CSS Fundamentals – Styling the Web

CS 390 – Web Application Development

J. Setpal

August 28, 2023

Outline

Outline

WIWYT – CSS

Recap from Monday:

- 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 is CSS?

Cascading **S**tyle **S**heets (CSS) is a **rule-based language**.

Reference: MDN

What is CSS?

Cascading **S**tyle **S**heets (CSS) is a **rule-based language**. Each rule is defined by specifying groups of styles that applied to particular elements or groups of elements on your webpage.

Reference: MDN

What is CSS?

Cascading **S**tyle **S**heets (CSS) is a **rule-based language**. Each rule is defined by specifying groups of styles that applied to particular elements or groups of elements on your webpage.

Three important concepts to keep in mind:

1. **Cascading:** The order of the rules matters for conflict resolution.

Reference: MDN

What is CSS?

Cascading **S**tyle **S**heets (CSS) is a **rule-based language**. Each rule is defined by specifying groups of styles that applied to particular elements or groups of elements on your webpage.

Three important concepts to keep in mind:

1. **Cascading:** The order of the rules matters for conflict resolution.
2. **Specificity:** Narrow Definitions $\propto$ Rule Importance.

Reference: MDN

What is CSS?

Cascading **S**tyle **S**heets (CSS) is a **rule-based language**. Each rule is defined by specifying groups of styles that applied to particular elements or groups of elements on your webpage.

Three important concepts to keep in mind:

1. **Cascading:** The order of the rules matters for conflict resolution.
2. **Specificity:** Narrow Definitions $\propto$ Rule Importance.
3. **Inheritance:** Some values are inherited from parents, some aren't.

Reference: MDN

What is CSS?

Cascading **S**tyle **S**heets (CSS) is a **rule-based language**. Each rule is defined by specifying groups of styles that applied to particular elements or groups of elements on your webpage.

Three important concepts to keep in mind:

1. **Cascading:** The order of the rules matters for conflict resolution.
2. **Specificity:** Narrow Definitions $\propto$ Rule Importance.
3. **Inheritance:** Some values are inherited from parents, some aren't. Both behaviors are overridable.

Reference: MDN

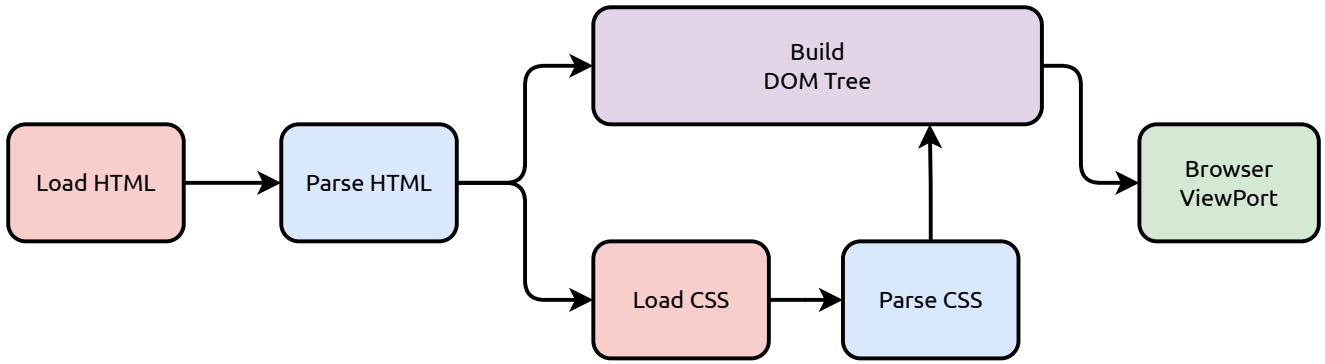
What is CSS?

Cascading **S**tyle **S**heets (CSS) is a **rule-based language**. Each rule is defined by specifying groups of styles that applied to particular elements or groups of elements on your webpage.

Three important concepts to keep in mind:

1. **Cascading:** The order of the rules matters for conflict resolution.
2. **Specificity:** Narrow Definitions $\propto$ Rule Importance.
3. **Inheritance:** Some values are inherited from parents, some aren't. Both behaviors are overridable.

