HTML

CS 390 – Web Application Development

J. Setpal

August 23, 2023

Outline

Outline

WIWYT – HTML

- HTML is the most basic building block of the web.[1]

[1] MDN

WIWYT – HTML

- HTML is the most basic building block of the web.[1]
- It establishes the structure of a given webpage, on top of which we add appearence and functionality.

[1] MDN

WIWYT – CSS

- HTML is awesome but without CSS, the webpage is ugly and in most cases unusable.

WIWYT – CSS

- HTML is awesome but without CSS, the webpage is ugly and in most cases unusable.
- CSS defines a set of rules applied based on HTML tags and attributes, that style and animate the webpage.

Outline

What happens when you click on a URL?

Q: Internally, what does your browser do when you request for a URL?

What happens when you click on a URL?

Q: Internally, what does your browser do when you request for a URL?
A: Think of it like a world-wide file browser.

What happens when you click on a URL?

Q: Internally, what does your browser do when you request for a URL?
A: Think of it like a world-wide file browser. When you request to open a file (click a URL):

1. The browser breaks down the request and understands it.

What happens when you click on a URL?

Q: Internally, what does your browser do when you request for a URL?
A: Think of it like a world-wide file browser. When you request to open a file (click a URL):

1. The browser breaks down the request and understands it.
2. It then resolves the domain.

What happens when you click on a URL?

Q: Internally, what does your browser do when you request for a URL?
A: Think of it like a world-wide file browser. When you request to open a file (click a URL):

1. The browser breaks down the request and understands it.
2. It then resolves the domain.
3. Finally, it sends a request to carry out the requested action.

What happens when you click on a URL?

Q: Internally, what does your browser do when you request for a URL?
A: Think of it like a world-wide file browser. When you request to open a file (click a URL):

1. The browser breaks down the request and understands it.
2. It then resolves the domain.
3. Finally, it sends a request to carry out the requested action.

Each URL has the following general syntax:

Structure:

`protocol://subdomain.domain.tld:port/path/?param=val#section`

What happens when you click on a URL?

Q: Internally, what does your browser do when you request for a URL?
A: Think of it like a world-wide file browser. When you request to open a file (click a URL):

1. The browser breaks down the request and understands it.
2. It then resolves the domain.
3. Finally, it sends a request to carry out the requested action.

Each URL has the following general syntax:

Structure:

`protocol://subdomain.domain.tld:port/path/?param=val#section`

Example:

`https://www.example.com:443/some/dir/?param=hi&lang=en#about`

What is HTML?

HTML (**H**yper**T**ext **M**arkup **L**anguage) contains an element-based grammar used to structure your webpage.

What is HTML?

HTML (**H**yper**T**ext **M**arkup **L**anguage) contains an element-based grammar used to structure your webpage.

Two important breakdowns of HTML:

1. **HyperText:** Documents link to one another (hence, *web*page).

What is HTML?

HTML (**H**yper**T**ext **M**arkup **L**anguage) contains an element-based grammar used to structure your webpage.

Two important breakdowns of HTML:

1. **HyperText:** Documents link to one another (hence, *web*page).
2. **Markup Language:** The document is declarative in nature.

What is HTML?

HTML (**H**yper**T**ext **M**arkup **L**anguage) contains an element-based grammar used to structure your webpage.

Two important breakdowns of HTML:

1. **HyperText:** Documents link to one another (hence, *web*page).
2. **Markup Language:** The document is declarative in nature.

We will discuss **HTML5**.

What is HTML?

HTML (**H**yper**T**ext **M**arkup **L**anguage) contains an element-based grammar used to structure your webpage.

Two important breakdowns of HTML:

1. **HyperText:** Documents link to one another (hence, *web*page).
2. **Markup Language:** The document is declarative in nature.

We will discuss **HTML5**. Fun facts:

1. HTML is not a case-sensitive language: `<html>` === `<HTML>`

What is HTML?

