Who Website Works?

A website works by following a sequence of steps when you enter its address in your browser:

- **Step 1: Address Lookup**
  - You type a website's URL (like www.example.com) into your browser.
  - The browser asks a **DNS (Domain Name System) server** to convert this human-readable address into the numerical IP address of the server where the website resides.
- **Step 2: Sending the Request**
  - Your browser uses the IP address to contact the web server that hosts the website.
  - The browser sends an **HTTP** request to the server, asking for the website's files (HTML, CSS, JavaScript, images, etc.).
- **Step 3: Server Response**
  - The web server processes the request. If the page exists, it responds with the requested files; if not, it may return an error (like "404 Not Found").
- **Step 4: Browser Renders the Page**
  - Your browser receives the HTML file and then requests additional files as needed (e.g., CSS, images, scripts) referenced by the HTML.
  - It assembles all these resources and **renders the page** visually for you. The HTML provides structure, CSS handles appearance, and JavaScript can add interactivity.
- **Website Components**
  - A **domain name** (the website address) points to the server.
  - **Web hosting** is where the site's files are stored.
  - The **homepage** is usually the main entry point.
  - Websites can be **static** (fixed content, faster) or **dynamic** (content changes or is generated on demand).

This entire process—from your address entry to the website display—typically happens in 1–2 seconds.

Building block of website -

The fundamental building blocks of a website are:

- **HTML (HyperText Markup Language):** Provides the structure and semantic organization of the webpage, such as headings, paragraphs, lists, images, links, and containers like <header>, <nav>, <main>, <section>, <article>, <aside>, and <footer>.
- **CSS (Cascading Style Sheets):** Controls the visual design and layout—colors, fonts, spacing, responsiveness, and alignment—separate from the HTML structure.
- **JavaScript:** Adds interactivity and dynamic behavior, allowing content updates without reloading the page, validating forms, animating elements, and more.
- **Content:** The actual information, such as text, images, videos, and links, that users come to consume or interact with.

Beyond the above technical foundations, a well-structured website typically includes these core sections or blocks in its page layout:

- **Header:** Usually includes the logo, site title, and primary navigation. This is often consistent across pages and located at the top.
- **Navigation (Menu):** Provides links to main sections and enables users to move around the site easily.
- **Main Content Area:** The central part of the page that holds unique content such as articles, videos, products, or blog posts. This varies from page to page.
- **Sidebar/Aside:** Contains supplementary information like related links, ads, or additional navigation. Not present on every site.
- **Footer:** Runs along the bottom and typically holds copyright notices, contact info, or additional navigation links.

Other important structural elements:

- **Homepage:** The primary entry point, helping users understand the site's purpose and guiding them to key sections.
- **Categories and Internal Links:** Organization of pages into groups and linking between them for better user navigation and SEO.
- **URL Structure and Sitemap:** Logical, user-friendly URLs and a sitemap for both user experience and search engine indexing.

So, the building blocks of a modern website include the combination of technologies (HTML, CSS, JavaScript), content, and a logical page structure made up of header, navigation, main, sidebar, and footer.

What is CSS?

CSS (Cascading Style Sheets) is a stylesheet language used to define the visual appearance and layout of web pages written in HTML or XML. Using CSS, you control properties like colors, fonts, spacing, borders, positioning, backgrounds, and responsive behavior for different devices.

- The main purpose of CSS is to separate content (HTML) from presentation (design & layout), making websites easier to style, maintain, and update.
- CSS rules can be applied in three ways: inline (in the HTML), internal (within a <style> tag), or external (in a separate .css file linked to the HTML).
- The "cascading" aspect means that when there are conflicting style rules, they are applied according to a predictable priority system, allowing layered and flexible designs.

In summary, **CSS is essential for creating professional, attractive, and responsive websites by controlling how HTML elements look and behave across devices**.

Creators and Maintainers of css?

The creators of CSS (Cascading Style Sheets) are **Håkon Wium Lie** and **Bert Bos**.

- Håkon Wium Lie first proposed CSS in 1994 while working with Tim Berners-Lee at CERN. He envisioned CSS as a way to separate content from presentation on the web, allowing web designers to control layout, colors, and fonts more effectively.
- Bert Bos joined the effort shortly afterward; his prior work on a style language helped shape CSS, and he is regarded as the co-creator alongside Lie.