CSS works by supplementing a DOM, following the below flowchart:



Reference: MDN

Syntax

CSS rules follow the below syntax:

```
target {
    key-1: value;
    key-2: value value2;
}
```

Syntax

CSS rules follow the below syntax:

```
target {
    key-1: value;
    key-2: value value2;
}
```

- **target:** The element / group of elements to update.

Syntax

CSS rules follow the below syntax:

```
target {
    key-1: value;
    key-2: value value2;
}
```

- **target:** The element / group of elements to update.
- **key-n:** The styling key whose value is to be updated.

Syntax

CSS rules follow the below syntax:

```
target {
    key-1: value;
    key-2: value value2;
}
```

- **target:** The element / group of elements to update.
- **key-n:** The styling key whose value is to be updated.
- **value:** The value(s) to set for the key.

Syntax

CSS rules follow the below syntax:

```
target {
    key-1: value;
    key-2: value value2;
}
```

- **target:** The element / group of elements to update.
- **key-n:** The styling key whose value is to be updated.
- **value:** The value(s) to set for the key. Multiple values are space-separated.

Syntax

CSS rules follow the below syntax:

```
target {
    key-1: value;
    key-2: value value2;
}
```

- **target:** The element / group of elements to update.
- **key-n:** The styling key whose value is to be updated.
- **value:** The value(s) to set for the key. Multiple values are space-separated.

Example: A rule that makes the body's font color blue looks like:

```
body {
    font-color: blue;
}
```

Syntax (contd.)

Comments: `/* Comments in CSS look like this */`

Syntax (contd.)

Comments: `/* Comments in CSS look like this */`

Useful Properties:

Key	Use-case
`display`	If and how an element is displayed

Syntax (contd.)

Comments: `/* Comments in CSS look like this */`

Useful Properties:

Key	Use-case
`display`	If and how an element is displayed
`position`	Configures the type of positioning

Syntax (contd.)

Comments: `/* Comments in CSS look like this */`
Useful Properties:

Key	Use-case
`display`	If and how an element is displayed
`position`	Configures the type of positioning
`width`	Width of an element

Syntax (contd.)

Comments: `/* Comments in CSS look like this */`
Useful Properties:

Key	Use-case
`display`	If and how an element is displayed
`position`	Configures the type of positioning
`width`	Width of an element
`height`	Height of an element

Syntax (contd.)

Comments: `/* Comments in CSS look like this */`
Useful Properties:

Key	Use-case
`display`	If and how an element is displayed
`position`	Configures the type of positioning
`width`	Width of an element
`height`	Height of an element
`margin`	All the margins of an element

Syntax (contd.)

Comments: `/* Comments in CSS look like this */`
Useful Properties:

Key	Use-case
`display`	If and how an element is displayed
`position`	Configures the type of positioning
`width`	Width of an element
`height`	Height of an element
`margin`	All the margins of an element
`padding`	Padding of an element

Syntax (contd.)

Comments: `/* Comments in CSS look like this */`
Useful Properties:

Key	Use-case
`display`	If and how an element is displayed
`position`	Configures the type of positioning
`width`	Width of an element
`height`	Height of an element
`margin`	All the margins of an element
`padding`	Padding of an element
`border`	Border of an element

Syntax (contd.)

Comments: `/* Comments in CSS look like this */`
Useful Properties:

Key	Use-case
`display`	If and how an element is displayed
`position`	Configures the type of positioning
`width`	Width of an element
`height`	Height of an element
`margin`	All the margins of an element
`padding`	Padding of an element
`border`	Border of an element
`background`	Configures all background look and behavior

Syntax (contd.)

Comments: `/* Comments in CSS look like this */`
Useful Properties:

Key	Use-case
`display`	If and how an element is displayed
`position`	Configures the type of positioning
`width`	Width of an element
`height`	Height of an element
`margin`	All the margins of an element
`padding`	Padding of an element
`border`	Border of an element
`background`	Configures all background look and behavior
`color`	Sets text color

Syntax (contd.)

Comments: `/* Comments in CSS look like this */`
Useful Properties:

Key	Use-case
`display`	If and how an element is displayed
`position`	Configures the type of positioning
`width`	Width of an element
`height`	Height of an element
`margin`	All the margins of an element
`padding`	Padding of an element
`border`	Border of an element
`background`	Configures all background look and behavior
`color`	Sets text color

These are just a small subset of important ones!
Extensive List: `https://www.w3schools.com/cssref/`

Inline, Internal & External CSS

There are three ways to integrate CSS with HTML:[1]

1. **Inline:** Directly adding CSS on a per-element basis.