HTML (**H**yper**T**ext **M**arkup **L**anguage) contains an element-based grammar used to structure your webpage.

Two important breakdowns of HTML:

1. **HyperText:** Documents link to one another (hence, *web*page).
2. **Markup Language:** The document is declarative in nature.

We will discuss **HTML5**. Fun facts:

1. HTML is not a case-sensitive language: `<html>` === `<HTML>`
2. Tags can be nested. `<x>intro<y>main</y>end</x>`

What is HTML?

HTML (**H**yper**T**ext **M**arkup **L**anguage) contains an element-based grammar used to structure your webpage.

Two important breakdowns of HTML:

1. **HyperText:** Documents link to one another (hence, *web*page).
2. **Markup Language:** The document is declarative in nature.

We will discuss **HTML5**. Fun facts:

1. HTML is not a case-sensitive language: `<html>` === `<HTML>`
2. Tags can be nested. `<x>intro<y>main</y>end</x>`
   However, nesting can't be broken: `<x>intro<y>main</x>end</y>`

What is HTML?

HTML (**H**yper**T**ext **M**arkup **L**anguage) contains an element-based grammar used to structure your webpage.

Two important breakdowns of HTML:

1. **HyperText:** Documents link to one another (hence, *web*page).
2. **Markup Language:** The document is declarative in nature.

We will discuss **HTML5**. Fun facts:

1. HTML is not a case-sensitive language: `<html>` === `<HTML>`
2. Tags can be nested. `<x>intro<y>main</y>end</x>`
   However, nesting can't be broken: `<x>intro<y>main</x>end</y>`
3. HTML uses entities to escape characters interpreted by the grammar and some additionals: `&<charcode>;`. Ex: ` `, `&lt;`, `&gt;`.

What is HTML?

HTML (**H**yper**T**ext **M**arkup **L**anguage) contains an element-based grammar used to structure your webpage.

Two important breakdowns of HTML:

1. **HyperText:** Documents link to one another (hence, *web*page).
2. **Markup Language:** The document is declarative in nature.

We will discuss **HTML5**. Fun facts:

1. HTML is not a case-sensitive language: `<html>` === `<HTML>`
2. Tags can be nested. `<x>intro<y>main</y>end</x>`
   However, nesting can't be broken: `<x>intro<y>main</x>end</y>`
3. HTML uses entities to escape characters interpreted by the grammar and some additionals: `&<charcode>;`. Ex: ` `, `&lt;`, `&gt;`.
4. `<!-- This is an HTML comment -->`.

The Anatomy of an HTML Element

```
<h1 style="color:blue">Hello World!</h1>
```

The Anatomy of an HTML Element

```
<h1 style="color:blue">Hello World!</h1>
```

- **Opening Tag:** `<h1 style="color:blue">`

The Anatomy of an HTML Element

```
<h1 style="color:blue">Hello World!</h1>
```

- **Opening Tag:** `<h1 style="color:blue">`
- **Attributes:** `style="color:blue"`

The Anatomy of an HTML Element

```
<h1 style="color:blue">Hello World!</h1>
```

- **Opening Tag:** `<h1 style="color:blue">`
- **Attributes:** `style="color:blue"`
- **Inline CSS:** `color:blue`

The Anatomy of an HTML Element

```
<h1 style="color:blue">Hello World!</h1>
```

- **Opening Tag:** `<h1 style="color:blue">`
- **Attributes:** `style="color:blue"`
- **Inline CSS:** `color:blue`
- **Content:** `Hello World!`

The Anatomy of an HTML Element

```
<h1 style="color:blue">Hello World!</h1>
```

- **Opening Tag:** `<h1 style="color:blue">`
- **Attributes:** `style="color:blue"`
- **Inline CSS:** `color:blue`
- **Content:** `Hello World!`
- **Closing Tag:** `</h1>`

