Week 2 - CSS

Semester 1
2024

ISCG6420
Internet & Website Development

Table of Contents

HTML Semantic Elements

Semantic Elements

There are over 100 semantic elements we can use in HTML. The purpose and flexibility of the elements vary significantly, which affects their popularity in web development.

Semantic HTML refers to utilising elements based on their intended purpose when creating HTML documents for a website.

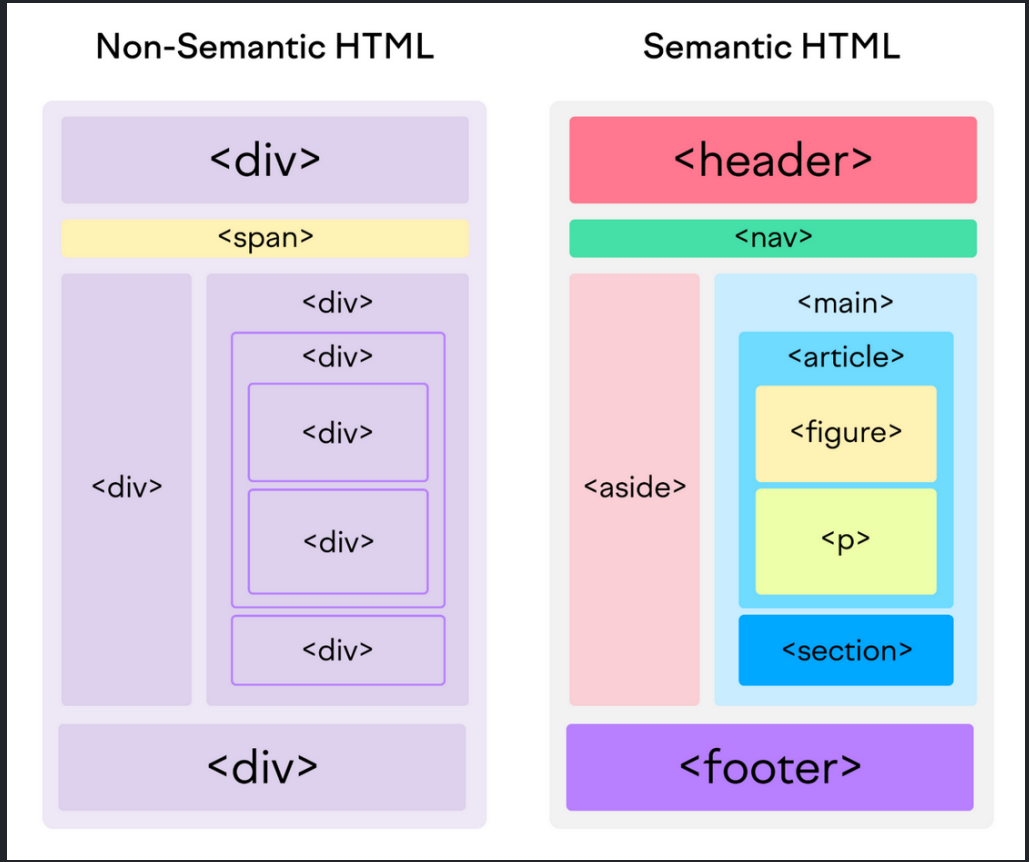


Image from Semrush

Example Elements

Element	Description
main	Specifies the main content of a document
section	Represents a generic section of a document or application
article	Represents an independent item section of content
header	Represents a group of introductory or navigational aids
footer	Represents a foot for its nearest ancestor sectioning content or sectioning root element
nav	Represents a section with navigational links
aside	Represents a section of a page that contists of content that is tangentially related to the content around the aside element, and which could be considered separate from that content
figure	Represents some flow content
figcaption	Represents a caption or legend for the contents of the parent figure element
summary	Represents a summary, caption, or legend for the rest of the contents of the summary element’s parent details element, if any.
body	Represents the main content of the document

Why use semantics? I can use divs!

Semantic elements are added to the HTML living standard with the intention of unifying development approaches to page structuring. This has a few benefits:

- **Clarifies the purpose of content**
- **Simplifies knowledge transfer across development approaches**
- **Makes compliant websites structurally uniform**
- **Improves compatibility with accessibility tools**
- **Helps search engines understand your website**

Semantic Nesting

Semantic elements can exist within other elements as long as the element is compliant with its intended purpose within scope of the parent element.

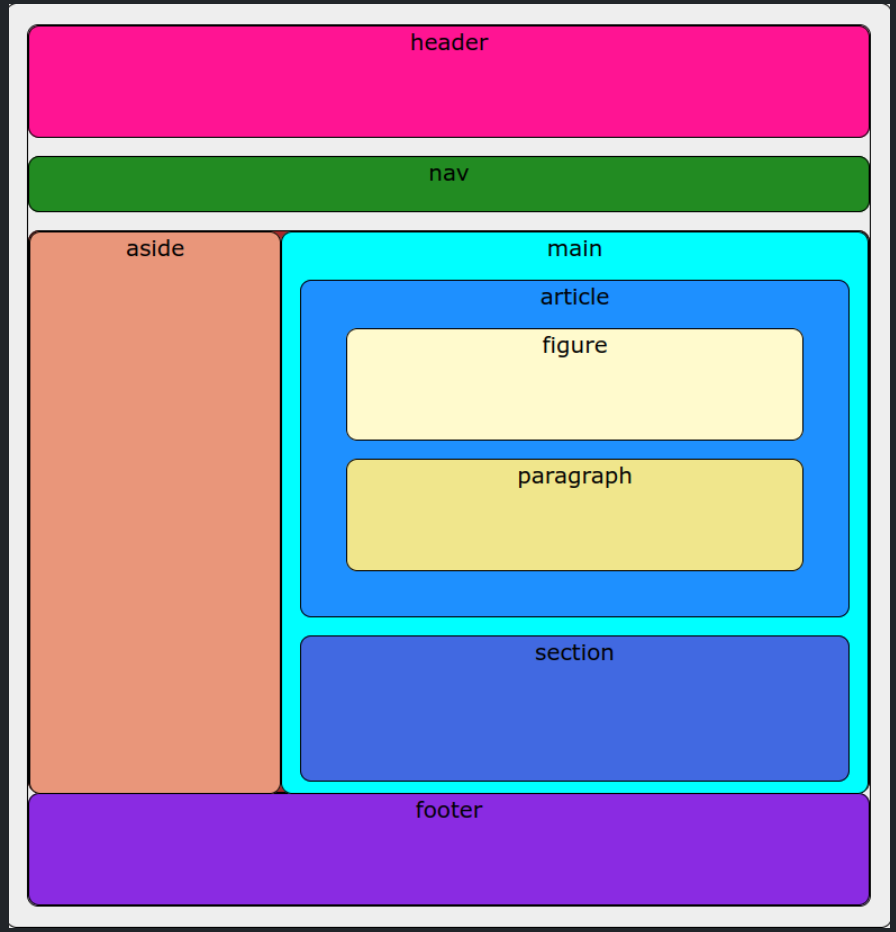
Nesting refers to elements existing inside another element. Inner elements are called **child** elements, and outer elements are **parent** elements. :. Parent – Child relationship.