[1] Ordered from bad to good.

Inline, Internal & External CSS

There are three ways to integrate CSS with HTML:[1]

1. **Inline:** Directly adding CSS on a per-element basis.
   Example: `<h1 style="color: blue">Hello World!</h1>`

[1] Ordered from bad to good.

Inline, Internal & External CSS

There are three ways to integrate CSS with HTML:[1]

1. **Inline:** Directly adding CSS on a per-element basis.
   Example: `<h1 style="color: blue">Hello World!</h1>`
2. **Internal:** Adding CSS to the `<head>` of an HTML file.

[1] Ordered from bad to good.

Inline, Internal & External CSS

There are three ways to integrate CSS with HTML:[1]

1. **Inline:** Directly adding CSS on a per-element basis.
   Example: `<h1 style="color: blue">Hello World!</h1>`
2. **Internal:** Adding CSS to the `<head>` of an HTML file.
   Example:

```
<head>
...
    <style type="text/css">
    h1 {
      color: blue;
    }
    </style>
</head>
...
    <h1>Hello World!</h1>
...
```

[1] Ordered from bad to good.

Inline, Internal & External CSS (contd.)

3. **External:** Severing the CSS to a new file.

Inline, Internal & External CSS (contd.)

3. **External:** Severing the CSS to a new file.
   Example:

style.css

```
h1 {
  color: blue;
}
```

index.html

```
<head>
...
   <link rel="stylesheet" \
       href="style.css">
</head>
...
   <h1>Hello World!</h1>
...
```

Rule Subsetting

Rules can be selected with varying levels of specificity:[2]

1. **Global:** Applies to everything.
   Syntax: `* { k: v; }`

[2] Ordered by least to most specific.

[3] `https://developer.mozilla.org/en-US/docs/Web/CSS/<element>#formal_definition`

Rule Subsetting

Rules can be selected with varying levels of specificity:[2]

1. **Global:** Applies to everything.
   Syntax: `* { k: v; }`
2. **Top-Level:** Applies to everything that can be inherited.[3]
   Syntax: `html { k: v; }`

[2] Ordered by least to most specific.

[3] `https://developer.mozilla.org/en-US/docs/Web/CSS/<element>#formal_definition`

Rule Subsetting

Rules can be selected with varying levels of specificity:[2]

1. **Global:** Applies to everything.
   Syntax: `* { k: v; }`
2. **Top-Level:** Applies to everything that can be inherited.[3]
   Syntax: `html { k: v; }`
3. **Tag-Level:** Applies to elements with a particular tag.
   Syntax: `<tagname> { k: v; }`

[2] Ordered by least to most specific.

[3] `https://developer.mozilla.org/en-US/docs/Web/CSS/<element>#formal_definition`

Rule Subsetting

Rules can be selected with varying levels of specificity:[2]

1. **Global:** Applies to everything.
   Syntax: `* { k: v; }`
2. **Top-Level:** Applies to everything that can be inherited.[3]
   Syntax: `html { k: v; }`
3. **Tag-Level:** Applies to elements with a particular tag.
   Syntax: `<tagname> { k: v; }`
4. **Class-Level:** Applies to elements with a particular class.
   Syntax: `.classname { k: v; }`

[2] Ordered by least to most specific.

[3] `https://developer.mozilla.org/en-US/docs/Web/CSS/<element>#formal_definition`

Rule Subsetting

Rules can be selected with varying levels of specificity:[2]

