CSCB20
Introduction to Databases and Web Application

Week 6 - HTML and CSS

Dr. Purva R Gawde

Three tier architecture

- The Internet is comprised of three interacting features.
  - Websites: a collection of files and information that we access through a computer and a server
  - Servers: the computers that store all the data of websites in a massive network
  - Browsers: the programs that load and display content on your computer
- Every website is comprised of the frontend (client side) and the backend (server side)

HTML

- HTML (HyperText Markup Language) is the basic programming language for web development.
- It provides the basic structure of a site such as words, titles, and paragraphs.
- HTML consists of a bunch of established tags, which represent different functions that then “translate” into readable information on the screen.
- These tags are written between angle brackets.
- the text surrounded by the   will be bold.
- The tag <h6> </h6>will make your text the smallest header size.

CSS

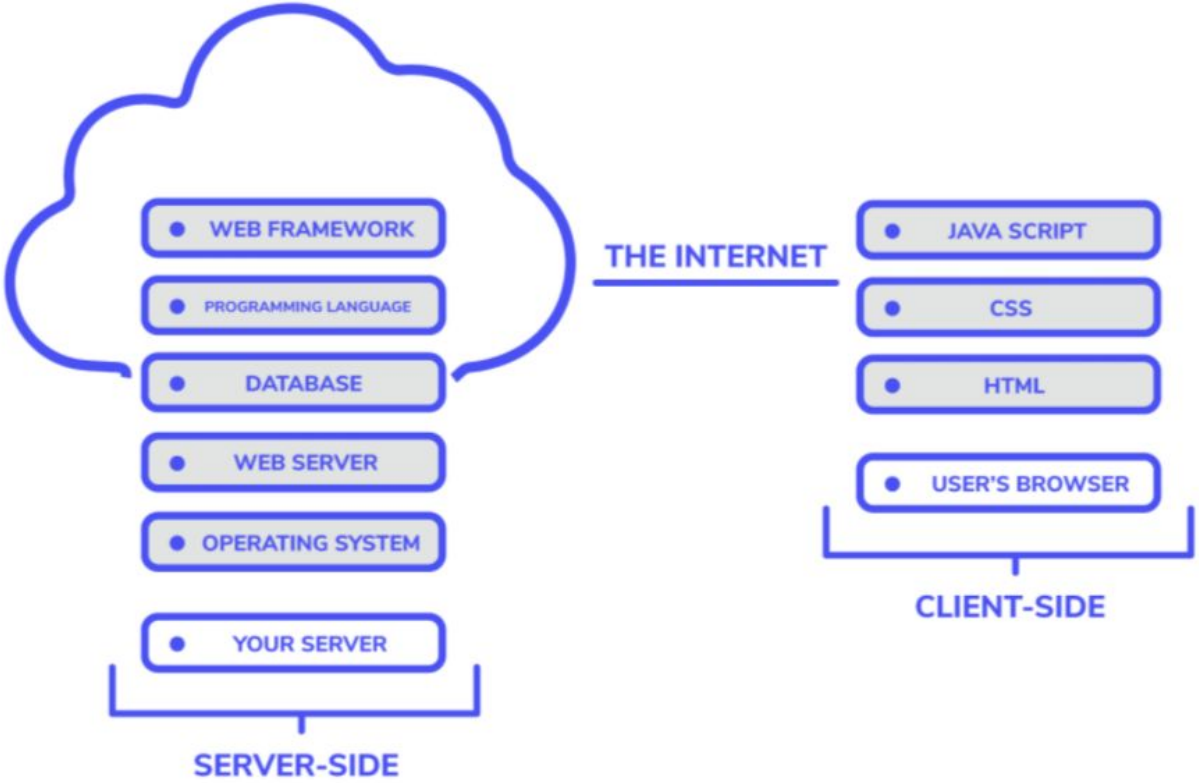
- CSS (Cascading Style Sheets) is a style sheet that essentially describes how HTML elements will appear on a webpage.
- You use CSS to control the presentation, style, and formatting of your site, like RGB values, border colors, background images, and more. CSS files declare a set of rules, that define a set of properties and their values.
- For example:
  - `<h4 style="color:Tomato;">Tomato Red</h4>`
  - determines that the text “Tomato Red” will appear in the color “Tomato”.

