FEWD

WELCOME TO FRONT-END WEB DEVELOPMENT

Please sit next to a different classmate
and write your name on your name tag.

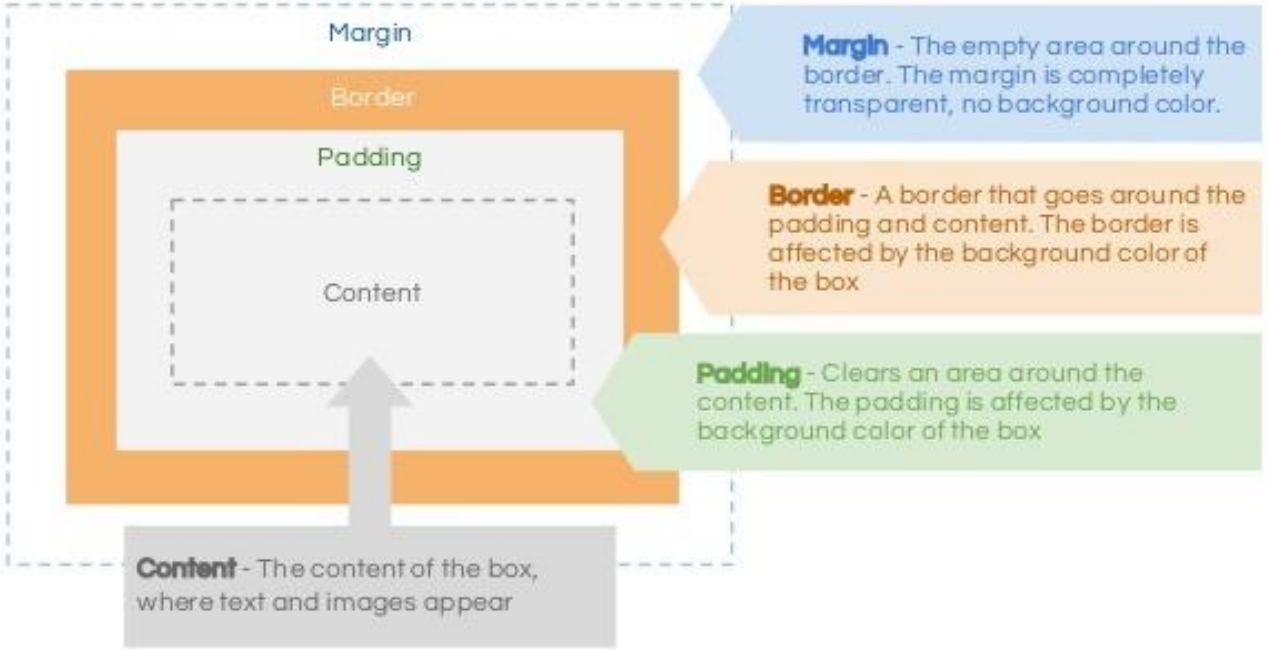
Wi-fi: GA-Guest
pw: yellowpencil

CSS

LESSON 05 & 06 ADVANCED CSS LAYOUTS RECAP

CSS Box Model

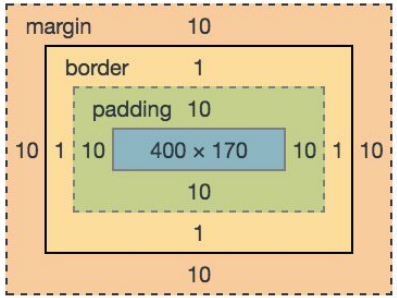
In an HTML document, each element is represented as a rectangular box, with the box's content, padding, border, and margin built up around one another **like the layers of an onion**.



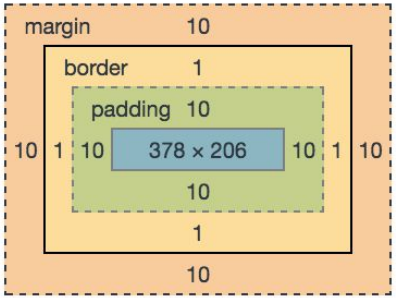
Understanding Box Sizing

```
div {
  width: 400px;
  padding: 10px;
  border: 1px solid black;
  margin: 10px;
}

.content-box {
  /* default box-sizing */
  box-sizing: content-box;
}

.border-box {
  box-sizing: border-box;
}
```

.content-box



.border-box



Codepen example

Applying border-box to all elements

Setting the layout model to `border-box` is very practical as it simplifies width and height calculations.

However, some third-party components or external framework elements might still be using the `content-box` layout model.

This ruleset ensures the best way to achieve using `border-box` across all modern browsers without breaking external components.

