FAKULTAS
ILMU
KOMPUTER

Web Design
Using HTML5 and CSS3

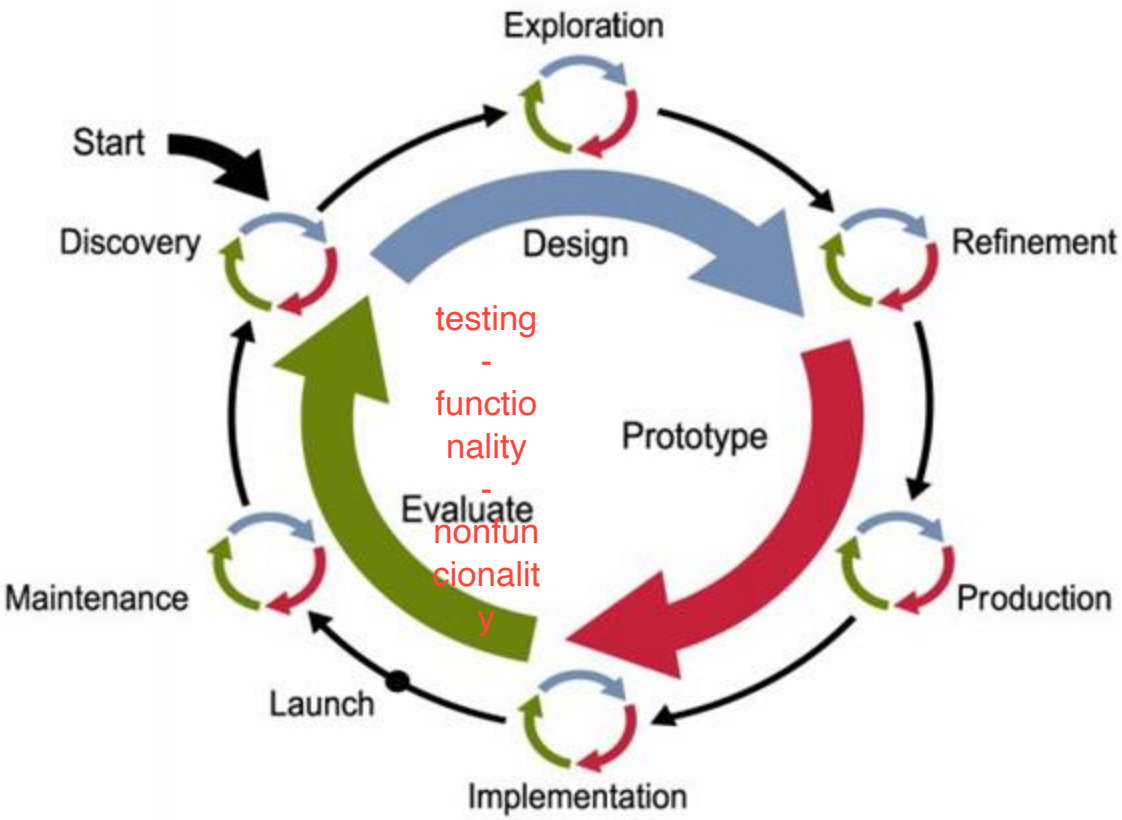
Tim Dosen PBP

Outline

- Web Development Cycle
- Interface Design
- HTML5
- CSS3
- Responsive Web Design
- Django Static Files (CSS, JavaScript, Images, etc)

Web Development Cycle

Web Development Cycle



Design Phase

Discovery

↓

Design Exploration

↓

Design Refinement

↓

Production

Value Preposition (VP) Questions

1. What are the purposes/visions of this website?
2. What problems this website will solve?
3. Whom will this website help?
4. How will this website help them?

