Dr. Wittawin Susutti

wittawin.sus@kmutt.ac.th

CSS233 WEB PROGRAMMING I

LECTURE 02 HTML & CSS I

Outline

- Introduction
- HTML
  - Structure
  - Component
  - Element
- HTML5
- Semantic HTML
- CSS
- CSS property
- CSS Selector
- Specificity

Introduction

- **HTML**
  - Structure and content of a webpage.
  - Nouns of a webpage.
- **CSS**
  - Style of HTML.
  - Adj. of a webpage.
- **JavaScript**
  - Adds logic and interactivity to a page.
  - Verb of a webpage.

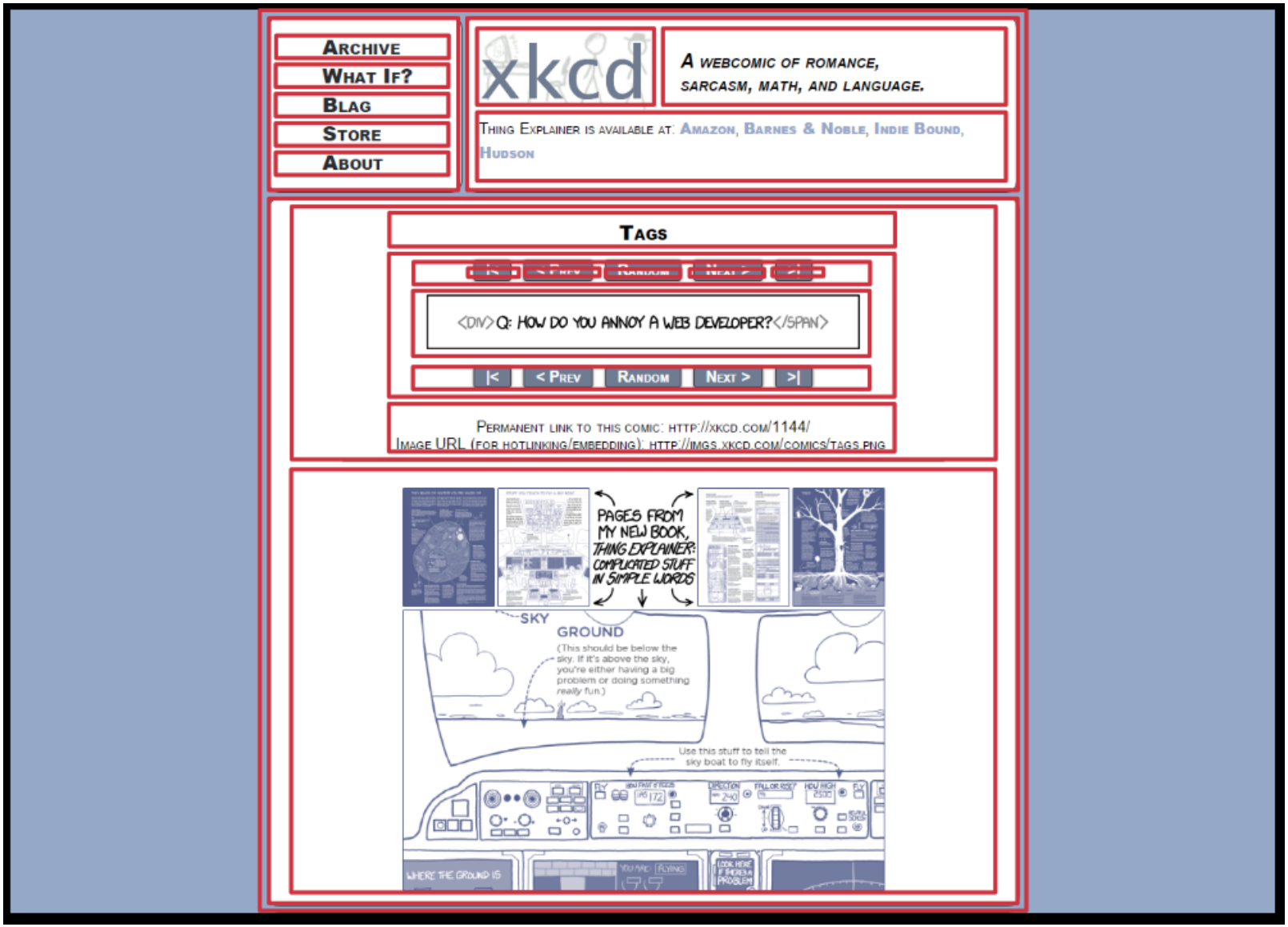

Archive
What If?
Blag
Store
About
xkcd
A webcomic of romance, sarcasm, math, and language.
Thing Explainer is available at: Amazon, Barnes & Noble, Indie Bound, Hudson
Tags
Q: How do you annoy a web developer?
|< < Prev Random Next > >|
Permanent link to this comic: http://xkcd.com/1144/
Image URL (for hotlinking/embedding): http://imgs.xkcd.com/comics/tags.png
Pages from my new book, Thing Explainer: Complicated stuff in simple words
SKY
GROUND
(This should be below the sky. If it's above the sky, you're either having a big problem or doing something really fun.)
Use this stuff to tell the sky boat to fly itself.
WHERE THE GROUND IS
YOU ARE: FLYING
LOOK HERE IF THERE'S A PROBLEM