Read more in Paul Irish's [“Box-sizing border-box FTW”]

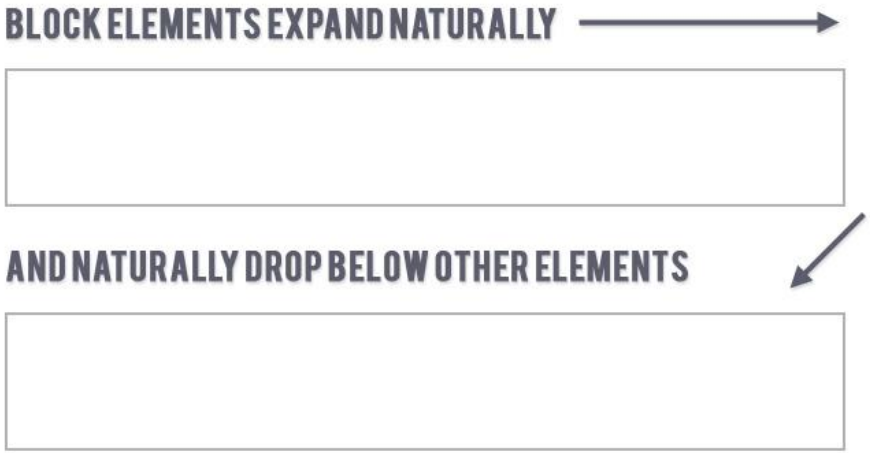
```
/* apply border-box to all elements */
html {
  box-sizing: border-box;
}

/* but allow components to change */
*, *:before, *:after {
  box-sizing: inherit;
}
```

Understanding block-level vs inline elements

Block-level elements:

- If no **width** is set, **will expand naturally** to fill its parent container
- If no **height** is set, **will expand naturally** to fit its child elements
- Can have margins and padding
- By default, will be placed below previous elements in the markup

Examples of block-level elements:
`<p>`, `<div>`, `<form>`, `<header>`, `<nav>`, `<ul>`, `<li>` and `<h1>`



Understanding block-level vs inline elements

Inline elements:

- Flows along with text content
- Will not clear previous content to drop to the next line like block elements
- Will ignore top and bottom margin settings, but will apply left and right margins, and any padding
- Will ignore the width and height properties

Examples of inline elements:
`<a>`, `<span>`, `<b>`, `<em>`, `<i>`, `<cite>`, `<mark>` and `<code>`

INLINE ELEMENTS FLOW WITH TEXT

PELLENTESQUE HABITANT MORBI TRISTIQUE SENECTUS ET NETUS ET MALESUADA FAMES AC TURPIS EGESTAS. VESTIBULUM INLINE ELEMENT VITAE, ULTRICIES EGET, TEMPOR SIT AMET, ANTE. DONEC EU LIBERO SIT AMET QUAM EGESTAS SEMPER. AENEAN ULTRICIES MI VITAE EST. MAURIS PLACERAT ELEIFEND LEO.

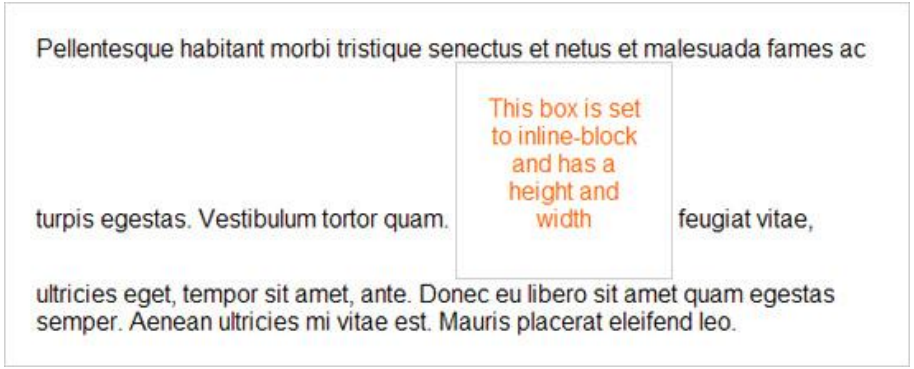
Introducing inline-block

Inline-block elements:

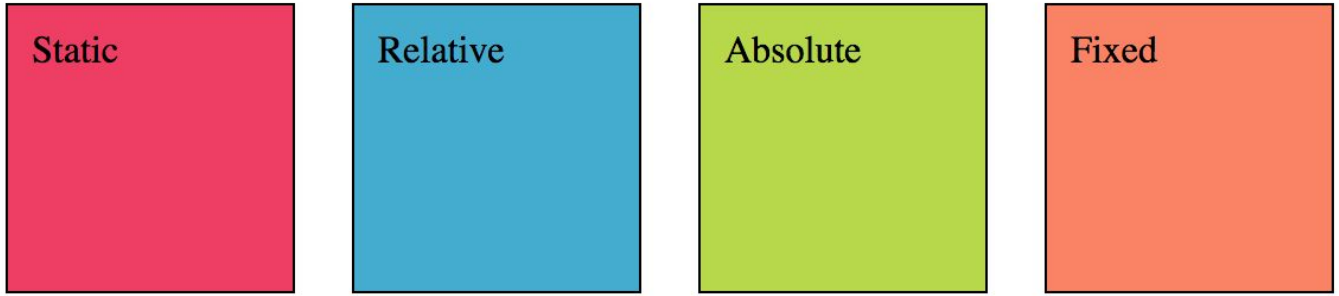
- Allow other elements to sit to their left and right
- Can have margins and padding
- Can have explicit height and width