Interface Design

The 8 Golden Rules of Interface Design

1. Strive for consistency
2. Seek universal usability
3. Offer informative feedback
4. Design dialogs to yield closure
5. Prevent errors
6. Permit easy reversal of actions
7. Keep users in control
8. Reduce short-term memory load

Source: https://www.cs.umd.edu/users/ben/goldenrules.html

Shneiderman's 8 Golden Rules of Interface Design

The principles	Questions to consider	Mark Complete
1. Strive for consistency	Is the style of this element maintained across your site/app? Is this content placed in the correct location according to the site hierarchy? Does this follow the conventions for your chosen platform? How can you make your designs more consistent?	☐
2. Enable frequent users to use shortcuts	Are there shortcuts available for your more experienced users? Who is this product designed for? Will there be a need to consider experienced users? How can you make it easier and quicker for experienced users?	☐
3. Offer informative feedback	Does the user know where they are at in the process? Does the user know what they have done after performing this action? How are you communicating this feedback to your user?	☐
4. Design dialogue to yield closure	Does the user have to do any guessing here? Is it clear and obvious enough for your intended audience? Are there any next steps for the user? How are you communicating the system status with the user?	☐
5. Offer simple error handling	Have you done everything imaginable to prevent this error from happening on your end? Is this error avoidable in the first place? If the user does make an error, how easy is it for them to fix it?	☐
6. Permit easy reversal of actions	How many steps does the user have to take to reverse their actions? Will the user quickly realize they need to reverse the action in the first place? How can you make your users detect the possibility of reversal?	☐
7. Support internal locus of control	Will the user feel in control at this specific touch point in your app? Will they be surprised in an unpleasant manner? Does the site feel easily navigable? Does the user feel safe and in control? How can you make the user feel more safe and in control??	☐
8. Reduce short-term memory load	Are there enough visual cues here for the user to find the functionality or item? Do they have to remember things to understand what's going on? How can you help the user recall?	☐

Source: https://public-media.interaction-design.org/pdf/Shneiderman.s.Eight.Golden.Rules.Worksheet.pdf

The Psychology of Color

RED

POSITIVE	NEGATIVE
Power	Anger
Passion	Danger
Energy	Warning
Fearlessness	Defiance
Strength	Aggression
Excitement	Pain

ORANGE

POSITIVE	NEGATIVE
Courage	Deprivation
Confidence	Frustration
Warmth	Frivolity
Innovation	Immaturity
Friendliness	Ignorance
Energy	Sluggishness

YELLOW

POSITIVE	NEGATIVE
Optimism	Irrationality
Warmth	Fear
Happiness	Caution
Creativity	Anxiety
Intellect	Frustration
Extraversion	Cowardice

GREEN

POSITIVE	NEGATIVE
Health	Boredom
Hope	Stagnation
Freshness	Envy
Nature	Blandness
Growth	Enervation
Prosperity	Sickness

TURQUOISE

POSITIVE	NEGATIVE
Communication	Boastfulness
Clarity	Secrecy
Calmness	Unreliability
Inspiration	Reticence
Self-expression	Fence-sitting
Healing	Aloofness

BLUE

POSITIVE	NEGATIVE
Trust	Coldness
Loyalty	Aloofness
Dependability	Emotionless
Logic	Unfriendliness
Serenity	Uncaring
Security	Unappetizing

PURPLE

POSITIVE	NEGATIVE
Wisdom	Introversion
Luxury	Decadence
Wealth	Suppression
Spirituality	Inferiority
Imaginative	Extravagance
Sophistication	Moodiness

MAGENTA

POSITIVE	NEGATIVE
Imaginative	Outrageousness
Passion	Nonconformity
Transformation	Flippancy
Creative	Impulsiveness
Innovation	Eccentricity
Balance	Ephemeralness

BROWN

POSITIVE	NEGATIVE
Seriousness	Humorlessness
Warmth	Heaviness
Earthiness	Unsophisticated
Reliability	Sadness
Support	Dirtiness
Authenticity	Conservativeness

BLACK

POSITIVE	NEGATIVE
Sophistication	Oppression
Security	Coldness
Power	Menace
Elegance	Heaviness
Authority	Evil
Substance	Mourning

GRAY

POSITIVE	NEGATIVE
Timelessness	Unconfident
Neutrality	Dampness
Reliability	Depression
Balance	Hibernation
Intelligence	Lack of energy
Strength	Blandness

WHITE

POSITIVE	NEGATIVE
Cleanness	Sterility
Clarity	Coldness
Purity	Unfriendliness
Simplicity	Elitism
Sophistication	Isolation
Freshness	Emptiness

Source: https://drive.google.com/file/d/1pzuKoz6Qx5Pedf24wPkkbmoO_lDHiObq/view

HTML5

Apakah HTML (Hypertext Markup Language) adalah bahasa pemrograman?