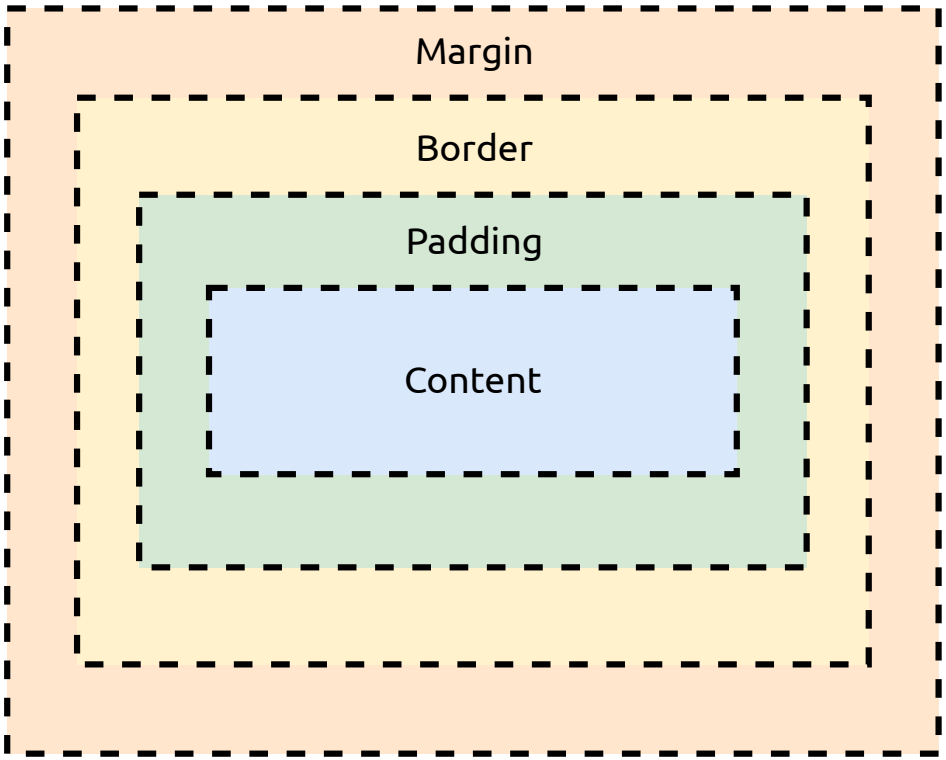
1. **Global:** Applies to everything.
   Syntax: `* { k: v; }`
2. **Top-Level:** Applies to everything that can be inherited.[3]
   Syntax: `html { k: v; }`
3. **Tag-Level:** Applies to elements with a particular tag.
   Syntax: `<tagname> { k: v; }`
4. **Class-Level:** Applies to elements with a particular class.
   Syntax: `.classname { k: v; }`
5. **ID-Level:** Applies to elements with a particular ID.
   Syntax: `#id { k: v; }`

[2] Ordered by least to most specific.

[3] `https://developer.mozilla.org/en-US/docs/Web/CSS/<element>#formal_definition`

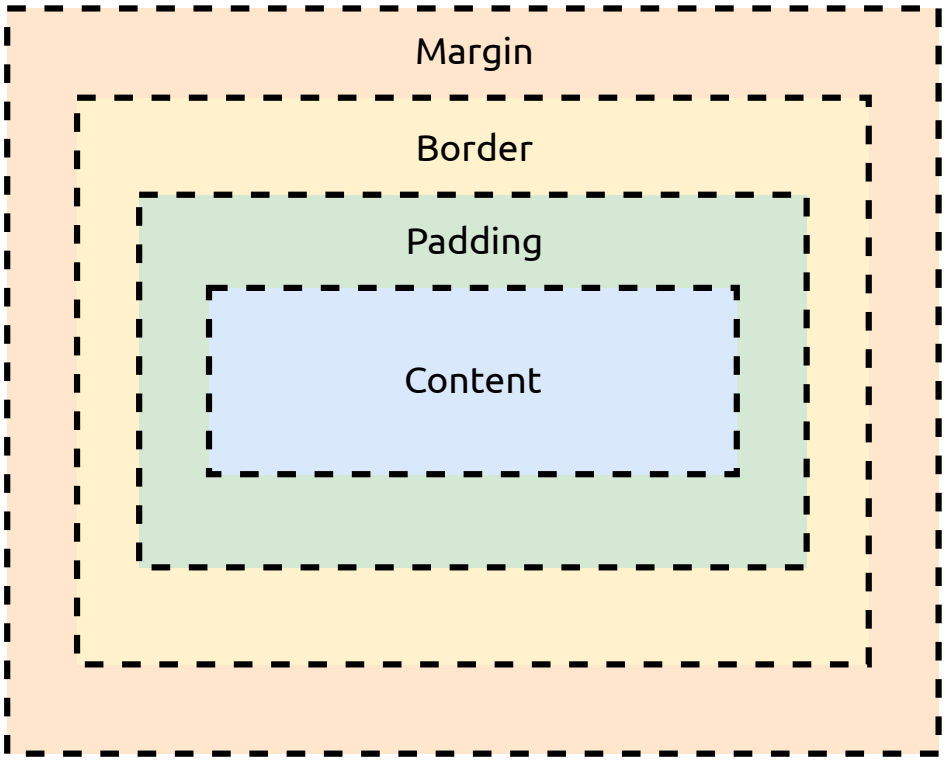
Box Model

The key idea here is that every element in CSS is a box.