Examples:

- <section> containing <article>
- <article> containing <figure>
- <main> containing <aside>
- <aside> containing <section>
- <details> containing <mark>
- <header> containing <nav>
- <footer> containing <figure>

Example

```
<body>
    <header>
        header
    </header>
    <nav>
        nav
    </nav>
    <content> <!-- custom -->
        <aside>
            aside
        </aside>
        <main>
            main
            <article>
                article
                <figure>
                    figure
                </figure>
                <p>
                    paragraph
                </p>
            </article>
            <section>
                section
            </section>
        </main>
    </content>
    <footer>
        footer
    </footer>
</body>
```



Div and Span

: Content Division element.

It is a generic container for **flow** content. It has no default styling or influence on page content or layout, but can be manipulated with CSS styling or by applying a layout model to its parent element.

 is typically used to define a block

```
<div>
    This is a block-level element, which
    represents a vertical segment of the parent.
    This <span>is an inline element</span>
    which represents a horizontal segment of a
    block element.
</div>
```

: Content Span element.

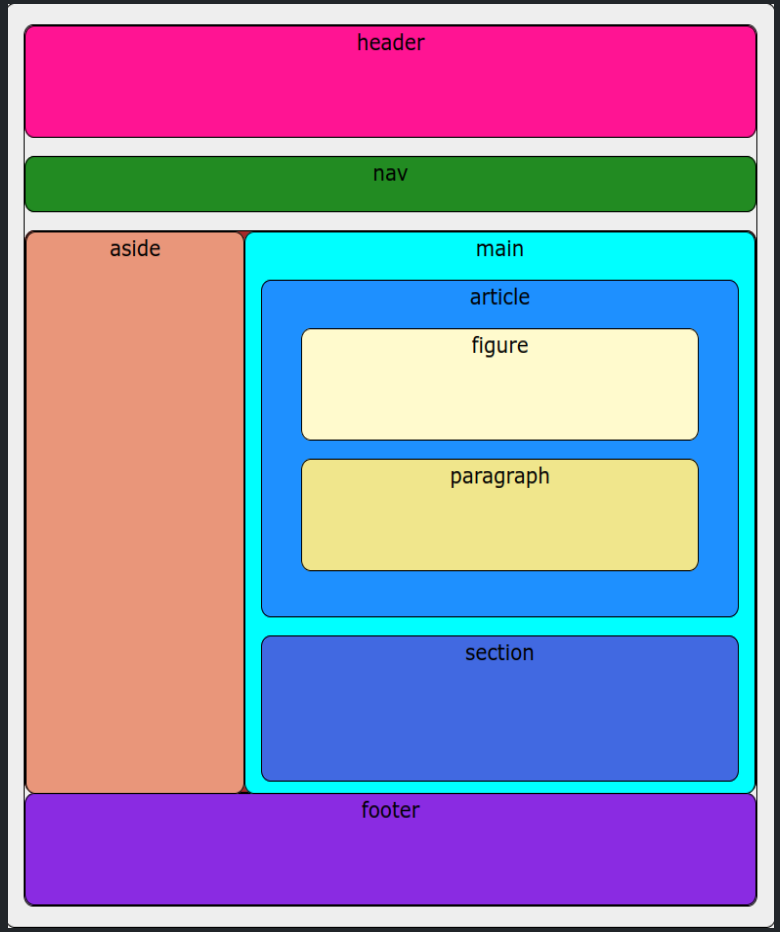
It is a generic **inline** container for **phrasing** content. It does not inherently represent anythting. It can be used to group elements for styling purposes, or because they share attribute values, such as lang.

 is typically used as an inline element

This is a block-level element, which represents a vertical segment of the parent. this is an inline element which represents a horizontal segment of a block element.

Example

```
<body>
    <div class="header">
        header
    </div>
    <div class="nav">
        nav
    </div>
    <div class="content">
        <div class="sidebar">
            aside
        </div>
        <div class="main">
            main
            <div class="article">
                article
                <div class="figure">
                    figure
                </div>
                <div class="paragraph">
                    paragraph
                </div>
            </div>
            <div class="section">
                section
            </div>
        </div>
    </div>
    <div class="footer">
        footer
    </div>
</body>
```



References

Mozilla Developer Network (MDN)

World Wide Web Consortium (W3C)

Planning ahead

We can make using semantic elements easier and more accurate by planning our web page layout. One common way to do this is to create a wireframe diagram.

