `language` attribute is **now obsolete** and should be removed.

```
<!-- HTML markup -->
<script></script>
```

The start attribute

In HTML 4.01, the `start` attribute **was removed**. This made it very hard to restart ordered lists without resorting to invalid markup or hacks.

HTML 5 **now allows** the `start` attribute to be defined within the `<ol>` element.

```
<!-- HTML markup -->
<ol start="3">
  <li>Eastern Grey Kangaroo</li>
  <li>Red Kangaroo</li>
</ol>
```

The accept attribute

The `accept` attribute is used to provide user agents **with a hint of what file types will be accepted**. This attribute now allows the values `audio/*`, `video/*` and `image/*`.

```
<input type="file" accept="image/*">
```

The action attribute

The `action` attribute on the `<form>` element is **no longer allowed to have an empty URL**.

```
<form action="#">
   ...
</form>
```

The border attribute

The `border` attribute on `<table>` only allows the values `"1"` and **the empty string**.

```
<table border="1">
  ...
</table>

<table border="">
  ...
</table>
```

The colspan attribute

The `colspan` attribute on `<td>` and `<th>` **now has to be greater than zero**.

```
<td colspan="2">...</td>
```

The defer attribute

The `defer` attribute on the `<script>` element **now explicitly makes the external script execute** when the page has finished parsing.

```
<script src="demo.js" defer></script>
```

Width and height

The `width` and `height` attributes on `<img>`, `<iframe>` and `<object>` elements are **no longer allowed to contain percentage values**.

These attributes are also **not allowed to be used to stretch the image** to a different aspect ratio than its intrinsic aspect ratio.

```
<img src="somthing.png" alt="Alt text" height="42"
width="42">
```

The href attribute

The `href` attribute on `<a>` elements is **no longer allowed to have an empty URL**.

```
<a href="#">Link text</a>
```

Exercise 2b: change attributes

Continue working in **exercise02-change/exercise2-start.htm**

Review your work against **exercise02-change/exercise2-finished.htm**

1. Add a `start` attribute and value
2. Change a `name` attribute to an `ID`
3. Remove a `language` attribute

New content categories

In HTML 4.01, there was a distinction between **block-level and inline** elements.

Block-level elements

Block-level elements are elements that are formatted visually as blocks.

```
<!-- Block-level elements -->
<article></article>
<aside></aside>
<blockquote></blockquote>
<div></div>
<footer></footer>
<form></form>
<h1></h1>
<header></header>
<p></p>
```

In almost all cases, block-level elements **can be nested** inside other block-level elements.

```
<!-- Nested block-level elements -->
<div>
  <div>
    <h1></h1>
    <p></p>
  </div>
</div>
```

However, block-level elements such as `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>` and `<p>` **cannot contain other block-level elements**.

```
<!-- Cannot contain blocks -->
<h1></h1>
<h2></h2>
<h3></h3>
<h4></h4>
<h5></h5>
<h6></h6>
<p></p>
```

Inline elements

Inline elements are elements that **do not form new blocks of content**; the content is distributed in lines.

```
<!-- Inline elements -->
<a></a>
<abbr></abbr>
<b></b>
<cite></cite>
<code></code>
<em></em>
<i></i>
<span></span>
<strong></strong>
```

In almost all cases, inline elements **can be nested inside** other inline elements.

```
<!-- Nested inline elements -->
<span>
  <span></span>
  <em></em>
</span>
```

Until HTML 5, inline elements **could not contain block-level elements**.

```
<!-- Block inside inline elements -->
<span>
  <div></div>
</span>
```

However, in HTML 5, the inline `<a>` element was redefined so that it could **wrap around block-level elements**.

```
<!-- Block inside inline elements -->
<a href="#">
  <div></div>
</a>
```

The `<a>` element can only wrap around block-level elements as long as there is **no interactive content inside** the block-level element.

```
<!-- Interactive elements inside -->
<a href="#">
  <div>
    <a href="#"></a>
    <button></button>
    <input type="text" />
    <textarea></textarea>
  </div>
</a>
```

Interactive content includes:

```
<!-- Interactive content 1 -->
<a href="#"></a> (if href present)
<audio controls></audio>
<button></button>
<details></details>
<embed></details>
<iframe></iframe>
<img usemap="#a"> (if usemap present)
```

```
<!-- Interactive content 2 -->
<input />
<label></label>
<object></object>
<select></select>
<textarea></textarea>
<video controls></video> (if controls present)
```

Some browsers **apply underlines** to all content inside the `<a>` element, other browsers do not.

You should always set these `<a>` elements to `display: block` and turn off underlines, so that they are **visually consistent across different browsers**.

New content categories

In HTML5, the binary distinction of block vs inline has been **replaced with a more complex set of content categories**.