The Anatomy of an HTML Element

```
<h1 style="color:blue">Hello World!</h1>
```

- **Opening Tag:** `<h1 style="color:blue">`
- **Attributes:** `style="color:blue"`
- **Inline CSS:** `color:blue`
- **Content:** `Hello World!`
- **Closing Tag:** `</h1>`

```
<img src="url" alt="a descriptive description" download>
```

The Anatomy of an HTML Element

```
<h1 style="color:blue">Hello World!</h1>
```

- **Opening Tag:** `<h1 style="color:blue">`
- **Attributes:** `style="color:blue"`
- **Inline CSS:** `color:blue`
- **Content:** `Hello World!`
- **Closing Tag:** `</h1>`

```
<img src="url" alt="a descriptive description" download>
```

- **Boolean Attribute**: `download`.

The Anatomy of an HTML Element

```
<h1 style="color:blue">Hello World!</h1>
```

- **Opening Tag:** `<h1 style="color:blue">`
- **Attributes:** `style="color:blue"`
- **Inline CSS:** `color:blue`
- **Content:** `Hello World!`
- **Closing Tag:** `</h1>`

```
<img src="url" alt="a descriptive description" download>
```

- **Boolean Attribute**: `download`.
- **Self-Contained Tag**: No content or closing tag.

The Anatomy of an HTML Element

```
<h1 style="color:blue">Hello World!</h1>
```

- **Opening Tag:** `<h1 style="color:blue">`
- **Attributes:** `style="color:blue"`
- **Inline CSS:** `color:blue`
- **Content:** `Hello World!`
- **Closing Tag:** `</h1>`

```
<img src="url" alt="a descriptive description" download>
```

- **Boolean Attribute**: `download`.
- **Self-Contained Tag**: No content or closing tag.
- Multiple attributes use space separators.

The Document Object Model (DOM)

HTML, CSS and JavaScript are three very different languages that need to work together to render a functional webpage.

The Document Object Model (DOM)

HTML, CSS and JavaScript are three very different languages that need to work together to render a functional webpage.

The Document Object Model or **DOM** is an API that represents the structure of the webpage in memory and can be used to easily manipulate the contents of an HTML page.

The Document Object Model (DOM)

HTML, CSS and JavaScript are three very different languages that need to work together to render a functional webpage.

The Document Object Model or **DOM** is an API that represents the structure of the webpage in memory and can be used to easily manipulate the contents of an HTML page.

The browser manages the DOM, ensuring the sync between HTML and the object-oriented representation.

The Document Object Model (DOM)

HTML, CSS and JavaScript are three very different languages that need to work together to render a functional webpage.

The Document Object Model or **DOM** is an API that represents the structure of the webpage in memory and can be used to easily manipulate the contents of an HTML page.

The browser manages the DOM, ensuring the sync between HTML and the object-oriented representation.

Additionally, we use it extensively when performing browser automation and using tools like Selenium.

The Document Object Model (DOM)

HTML, CSS and JavaScript are three very different languages that need to work together to render a functional webpage.

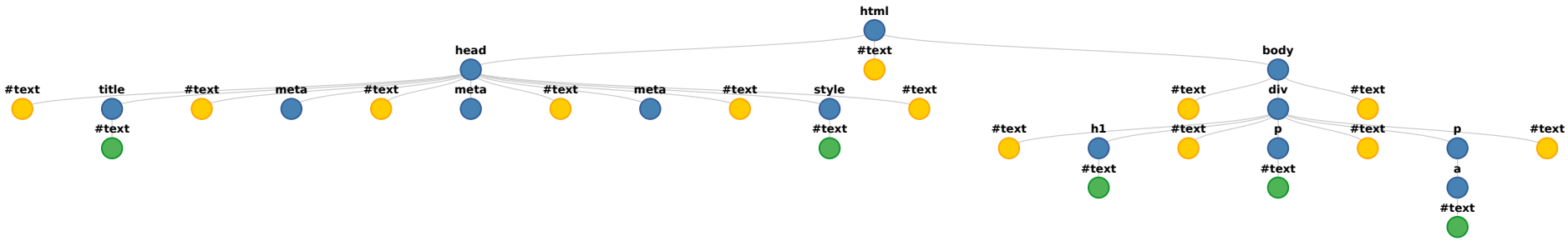
The Document Object Model or **DOM** is an API that represents the structure of the webpage in memory and can be used to easily manipulate the contents of an HTML page.