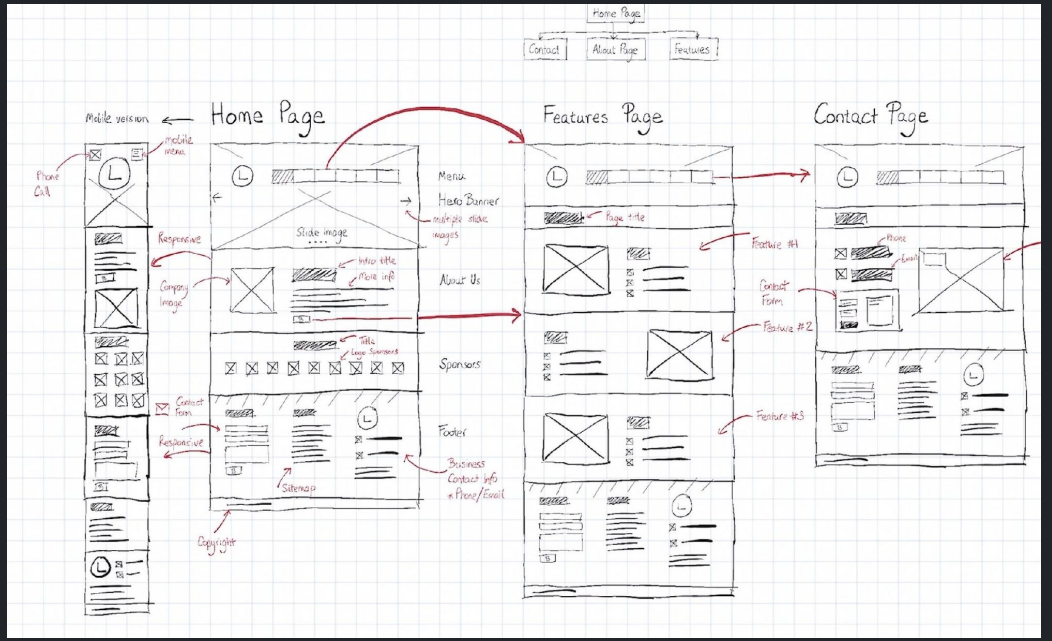


Image from FCC

2 Software Dev Life Cycle

SDLC

Software Development Life Cycle is a representation of the phases or stages a software development project undertakes. The process is cyclical as output from the last phase are compatible with inputs for the first phase, albeit with a narrower scope.

Diagrams like wireframes and other mockup designs are artifacts created during the Design phase of the SDLC.

Developer career paths typically begin at phase 6 and progress towards phase 1

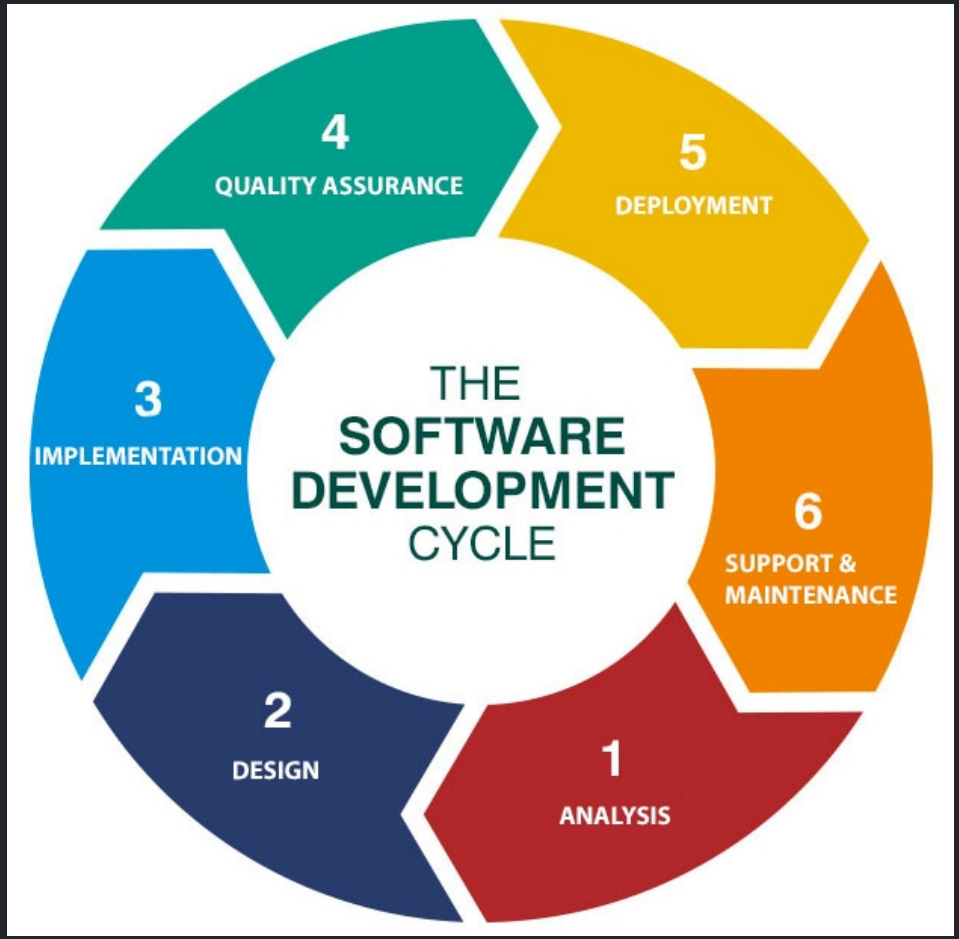


Image from Lvivity

SDLC
Phase 1: Analysis

The Planning / Analysis phase **defines inital scope** for the current cycle of development. The scope provides bounds to design within in phase 2.

Typical tasks include **cost-benefit analysis**, **scheduling**, **resource estimation**, and **allocation**.
The development team **collects requirements** from several stakeholders such as customers, internal and external experts, and managers to create a software requirements specification document.