Evolution of HTML

1993	HTML 1.0 - Developed by Tim Berners Lee to link the document
1995	HTML 2.0 - Developed by Internet Engineering Task Force RFC to include stylized text and tables
1996	CSS1
1997	HTML 3.2 - Developed by W3C and included browser specific feature
1997	HTML 4.0 - A move back to normalizing the pages across platforms
1998	CSS2
1999	HTML 4.01 - Introduced different document types
2012	HTML 5 - Back to HTML plus multimedia and semantic tags

HTML Page Example


```html
<!DOCTYPE html>
<html>
    <head>
        <title>Judul Halaman</title>
    </head>
    <body>
        <h1>Judul Artikel</h1>
        <p>Paragraf pertama.</p>
    </body>
</html>
```

HTML Elements

```
<title>Page Title</title>
```

start tag

element content

end tag

empty element

```
<img src="img_girl.jpg"/>
```

Self closing tag

HTML Attributes

- Attributes provide additional information about elements. Attributes are always specified in the start tag.
- Attributes usually come in name/value pairs like: name="value"
- Example:

```
<img src="img_girl.jpg" width="500" height="600"/>
```

HTML and Browser

	MAC				WIN								
	FIREFOX	OPERA	CHROME	SAFARI	FIREFOX	OPERA	CHROME	SAFARI	IE				
	11	11.62	18	5.1	11	11.61	18	5.1	6	7	8	9	
Canvas	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	90%
Canvas Text	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	89%
SVG	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	88%
SVG Clipping Paths	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	88%
SVG Inline	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	54%
SMIL	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	69%
WebGL	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	61%
Audio	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	88%
Video	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	88%

HTML5

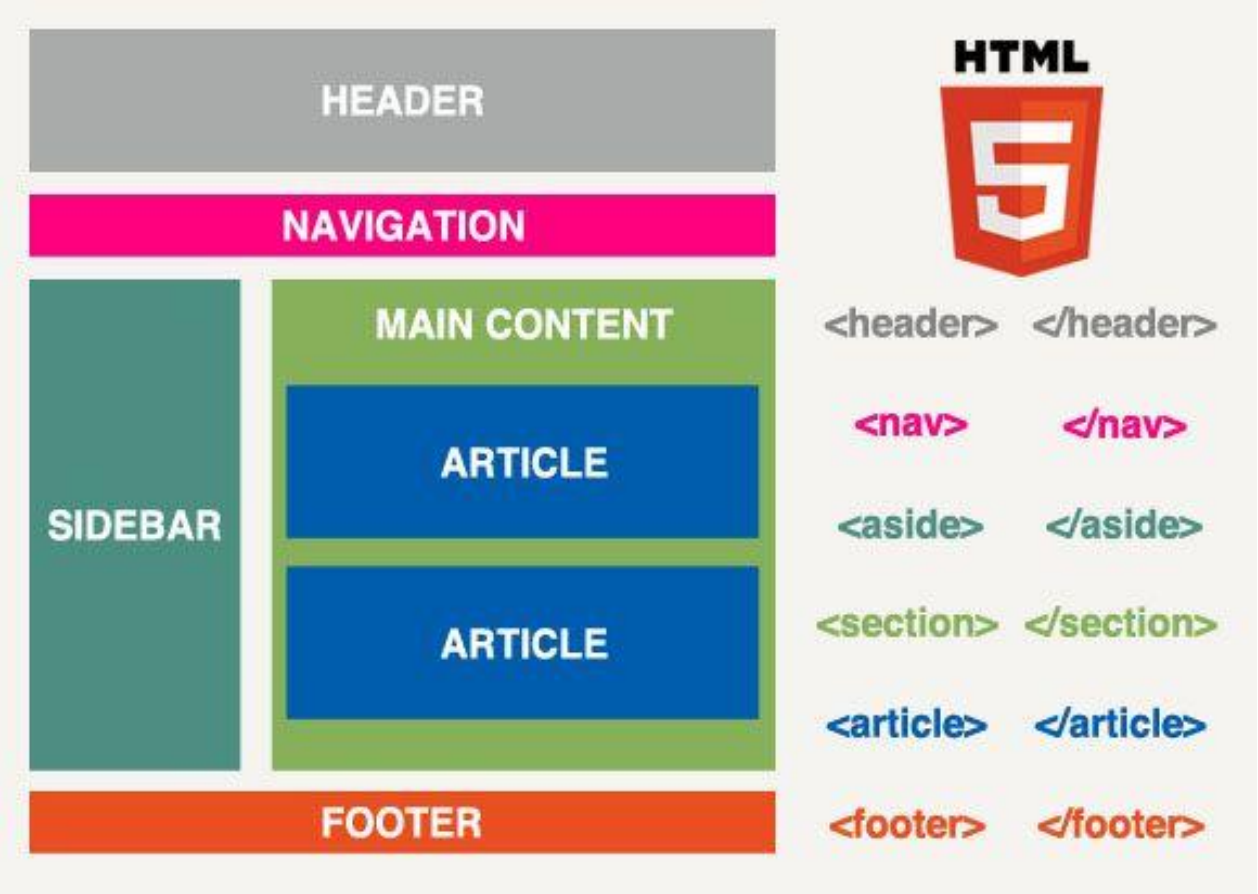
HTML5 is the latest specification of the HTML language, with objectives primarily include:

- Encouraging semantic (meaningful) markup
- Separating design from content
- Promoting accessibility and design responsiveness
- Reducing the overlap between HTML, CSS, and JavaScript
- Supporting rich media experiences while eliminating the need for plugins such as Flash or Java

Read more:
https://html.com/html5/#ixzz77q499mnd

- New semantic elements like <nav>, <header>, <footer>, <article>, <section>
- New attributes of form elements like datalist, keygen, output
- New input types: datetime, number, email, month, url, color, etc.
- New input attributes: required, placeholder, autofocus
- New graphic elements: <svg> and <canvas>
- New multimedia elements: <audio> and <video>

HTML5


HEADER
NAVIGATION
SIDEBAR
MAIN CONTENT
ARTICLE
ARTICLE
FOOTER
HTML
<header> </header>
<nav> </nav>
<aside> </aside>
<section> </section>
<article> </article>
<footer> </footer>

HTML + CSS

HTML + CSS

HTML

CSS3

tampilan

CSS (Cascading Style Sheets)

- CSS is a language that describes the style of an HTML document.
- CSS describes how selected HTML elements should be displayed.

CSS Syntax


```
h1 {
        color:blue;
        font-size:12px;
}
```

How to apply CSS

- Inline style: inline tag of html
- Internal style sheet: inside html
- External style sheet: separated file

Tips:

For better maintenance, use:
external style sheet

CSS Selectors

- The HTML element can be selected by:
  - Element Selector (without leading # or .)
  - Class Selector (with leading .)
  - ID Selector (with leading #)

selector.css

```
h1 {
    color:blue;
    text-align:center;
}

h2, h3 {
    color:red;
    text-align:center;
}

p {
    background-color:pink;
}

.utama {
    background-color:yellow;
}

#p1 {
    background-color:aqua;
}
```

css_selector.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Judul Halaman</title>
        <link href="selector.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <h1>Elemen h1</h1>
        <h2>Elemen h2</h2>
        <h3>Elemen h3</h3>
        <p id="p1" class="utama">Paragraf satu.</p>
        <p id="p2">Paragraf dua.</p>
        <p id="p3" class="utama">Paragraf tiga.</p>
    </body>
</html>
```

CSS3 Flexbox

- CSS3 is the latest standard in CSS and it is backward compatible
- Flexbox is a (new) layout mode in CSS3
  - https://www.w3schools.com/css/css3_flexbox.asp
  - https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Try:

- https://www.w3schools.com/css/tryit.asp?filename=trycss3_column-count
- https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_wrap-reverse

Where is the “Cascading” part?

If there are two rules still in conflict at a given weight, the algorithm will continue to “cascade down” and check attribute priorities until it finds one that wins.

Priorities (highest order wins)

1. Origin & Importance
2. Selector Specificity
3. Order of Appearance
4. Initial & Inherited Properties (default values)

1. Origin & importance

- **Origin**
  1. **Author**: CSS comes from front-end developer.
  2. **User**: The user of browser can manually set the style for their browser, such as its font style and color.
  3. **User-Agent**: The browser serves default style. Different browser provides different style
- The importance of a CSS declaration is determined by the `!important` syntax.

  \+ use `!important` as an escape hatch for when all else fails (such as when working with 3rd-party styles)

  \- can make your CSS more brittle

2. Selector Specificity (Selectors)

- CSS selectors can belong to one of the following order. The highest order wins.
    1. Inline styles (anything inside a style tag)
    2. ID selectors
    3. Classes selector
    4. Element selector

Selector Specificity (Selectors)