The **maintainers of CSS** are the **CSS Working Group**, part of the World Wide Web Consortium (**W3C**). This group is responsible for developing and maintaining the CSS specifications. They work on evolving CSS standards, ensuring compatibility, naming consistency, and producing test suites for CSS features. The W3C CSS Working Group guides CSS development to ensure it remains interoperable and widely implemented across different browsers and platforms.

In summary:

- **Creators:** Håkon Wium Lie (primary proposer) and Bert Bos (co-creator)
- **Maintainers:** CSS Working Group at W3C

This group continues to develop and maintain CSS as a cornerstone technology for web design and layout.

CSS History -

CSS (Cascading Style Sheets) History Overview:

- **Origin (1994):** CSS was first proposed in October 1994 by **Håkon Wium Lie** while working with Tim Berners-Lee at CERN. The goal was to separate content (HTML) from presentation (styling) to enable better control over the appearance of web pages and ease web development.
- **Co-creator:** Bert Bos, a Dutch programmer, joined the effort, and together they developed the CSS specification. The term "Cascading Style Sheets" was chosen because the styles could cascade (combine) from multiple sources.
- **First official specification (CSS1, 1996):** Published in December 1996 by the W3C. It introduced basic features such as styling fonts, colors, backgrounds, margins, padding, text alignment, and the box model. It laid the foundation for styling web documents in a separate language from HTML.

- **CSS2 (1998):** Released in May 1998, CSS2 expanded CSS1 with advanced features including absolute, relative, fixed positioning, z-index layering, media types for different devices, bidirectional text, and more font properties like shadows. It started supporting print and aural style sheets.
- **CSS2.1 (2011):** A revision aiming to fix errors and improve stability and compatibility among browsers. CSS2.1 became a W3C recommendation in 2011 after extensive review and iterations.
- **CSS3 (started late 1990s/early 2000s, modular ongoing):** CSS3 departed from monolithic versions to a modular structure, allowing features like animations, gradients, flexible box layouts (Flexbox), and media queries to be developed and adopted incrementally. CSS3 continues being developed with modern styling features.
- **CSS4 (in development):** Not a formal CSS4 specification but a set of proposed future CSS modules focused on enhancing web design capabilities with features like responsive design improvements, variable fonts, and new layout systems.

Summary timeline:

Year	Event
1994	CSS concept proposed by Håkon Wium Lie
1996	CSS1 specification published by W3C
1998	CSS2 released with advanced features
2011	CSS2.1 finalized for stability and compatibility
2000s–present	CSS3 modular evolution ongoing
Future	CSS4 modules under development

The development of CSS was driven by a need to improve control over web page styling, separate it from HTML structure, and support an evolving variety of devices and layouts. The CSS Working Group at W3C maintains and advances the specifications.

CSS Rule Structure -

A **CSS rule** (or ruleset) has a simple and structured format consisting of two main parts:

1. **Selector**:
   This specifies which HTML element(s) the rule applies to. It can be an element name, class, id, attribute, or a combination of these. For example, p targets all paragraphs, .className targets elements with that class, and #idName targets a unique element by id.
2. **Declaration Block**:
   This is enclosed in curly braces { } and contains one or more **declarations**. Each declaration is a pair of a **property** and a **value**, separated by a colon : and ended with a semicolon ;. The property is the CSS feature you want to change (e.g., color, font-size), and the value is the setting you want to give that feature (e.g., red, 16px).

Basic CSS Rule Structure

```
selector {

  property1: value1;

  property2: value2;

  /* more declarations as needed */

}
```

Example

```
p {

  color: red;

  text-align: center;

}
```

- **Selector:** p — targets all <p> elements.
- **Declarations:**
  - color is the property, red is the value, setting the text color.
  - text-align is the property, center is the value, centering the text.
- Declarations inside {} are separated by semicolons.

Summary

Part	Description
Selector	Defines which HTML elements are affected by the rule
Declaration block	Contains CSS declarations inside {} braces
Declaration	Property-value pair defining a style (e.g., color: blue;)
Property	The aspect to style (e.g., color, background, font-size)
Value	The setting for the property (e.g., blue, 16px, bold)

This structure allows CSS to apply styles clearly and efficiently to web page elements.