Box Model

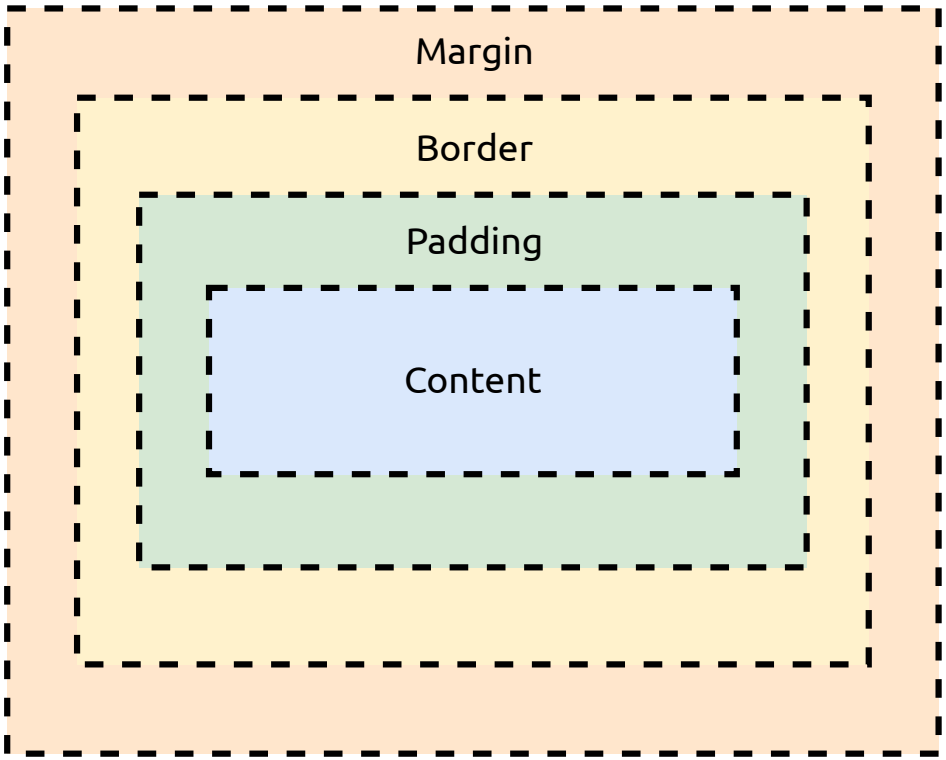
The key idea here is that every element in CSS is a box.



1. **Content:** Contains the value within the tag.

Box Model

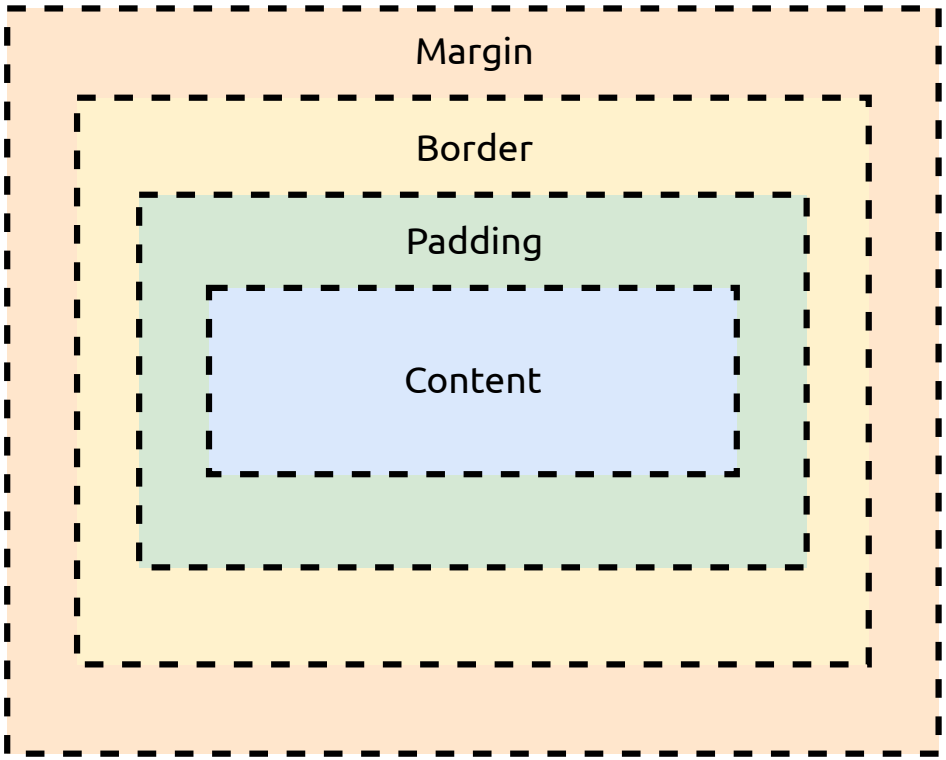
The key idea here is that every element in CSS is a box.