The browser manages the DOM, ensuring the sync between HTML and the object-oriented representation.

Additionally, we use it extensively when performing browser automation and using tools like Selenium. We'll talk more about DOM Traversal and Manipulation during the JavaScript module.

DOM (Contd.)

DOMs are often represented as **trees**. Below is the tree for `https://example.com/`:



HTML4 vs HTML5

What's new?

- **Semantic HTML:** HTML5 introduces new semantic tags: `<header>`, `<footer>`, `<aside>`, `<figure>`, `<figcaption>` for bots and screenreaders.

Further reading: `https://www.w3.org/TR/html5-diff/`

HTML4 vs HTML5

What's new?

- **Semantic HTML:** HTML5 introduces new semantic tags: `<header>`, `<footer>`, `<aside>`, `<figure>`, `<figcaption>` for bots and screenreaders.
- **Smarter Forms:** Forms in HTML5 have in-built input validation, placeholders, date pickers, and numeric sliders.

Further reading: `https://www.w3.org/TR/html5-diff/`

HTML4 vs HTML5

What's new?

- **Semantic HTML:** HTML5 introduces new semantic tags: `<header>`, `<footer>`, `<aside>`, `<figure>`, `<figcaption>` for bots and screenreaders.
- **Smarter Forms:** Forms in HTML5 have in-built input validation, placeholders, date pickers, and numeric sliders.
- **Better Storage:** Uses local storage instead of cookies.

Further reading: `https://www.w3.org/TR/html5-diff/`

HTML4 vs HTML5

What's new?

- **Semantic HTML:** HTML5 introduces new semantic tags: `<header>`, `<footer>`, `<aside>`, `<figure>`, `<figcaption>` for bots and screenreaders.
- **Smarter Forms:** Forms in HTML5 have in-built input validation, placeholders, date pickers, and numeric sliders.
- **Better Storage:** Uses local storage instead of cookies. Adds Application Cache, Web SQL DB and Web Storage.

Further reading: `https://www.w3.org/TR/html5-diff/`

HTML4 vs HTML5

What's new?

- **Semantic HTML:** HTML5 introduces new semantic tags: `<header>`, `<footer>`, `<aside>`, `<figure>`, `<figcaption>` for bots and screenreaders.
- **Smarter Forms:** Forms in HTML5 have in-built input validation, placeholders, date pickers, and numeric sliders.
- **Better Storage:** Uses local storage instead of cookies. Adds Application Cache, Web SQL DB and Web Storage.
- **In-built SVG:** Previously required browser plugins like Flash, but is now baked into HTML5.

Further reading: `https://www.w3.org/TR/html5-diff/`

HTML4 vs HTML5

What's new?

- **Semantic HTML:** HTML5 introduces new semantic tags: `<header>`, `<footer>`, `<aside>`, `<figure>`, `<figcaption>` for bots and screenreaders.
- **Smarter Forms:** Forms in HTML5 have in-built input validation, placeholders, date pickers, and numeric sliders.
- **Better Storage:** Uses local storage instead of cookies. Adds Application Cache, Web SQL DB and Web Storage.
- **In-built SVG:** Previously required browser plugins like Flash, but is now baked into HTML5.
- **Multimedia Support:** Integrates the open-source WebM format for plugin-free playback.