Image: MIT

HTML

- **HTML (Hypertext Markup Language)**
  - Describes the **content and structure of a webpage**
  - **NOT** a programming language.
  - Made up of building blocks called **elements.**

- Basic HTML page structure
- Saved in a *filename*.***html*** file.

```
<!DOCTYPE html>
<html>
  <head>
    <title>title</title>
  </head>
  <body>

  </body>
</html>
```

Metadata that doesn't appear in the viewport of the browser

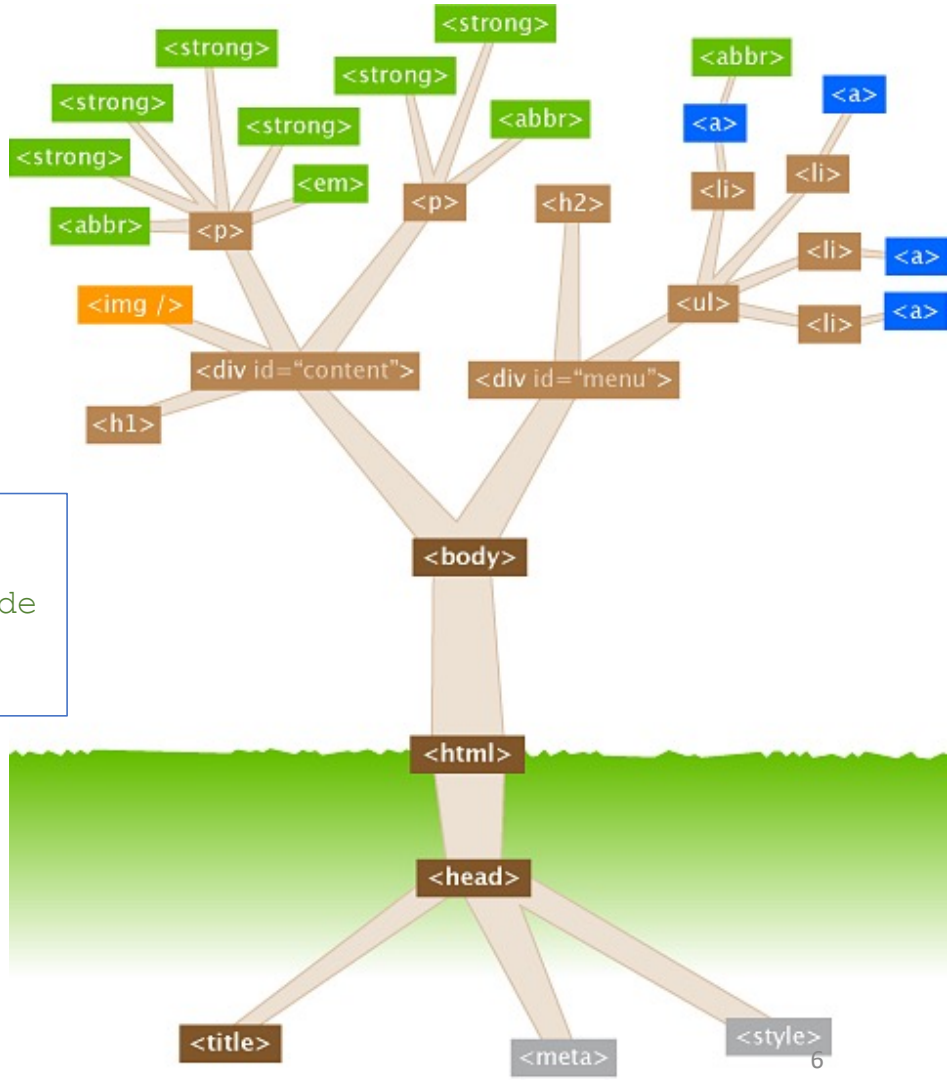
Contents that render in the viewport of the browser

HTML Structure

- HTML is a tree structure
  - Internal nodes represent structure
  - Leaf nodes represent content
- Specified textually as a tree

```
<node>
    <subnode field='value'>
        <leafnode /> -> Text in a  leaf node
    </subnode>
</node>
```

- Maintained internally as a tree (DOM - Document Object Model)
- Nodes have names, attributes
- Text may appear at leaves



Source: http://watershedcreative.com/

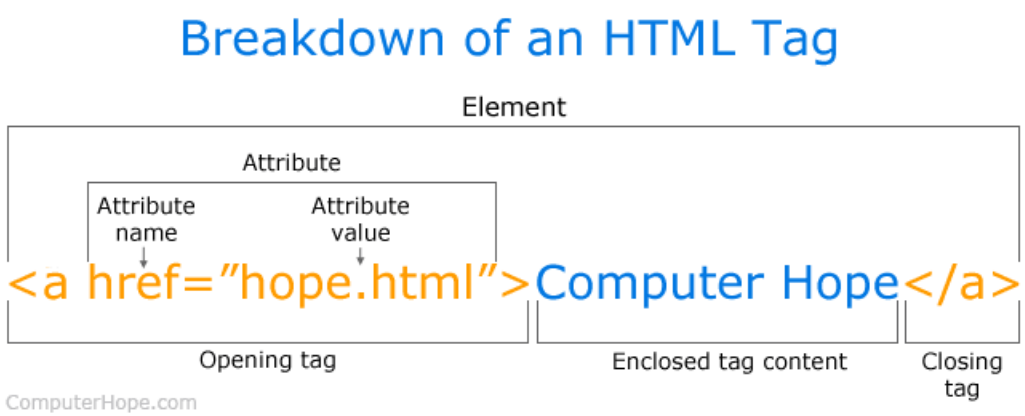
HTML Components

- Header: basic information about the page
  - Styles (CSS): information on how to display
    - Can be in separate files
  - Scripts (JavaScript)
    - Dynamic interactivity
    - Can be in separate files
- Body: the data to display
  - Description of what should be presented on the page