How to use css in html?

There are three common ways to use CSS in HTML to style your web pages:

1. Inline CSS

- CSS rules are written directly inside an HTML element using the style attribute.
- Useful for quick, specific styling on a single element.
- Not recommended for larger projects because it mixes content with styling and is harder to maintain.

Example:

```
<h1 style="color: blue; font-size: 24px;">This is a blue heading</h1>
```

```
<p style="color: red;">This paragraph is red.</p>
```

2. Internal CSS (Embedded)

- CSS is included within a <style> element inside the <head> section of the HTML document.
- Styles apply to the whole HTML document.
- Good for single-page sites or quick testing.

Example:

```
<head>
  <style>
    body {
      background-color: lightgray;
    }
    h1 {
      color: blue;
      font-size: 30px;
    }
    p {
      color: red;
      font-size: 18px;
    }
  </style>
</head>
<body>
  <h1>Hello World</h1>
  <p>This is a paragraph.</p>
</body>
```

3. External CSS

- CSS rules are written in a separate .css file.
- The CSS file is linked to the HTML using the <link> tag inside the <head> section.
- Best for multi-page websites because you can apply one stylesheet to many pages and maintain styles centrally.

Example:

styles.css (CSS file):

```
body {
  background-color: lightyellow;
}
h1 {
  color: green;
  font-size: 28px;
}
p {
  color: purple;
  font-size: 16px;
}
```

index.html (HTML file):

```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Welcome</h1>
  <p>This is a styled paragraph using external CSS.</p>
</body>
```

Summary Table of CSS Usage Methods

Method	Where CSS is Written	Scope	Use Case
Inline CSS	Inside HTML element's style	Single element only	Quick styles, overrides, testing
Internal CSS	Inside <style> in <head>	Single HTML page	Single page styling, development/testin g
External CSS	Separate .css file linked via <link>	Multiple pages	Large sites, reusable centralized styling

Using external CSS is generally recommended for better maintainability and scalability once your project grows.

Types of selector -

Here are the common types of CSS selectors used to target HTML elements for styling:

1. Element (Type) Selector

- Selects all elements of a specific type.
- Example: p { color: blue; } styles all <p> elements.

2. ID Selector

- Selects a single element with a specific id attribute.
- Syntax: #idName
- Example: #header { background: gray; } styles the element with id="header".

3. Class Selector

- Selects all elements with a specific class attribute.
- Syntax: .className
- Example: .highlight { color: red; } styles all elements with class="highlight".

4. Universal Selector

- Selects **all** elements on the page.

- Syntax: *
- Example: * { margin: 0; padding: 0; } resets margin and padding for all elements.

5. Group Selector

- Allows applying the same style to multiple selectors at once.
- Syntax: selector1, selector2, selector3
- Example: h1, h2, p { color: green; } styles all <h1>, <h2>, and <p> elements.

6. Attribute Selector

- Selects elements with a specific attribute or attribute value.
- Examples:
  - [type] { border: 1px solid black; } targets elements with a type attribute.
  - [type="text"] { background: yellow; } targets elements with type="text".

Summary Table

Selector Type	Syntax	Description	Example
Element Selector	element	Selects all elements of that type	p { color: blue; }
ID Selector	#id	Selects one unique element by id	#header { background: gray; }
Class Selector	.class	Selects all elements with that class	.highlight { color: red; }
Universal Selector	*	Selects all elements	* { margin: 0; }
Group Selector	selector1,selector2	Applies styles to multiple selectors at once	h1, h2, p { color: green; }
Attribute Selector	[attr=value]	Selects elements based on attribute value	input[type="text"] { background: yellow; }

These selectors form the foundation of targeting elements in CSS. There are many more advanced selectors like pseudo-classes, combinators, and pseudo-elements, but these are the most commonly used basic types.

Advanced Selectors -

Here is an explanation of **advanced CSS selectors** including their types, syntax, and examples to help you target elements more precisely:

1. **Adjacent Sibling Selector +**

- Selects an element that is **immediately after** a specified sibling element.
- Syntax: A + B selects element B that immediately follows element A.