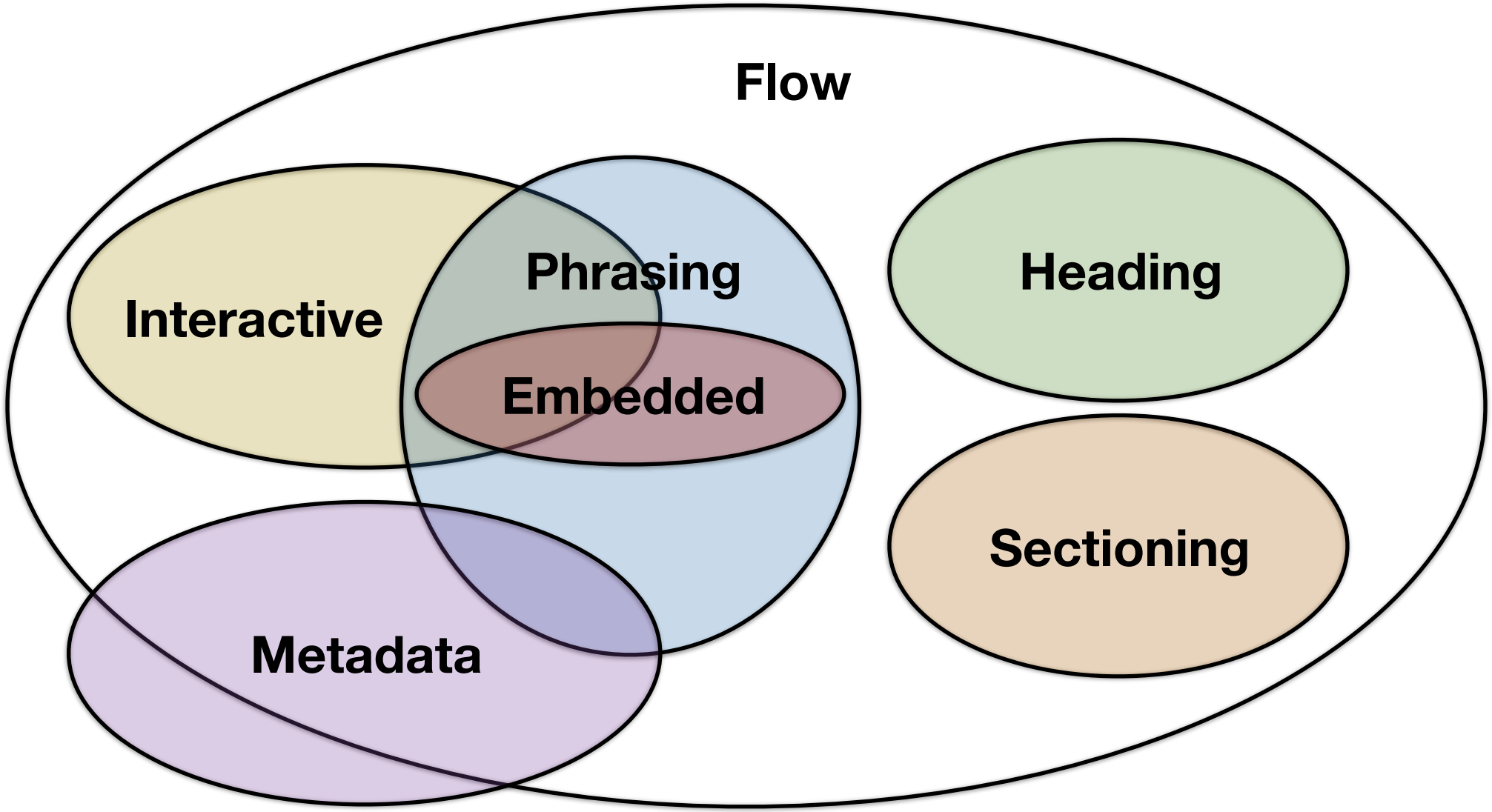
Each element in HTML falls into **zero or more categories** that group elements with similar characteristics together.

The following **broad categories** are used in this specification:

1. Metadata content
2. Flow content
3. Sectioning content
4. Heading content
5. Phrasing content
6. Embedded content
7. Interactive content

The older “block-level” category roughly corresponds to the HTML 5 category of **flow content**.

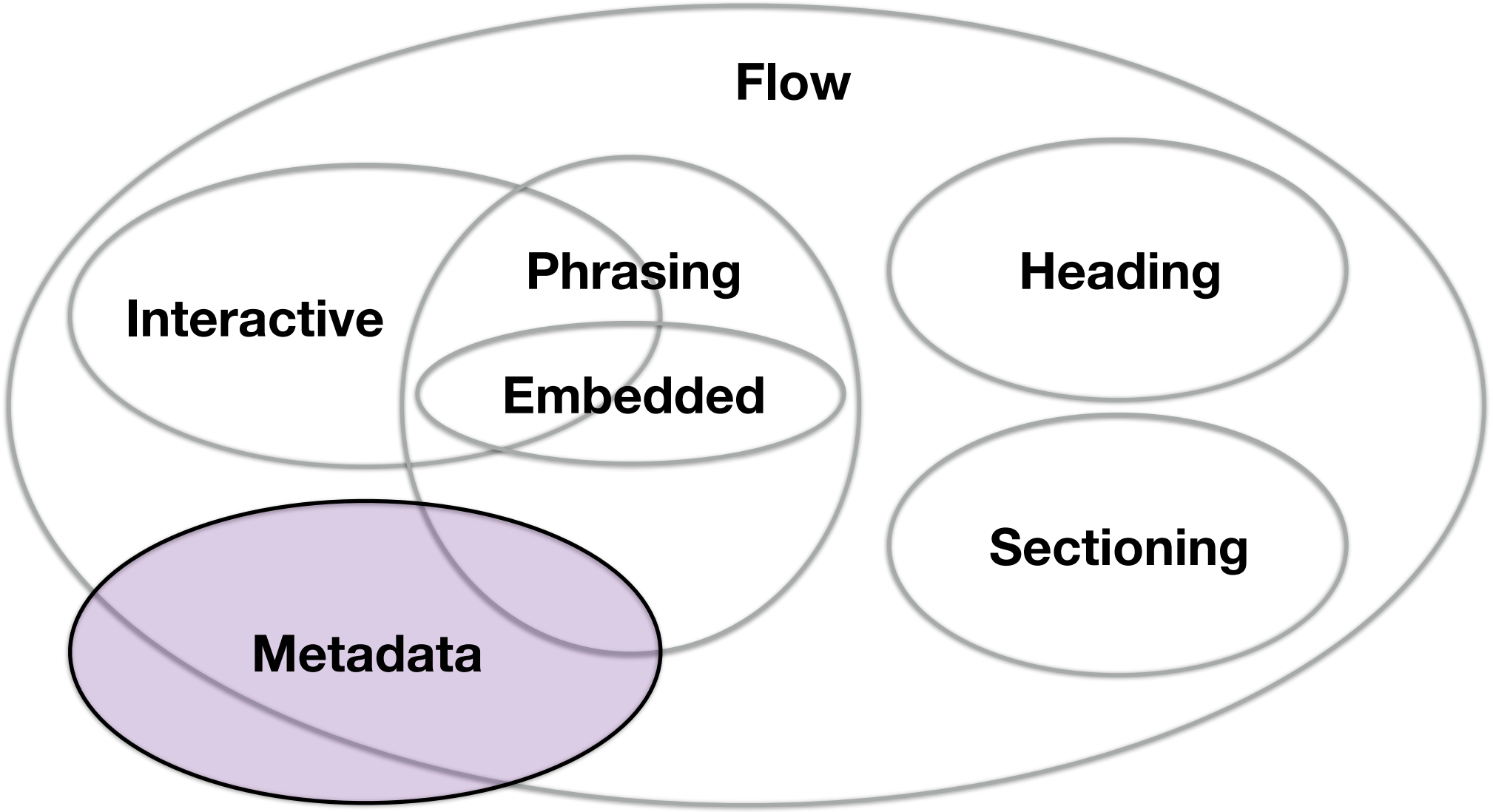
The older “inline” category corresponds to **phrasing content**.