Javascript

- JavaScript is how you control the behavior of your webpage.
- JavaScript makes websites interactive by manipulating your various HTML and CSS features
- With JavaScript, a user can click on a button, scroll to the bottom of a page, or display photos in a moving carousel.
- Example:

```
<button type="button"
onclick="document.getElementById('demo').style.fontSize='35px'">
Click here
</button>
```

Technology Stack



Frameworks

- Frameworks:
  - prepackaged structure of pre-written code for your website that determines how programs should interact.

- Backend frameworks
  - Express JS: used by IBM and Uber;
  - Django: used by Google, Instagram; tons of built-in features; based on Python
  - Spring Boot: easy to use; good for large scale cloud projects; based on Java
  - Ruby on Rails: used by SoundCloud; good for small projects; based on Ruby
  - Flask: used by Lyft; easy to set up; based on Python

- Frontend frameworks
  - Ember: used by Netflix, LinkedIn; based in JavaScript
  - React: easy DOM manipulation; based in JavaScript
  - Backbone: lightweight; helps with code organization; based in JavaScript
  - Vue: progressive frameworks; easy to understand; based in JavaScript
  - Angular: good for Single-Page applications; not SEO friendly; based in TypeScript

Libraries

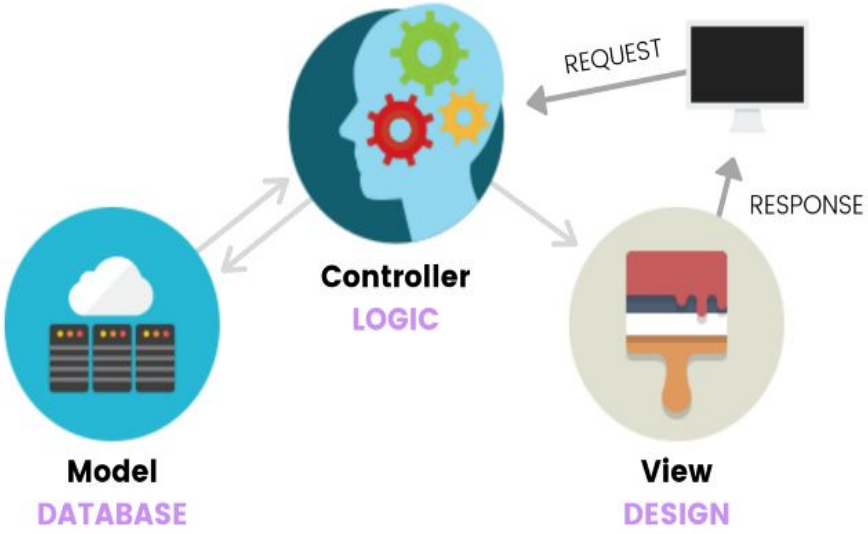
- What are libraries?
  - A library is a collection of specific tools and features that you can add to your website for functionality.
  - A library does not offer any structure but rather implements different behaviors and actions on your webpage.
    - jQuery: for manipulating HTML, DOM, and CSS
    - React.js: for creating interactive UIs
    - Chart.js: for making charts using JavaScript
    - Wow.js: for showing animations as you scroll
    - Scrolline.js: for showing how far you've scrolled on a page

Python Flask

- #1: What is Flask? Why Should You Care?
  - Flask is a lightweight and extensible Python web framework
- #2: Flask Structure
  - Instead of cramming all your code into one place, Flask helps you organize
    - (1) your logic, (2) design, and (3) database into separate files.