Example:

```
h4 + ul {

  border: 2px solid red;

}
```

This targets any <ul> that directly follows an <h4>.

2. **General Sibling Selector ~**

- Selects all sibling elements that follow a specified element (not necessarily immediately).
- Syntax: A ~ B selects all B siblings after an A.

3. **Child Selector >**

- Selects elements that are **direct children** of a specified parent.
- Syntax: parent > child

Example:

```
div > p {

  color: blue;

}
```

This styles <p> elements that are **direct children** of a <div>, but not deeper nested.

4. **Attribute Selectors**

- Select elements based on the existence or value of attributes.
- Syntax and examples:

Syntax	Description	Example
[attr]	Selects elements with the attribute	input[type]
[attr="value"]	Selects elements where attribute equals value	input[type="checkbox"]
[attr^="value"]	Selects elements where attribute starts with value	a[href^="https"]
[attr$="value"]	Selects elements where attribute ends with value	img[src$=".jpg"]
[attr*="value"]	Selects elements where attribute contains value	div[class*="container"]

5. **Pseudo-class Selectors**

- Target elements in a specific **state** or based on their position in the DOM.

Common pseudo-classes:

Pseudo-class	Description	Example
:hover	When the user hovers over an element	button:hover
:focus	When an element gains focus	input:focus
:nth-child(n)	Select the nth child of a parent	li:nth-child(2)
:nth-of-type(n)	Select the nth child of its type	p:nth-of-type(3)
:first-child	Selects the first child	div:first-child
:last-child	Selects the last child	p:last-child
:not(selector)	Selects elements NOT matching the selector	div:not(.active)

Example:

```
li:nth-child(even) {

  background-color: #eee;

}
```

Styles all even list items.

6. **Pseudo-element Selectors**

- Allow styling of **parts** of elements or inserting content.

Common pseudo-elements:

Pseudo-element	Description	Example
::before	Inserts content before an element's content	p::before
::after	Inserts content after an element's content	p::after
::first-letter	Styles the first letter in a block of text	p::first-letter
::first-line	Styles the first line of text	p::first-line

Example:

```
p::before {
  content: "Note: ";
  font-weight: bold;
}
```

This adds the word "Note:" before every paragraph.

7. **Grouping Selector ,**

- Allows you to group multiple selectors to apply the same styles.
- Syntax: selector1, selector2 { ... }

Example:

```
h1, h2, p {
  color: green;
}
```

Styles all h1, h2, and p elements with green text.

Summary of Common Advanced CSS Selectors:

Selector Type	Syntax Example	What It Selects
Adjacent Sibling	h4 + ul	<ul> immediately following <h4>
General Sibling	h2 ~ p	All <p> siblings after <h2>
Child	div > p	<p> that are direct children of <div>
Attribute	input[type="text"]	<input> elements of type text
Pseudo-class	li:nth-child(odd)	Odd list items
Pseudo-element	p::before	Content before paragraphs
Grouping	h1, h2, p	All h1, h2, and p elements

What is declaration ?

A **CSS declaration** is a fundamental part of a CSS rule that specifies a single style property and its assigned value to apply to HTML elements.

Key points about a CSS declaration:

- It is a **property-value pair** separated by a colon (:).
- The **property** defines the style aspect you want to change (e.g., color, font-size, margin).
- The **value** sets how that property should be applied (e.g., red, 16px, 10px).
- Each declaration ends with a **semicolon** (;) to separate it from other declarations.
- Declarations are placed inside curly braces {} in a **declaration block**, which together with a selector forms a CSS rule.

Example of declarations:

```
color: red;

font-size: 16px;

background-color: blue;
```

In this example:

- color is the property, and red is the value — it sets the text color to red.
- font-size is the property, and 16px sets the font size to 16 pixels.
- background-color is the property, and blue sets the background color to blue.

How it fits in CSS syntax:

```
selector {

  property: value;

  property2: value2;

}
```

For example:

```
p {

  color: red;

  font-size: 16px;

}
```

This rule styles all <p> elements with red text and a font size of 16 pixels.

In summary, a **CSS declaration** defines exactly *what* style change to apply (property) and *how* to apply it (value) for elements targeted by the selector within the CSS rule.