1. **Content:** Contains the value within the tag.
2. **Padding:** Adds whitespace between the content and the border.

Box Model

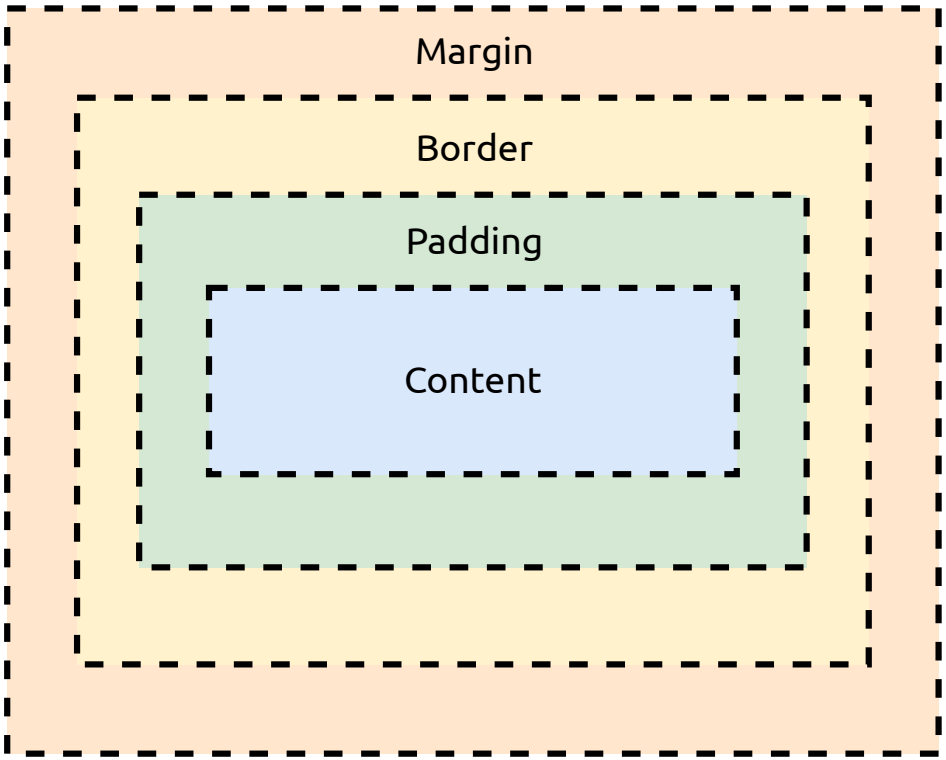
The key idea here is that every element in CSS is a box.



1. **Content:** Contains the value within the tag.
2. **Padding:** Adds whitespace between the content and the border.
3. **Border:** Separates padding from the margin.