Basic HTML Body Components

- Text
- Descriptions of how to display text
  - `<em>text</em>`, `<span class='emph'>text</span>`
  - Managed by CSS
- Text organization
  - Headers, paragraphs, blocks, tables
- Page layout and organization
  - `DIV, LIST, TABLE, FRAME`
- Interactive regions
  - Forms: text fields, buttons, …
  - Canvas, SVG regions

HTML elements



- An element can have attributes (**`href="hope.html"`**)
- Elements can contain other elements (**`p`** contains **`em`** and **`img`**)

```
<p>
        Hello Kitty is <em>cute</em>
        <img src="kitty.png">
</p>
```

- An element can be self-closing (**`img`**)

Some HTML elements

Heading `h1, h2, ... h6`	`<h1>CSS233</h1>`
Paragraph	`<p>Hello Kitty is sexy.</p>`

Division container	`<div class="shadowbox">`
	`  <p>This's a very interesting topic.</p>`
	`</div>`

Inline container	`<p>Some text is <span>red</span></p>`
Line break	`This is the first line. ` `This is second line.`
Image	`<img src="kitty.png">`
Link	`<a href="google.com">Go to Google!</a>`
Strong (bold)	`<strong>BOLD</strong>`
Emphasis (italic)	`Here is <em>italic</em>.`

title Element

- The head element contains two types of elements—**meta** and **title**.
- The title element's contained data specifies the label that appears in the browser window's **title bar**.
- Besides providing a label for your browser window's title bar, what's the purpose of the title element?
  - It provides documentation for someone trying to maintain your web page
  - It helps web search engines find your web page

meta Element

- The meta elements provide **information** about the web page.
- There are many different types of meta elements
  - some you should always include, but most are just optional.
- *The meta element does not have an end tag.*
- Attributes:
  - charset
  - name
  - content

```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Travelling through Surrey, the
best pubs in Egham">
  <meta name="keywords" content="Egham,Surrey,Pubs, travel">
  <meta name="author" content="Joe Bloggs">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
</head>
```

Image:webtrends-optimize.com

List tag

- **`<ol>`: The Ordered List element**

```
<ol>
  <li>Iron man</li>
  <li>Captain america</li>
  <li>Thor</li>
  <li>Hulk</li>
  <li>Black widow</li>
  <li>Hawkeye</li>
</ol>
```

- **`<ul>`: The Unordered List element**

```
<ul>
  <li>Iron man</li>
  <li>Captain america</li>
  <li>Thor</li>
  <li>Hulk</li>
  <li>Black widow</li>
  <li>Hawkeye</li>
</ul>
```

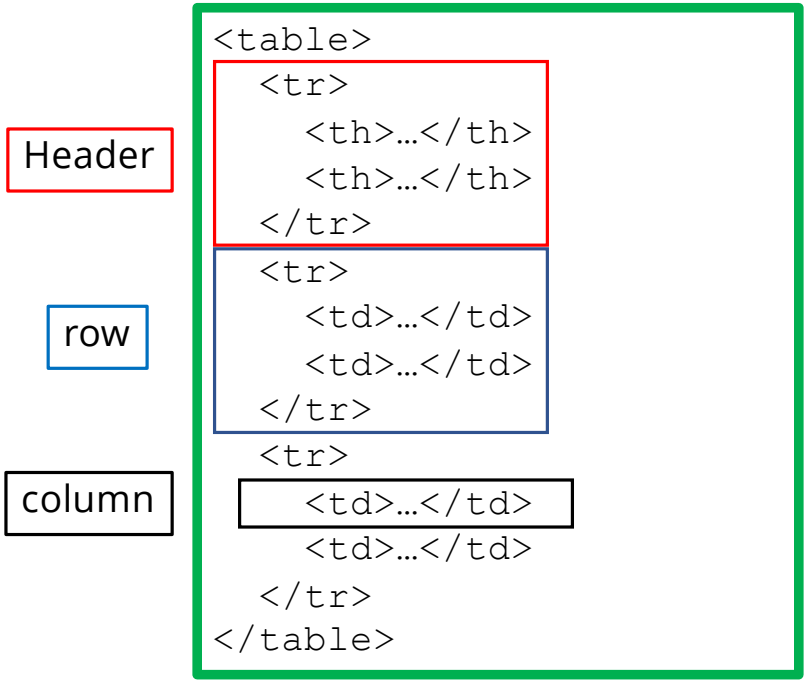
`<img>`: The Image Embed element

- Main attribute
  - `src`
  - `alt`
  - `height`
  - `width`

```
<img src="Hello_kitty_character_portait.png"
alt="Hello kitty with blue bg." height="100px">
```