CSS Comments -

In CSS, **comments** are used to add notes, explanations, or temporarily disable code without affecting how the page looks. Comments are ignored by the browser—they are only there for developers.

Syntax for CSS Comments

CSS comments start with /* and end with */.

```
/* This is a CSS comment */
```

Examples

1. Single-line comment:

```
p {
  color: red; /* Makes all paragraphs red */
}
```

2. Multi-line comment:

```
/*
This is a multi-line comment.
You can write notes here for developers.
*/
p {
  font-size: 16px;
}
```

Key Points

- CSS **does not** support // style comments (like JavaScript).
- Comments can be placed **anywhere** in CSS: before a selector, inside a declaration block, or between rules.
- Useful for:
  - Documenting your stylesheet
  - Explaining specific style choices
  - Temporarily disabling rules while testing:
  - */**
  - *h1 {*
  - * color: blue;*
  - *}*
  - **/*
- Comments work in **both** internal <style> blocks and external .css files.

✅ **Best Practice:** Use comments to organize CSS into sections, especially for large projects.

Colors in css -

In CSS, the **color** property is used to set the color of the text (foreground color) of an HTML element.

Syntax:

```
selector {

  color: value;

}
```

Ways to specify color values in CSS:

1. **Color Keywords (Names)**
   - Example: color: red;, color: blue;, color: orange;, color: rebeccapurple;
   - CSS supports around 140 named colors.
2. **Hexadecimal (HEX) Notation**
   - Format: #RRGGBB or shorthand #RGB
   - Example: color: #ff0000; (red), color: #00ff00; (green), color: #0000ff; (blue)
3. **RGB and RGBA Functions**
   - RGB defines colors by Red, Green, Blue components (0–255)
   - RGBA includes Alpha (opacity) value (0.0 to 1.0)
   - Examples:
     - color: rgb(255, 0, 0); (red)
     - color: rgba(255, 0, 0, 0.5); (semi-transparent red)
4. **HSL and HSLA Functions**
   - HSL stands for Hue (degrees 0-360), Saturation (0%-100%), Lightness (0%-100%)
   - HSLA adds Alpha (opacity)
   - Examples:
     - color: hsl(0, 100%, 50%); (red)
     - color: hsla(0, 100%, 50%, 0.3); (semi-transparent red)

Example usage in CSS:

```
p {

  color: blue;

}



h1 {

  color: #ff6347; /* tomato */

}



div {

  color: rgb(34, 139, 34); /* forestgreen */
}


span {
  color: rgba(255, 165, 0, 0.7); /* semi-transparent orange */
}


a {
  color: hsl(240, 100%, 50%); /* blue */
}
```

Additional notes:

- The color property affects text and text decorations.
- It's inherited by default from parent elements.
- Use combinations of foreground color and background color for readability.
- You can also use CSS variables (custom properties) for colors to maintain consistency.

This allows flexible and precise control of text coloring across your webpages.

Backgrounds -

The CSS **background** property is a versatile way to set the background of an HTML element, including colors, images, positioning, and other stylistic options. It can be used as a shorthand to control multiple background-related properties at once or you can set them individually.

Key Background Properties in CSS

Property	Description
background-color	Sets a solid background color (e.g., red, #ff0000, rgb(255,0,0))
background-image	Defines an image or gradient as the background (e.g., url('image.jpg'))
background-repeat	Controls repetition of the background image (repeat, no-repeat, repeat-x, repeat-y)
background-position	Sets starting position of background image (center, top left, 50% 50%, etc.)
background-size	Defines size of background image (auto, cover, contain, or specific dimensions)
background-attachment	Defines if background scrolls with content (scroll) or stays fixed (fixed)
background-origin	Specifies positioning area for background (padding-box, border-box, content-box)
background-clip	Defines the painting area of background (similar to background-origin)

Background Shorthand Syntax

You can set all of the above properties together using one shorthand:

```
background: [background-color] [background-image] [background-position] / [background-size] [background-repeat] [background-attachment] [background-origin] [background-clip];
```

Example:

```
body {
  background: #cccccc url('bg.jpg') no-repeat center center / cover fixed;
}
```

This sets a gray background color, with a non-repeating background image centered and scaled to cover the entire area, and that image stays fixed when scrolling.