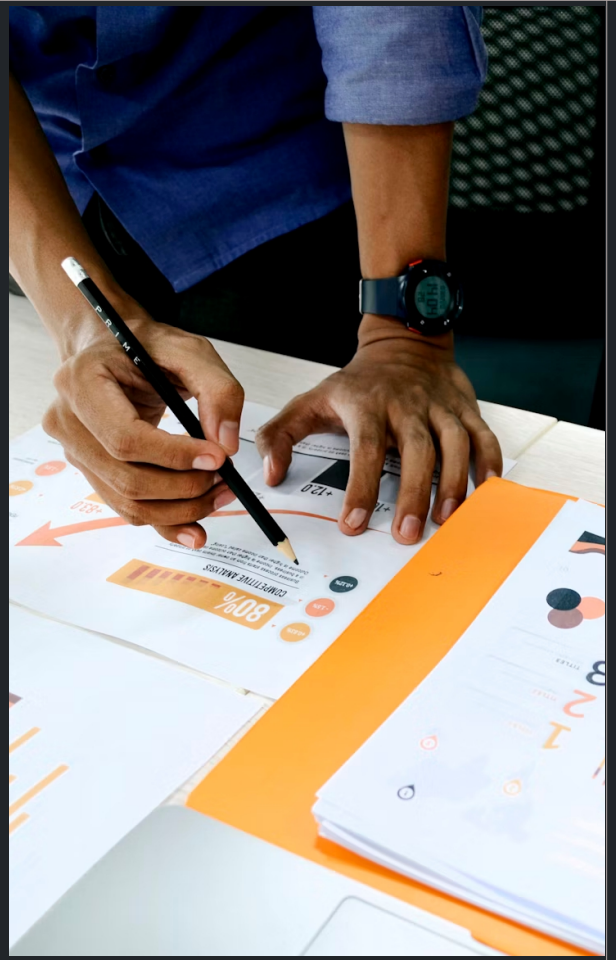
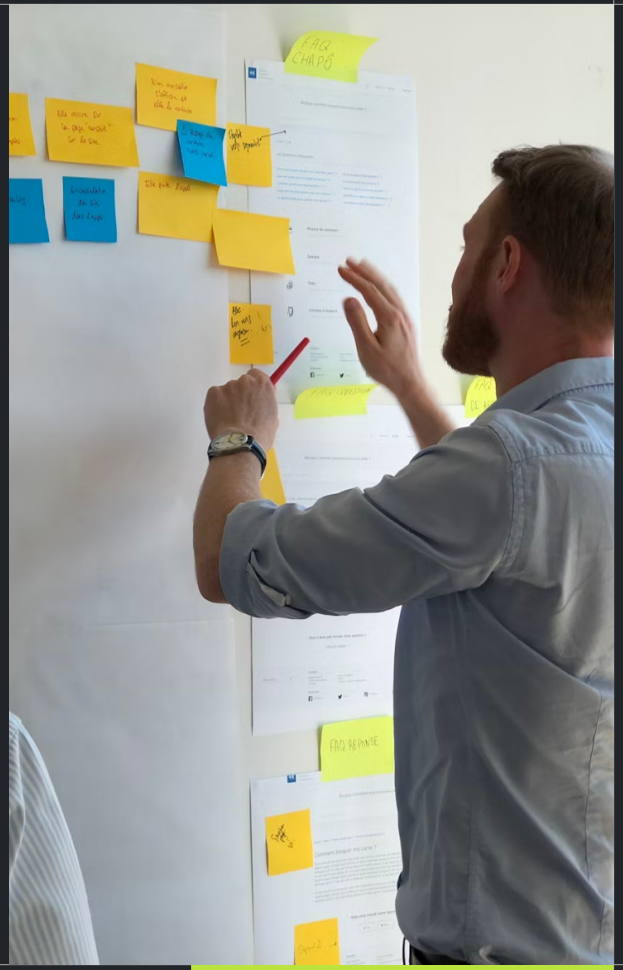


Photo by uxindo on Unsplash

https://aws.amazon.com/what-is/sdlc/

SDLC
Phase 2: Design

In the design phase, software engineers **analyze requirements** and identify the best solutions to create the software. For example they may consider integrating pre-existing modules, **make technology choices**, and identify development tools. They will look at how to best integrate the new software into any existing IT infrastructure the organisation may have.

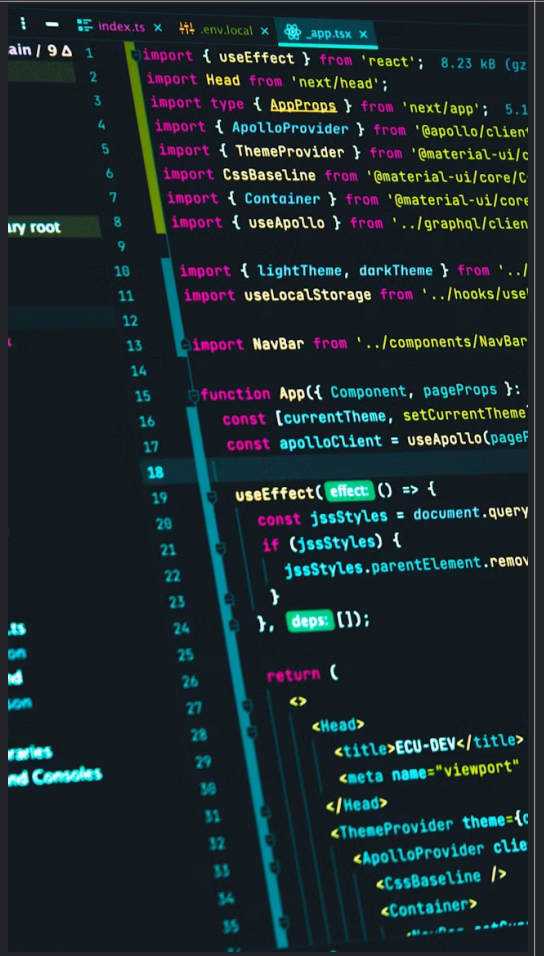


https://aws.amazon.com/what-is/sdlc/

Photo by Neonbrand on Unsplash

SDLC Phase 3: Implementation

In the implementation phase, the development team codes the product. They analyze the requirements to identify smaller coding tasks they can do daily to achieve the final result.