Flow
Interactive
Phrasing
Embedded
Heading
Sectioning
Metadata

A chart of **all elements in their content types** is available here:

https://russmaxdesign.github.io/html5-content-types/

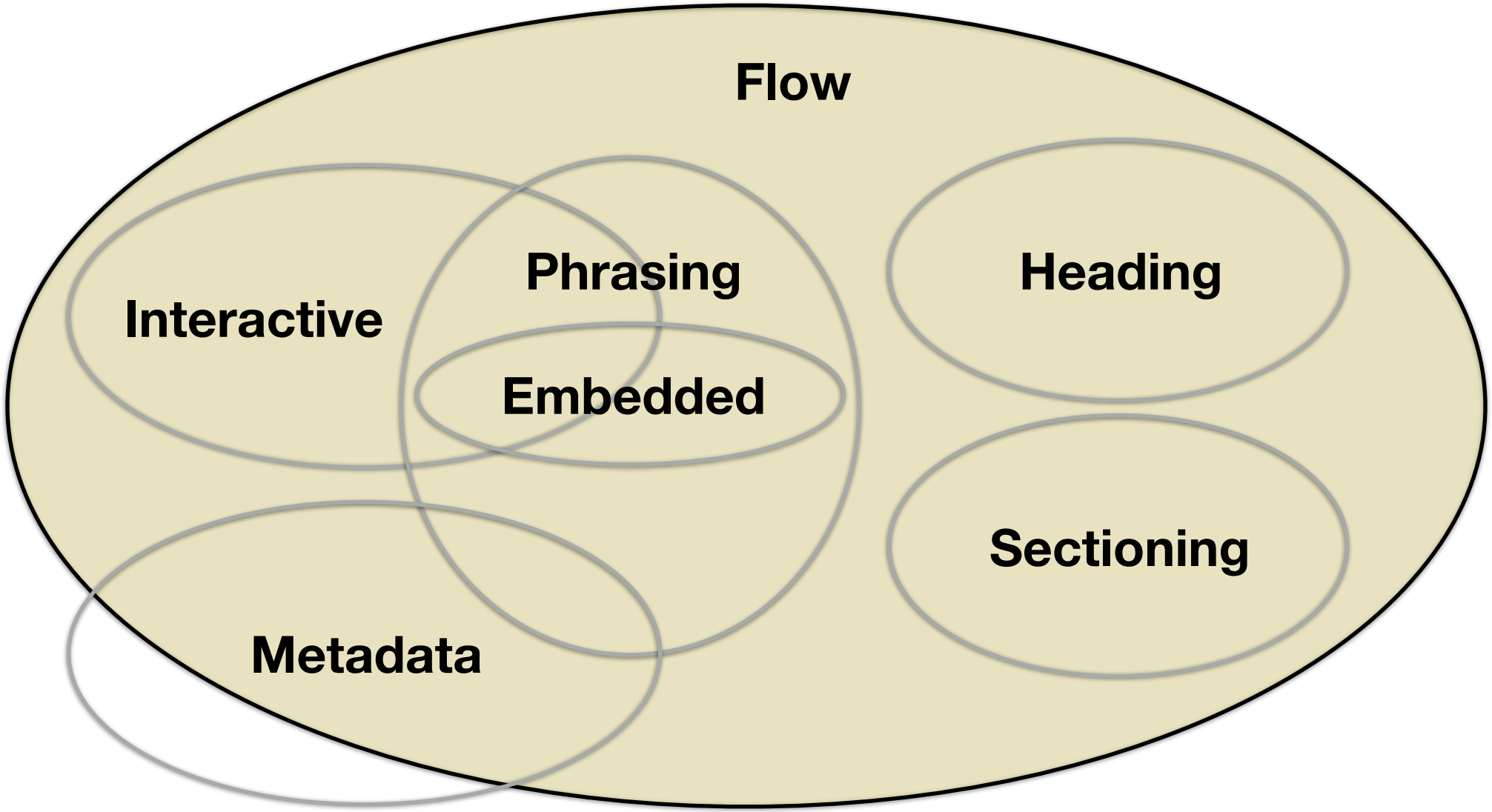
Metadata content


Flow
Interactive
Phrasing
Embedded
Heading
Sectioning
Metadata

Metadata content is content that **sets up the presentation or behaviour of the rest of the content**, or that sets up the relationship of the document with other documents.

```
<!-- Meta content -->
<base />
<link />
<meta />
<noscript></noscript>
<script></script>
<style></style>
<template></template>
<title></title>
```

Flow content


Flow
Interactive
Phrasing
Embedded
Heading
Sectioning
Metadata

Most elements that are used in the body of documents **are categorised as flow content**.