Box Model

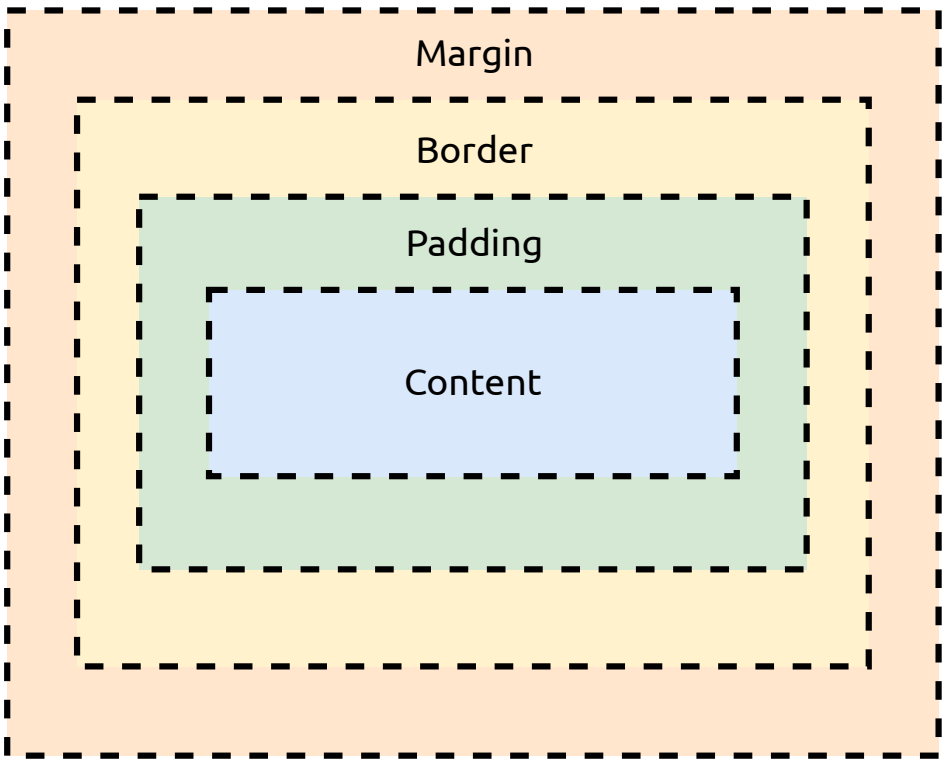
The key idea here is that every element in CSS is a box.



1. **Content:** Contains the value within the tag.
2. **Padding:** Adds whitespace between the content and the border.
3. **Border:** Separates padding from the margin.
4. **Margin:** Separates elements from one another.

Box Model

The key idea here is that every element in CSS is a box.



1. **Content:** Contains the value within the tag.
2. **Padding:** Adds whitespace between the content and the border.
3. **Border:** Separates padding from the margin.
4. **Margin:** Separates elements from one another.

Bonus Component

Outline: `border`, but floating; not part of the box model.

Sidenote: Pseudo-classes

In addition to default classes, there are also **Pseudo-classes**, that selects elements in a specifc state.

Syntax: `target:state { k: v; }`

Sidenote: Pseudo-classes

In addition to default classes, there are also **Pseudo-classes**, that selects elements in a specifc state.

Syntax: `target:state { k: v; }`

We'll cover this and pseudo-elements in more detail, for today's exercise you will need to know these targets:

1. `a:link` – unvisited link
2. `a:visited` – visited link

Units

There are two types of units: **absolute** and **relative**.

Units

There are two types of units: **absolute** and **relative**.

Absolute:

Unit	Definition	Length
`px`	# of pixels	1/96 inches

Units

There are two types of units: **absolute** and **relative**.

Absolute:

Unit	Definition	Length
`px`	# of pixels	1/96 inches
`pt`	# of points	1/72 inches

Units

There are two types of units: **absolute** and **relative**.

Absolute:

Unit	Definition	Length
`px`	# of pixels	1/96 inches
`pt`	# of points	1/72 inches
`in`	# of inches	1 inch

Units

There are two types of units: **absolute** and **relative**.

Absolute:

Unit	Definition	Length
`px`	# of pixels	1/96 inches
`pt`	# of points	1/72 inches
`in`	# of inches	1 inch

Relative:

Unit	Definition	Relative To
%	Percentage of	Parent

Units

There are two types of units: **absolute** and **relative**.

Absolute:

Unit	Definition	Length
`px`	# of pixels	1/96 inches
`pt`	# of points	1/72 inches
`in`	# of inches	1 inch

Relative:

Unit	Definition	Relative To
`%`	Percentage of	Parent
`em`	Ephemeral	Parent Font-Size

Units

There are two types of units: **absolute** and **relative**.

Absolute:

Unit	Definition	Length
`px`	# of pixels	1/96 inches
`pt`	# of points	1/72 inches
`in`	# of inches	1 inch

Relative:

Unit	Definition	Relative To
`%`	Percentage of	Parent
`em`	Ephemeral	Parent Font-Size
`rem`	Root Ephemeral	HTML Element's Font-Size

Units

There are two types of units: **absolute** and **relative**.