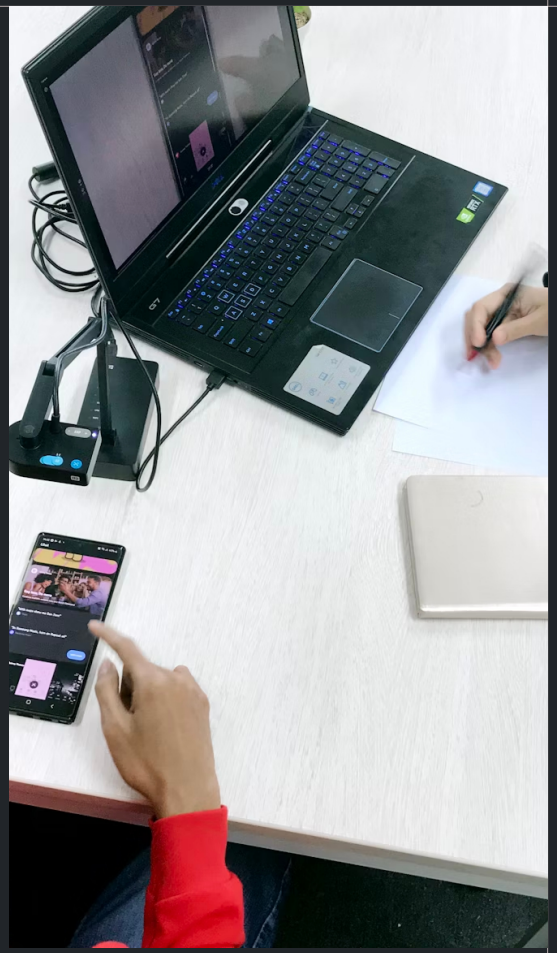


https://aws.amazon.com/what-is/sdlc/

Photo by juanjodev02 on Unsplash

SDLC
Phase 4: QA

The development team combines **automation** and **manual testing** to check for software bugs. Quality Analysis includes **testing the software for errors** and checking if it **meets customer requirements**. Because many teams immediately test the code they write, the testing phase often **runs parallel to the development phase**.



https://aws.amazon.com/what-is/sdlc/

Photo by uxindo on Unsplash

SDLC Phase 5: Deployment

When teams develop software, the code and test on a different copy of the software than the one that the users have access to. The software that customers use is called **Production**, while other copies are said to be in the **Build Environment** or testing environment.

Having separate build and production environments ensures that customers can continue to use the software even while it is being changed or upgraded. The deployment phase includes several tasks to move the latest build copy to the production environment, such as packaging, environment configuration, and installation.

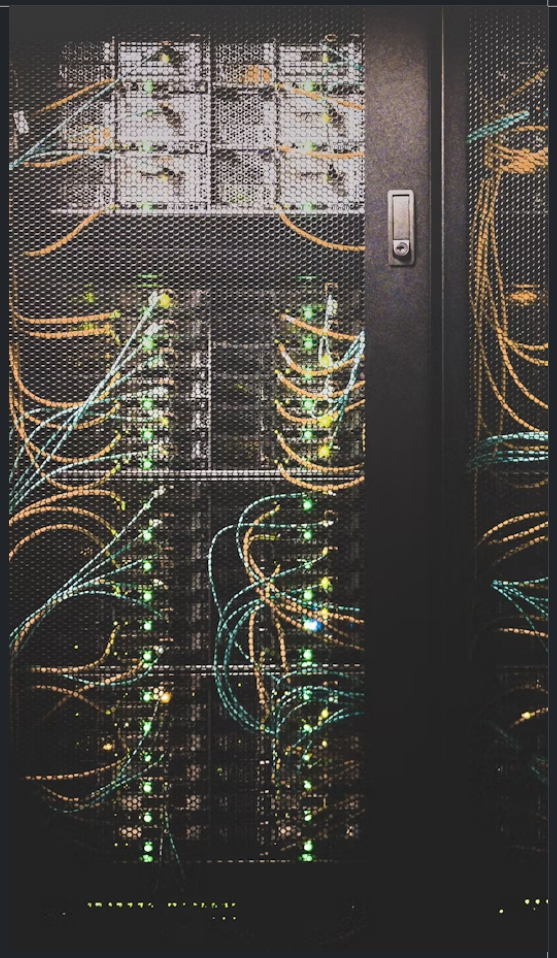


Photo by tvick on Unsplash

https://aws.amazon.com/what-is/sdlc/

SDLC
Phase 6: Support

In the support or maintenance phase, the team **fixes bugs**, resolves customer **issues**, and manages software changes. In addition, the team monitors overall system performance, security, and user experience to identify new ways to improve the existing software.

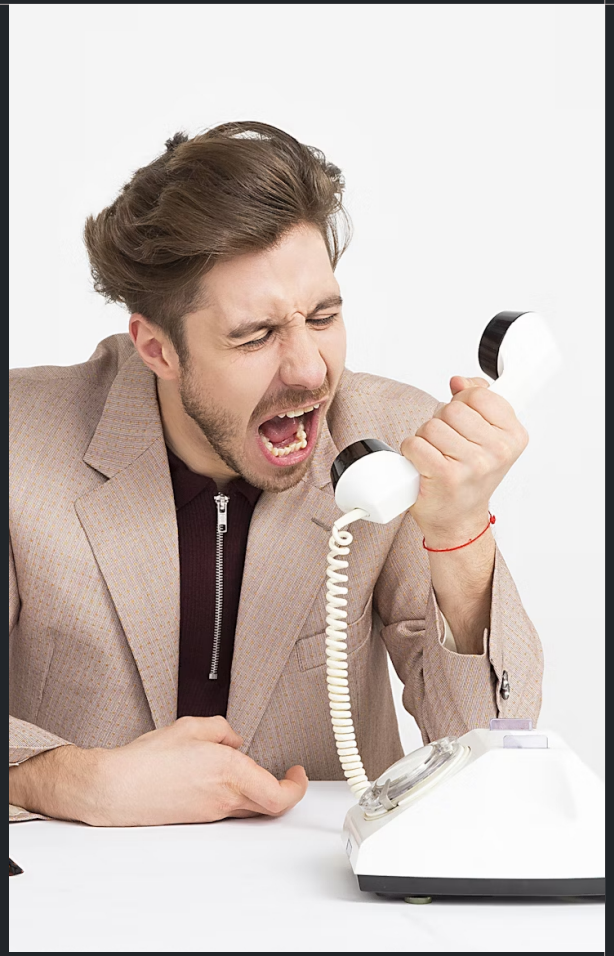


Photo by icons8 on Unsplash

https://aws.amazon.com/what-is/sdlc/

SDLC vs Project Model

Project management methodology models such as Waterfall and Agile are not alternatives to SDLC. They are implementation variations of SDLC.

Waterfall methodology is taught in Project Planning and Control (ISCG6411)

Agile methodology is taught in Agile and Lean Software Delivery (ISCG7427)

01 Waterfall

Highly disciplined
Low flexibility
Suitable for small, well scoped projects

02 Agile

Rapid reflection timelines
Highly flexible
Susceptible to scope creep

03 Spiral / Evolutionary

Combines Waterfall & Agile
Risk analysis focused
Suitable for large and complex projects

Design:
Site Map Diagram

The Site Map diagram or Visual Site Map is a design-phase planning tool that outlines the structure of a website's content and pages in a graphical format.