HTML Table

`<table>`: The Table element



<form> tag

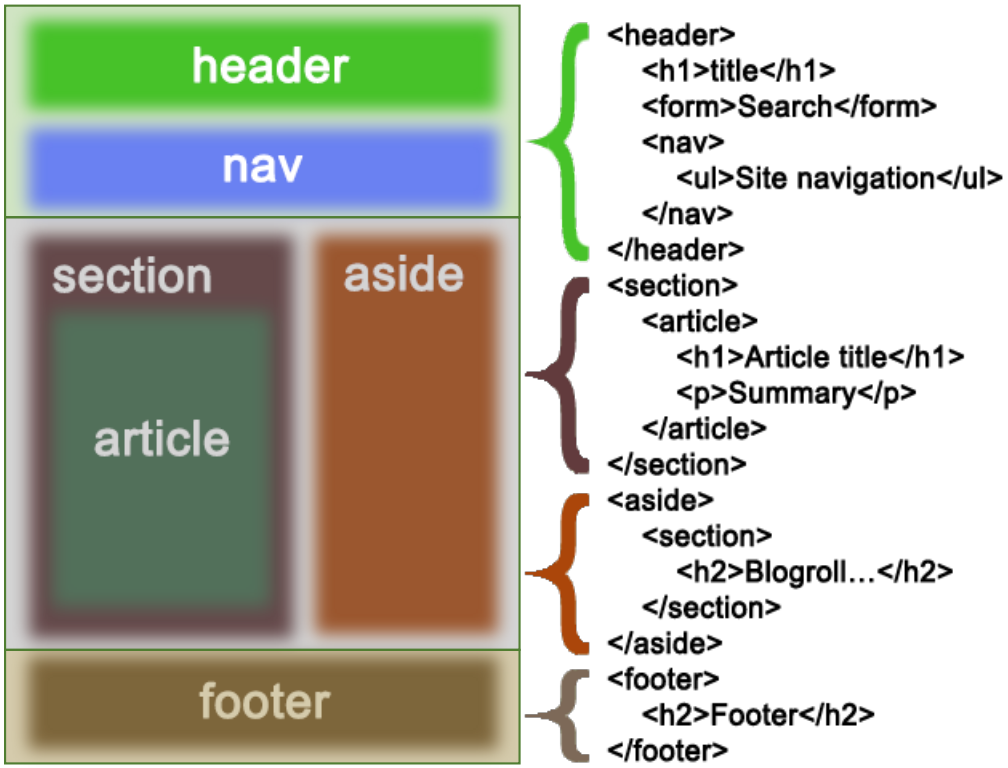
- <form> tag
  - action
  - method
- <input> tag
  - type – text, color, radio, password, and more types here!
  - placeholder
  - name
- <button> tag
- <label>
- Validation

The HTML5 standard

- **Audio and video**—The audio and video elements allow users to play music and video files directly from their browsers without the need of a plug-in.
- **Canvas**—The canvas element provides a drawing area and a set of commands that a web programmer can use to draw two-dimensional shapes and animate them.
- **Drag and drop functionality**—The drag and drop constructs provide the ability to drag elements within a web page.
- **Web storage functionality**—The web storage constructs provide the ability to permanently store data on the browser's computer.
- **Geolocation functionality**—The geolocation constructs provide the ability to locate the browser's computer.

The HTML5 standard

Structural organization elements



Source:htmlweblearn

Semantic HTML

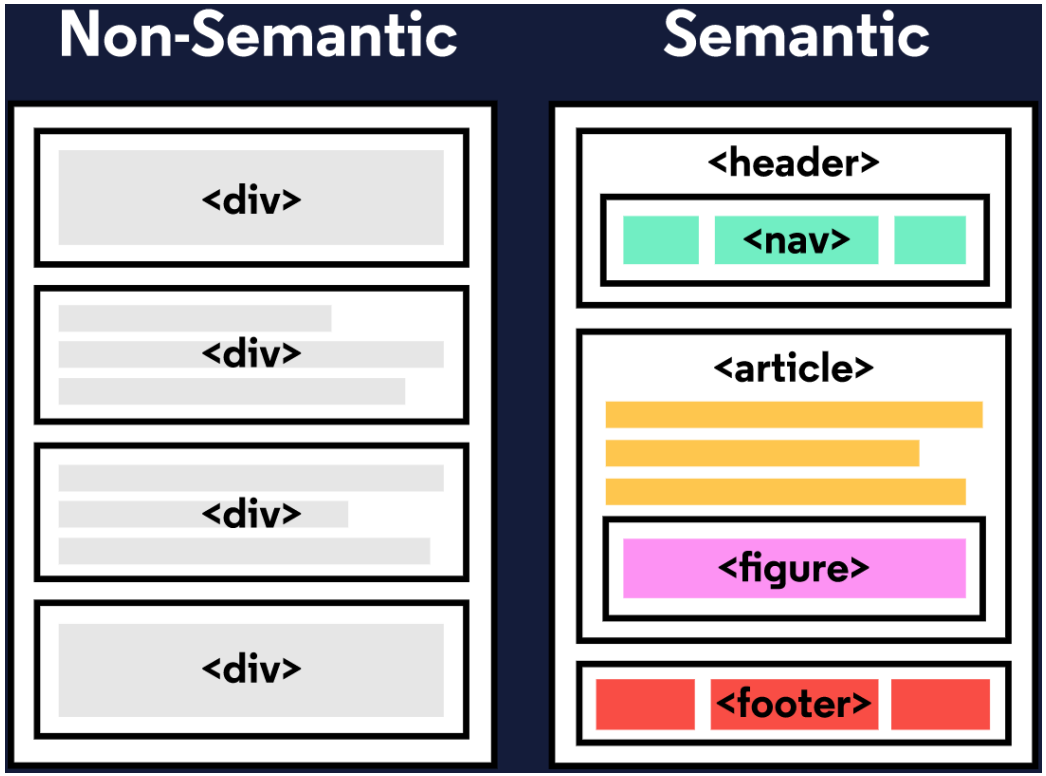
```
<!--Non Semantic HTML-->
<div id="footer">
 <p>this is a footer</p>
</div>

<!--Semantic HTML-->
<footer>
 <p>this is a footer</p>
</footer>
```