Absolute:

Unit	Definition	Length
`px`	# of pixels	1/96 inches
`pt`	# of points	1/72 inches
`in`	# of inches	1 inch

Relative:

Unit	Definition	Relative To
`%`	Percentage of	Parent
`em`	Ephemeral	Parent Font-Size
`rem`	Root Ephemeral	HTML Element's Font-Size
`vw`	view width	0.01 of the width of the viewport

Units

There are two types of units: **absolute** and **relative**.

Absolute:

Unit	Definition	Length
`px`	# of pixels	1/96 inches
`pt`	# of points	1/72 inches
`in`	# of inches	1 inch

Relative:

Unit	Definition	Relative To
`%`	Percentage of	Parent
`em`	Ephemeral	Parent Font-Size
`rem`	Root Ephemeral	HTML Element's Font-Size
`vw`	view width	0.01 of the width of the viewport
`vh`	view height	0.01 of the height of the viewport

CSS Reset & Normalization

Each browser has a predefined set of **CSS defaults**. As a consequence, webpages appear differently.

CSS Reset & Normalization

Each browser has a predefined set of **CSS defaults**. As a consequence, webpages appear differently.

We can fix this using:

1. **CSS Resets:** Applying global rules using the wildcard. **Example:**

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
 }
```

CSS Reset & Normalization

Each browser has a predefined set of **CSS defaults**. As a consequence, webpages appear differently.

We can fix this using:

1. **CSS Resets:** Applying global rules using the wildcard. **Example:**

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
 }
```

2. **Normalization Libraries:** A CSS rule preset you can import. We'll cover this in more detail later, one great example is normalize.css.

Fonts

Font styles can be setup by setting a series of fonts, that is checked in priority order.

Syntax: `target { font-family:  font-1, "Backup Font"}`

Fonts

Font styles can be setup by setting a series of fonts, that is checked in priority order.

Syntax: `target { font-family: font-1, "Backup Font"}`

We can also modify text decorations, such as the emphasis, weight, or if it is struck through.

Syntax: `target { text-decoration: underline blue}`

Let's ~~Build~~ Design a Webpage!

Objective: Re-create the style present on `example.com`. I'll be coming around for help, if needed.

Rules & Specifics:

1. **DO NOT VIEW OR USE THE ORIGINAL CSS.** Play fair.
2. Dialog box color: `#fdfdff`, background color: `#f0f0f2`.
3. To create a shadow, use `box-shadow` (look it up!).
4. Hyperlink color is always: `#38488f`.
5. Use 'Arial' with 'sans-serif' as a fallback.
6. Use External CSS.
7. The HTML code is in the attached tarball on Brightspace.
8. Submit just the `styles.css` file.

Let's ~~Build~~ Design a Webpage!

Objective: Re-create the style present on `example.com`. I'll be coming around for help, if needed.

Rules & Specifics:

1. **DO NOT VIEW OR USE THE ORIGINAL CSS.** Play fair.
2. Dialog box color: `#fdfdff`, background color: `#f0f0f2`.
3. To create a shadow, use `box-shadow` (look it up!).
4. Hyperlink color is always: `#38488f`.
5. Use 'Arial' with 'sans-serif' as a fallback.
6. Use External CSS.
7. The HTML code is in the attached tarball on Brightspace.
8. Submit just the `styles.css` file.

Winner (closest & fastest solution) gets a mini cheese pizza! (on Wednesday, after evaluation). This also doubles as the attendance survey.

Outline

Course Updates

- Module 1 was supposed to end today, but we're extending it into Wednesday.

Course Updates

- Module 1 was supposed to end today, but we're extending it into Wednesday.
- So, the homework is also delayed to Wednesday.

Course Updates

- Module 1 was supposed to end today, but we're extending it into Wednesday.
- So, the homework is also delayed to Wednesday.
- We'll have a **guided PSO on git tomorrow**. Optional and will be recorded.

Course Updates

- Module 1 was supposed to end today, but we're extending it into Wednesday.
- So, the homework is also delayed to Wednesday.
- We'll have a **guided PSO on git tomorrow**. Optional and will be recorded.
- Mikail will begin teaching JavaScript on Wednesday!

Thank you!

Have an awesome rest of your day!

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

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