It provides a bird's eye view of all your web pages and their interconnections. This allows you to plan the user experience (UX) and your website's overall design

Photo by seagrave on Unsplash

https://miro.com/diagramming/how-to-create-a-sitemap/

Site Map Example

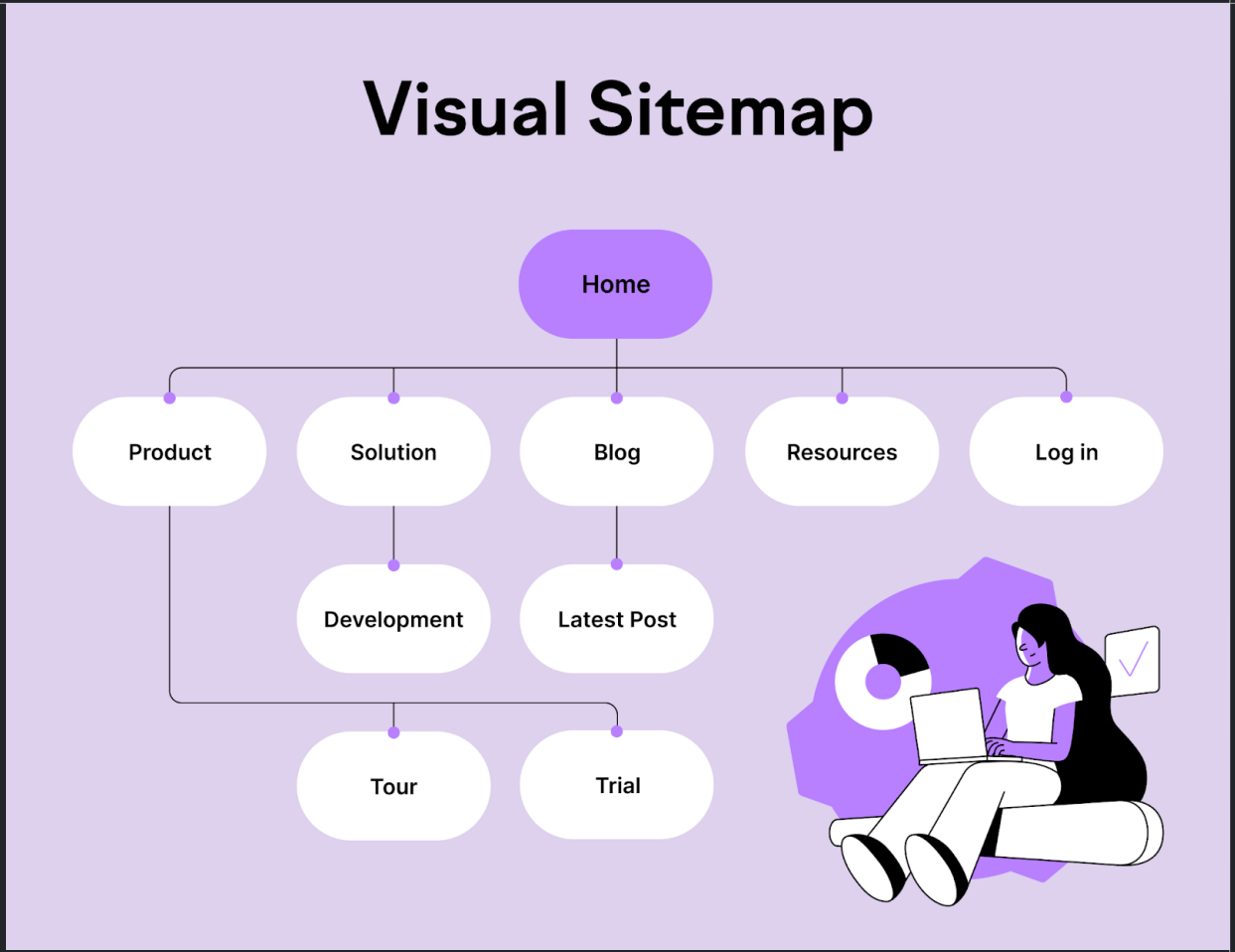


Image from Semrush

Design:
Wireframe Diagram

The Wireframe diagram is a design tool that allows for rapid prototyping of interface layouts. In web development they are used to create layouts of web pages.

Wireframe diagrams should be quick to create, flexible to changes and require little resources to deliver.

Your diagram should be designed well enough to convey the semantics of your design without the resource requirements of a true website layout implementation.