Further reading: `https://www.w3.org/TR/html5-diff/`

Outline

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container
`<a>`	Defines a hyperlink

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container
`<a>`	Defines a hyperlink
` `	Adds line breaks

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container
`<a>`	Defines a hyperlink
` `	Adds line breaks
`<p>`	Paragraph

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container
`<a>`	Defines a hyperlink
` `	Adds line breaks
`<p>`	Paragraph
`<pre>`	Pre-formatted Text

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container
`<a>`	Defines a hyperlink
` `	Adds line breaks
`<p>`	Paragraph
`<pre>`	Pre-formatted Text
`<script>`	Insert JavaScript

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container
`<a>`	Defines a hyperlink
` `	Adds line breaks
`<p>`	Paragraph
`<pre>`	Pre-formatted Text
`<script>`	Insert JavaScript
`<div>`	Defines a Section

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container
`<a>`	Defines a hyperlink
` `	Adds line breaks
`<p>`	Paragraph
`<pre>`	Pre-formatted Text
`<script>`	Insert JavaScript
`<div>`	Defines a Section
`<span>`	Defines an inline section

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container
`<a>`	Defines a hyperlink
` `	Adds line breaks
`<p>`	Paragraph
`<pre>`	Pre-formatted Text
`<script>`	Insert JavaScript
`<div>`	Defines a Section
`<span>`	Defines an inline section
`<strong>`	Emphasises a phrase

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container
`<a>`	Defines a hyperlink
` `	Adds line breaks
`<p>`	Paragraph
`<pre>`	Pre-formatted Text
`<script>`	Insert JavaScript
`<div>`	Defines a Section
`<span>`	Defines an inline section
`<strong>`	Emphasises a phrase
`<emph>`	Also emphasises a phrase

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container
`<a>`	Defines a hyperlink
` `	Adds line breaks
`<p>`	Paragraph
`<pre>`	Pre-formatted Text
`<script>`	Insert JavaScript
`<div>`	Defines a Section
`<span>`	Defines an inline section
`<strong>`	Emphasises a phrase
`<emph>`	Also emphasises a phrase
`<ul>`	Unordered List

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container
`<a>`	Defines a hyperlink
` `	Adds line breaks
`<p>`	Paragraph
`<pre>`	Pre-formatted Text
`<script>`	Insert JavaScript
`<div>`	Defines a Section
`<span>`	Defines an inline section
`<strong>`	Emphasises a phrase
`<emph>`	Also emphasises a phrase
`<ul>`	Unordered List
`<ol>`	Ordered List

HTML Tag Rundown

Tag	Use-case
`<head>`	Webpage Metadata Container
`<body>`	Rendered Content Container
`<a>`	Defines a hyperlink
` `	Adds line breaks
`<p>`	Paragraph
`<pre>`	Pre-formatted Text
`<script>`	Insert JavaScript
`<div>`	Defines a Section
`<span>`	Defines an inline section
`<strong>`	Emphasises a phrase
`<emph>`	Also emphasises a phrase
`<ul>`	Unordered List
`<ol>`	Ordered List
`<li>`	Single list element

HTML Attributes

HTML Attributes define the characteristics of a given element.

HTML Attributes

HTML Attributes define the characteristics of a given element. They each have default parameters that you can override.

HTML Attributes

HTML Attributes define the characteristics of a given element. They each have default parameters that you can override.

Elements can support any number of unique attributes.

HTML Attributes

HTML Attributes define the characteristics of a given element. They each have default parameters that you can override.

Elements can support any number of unique attributes. Attributes are space separated and a single attribute **cannot have spaces** within it.

HTML for Structural Heirarchy Management

Important: HTML's purpose is to manage the structure of the webpage.

HTML for Structural Heirarchy Management

Important: HTML's purpose is to manage the structure of the webpage. Mistakes to avoid / points to note:

- Use `<h1-6>` for heirarchy only, **not for size**.