The only elements that **are not considered flow elements** are a subset of metadata elements.

However, some content **only exists in the flow category**. Previously, this content would have been referred to as block-level. This content includes:

```
<!-- Flow-specific content 1 -->
<address></address>
<article></article>
<aside></aside>
<blockquote></blockquote>
<details></details>
<dialog></dialog>
<div></div>
```

```
<!-- Flow-specific content 2 -->
<dl></dl>
<fieldset></fieldset>
<figure></figure>
<footer></footer>
<form></form>
<h1></h1>
<h2></h2>
```

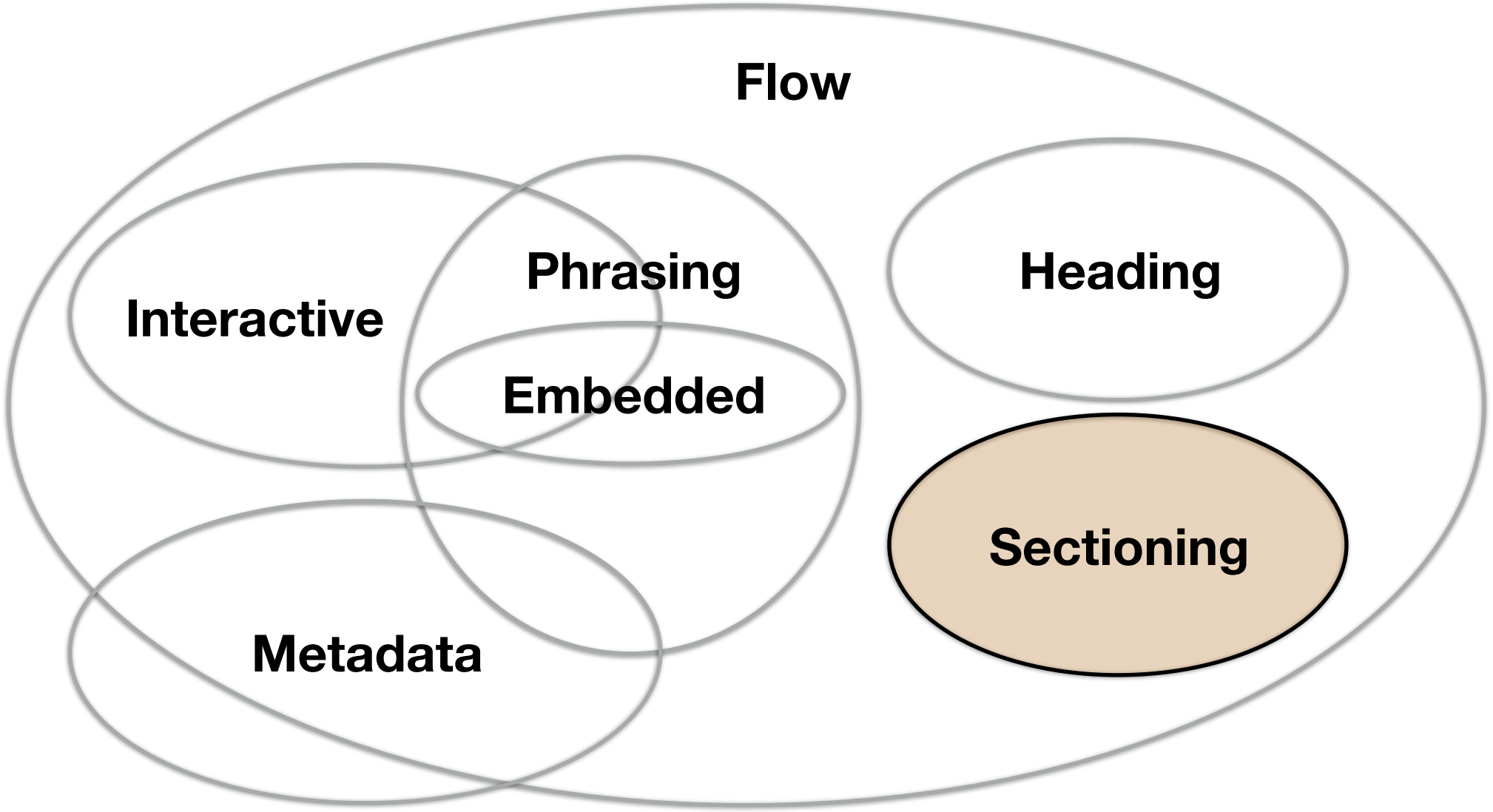
```
<!-- Flow-specific content 3 -->
<h3></h3>
<h4></h4>
<h5></h5>
<h6></h6>
<header></header>
<hr></header>
<main></main>
```

```
<!-- Flow-specific content 4 -->
<menu></menu>
<nav></nav>
<ol></ol>
<p></p>
<pre></pre>
<section></section>
<table></table>
<ul></ul>
```

The following elements are **considered flow** if the following conditions are met:

<area> (if it is a descendant of a <map> element)
<link> (if the itemprop attribute is present)
<meta> (if the itemprop attribute is present)
<style> (if the scoped attribute is present)
<a> (if it contains only phrasing content)