Photo by fazurrehman on Unsplash

Example

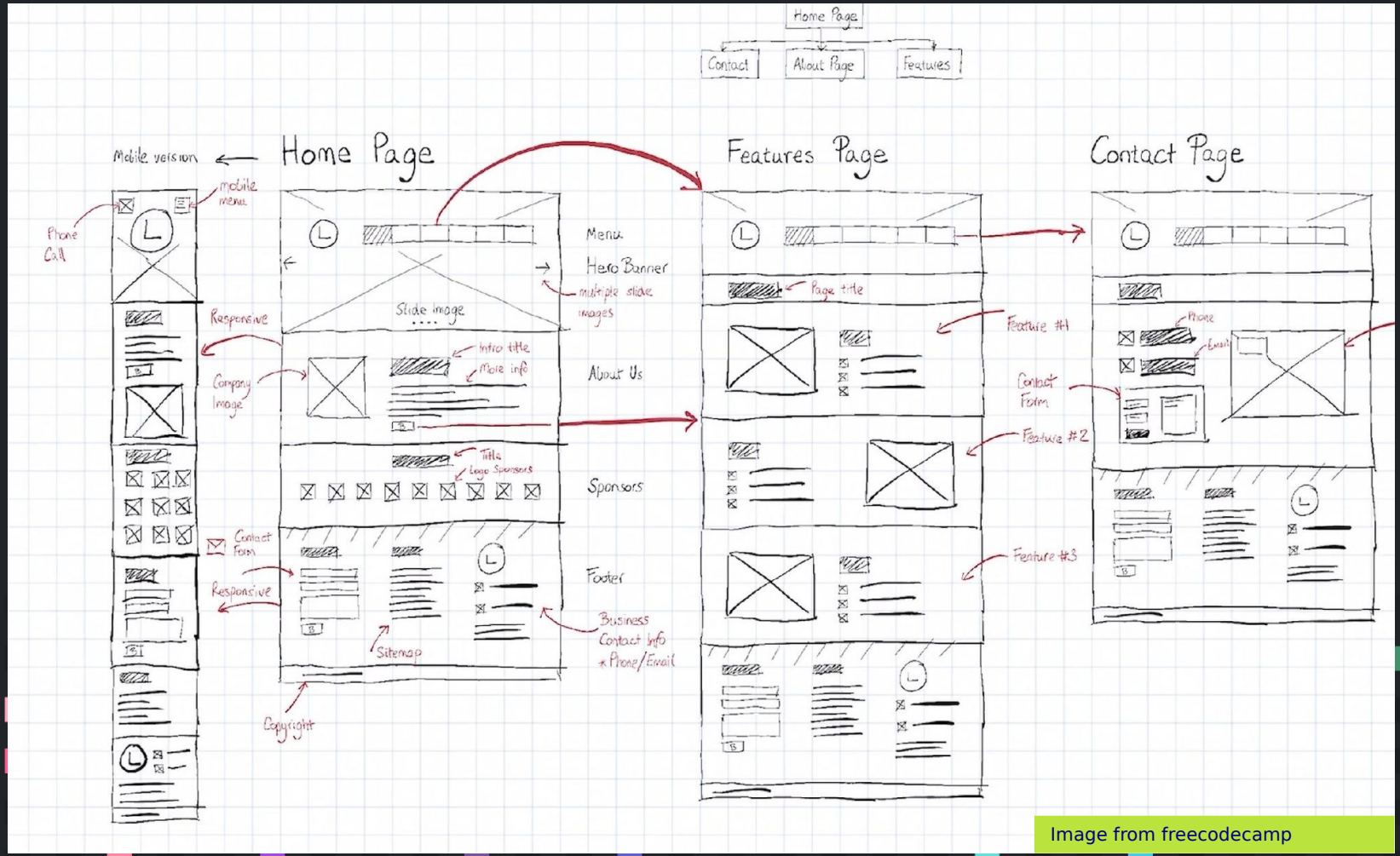
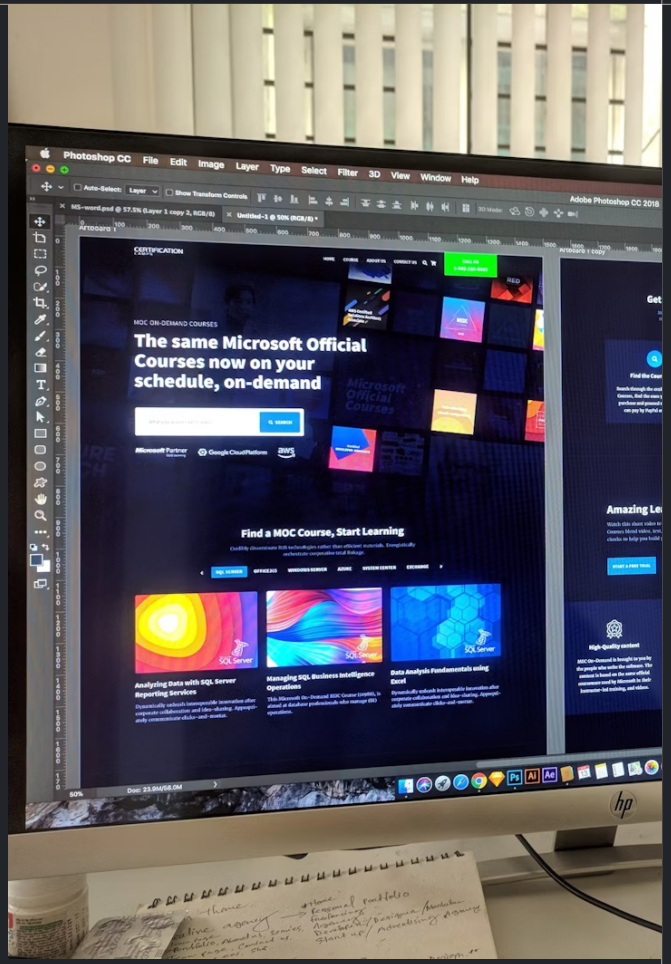


Image from freecodecamp

Design: Mockup

A website mockup is a static visual model of what a web page, website, or web application will look like in its final form. A mockup is designed to resemble the final product, but it is not yet functional (i.e., you can't interact with it).

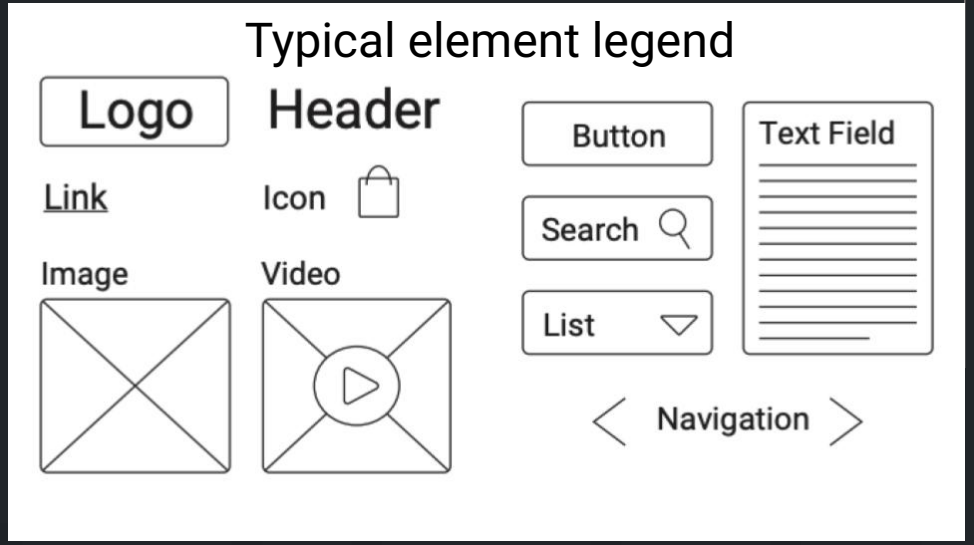
This process is typically performed after the wireframe diagramming process, due to the increased resources requiremed to deliver a Mockup.