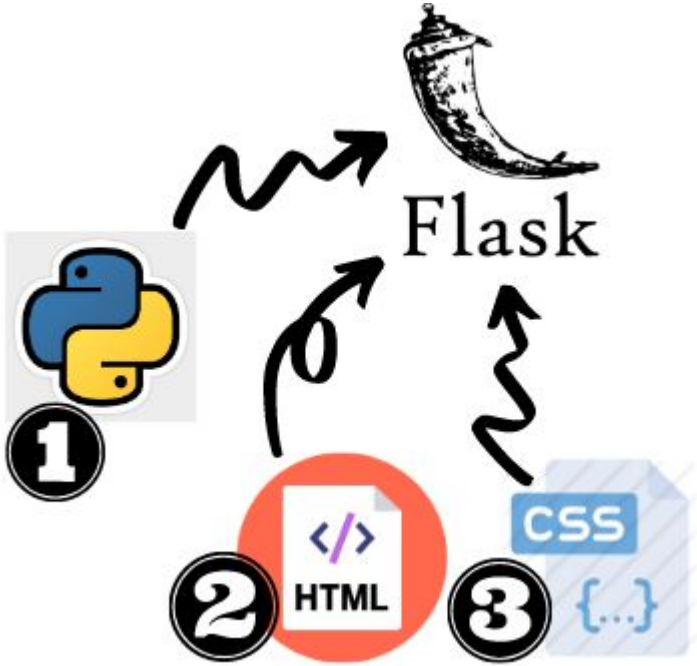
Python Flask Structure

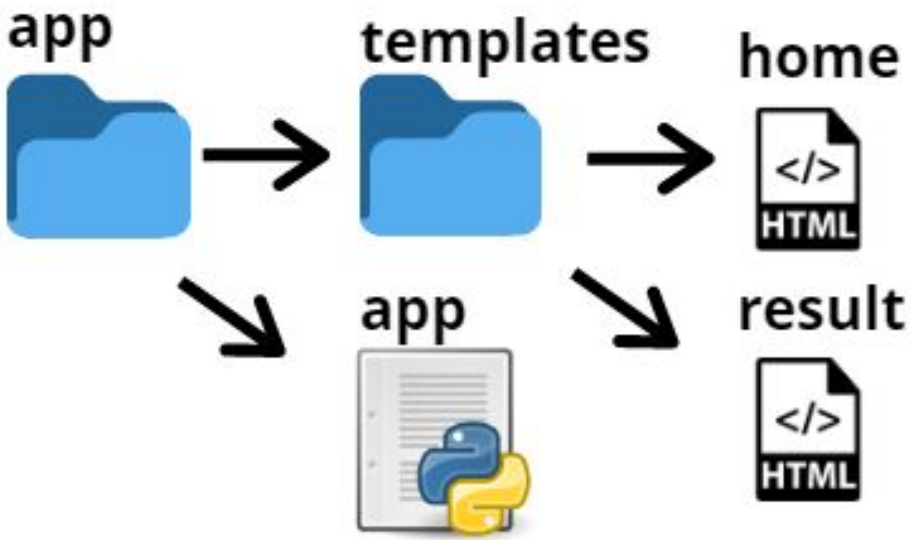
- Logic:
  - ‘main.py’ or ‘app.py’ imports the Flask module, creates a web server, creates an instance of the Flask class
- Design:
  - HTML files in templates
  - CSS in static
- Database:
  - SQLAlchemy supports a long list of database engines, including the SQLite (Grinberg).



Source: from Web Programming with Flask — Intro to Computer Science — Harvard’s CS50 (2018)