```
...
<h1 class="class1" style="color: red;" id="title1">HTML5 Example Page</h1>
...
```

```
style.css
h1 {
    color: blue;
}
#title1 {
    color: aqua;
}
.class1 {
    color: cadetblue;
}
```

Apa warna tulisan
“HTML5 Example Page”?

3. Source Order

If you’ve got 2 stylesheets linked in the head of your HTML document, the second stylesheet will override rules in the first stylesheet.

First CSS loaded

```
<html>
  <head>
    <link href="file.css"
     rel="stylesheet"
     type="text/css">

    <style type=”text/css”>
      table {
              color: red;
      }
    </style>
  </head>
</html>
```

Second CSS loaded. If there are same element selectors named “table”, then the Second CSS will override the style.

4. Initial & inherited properties

- In the following example, the <p> tag will render with a monospace font & red text, since its parent node contains those styles.
- The tag will render with yellow background, inherited from style from its parent (the style for <body> tag)
- The “inherit” property will automatically use the value from its parent. When the parent change its style, their child inherit all changed style.
- For non-inherited properties, each element has a set of initial values.

```
<html>
  <head>
    <style type=text/css>
      div {
        background-color: initial;
        color: inherit;
      }
      body {
        background-color: yellow;
      }
    </style>
  </head>
  <body>
    <div style="font-family: monospace; color: red;">
      <p>inheritance can be super useful!</p>
    </div>
  </body>
</html>
```

Framework to Help You Design Better

Prettification: Using a CSS Framework

- Design is hard, and doubly so now that we have to deal with mobile, tablets, and so forth.
- That's why many programmers are turning to CSS frameworks to solve some of those problems for them.
- There are lots of frameworks out there, but one of the earliest and most popular is Twitter's Bootstrap.

Responsive Design using Bootstrap

- Framework for building responsive, mobile-first sites, with the BootstrapCDN and a template starter page.
- Bootstrap also provides several animated and interactive elements built by JavaScript.

Learning Source:

- https://getbootstrap.com/docs/5.3/getting-started/introduction/

How to use Bootstrap?

1. Copy the Bootstrap source link to your HTML code or download Bootstrap and add the Bootstrap folder to your CSS Folder.
2. Find the Bootstrap style on the documentation.
3. Put the Bootstrap class to your code.

Example: 1. Copy the Bootstrap source link to your HTML code

Include Bootstrap's CSS and JS. Place the `<link>` tag in the `<head>` for our CSS, and the `<script>` tag for our JavaScript bundle (including Popper for positioning dropdowns, poppers, and tooltips) before the closing `</body>`. Learn more about our CDN links.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" re
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/js/bootstrap.bundle.min.
  </body>
</html>
```

Example: 2. Find the Bootstrap style on the documentation

Buttons

View on GitHub

Use Bootstrap's custom button styles for actions in forms, dialogs, and more with support for multiple sizes, states, and more.

Examples

Bootstrap includes several predefined button styles, each serving its own semantic purpose, with a few extras thrown in for more control.

Primary Secondary Success Danger Warning Info Light Dark Link

HTML

```
<button type="button" class="btn btn-primary">Primary</button>
<button type="button" class="btn btn-secondary">Secondary</button>
<button type="button" class="btn btn-success">Success</button>
```