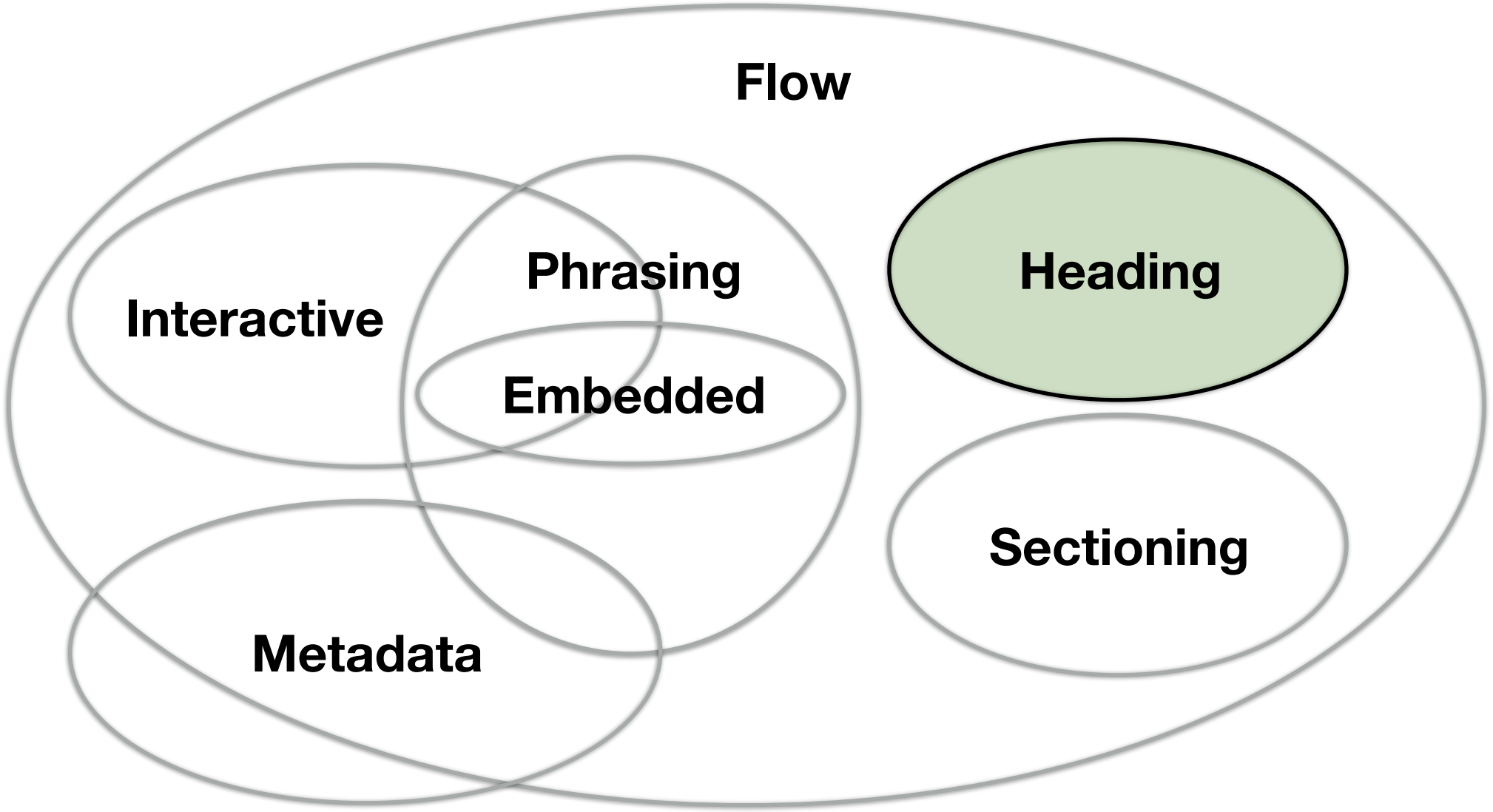
Sectioning content


Flow
Interactive
Phrasing
Embedded
Heading
Sectioning
Metadata

Sectioning content is content that **defines the scope of headings and footers**.

```
<!-- Sectioning content -->
<article></article>
<aside></aside>
<nav></nav>
<section></section>
```