Flask + HTML + CSS



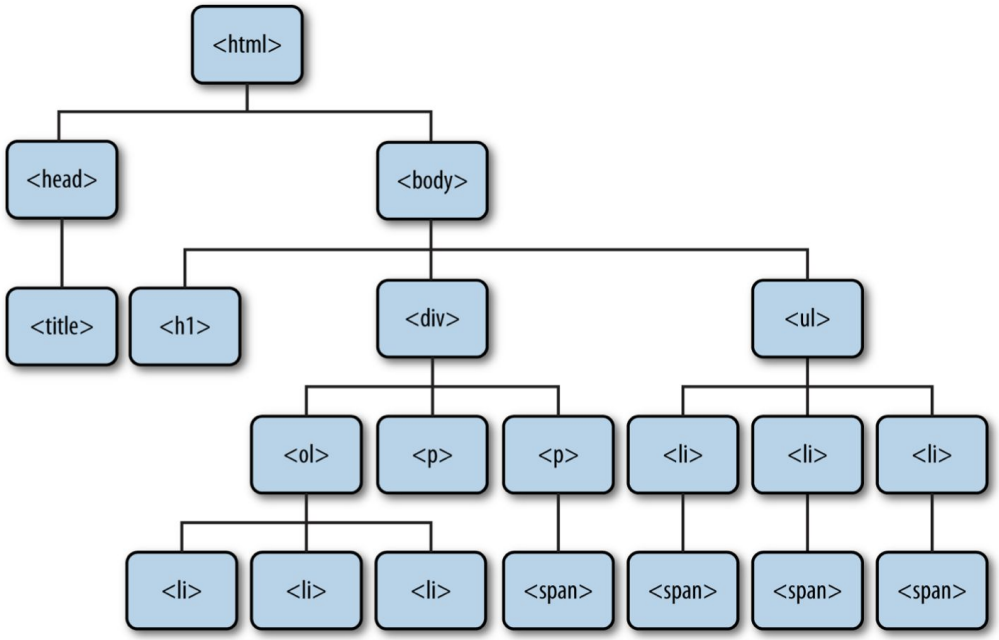


HTML DOM

Document Object Model

- HTML tags define a hierarchical structure called the Document Object Model, or DOM for short.

```
<!doctype html>
<html>
  <head>
    <title>Hello World!</title>
  </head>
  <body>
    <h1>Hello World!</h1>

    <div>
      <ol>
        <li>List Item</li>
        <li>List Item</li>
        <li>List Item</li>
      </ol>

      <p>This is a paragraph.</p>

      <p>This is a <span>second</span> paragraph.</p>

    </div>

    <ul>
      <li>List Item <span>1</span></li>
      <li>List Item <span>2</span></li>
      <li>List Item <span>3</span></li>
    </ul>
  </body>
</html>
```



HTML validation to Identify problems

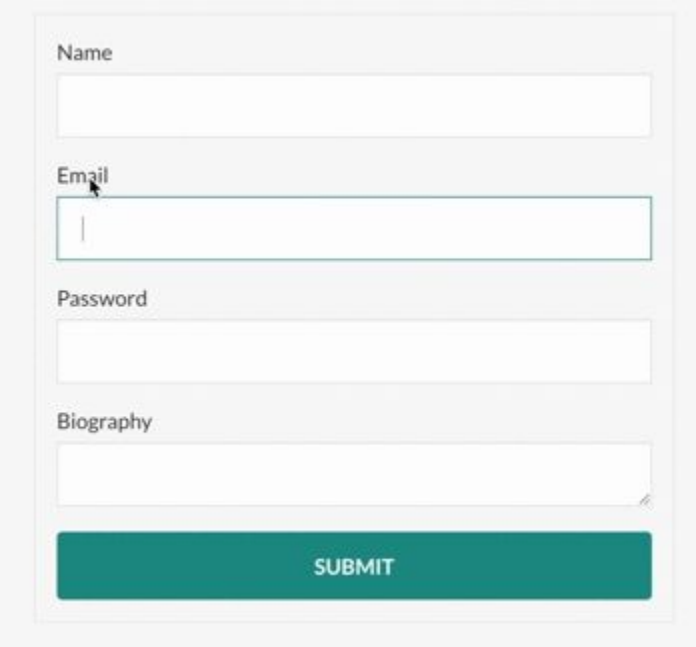
- Validation Program: automatically checks to see if your code conforms to certain basic standards
  - https://validator.w3.org/
- why do we care?
  - the only way we can guarantee that it will always look the same in every browser is if our HTML is correct

```html
<!doctype html>
<html>
  <head>
    <title>My First Web App</title>
  </head>

  <body>
    <h1>Hello, World!</h1>

    <nav>
      <div>Login</div>
      <div>FAQ</div>
      <div>About Us</div>
    </nav>

    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
       tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
       veniam, <span>quis nostrud exercitation</span> ullamco laboris nisi ut
       aliquip ex ea commodo consequat.</p>

    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
       tempor incididunt ut labore et dolore magna aliqua. Ut enim <span>ad
       minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip
       ex ea commodo consequat. Duis aute irure dolor in reprehenderit in
       voluptate <span>velit esse cillum dolore eu fugiat</span> nulla
       pariatur.</p>

    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
       tempor incididunt ut labore et dolore magna aliqua.</p>

  </body>
</html>
```

HTML forms

HTML Forms

- HTML forms are required when you want to collect some data from the person who visits your website.
- For example, when you register/login with information like Name, Email, and Password through HTML forms.



Source: https://miro.medium.com/max/512/1*7dd2MZ78ekF3bA3N0Jlvmw.gif

Form element

- Wraps all the other elements that go inside of our form.
- Attributes:
  - action: The action attribute is the web address (URL) of a program that processes the information submitted by our form.
  - method: It is the HTTP method that the browser uses to submit the form, the possible values are POST and GET.
  - POST — Used to send data to a server to create/update a resource
  - GET — Used to request data from a specified resource.