HTML for Structural Heirarchy Management

Important: HTML's purpose is to manage the structure of the webpage. Mistakes to avoid / points to note:

- Use `<h1-6>` for heirarchy only, **not for size**.
- `<b>`, `<i>` are considered **deprecated**.

HTML for Structural Heirarchy Management

Important: HTML's purpose is to manage the structure of the webpage. Mistakes to avoid / points to note:

- Use `<h1-6>` for heirarchy only, **not for size**.
- `<b>`, `<i>` are considered **deprecated**. Instead, use `<strong>`, `<em>` as styling elements, with CSS updating the style.

HTML for Structural Heirarchy Management

Important: HTML's purpose is to manage the structure of the webpage. Mistakes to avoid / points to note:

- Use `<h1-6>` for heirarchy only, **not for size**.
- `<b>`, `<i>` are considered **deprecated**. Instead, use `<strong>`, `<em>` as styling elements, with CSS updating the style.
- Additional **now-unsupported** styling elements: `<basefont>`, `<big>`, `<center>`, `<font>`, `<tt>`.

Hyperlinks

A hyperlink is defined using the anchor (<a>) element, by setting the `href` attribute

Hyperlinks

A hyperlink is defined using the anchor (<a>) element, by setting the `href` attribute in one of the following ways:

- Fragments: `href="#section-id"`

Hyperlinks

A hyperlink is defined using the anchor (<a>) element, by setting the `href` attribute in one of the following ways:

- Fragments: `href="#section-id"`
- Relative: `href="path/to/file.html"` **or** `href="../../file.html"`

Hyperlinks

A hyperlink is defined using the anchor (<a>) element, by setting the `href` attribute in one of the following ways:

- Fragments: `href="#section-id"`
- Relative: `href="path/to/file.html"` **or** `href="../../file.html"`
- Absolute: `href="/path/to/file.html"`

Hyperlinks

A hyperlink is defined using the anchor (<a>) element, by setting the `href` attribute in one of the following ways:

- Fragments: `href="#section-id"`
- Relative: `href="path/to/file.html"` **or** `href="../../file.html"`
- Absolute: `href="/path/to/file.html"`
- Remote URL: `href="https://example.com/"`

Hyperlinks

A hyperlink is defined using the anchor (<a>) element, by setting the `href` attribute in one of the following ways:

- Fragments: `href="#section-id"`
- Relative: `href="path/to/file.html"` **or** `href="../../file.html"`
- Absolute: `href="/path/to/file.html"`
- Remote URL: `href="https://example.com/"`

Protip: To make a hyperlink spawn a new tab, use `target='_blank'`.

Document Validators

As we've seen before, the web hates throwing errors.

Document Validators

As we've seen before, the web hates throwing errors.

In cases with non-terminal errors, HTML partially renders the page without making the errors obvious.

Document Validators

As we've seen before, the web hates throwing errors.

In cases with non-terminal errors, HTML partially renders the page without making the errors obvious.

To identify errors, we can use validators by W3C:

- **HTML Validator:** `https://validator.w3.org/`
- **CSS Validator:** `https://jigsaw.w3.org/css-validator/`

Document Validators

As we've seen before, the web hates throwing errors.

In cases with non-terminal errors, HTML partially renders the page without making the errors obvious.

To identify errors, we can use validators by W3C:

- **HTML Validator:** `https://validator.w3.org/`
- **CSS Validator:** `https://jigsaw.w3.org/css-validator/`

Document validators are a great first diagnostic step for a broken page.

Let's Build a Webpage!

If you can view this screen, I am making a mistake.

Outline

Piazza

Our Piazza forum is up!
Please join: `https://piazza.com/purdue/fall2023/cs390`

Thank you!

Have an awesome rest of your day!

Slides: `https://cs.purdue.edu/homes/jsetpal/slides/html.pdf`

If anything's incorrect or unclear, please ping `jsetpal@purdue.edu`
I'll patch it ASAP.