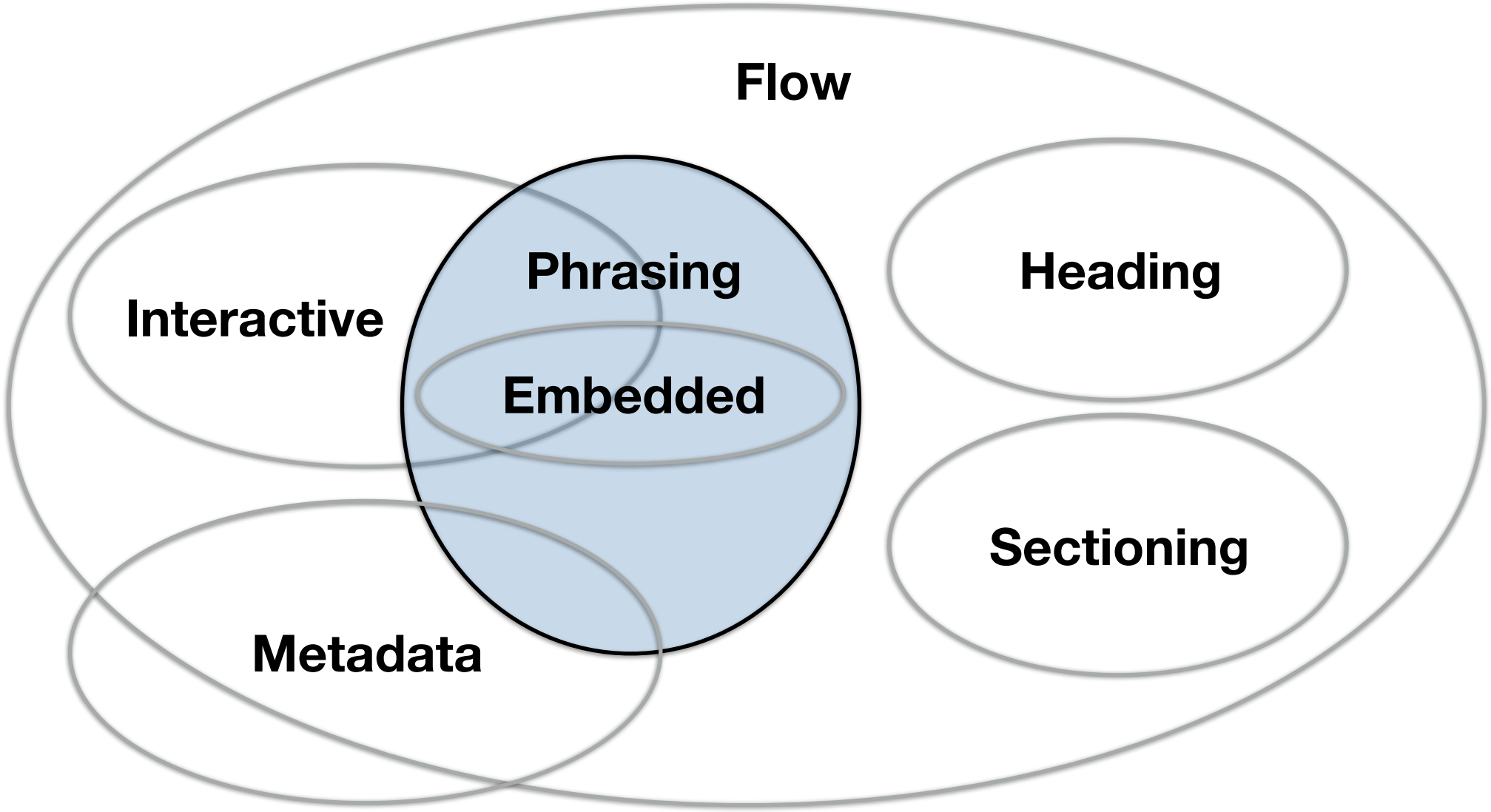
Heading content


Flow
Interactive
Phrasing
Embedded
Heading
Sectioning
Metadata

Heading content **defines the header of a section** (whether explicitly marked up using sectioning content elements, or implied by the heading content itself).

```
<!-- Heading content -->
<h1></h1>
<h2></h2>
<h3></h3>
<h4></h4>
<h5></h5>
<h6></h6>
```

Phrasing content


Flow
Interactive
Phrasing
Embedded
Heading
Sectioning
Metadata

Phrasing content is the **text of the document** (previously referred to as inline content).

```
<!-- Phrasing content 1 -->
<a></a>
<abbr></abbr>
<audio></audio>
<b></b>
<bdi></bdi>
<bdo></bdo>
```

```
<!-- Phrasing content 2 -->
<br />
<canvas></canvas>
<cite></cite>
<code></code>
<data></data>
<datalist></datalist>
<del></del>
```

```