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Sign Up Form</title>
    </head>
    <body>
      <form action="index.html" method="post">
      </form>
    </body>
</html>
```

Source: https://miro.medium.com/max/512/1*7dd2MZ78ekF3bA3N0Jlvmw.gif

Input element

- Used to make a text field where the user can type some information for example email, password etc.
- attributes:
    - type: The type attribute indicates what kind of input we want.
    - id: The ID attribute is not mandatory, but it's a good idea to add one
    - name: The name attribute is necessary. When a form is submitted to the server side code, the server can understand the form data and process the values appropriately.

```html
<form action="index.html" method="post">
  <input type="text" id="name" name="student_name">
  <input type="email" id="mail" name="student_email">
  <input type="password" id="password" name="student_password">
</form>
```

Source: https://miro.medium.com/max/512/1*7dd2MZ78ekF3bA3N0Jlvmw.gif

Textarea element

- Sometimes a single line of text is not enough and a simple input element won't work.
- The <textarea> is not a self-closing tag
- Attributes:
  - id: Same as mentioned in <input/> element.
  - name: Same as mentioned in <input/> element.
  - cols: Specifies the visible width of a text area.
  - rows: Specifies the visible number of lines in a text area.

```
<form action="index.html" method="post">
    <textarea id="bio" name="student_bio"></textarea>
  </form>
```

Button element

- Without a button you cannot submit any form to the server for processing.
- Attributes:
  - type=”reset”: It will clear all the form data when it’s clicked.
  - type=”button”: Used with JavaScript to program it for custom behavior.
  - type=”submit”: Submit the form and send all the data over to the server.

```html
<form action="index.html" method="post">
    <input type="text" id="name" name="student_name">
    <input type="email" id="mail" name="student_email">
    <input type="password" id="password" name="student_password">

    <textarea id="bio" name="student_bio"></textarea>

    <button type="submit">Submit</button>
</form>
```

Label element

- We can label each one of our forms controls using the label element.
- The most used attribute with a label is for.
- Attributes:
  - for: The for attribute associates the label with a particular form element.
    - The way it matches is by ID.

```html
<form action="index.html" method="post">
    <label for="name">Name</label>
    <input type="text" id="name" name="student_name">
    <br>

    <label for="mail">Email</label>
    <input type="email" id="mail" name="student_email">
    <br>

    <label for="password">Password</label>
    <input type="password" id="password" name="student_password">
    <br>

    <label for="bio">Biography</label>
    <textarea id="bio" name="student_bio"></textarea>
    <br>

    <button type="submit">Submit</button>
</form>
```

CSS selectors

CSS Selectors

- The Element Selectors: select and apply styling to all elements with the same specified element name.

```
p {
  background-color: yellow;
}
```

- #id selector: apply styling to all specified elements with a selected id

```
#dog {
  background-color: purple;
}
```

- .class selector: applies styling to all elements with a specified class attribute

```
li.roomy {
  margin: 2em;
}
```

- Attribute selector: applies styling to all elements with a specified class attribute