Inline-block level elements are defined:

```
element {
  Display: inline-block;
}
```



Understanding Positioning



Further reading

Codepen example

Understanding Floating

Further reading

Codepen example

More advanced layout techniques

In recent years, more **advanced layout techniques have been introduced** and are now starting to be **well supported by modern browsers**.

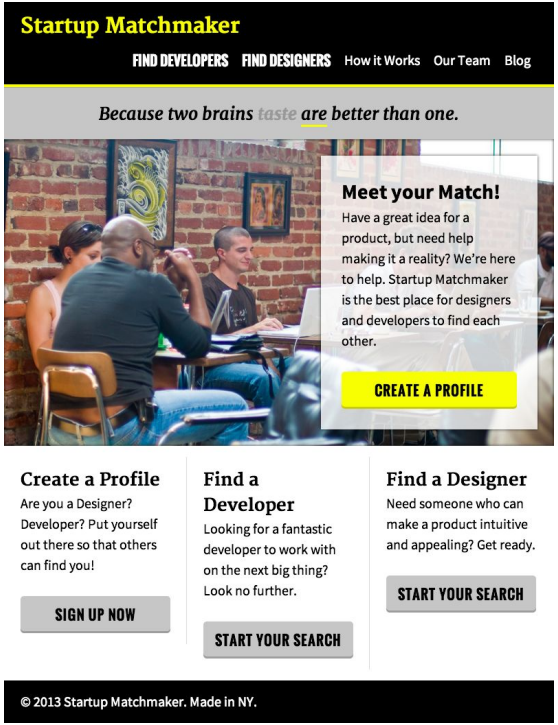
Though outside of the context of this course,
you're encouraged to look into **CSS Flexbox and Grid**.

This will improve your knowledge and allow you to create more advanced layouts.

CSS Flexbox

CSS Grid

Advanced layout exercise



LESSON 07
INTRO TO PROGRAMMING RECAP

Three basic layers of web-development

More important than learning the syntax and the particularities of a programming language is to **learn the basics of programming** and **understanding programmatic thinking***

* fancy way of saying “how computers think”

Programming basics

Computer programs are constructed using:

- **input data**: hardcoded in the program or provided by the user (e.g. *“Date of birth?”*)
- **output data**: normally the result of processed input data (e.g. *“You are over 21.”*)
- **variables**: markers that store values (e.g. `dateOfBirth`)
- **basic arithmetic**: addition, subtraction, multiplication, division (e.g. `currentDate - dateOfBirth`)
- **simple comparisons**: boolean logic (e.g. `userAge >= ageLimit`)
  - Equals =, Greater than >, Less than <, Greater than or equal to >=, Less than or equal to <=, Does not equal <>, And &&, Or ||, Not !
- **logical operations**
  - **if** *condition true* **then** *output* **else** *alternative output*
  - **while** *condition true* **do** *operation*

Programming in pseudo-code

Pseudo-code is:

- not a programming language
- a simple way of describing a set of instructions that solve a specific problem
- the perfect way of planning the development of a computer program
- varied in style: it can be very close to real programming language or look like prose

```
Put water into kettle;
Turn kettle on;
Wait for water to boil;
Put teabag in cup;
Add water to cup;
Remove teabag;
Serve;
```

```
IF X > Y
  PRINT "X is greater than Y"
ELSE
  PRINT "Y is greater than X"
END
```

Basic problem solving

Comparing two numbers

Take two numbers, X and Y. If X is greater than Y, print out X, otherwise print out B.

Solution

```
IF X > Y
  PRINT "X is greater than Y"
ELSE
  PRINT "Y is greater than X"
END
```

Basic problem solving

Double a number

Prompt user for a number and print out its double.

Solution

```
X = PROMPT "Enter a number"
Y = X * 2
PRINT The double of X is Y
```

Basic problem solving

Print a list of numbers

Print a list of numbers 1 through 15.

Solution

```
A = 1
WHILE (A < 16)
  PRINT A
  A + 1
ENDWHILE
```

Basic problem solving

Prompt user for a greater number

Present the user with a random number, X, and prompt user to enter a number, Y, greater than that. Print "Good job!" if the user gets it right or continue prompting until the user enters a correct response.

Solution

```
X = Random number
WHILE Y < X THEN
  Y = PROMPT "Enter a number greater than X"
  IF Y > X
    PRINT 'Good job!'
  ENDIF
ENDWHILE
```

LESSON 08
INTRO TO jQUERY