<!-- Phrasing content 3 -->
<dfn></dfn>
<em></em>
<embed></embed>
<i></i>
<iframe></iframe>
<img />
<input />
```

```
<!-- Phrasing content 4 -->
<ins></ins>
<kbd></kbd>
<label></label>
<link></link>
<map></map>
<mark></mark>
<math></math>
```

```
<!-- Phrasing content 5 -->
<meter></meter>
<noscript></noscript>
<object></object>
<output></output>
<picture></picture>
<progress></progress>
<q></q>
```

```
<!-- Phrasing content 6 -->
<ruby></ruby>
<s></s>
<samp></samp>
<script></script>
<select></select>
<small></small>
<span></span>
```

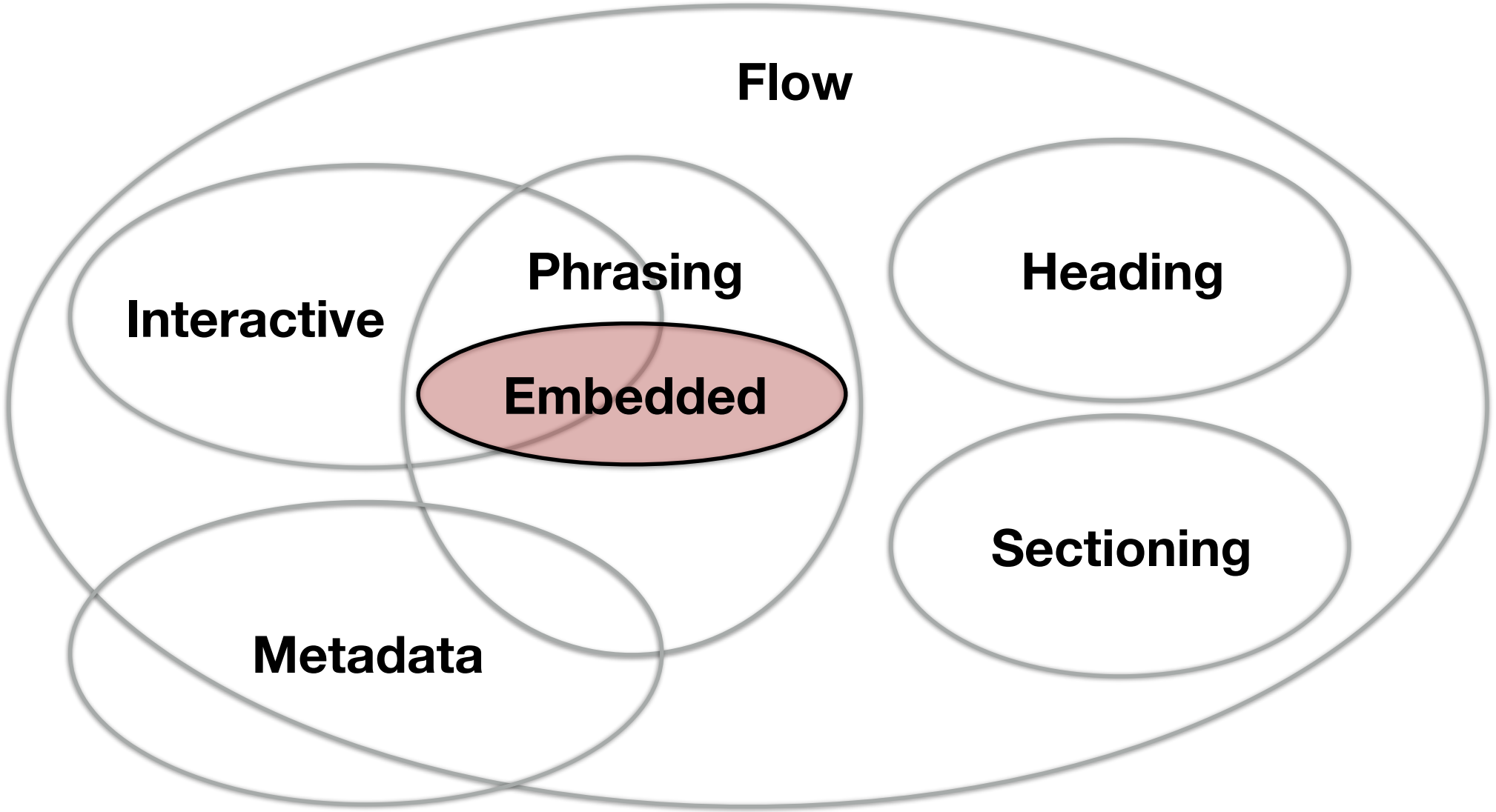
```
<!-- Phrasing content 7 -->
<strong></strong>
<sub></sub>
<sup></sup>
<svg></svg>
<template></template>
<textarea></textarea>
<time></time>
```

```
<!-- Phrasing content 8 -->
<u></u>
<var></var>
<video></video>
<wbr>
```