Usage Examples

1. Background Color Only

```
div {
  background-color: lightblue;
}
```

2. Background Image with Repeat

```
section {
  background-image: url('pattern.png');
  background-repeat: repeat;
}
```

3. Non-repeating Background Image Positioned Top Left

```
header {
  background-image: url('logo.png');
  background-repeat: no-repeat;
  background-position: top left;
}
```

4. Fixed Background Image with Cover Size

```
body {
  background: url('mountains.jpg') no-repeat center center fixed;
  background-size: cover;
}
```

5. Gradient Background

```
.container {
  background: linear-gradient(to right, #22c1c3, #fdbb2d);
}
```

Important Notes

- If you use the shorthand background, the order of values matters and some values can be omitted (they will take default values).
- For background-size to work in shorthand syntax, use a slash / between position and size, e.g., center/cover.
- background-color is usually placed last in the shorthand.
- You can use multiple background images by separating them with commas.

In summary:
The CSS background property allows you to control the visual backdrop of any HTML element, whether by solid color, image, gradients, or combinations, with extensive control over how and where the background is displayed.

Units -

In CSS, **units** specify the measurement for lengths, sizes, spacing, and other dimensional properties. They tell the browser how big or small something should be. CSS units are generally classified into **three main types: absolute, relative, and viewport units**.

1. Absolute Units

Absolute units are fixed and do not change relative to anything else. They provide precise measurements.

- **px** (pixels): One pixel, relative to the display device. (1px = 1/96th of 1 inch roughly)
- **cm** (centimeters): 1cm = 96px/2.54
- **mm** (millimeters): 1mm = 1/10th of 1cm
- **in** (inches): 1in = 2.54cm = 96px
- **pt** (points): 1pt = 1/72 of 1 inch
- **pc** (picas): 1pc = 12 points or 1/6 of an inch

Note: Absolute units are often better suited for print designs and are less used for screens because of varying screen sizes.

2. Relative Units

Relative units scale in relation to something else, like the parent element or root element size. They are beneficial for responsive and flexible designs.

- **em**: Relative to the *font size* of the parent element.
- **rem**: Relative to the *font size* of the root <html> element.
- **%** (percentage): Relative to the *parent element's* dimension (often width or height).

- **ex**: Relative to the height of the lowercase letter "x" in the current font.
- **ch**: Relative to the width of the "0" (zero) character in the current font.
- **fr**: Fractional unit used in CSS Grid layouts, representing a fraction of available space.

3. Viewport Units

These units are relative to the size of the browser viewport (visible area of the webpage).

- **vw**: 1% of the viewport's width
- **vh**: 1% of the viewport's height
- **vmin**: 1% of the smaller dimension of the viewport (width or height)
- **vmax**: 1% of the larger dimension of the viewport (width or height)

Summary Table

Unit	Type	Relative To	Description
px	Absolute	Device pixels	Fixed pixel measurement
cm	Absolute	Physical centimeters	1cm = 96px/2.54
mm	Absolute	Physical millimeters	1mm = 1/10 cm
in	Absolute	Inches	1in = 2.54 cm
pt	Absolute	Points (1/72 inch)	Used in typography
pc	Absolute	Picas (12 points)	Mainly for print
em	Relative	Parent element's font size	Scales with font sizes
rem	Relative	Root element's font size	Consistent scaling across page
%	Relative	Parent element's dimension	Percentage-based sizing
ex	Relative	x-height of font	Height of lowercase "x"
ch	Relative	Width of "0" character	Width of character zero
fr	Relative	Fraction of grid container	Fractional unit for CSS Grid
vw	Viewport	1% viewport width	Responsive width-based sizing
vh	Viewport	1% viewport height	Responsive height-based sizing

vmin	Viewport	1% smaller viewport side	Responsive to smaller dimension
vmax	Viewport	1% larger viewport side	Responsive to larger dimension

Choosing units depends on context:

- Use **px** for fixed sizes.
- Use **em** and **rem** for scalable fonts and spacing.
- Use **%**, **vw**, **vh** for responsive layouts.
- Use absolute units when precise, unchanging measurements are needed (like print).

This flexible system of CSS units helps create designs that can adapt beautifully across different screens and devices.