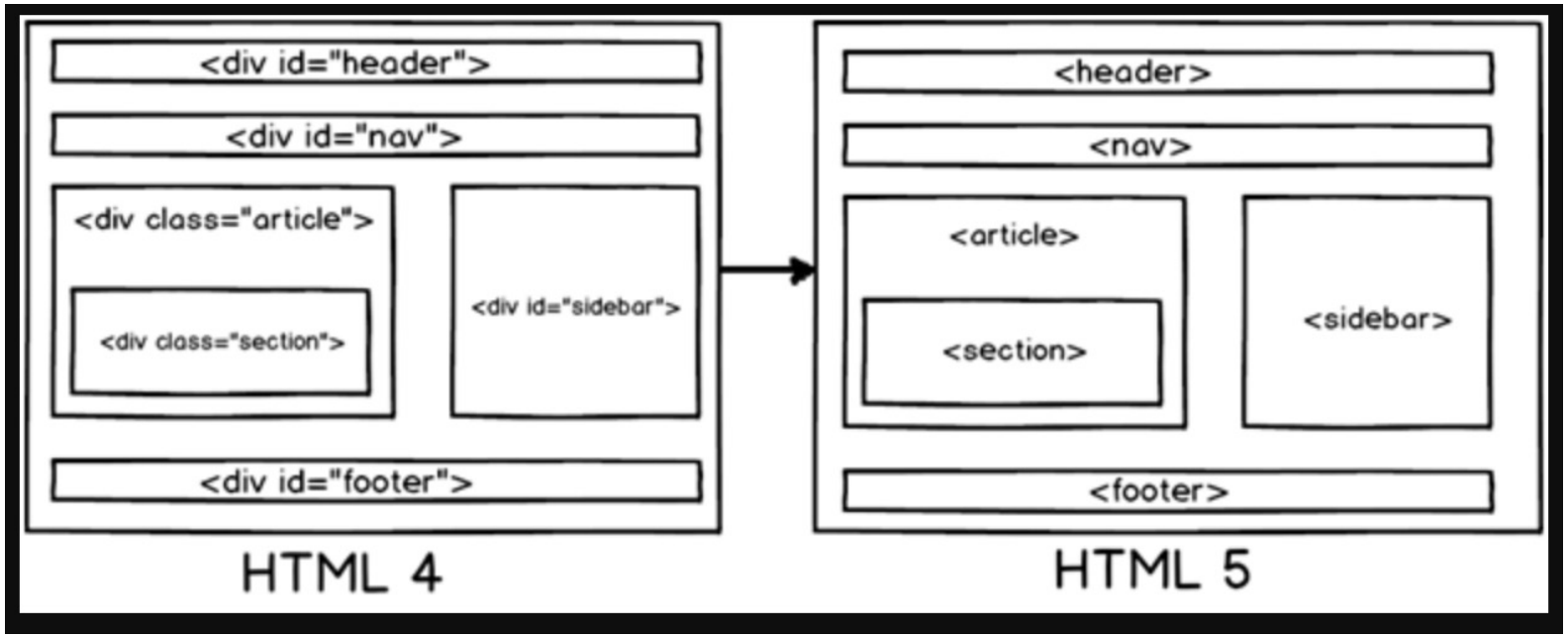
?

Semantic HTML

- When building web pages, we use a **combination** of non-semantic HTML and Semantic HTML.
- The semantic elements provide information about the content between the opening and closing tags.
- By using Semantic HTML, we select HTML elements based on their meaning, not on how they are presented.
- For example, instead of using a **`<div>`** element to contain our header information, we could use a **`<header>`** element, which is used as a heading section.



Semantic HTML



Why use Semantic HTML?

- **Accessibility**
  - Semantic HTML makes webpages accessible for mobile devices and for people with disabilities as well.
  - This is because screen readers and browsers can interpret the code better.
- **SEO**
  - It improves the website SEO, or ***Search Engine Optimization***, which is the process of increasing the number of people that visit your webpage.
  - With better SEO, search engines are better able to identify the content of your website and weight the most important content appropriately.
- **Easy to Understand**
  - Semantic HTML also makes the website's source code easier to read for other web developers.