These elements are **considered phrasing** if the following conditions are met:

<area> (if it is a descendant of a <map> element)
<del> (if it contains only phrasing content)
<ins> (if it contains only phrasing content)
<link> (if the itemprop attribute is present)
<map> (if it contains only phrasing content)
<meta> (if the itemprop attribute is present)

Embedded content

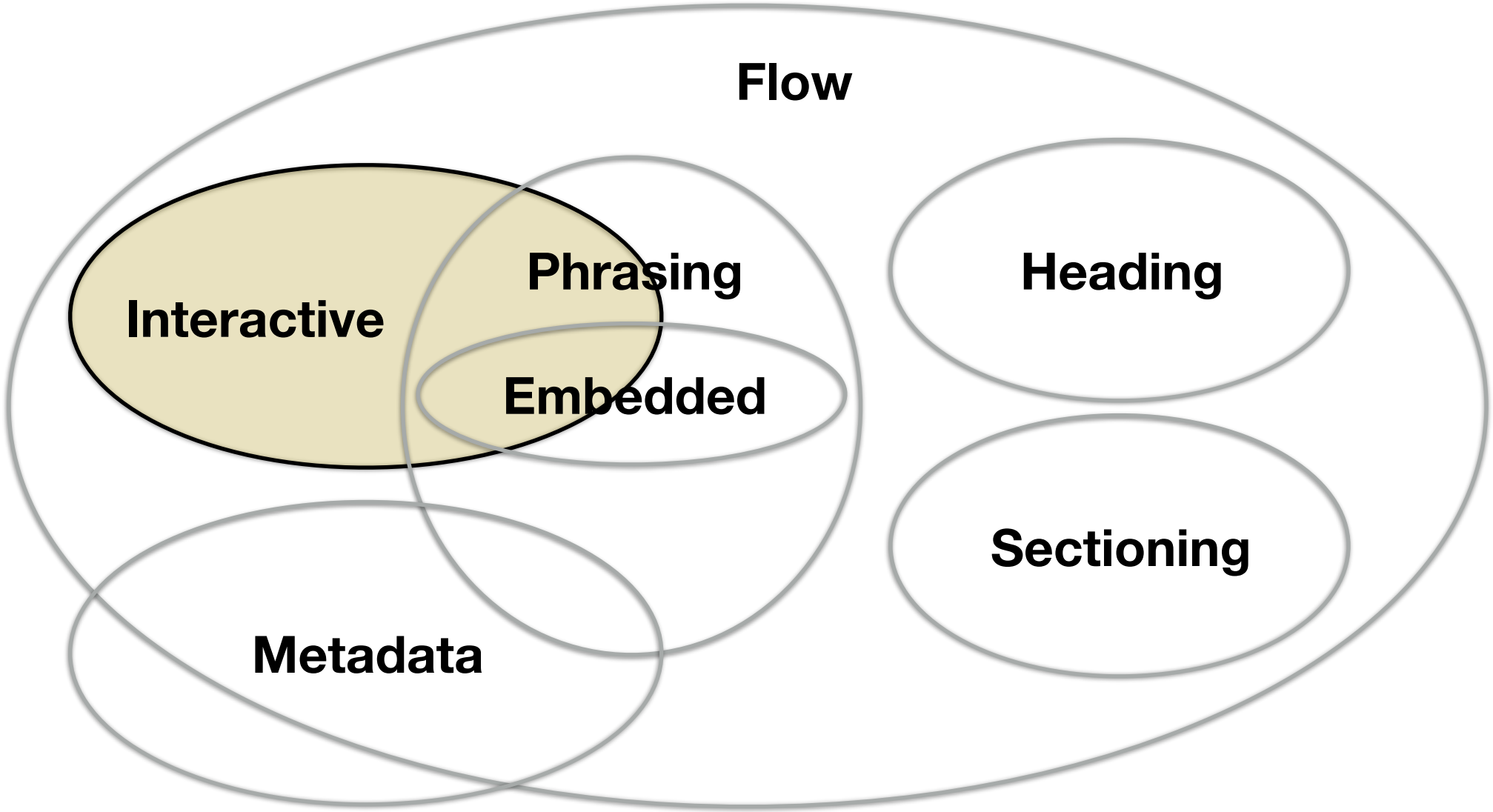

Flow
Interactive
Phrasing
Embedded
Heading
Sectioning
Metadata

Embedded content is content that **imports another resource into the document**, or content from another vocabulary that is inserted into the document.

```
<!-- Embedded content 1 -->
<audio></audio>
<canvas></canvas>
<embed />
<iframe></iframe>
<img />
<math></math>
<object></object>
```

```
<!-- Embedded content 2 -->
<picture></picture>
<svg></svg>
<video></video>
```

Interactive content


Flow
Interactive
Phrasing
Embedded
Heading
Sectioning
Metadata

Interactive content is content that is **specifically intended for user interaction**.

```
<!-- Interactive content 1 -->
<a href="#"></a> (if href present)
<audio controls></audio>
<button></button>
<details></details>
<embed></details>
<iframe></iframe>
<img usemap="#a"> (if usemap present)
```

```
<!-- Interactive content 2 -->
<input /> if the type is not hidden
<label></label>
<menu type="toolbar"></menu> (if toolbar present)
<object usemap="#a"></object> (if usemap present)
<select></select>
<textarea></textarea>
<video controls></video> (if controls present)
```

Exercise 2c: change elements

Continue working in **exercise02-change/exercise2-start.htm**

Review your work against **exercise02-change/exercise2-finished.htm**

1. Wrap relevant `<a>`'s around a block level element to avoid duplicate links.

New elements

`<audio>` can be used for **audio-based multimedia content**.

`<bdi>` represents **a span of text that is to be isolated from its surroundings** for the purposes of bidirectional text formatting.

`<canvas>` is used for **rendering dynamic bitmap graphics** on the fly, such as graphs or games.

`<datalist>` together with the a new `list` attribute for the `<input>` element **can be used to make comboboxes**.

`<embed>` is used for **plugin content**.

<mark> represents **a run of text in one document** marked or highlighted for reference purposes, due to its relevance in another context.

`<meter>` represents a **measurement**, such as disk usage.

`<output>` represents **some type of output**, such as from a calculation done through scripting.

`<progress>` represents a **completion of a task**, such as downloading or when performing a series of expensive operations.

`<ruby>`, `<rt>`, and `<rp>` allow for **marking up ruby annotations**.

`<track>` provides **text tracks for the video element**.

`<template>` can be used to **declare fragments of HTML** that can be cloned and inserted in the document by script.

`<video>` can be used for **video-based multimedia content**.

`<wbr>` represents a **line break** opportunity.

The <time> element

`<time>` represents a **date and/or time**.

```
<time>October 19</time>
```

If the element needs a date, and the `datetime` attribute is present, then the attribute's value must be a **valid date string with optional time**.

```
<time datetime="2018-10-19">October 19</time>
```

A **time** consists of a specific time with no time-zone information, consisting of an hour, a minute, a second, and a fraction of a second.

```
<!-- Valid year, month, date strings -->
<time datetime="2018">xxx</time>
<time datetime="2018-10">xxx</time>
<time datetime="10-19">xxx</time>
```

```
<!-- Valid time strings -->
<time datetime="14:54">xxx</time>
<time datetime="14:54:39">xxx</time>
<time datetime="14:54:39.929">xxx</time>
```

```
<!-- Valid year, month, date and time string -->
<time datetime="2018-10-19T14:54:39.929">xxx</time>
```

```
<!-- Time zones -->
<time datetime="2018-10-19T14:54:39.929+00:00">xxx</
time>
```

The <dialog> element

Newly released in HTML 5.2, the `<dialog>` element aims to provide a simple way to **include modal dialogs to web pages**.

```
<dialog>
  <h2>Dialog Title</h2>
  <p>Dialog content</p>
</dialog>
```

By default, the `<dialog>` is **hidden from view** (and from DOM access) unless the **open** attribute is applied.

```
<dialog open>
  <h2>Dialog Title</h2>
  <p>Dialog content</p>
</dialog>
```

The `open` attribute **can be toggled** by calling the `show()` and `close()` methods, which is available to any `HTMLDialogElement`.

New structural elements

A series of new elements have also been introduced to **allow authors to produce better structure** within their HTML markup.

These include: `<header>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<aside>`, `<footer>`, `<figure>` and `<figcaption>`.

We will look at these **in detail** soon.

New attributes

A new `autofocus` attribute provides **a declarative way to focus a form control** during page load.

A new `placeholder` attribute represents **a hint intended to aid the user with data entry**.

The new `form` attribute allows form controls to be associated with a `<form>` element **anywhere on a page**, not just as descendants of their parent `<form>` element.

The new `required` attribute **defines the relevant form control as required**.

The `<fieldset>` element **now allows** the `disabled` attribute to be applied.

The `<input>` element has **several new attributes to specify constraints**: `autocomplete`, `min`, `max`, `multiple`, `pattern` and `step`.

The `<textarea>` element **has three new attributes**: `maxlength`, `minlength` and `wrap` which control maximum and minimum input length and submitted line wrapping behaviour.

The `<input>` element has had support for the `maxlength` attribute since HTML 4.01. This element **now supports the new** `minlength` attribute as well.

The `<ol>` element has a new attribute called `reversed`. When present, it **indicates that the list order is descending**.

We will cover some of these attributes in more details **when we look at forms**.