```
input[type="submit"] {
  color: purple;
}
```

Descendant selector

- The descendant selector is the first of the CSS combination selectors family.
- This family lets you mix simple selectors with a specified logic.
- Applies styles to all elements that are descendants of a specified element.
- Selecting all <h1> elements nested inside elements looks like this.

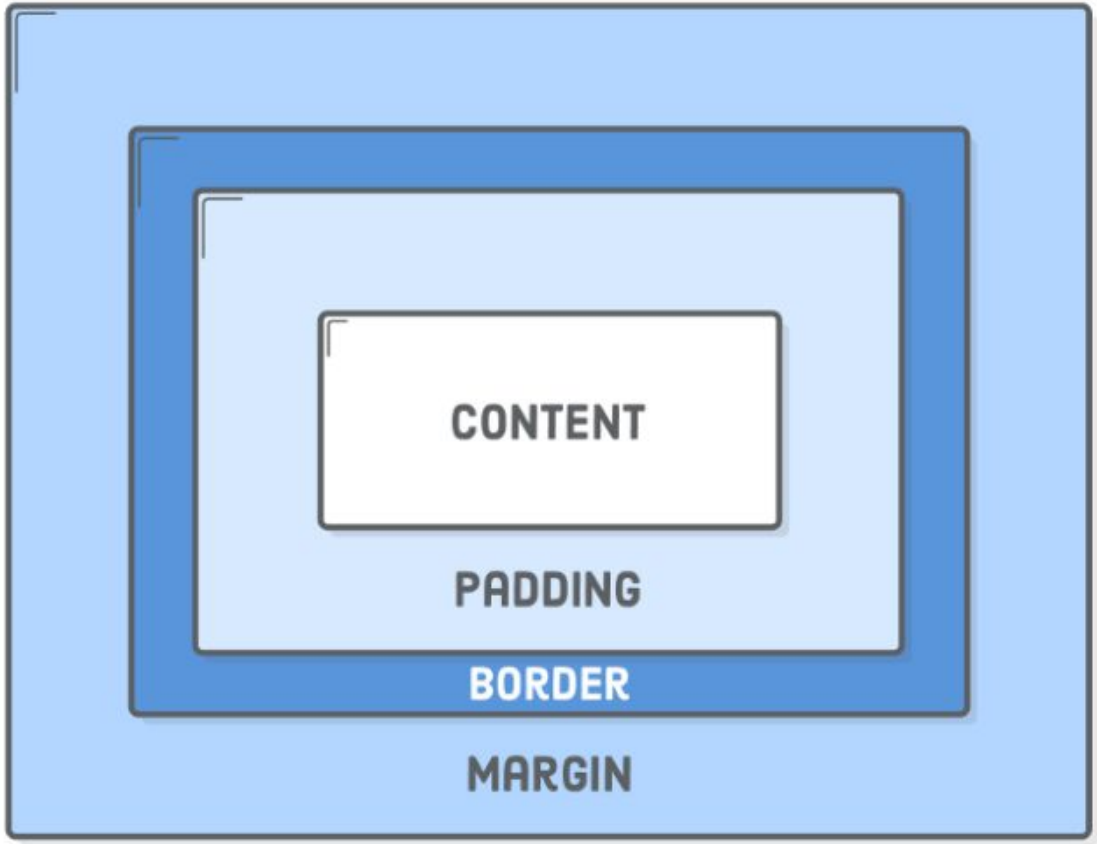
```
div h1 {
  background-color: purple;
}
```

Selector	Example	Example description
element element	div p	Selects all <p> elements inside elements
element>element	div > p	Selects all <p> elements where the parent is a element
element+element	div + p	Selects all <p> elements that are placed immediately after elements
element1~element2	p ~ ul	Selects every element that are preceded by a <p> element

CSS Box Model

The CSS Box Model

- The ‘CSS Box Model” is a set of rules that defines how every page on the internet is rendered.
- The Box Model is made up of 4 core element
- Content - The text, image, or other media content in the element.
- Padding- The space between the box’s content and its border.
- Borders - The line between the box’s padding and margin.
- Margins - The space between the box and surrounding boxes.



```
.container{
  display: flex;
}
```

CSS Flexbox Model (1 dimensional)

Flexbox Layout

- flexbox gives us complete control over the alignment, direction, order, and size of our boxes.
- two types of boxes :
  "flex containers" and "flex items".
- The job of a flex container is to group a bunch of flex items together and define how they're positioned.
- Flexbox layout is most appropriate to the components of an application, and small-scale layouts

ALIGNMENT

DIRECTION

ORDER

SIZE

"FLEX CONTAINER"

"FLEX ITEMS"

Properties for the Parent (flex container)

Properties

- Display
- flex-direction
- Flex-wrap
- flex-flow

- justify-content
- align-items
- align-content

"FLEX CONTAINER"

"FLEX ITEMS"

- justify-content
  - center
  - flex-start
  - flex-end

FLEX-START

CENTER

FLEX-END

Properties for the Children (flex items)

Properties

- Order
- flex-grow
- flex-shrink
- flex-basis
- **flex**
- align-self

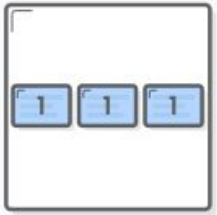
"FLEX CONTAINER"

"FLEX ITEMS"

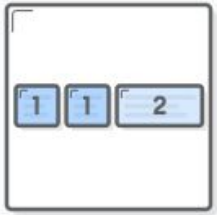
- **flex**

defines the width of individual items in a flex container

NO FLEX



EQUAL FLEX



UNEQUAL FLEX

CSS Flexbox Model
(2 dimensional)