Header

- A **<header>** is a container usually for either navigational links or introductory content containing **<h1>** to **<h6>** headings.
- By using a **<header>** tag, the code becomes easier to read.
- It is much easier to identify what is inside of the **<h1>**'s parent tags, as opposed to a ** ** tag which would provide no details as to what was inside of the tag.

```
<html>
  <body>
    <div>
      <h1>Navigational Links</h1>
      <div>
        <ul>
          <li><a href="#home">Home</a></li>
          <li><a href="#posts">Posts</a></li>
          <li><a href="#contact">Contact</a></li>
        </ul>
      </div>
    </div>
  </body>
</html>
```

```
<html>
  <body>
    <header>
      <h1>Navigational Links</h1>
      <div>
        <ul>
          <li><a href="#home">Home</a></li>
          <li><a href="#posts">Posts</a></li>
          <li><a href="#contact">Contact</a></li>
        </ul>
      </div>
    </header>
  </body>
</html>
```

Nav

- A **<nav>** is used to define a block of navigation links such as menus and tables of contents.
- It is important to note that **<nav>** can be used inside of the **<header>** element but can also be used on its own.
- By using **<nav>** to label the navigation links, it will be easier for web browsers and screen readers to read the code.

```
<html>
  <body>
    <header>
      <h1>Navigational Links</h1>
      <div>
        <ul>
          <li><a href="#home">Home</a></li>
          <li><a href="#posts">Posts</a></li>
          <li><a href="#contact">Contact</a></li>
        </ul>
      </div>
    </header>
  </body>
</html>
```

```
<html>
  <body>
    <header>
      <h1>Navigational Links</h1>
      <nav>
        <ul>
          <li><a href="#home">Home</a></li>
          <li><a href="#posts">Posts</a></li>
          <li><a href="#contact">Contact</a></li>
        </ul>
      </nav>
    </header>
  </body>
</html>
```

Main and Footer

- Two more structural elements are **`<main>`** and **`<footer>`**.
- The element **`<main>`** is used to encapsulate the dominant content within a webpage.
- This tag is separate from the **`<footer>`** and the **`<nav>`** of a web page since these elements don't contain the principal content.
- By using **`<main>`** as opposed to a **`<div>`** element, screen readers and web browsers are better able to identify that whatever is inside of the tag is the bulk of the content.

Main and Footer

- The content at the bottom of the subject information is known as the footer, indicated by the **`<footer>`** element.
- The footer contains information such as:
  - Contact information
  - Copyright information
  - Terms of use
  - Site Map
  - Reference to top of page links
- The **`<footer>`** tag is separate from the **`<main>`** element and typically located at the bottom of the content.

Article and Section

- **`<section>`** defines elements in a document, such as chapters, headings, or any other area of the document with the same theme.
- For example, content with the same theme such as articles about something can go under a single **`<section>`**.
- A website's home page could be split into sections for the introduction, news items, and contact information.
- The **`<article>`** element holds content that makes sense on its own.
- **`<article>`** can hold content such as articles, blogs, comments, magazines, etc.
- An **`<article>`** tag would help someone using a screen reader understand where the article content begins and ends.

The Aside Element

- The **<aside>** element is used to mark additional information that can enhance another element but isn't required in order to understand the main content.
- This element can be used alongside other elements such as **<article>** or **<section>**.
- Some common uses of the **<aside>** element are for:
  - Bibliographies
  - Endnotes
  - Comments
  - Pull quotes
  - Editorial sidebars
  - Additional information

```
<article>
  <p>The first World Series was played between
Pittsburgh and Boston in 1903 and was a nine-game
series.</p>
</article>
<aside>
  <p>
   Babe Ruth once stated, "Heroes get remembered, but
legends never die."
  </p>
</aside>
```

Figure and Figcaption

- **`<figure>`** is an element used to encapsulate media such as an image, illustration, diagram, code snippet, etc, which is referenced in the main flow of the document.
- It's possible to add a caption to the image by using **`<figcaption>`.**
- **`<figcaption>`** is an element used to describe the media in the **`<figure>`** tag.
- Usually, **`<figcaption>`** will go inside **`<figure>`.**
- This is useful for grouping an image with a caption.

```
<figure>
  <img src="overwatch.jpg">
  <figcaption>This picture shows characters from
Overwatch.</figcaption>
</figure>
```

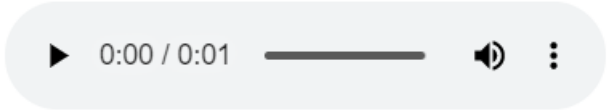
Audio and Attributes

- The **<audio>** element is used to embed audio content into a document.
- Like **<video>, <audio>** uses **src** to link the audio source.

```
<audio>
  <source src="iAmAnAudioFile.mp3" type="audio/mp3">
</audio>
```

- Specifying the type is recommended as it helps the browser identify more easily and determine if that type of audio file is supported by the browser.
- Attributes provide additional information about an element.

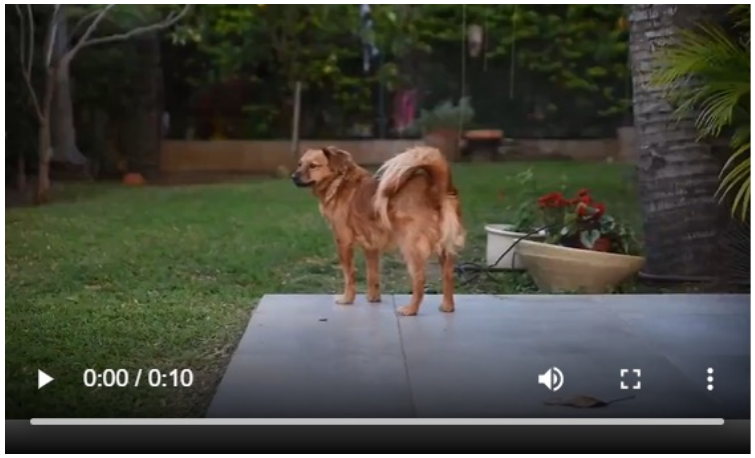
```
<audio autoplay controls>
```



Video and Embed

- By using a **<video>** element, we can add videos to our website.
- The **<video>** element makes it clear that a developer is attempting to display a video to the user.
- The **<embed>** tag can embed any media content including videos, audio files, and gifs from an external source.

```
<video src="coding.mp4" controls>Video not
supported</video>
```