Example: 3. Put the Bootstrap class to your code

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1Bx" crossorigin="anonymous">
  </head>
  <body>
    <button>
                                        Button (without Bootstrap)
                        </button>
    <button type="button" class="btn btn-primary">
                                        Button (with Bootstrap)
                        </button>
  </body>
</html>
```

Example

Carousel

View on GitHub

A slideshow component for cycling through elements—images or slides of text—like a carousel.

How it works

The carousel is a slideshow for cycling through a series of content, built with CSS 3D transforms and a bit of JavaScript. It works with a series of images, text, or custom markup. It also includes support for previous/next controls and indicators.

In browsers where the Page Visibility API is supported, the carousel will avoid sliding when the webpage is not visible to the user (such as when the browser tab is inactive, the browser window is minimized, etc.).

> The animation effect of this component is dependent on the `prefers-reduced-motion` media query. See the reduced motion section of our accessibility documentation.

Please be aware that nested carousels are not supported, and carousels are generally not compliant with accessibility

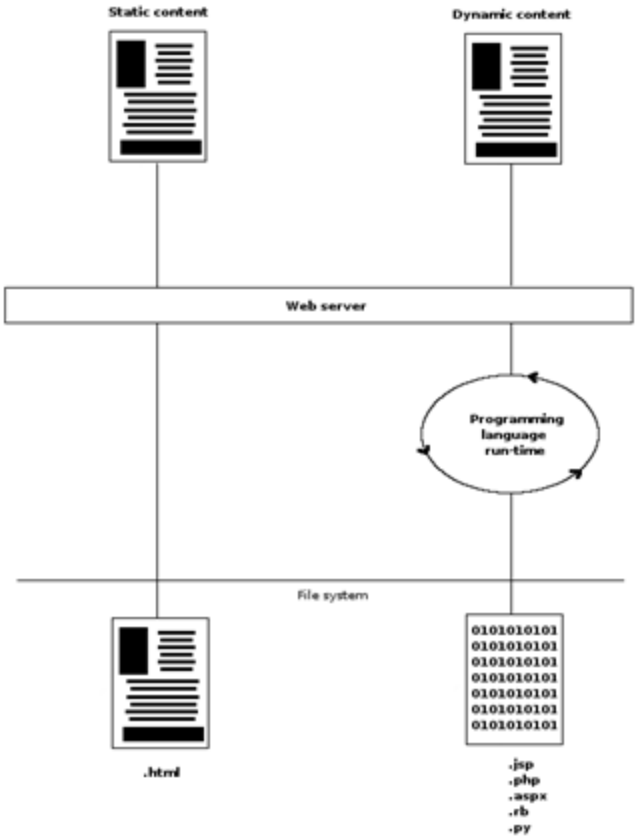
Django Static Files

Managing Static Files

- Websites generally need to serve additional files such as images, JavaScript, or CSS. Django refers to these files as “static files”.
- Django provides **`django.contrib.staticfiles`** to help you manage them.

Static files are handled differently

- Actually it is the dynamic files (or program files) which are handled specifically.
- They are differentiated to improve web server performance and also security.



Benefits of Static Files

- **They are static**: These files don't change until the developer replace them with a new one. Thus, the server just fetches them from the disk, taking a minimum amount of time.
- **Static files are easier to cache**: They don't change and are not modified by the server. That makes the performance faster.
- **Static files are energy efficient**: Static files are fetched from the disk when required. They require no processing which saves the processing overhead and website response becomes fast.

Static Files in Django

Django, and indeed any web server, needs to know two things to deal with static files:

- How to tell when a URL request is for a static file, as opposed to for some HTML that's going to be served via a view function
- Where to find the static file the user wants

Static Files in Django

- In other words, static files are a mapping from URLs to files on disk. Static files should not be part of your repository.
- Django lets us define a URL "prefix" to say that any URLs which start with that prefix should be treated as requests for static files. By default, the prefix is /static/. It was defined in settings.py.

Configuring Static Files

1. Make sure that **`django.contrib.staticfiles`** is included in your **`INSTALLED_APPS`**.
2. In your settings file, define **`STATIC_URL`**, for example:
   ```
   STATIC_URL = '/static/'
   ```
3. In your templates, use the **`static`** template tag to build the URL for the given relative path using the configured **`STATICFILES_STORAGE`**.
   ```
   {% load static %}
   <img src="{% static 'my_app/example.jpg' %}" alt="My image">
   ```

4. Store your static files in a folder called **`static`** in your app. For example **`my_app/static/my_app/example.jpg`**. Or you can simply command **`collectstatic`**

References

- Django Documentation: https://docs.djangoproject.com/en/5.0/
- CSS3 Flexbox:
  - https://css-tricks.com/snippets/css/a-guide-to-flexbox/
  - https://www.w3schools.com/css/css3_flexbox.asp
  - https://www.w3schools.com/css/tryit.asp?filename=trycss3_column-count
  - https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_wrap-reverse
- Bootstrap: https://getbootstrap.com/docs/5.3/getting-started/introduction/
- https://html.com/html5/#ixzz77q499mnd