```
.container{
  display: grid;
}
```

Grid Layout

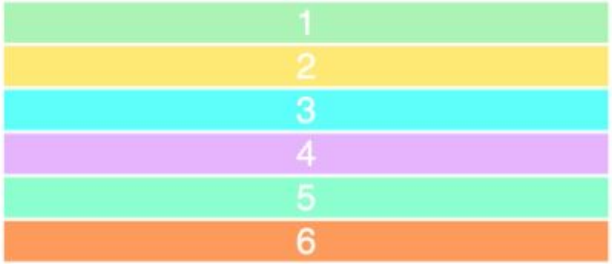
- The two core ingredients of a CSS Grid are the wrapper (parent) and the items (children).
- The wrapper is the actual grid and the items are the content inside the grid.

HTML

```html
<div class="wrapper">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>
```

CSS

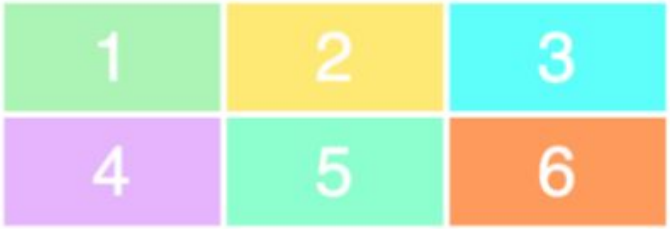
```css
.wrapper {
    display: grid;
}
```



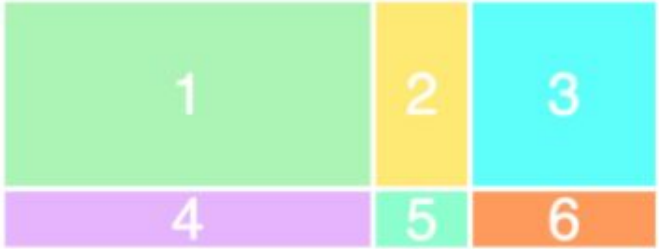
Columns and rows

- Two-dimensional using grid-template-row and grid-template-column properties.

```
.wrapper {
    display: grid;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows: 50px 50px;
}
```



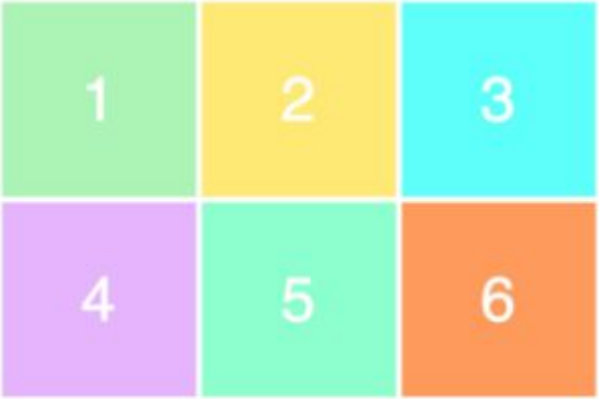

```
.wrapper {
    display: grid;
    grid-template-columns: 200px 50px 100px;
    grid-template-rows: 100px 30px;
}
```



Placing the items

- Placing the items on the grid.

```
.wrapper {
    display: grid;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows: 100px 100px 100px;
}
```

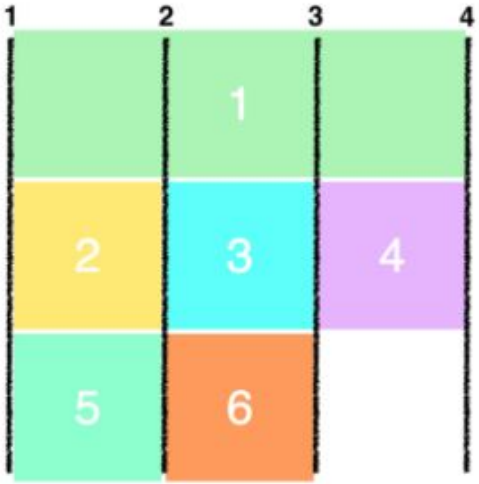


- Notice, we only see a 3x2 grid on the page, while we defined it as a 3x3 grid. That's because we only have six items to fill the grid with.

Placing the items

- Placing the items on the grid.

```
.item1 {
    grid-column-start: 1;
    grid-column-end: 4;
}
```




```
.item1 {
    grid-column: 1 / 4;
}
```