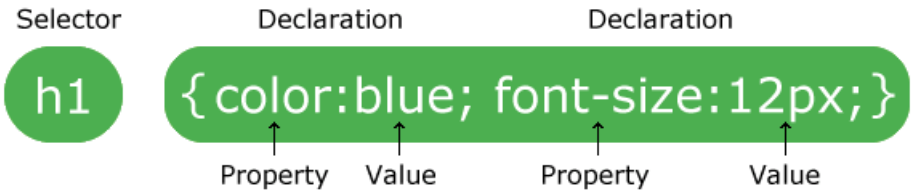


Semantic HTML tag

Tag	Function
`<header>...</header>`	Contains introductory information.
`<footer>...</footer>`	Contains supplementary material for its containing element (commonly a copyright notice or author information).
`<nav>...</nav>`	Contains navigational elements.
`<section>...</section>`	Contains thematically similar content, such as a chapter of a book or a section of a page.
`<article>...</article>`	Contains content that is a standalone body of work, such as a news article.
`<aside>...</aside>`	Contains secondary information for its containing element.
`<address>...</address>`	Contains address information related to its nearest `<article>` or `<body>` element, often contained within a `<footer>` element.

CSS

- **CSS: Cascading Style Sheets**
- Describes the **appearance** and **layout** of a web page
- Composed of CSS rules, which define sets of styles
- CSS Syntax



Credit: w3schools

- Saved in a *filename*`.css` file.

How CSS works

- A ***style rule*** is a formatting instruction that can be applied to an element on a web page.
- A style rule consists of one or more style properties and their associated values.
- The ***cascading*** part of the name CSS refers to the manner in which styles in a CSS style sheet form a hierarchy in which **more specific styles override more general styles**.

```
<div style="color:green;">
      This text is green.
      <p style="color:blue;">This text is blue.</p>
      <p>This text is still green.</p>
</div>
```

This text is green.

This text is blue.

This text is still green.

Linking CSS in HTML

- **Inline** - by using the style attribute in HTML elements

```
<h1 style="color:blue;">This is a Blue Heading</h1>
```

- **External** - by using an external CSS file and add a link in <head> section

```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
```

- **Internal** - by using a <style> tag in the <head> section

```
<head>
 <style>
   body {background-color: powderblue;}
   h1   {color: blue;}
   p    {color: red;}
 </style>
</head>
```

Absolute and Relative paths

- **Absolute path** - provide the full website address.
  - ***always*** include the domain name of the website.
  - must use if link to a location on another website.

Absolute Paths
http://www.mysite.com
http://www.mysite.com/graphics/image.png
http://www.mysite.com/articles/webpage.html

- **Relative path** – provide file path or folder
  - only point to a file or a file path.
  - easy when you change your domain name.

Relative Paths
index.html
/graphics/image.png
/articles/webpage.html

A CSS Style Primer

- The style properties in CSS can be generally grouped into two major categories:
  - **Layout properties**—Properties that affect the positioning of elements on a web page, such as margins, padding, and alignment
  - **Formatting properties**—Properties that affect the visual display of elements in a website, such as the font type, size, and color

Some CSS properties

There are over 500 CSS properties!

Font typeface	`font-family: Helvetica;`
Font color	`color: gray;`
Font style	`font-style: italic;`
Font weight	`font-weight: normal;`
Background color	`background-color: red;`
Border	`border: 3px solid green;`
Text alignment	`text-align: center;`

CSS Selectors

The type selector

```
p {
  text-align: center;
  color: red;
}
```

The class selector

```
.class {
  text-align: center;
  color: red;
}
```

The ID selector

```
#id1 {
  text-align: center;
  color: red;
}
```

Grouping Selector

```
h1 {
  text-align: center;
  color: red;
}
p {
  text-align: center;
  color: red;
}
```

```
h1, p {
  text-align: center;
  color: red;
}
```

CSS Selectors

Class

- Can use the same class on multiple elements
- Can use multiple classes on the same element
- Multiple classes

```
<span class="subject new">CSS</span>
```

ID

- Each element can have only one ID
- Each page can have only one element with that ID

ID Selector	Class Selector
• ID selector uses ID to select elements	• The class selector uses the CSS class to select elements.
• When you just need to select only one element, use ID selector.	• When you want to select a group of elements, having the same CSS class
• ID selector uses # character.	• The class selector uses “.” character.

CSS pseudo-classes

- **pseudo-classes**: special keywords you can append to selectors, specifying a *state* or *property* of the selector

Syntax	Explanation
`a`	All anchor tags (links) in all states
`a:visited`	A visited link
`a:link`	An unvisited link
`a:hover`	The style when you hover over a link
`a:active`	The style when you have "activated" a link (downclick)

CSS Pseudo-elements

- A CSS pseudo-element is used to style specified parts of an element.
- For example, it can be used to:
  - Style the first letter, or line, of an element
  - Insert content before, or after, the content of an element

```
selector::pseudo-element {
  property: value;
}
```

- All CSS Pseudo Elements

Selector	Example	Description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of each <p> element
::first-line	p::first-line	Selects the first line of each <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user

Selector summary

Example	Description
`p`	All `<p>` elements
`.abc`	All elements with the **abc class**, i.e., `class="abc"`
`#abc`	Element with the **abc id**, i.e., `id="abc"`
`p.abc`	`<p>` elements with **abc class**
`p#abc`	`<p>` element with **abc id** (`p` is redundant)
`div strong`	`<strong>` elements that are **descendants** (inside) of a `<div>`
`h2, div`	`<h2>` elements and `<div>`
`*`	All elements
`h2 + h3`	`<h3>` elements adjacent to `<h2>`
`ul > li`	`<li>` elements which are direct children of a `<ul>`
`p ~ img`	All `<img>` elements that come *anywhere* after `<p>` elements

Grouping selectors

2 Common bugs:

- `p.abc` **vs** `p .abc`

A `<p>` element with the **abc** class
vs
An element with the **abc** class that descends from **<p>**

- `p .abc` **vs** `p, .abc`

An element with the **abc** class that descends from **<p>**
vs
All **<p>** elements ***and*** all elements with the **abc** class

Colliding styles

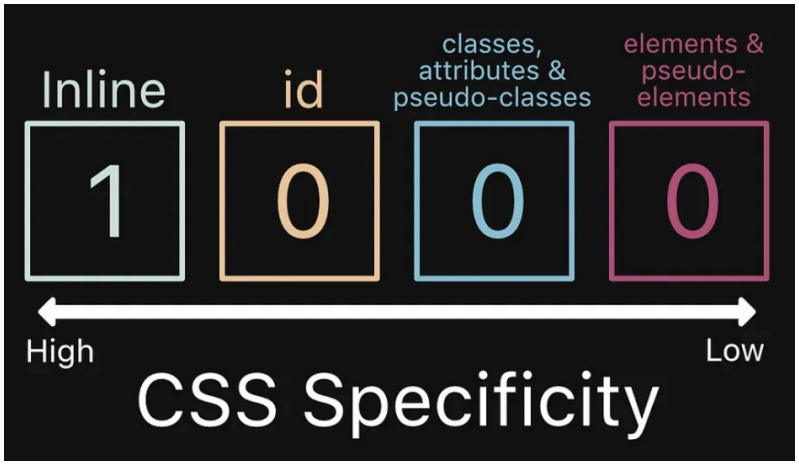
```
strong { color: red; }
strong { color: blue; }

<div>
    <strong>What color am I?</strong>
</div>
```

```
div strong { color: red; }
strong { color: blue; }

<div>
  <strong>What color am I?</strong>
</div>
```

How is specificity calculated?



- For each **ID** in a matching selector, add **1-0-0** to the weight value.
- For each **class**, **attribute selector**, or **pseudo-class** in a matching selector, add **0-1-0** to the weight value.
- For each **type** or **pseudo-element** in a matching selector, add **0-0-1** to the weight value.
- The universal selector (*) and the pseudo-class :where() and its parameters aren't counted, their value is 0-0-0, but they do match elements.
- Combinators, such as +, >, ~, " ", and ||, may make a selector more specific but they don't add any value to the specificity weight.
- The negation pseudo-class, :not(), itself has no weight. Neither do the :is() or the :has() pseudo-classes.
- Inline styles always overwrite any normal styles in author stylesheets. Think of inline styles as having a specificity weight of 1-0-0-0.

How is specificity calculated?

```
#myElement {
	color: green;
}
.bodyClass .sectionClass .parentClass [id="myElement"] {
	color: yellow;
}
```

```
input.myClass {
	color: yellow;
}
:root input {
	color: green;
}
```

```
#myElement {
	color: yellow;
}
#myApp [id="myElement"] {
	color: green;
}
```

```
:root input {
	color: green;
}
html body main input {
	color: yellow;
}
```

Colliding styles

- When styles collide, there are following these rules:
  1. `!important` (Bad for maintenance)
  2. The most specific rule
  3. Source order

- Specificity precedence rules:
  - Inline styles are the most specific. (Not a good practice)
  - Ids are more specific than classes.
  - Classes are more specific than element names.
  - Style rules that directly target elements are more specific than style rules that are inherited.
  - The universal selector (`*`) has no specificity value.

Inheritance

- Some CSS styles are inherited from parent to child.

Instead of selecting all elements individually:

```
a, h1, p, strong {
 font-family: Helvetica;
}
```

You can style the parent and the children will inherit the styles.

```
body {
 font-family: Helvetica;
}
```

You can override this style via specificity:

```
h1, h2 {
 font-family: Consolas;
}
```

- Not all CSS properties are inherited.

User-agent styles

- The browser has its own default styles.
- Browser loads its own default stylesheet on every webpage.
- User-agent stylesheets can vary between different browsers and versions of browsers and can also be influenced by the operating system being used.
- Web developers can override the styles defined by the user agent stylesheet by defining their own styles in a stylesheet.

Online references and tools

- Reference
  - https://www.w3schools.com/
  - https://developer.mozilla.org
- Online editor
  - https://jsbin.com
  - https://codepen.io
  - https://codesandbox.io/
- CSS design
  - https://csszengarden.com/
- CSS design awards
  - https://www.cssdesignawards.com/