https://blog.hubspot.com/website/website-mockup

Photo by easiblu on Unsplash

How to Wireframe



Wireframe.cc https://www.youtube.com/watch?v=qpH7-KFWZRI

Process https://www.youtube.com/watch?v=e2Oynq-mOLk

Image from codeacademy

Exercise

Complete exercise 1 & 2. PDF documents are available on Moodle – Week 2, and on Github

01 Exercise – wireframe 1

Mostly independent exercise
Discussion with peers beneficial

02 Exercise – wireframe 2

Group exercise
Groups of 2 or more

Cascading Style Sheets

Why CSS?

CSS exists as a means of styling websites without modifying the structure of the HTML. Separating HTML from CSS allows modification of each language to be handled independently and in parallel without affecting the other.

CSS is syntactically lighter than HTML, reducing space and time resource consumption during development and during client access.

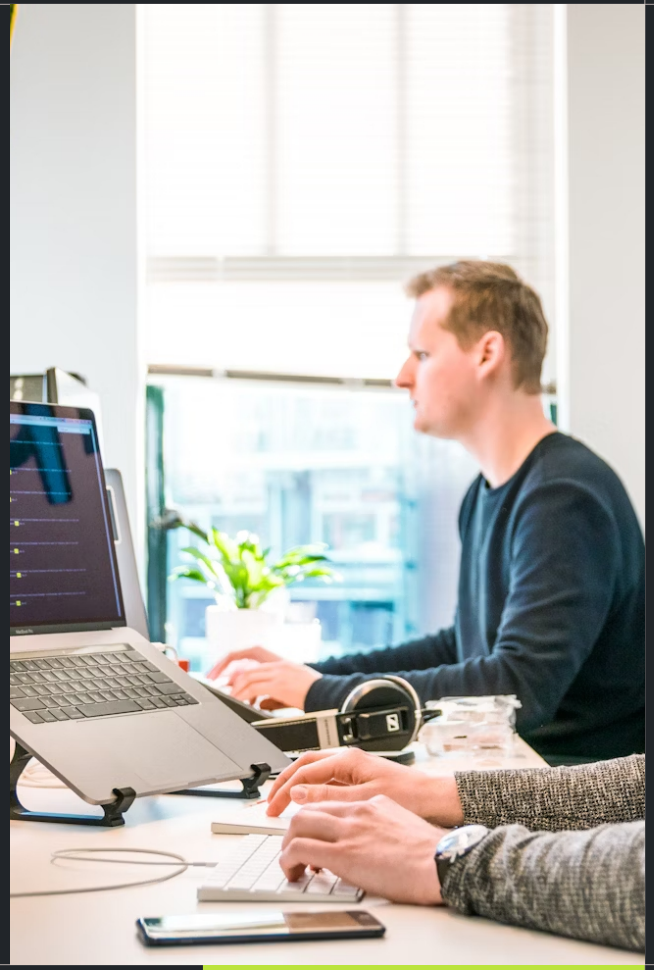


Photo by Tim on Unsplash

Using CSS

CSS can be added to your website in a few ways:

External CSS: from file with a <link> relation
Internal CSS: from <style> element
Inline CSS: from element style attribute

```
<!-- External CSS -->
<link rel="stylesheet" href="style.css">



<!-- Internet CSS -->
<style>
    div {
        color: #fff;
        text-align: center;
    }
</style>



<!-- Inline CSS -->
<div style="color: #fff; text-align: center">
    div content
</div>
```

Cascading...

Styles can be applied multiple times, even with conflicting or overwriting changes.

Order of application matters, as the last change made to an element's style will be what is displayed on the web page.

```
<!-- Internet CSS -->
<style>
    div {
        color: #555;
        text-align: right;
    }
</style>



<!-- Inline CSS -->
<div style="color: #fff; text-align: center">
    div content
</div>
```

...Style...

Syntax for CSS styles are as follows:

Selector: Identifies the affected element(s)

Declaration: Block containing style changes

Property: Identifies the style to be changed

Value: Sets the new property value

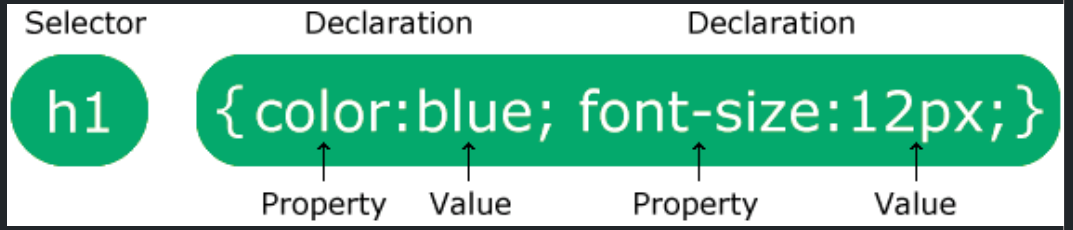


Image from w3schools

...Sheet

Stylesheets contain one or more style declared independently. A style can apply to one or more elements, and can contain changes that affect one or more style properties.

Styles are typically ordered by scope descending. The styles with the broadest scope are listed before styles with narrower scope to follow cascading modification order.

style.css > ...

```
1   html {
2       box-sizing: border-box;
3   }
4   *, *:before, *:after {
5       box-sizing: inherit;
6   }
7
8   html,body {
9       font-family: Verdana;
10      font-size: 15px;
11      line-height: 1.5;
12  }
13
14  body {
15      float: left;
16  }
17
18  h1 {
19      font-size: 36px;
20  }
```

Selectors

There are infinite combinations of selectors for targeting elements. They are categorised as:

- Simple
- Combinator
- Pseudo-class
- Pseudo-element
- Attribute

```
/* Simple Selector */
body {
    color: #fff;
}

/* Combinator Selector */
div p {
    color: #fff;
}

/* Pseudo-class Selector */
a:hover {
    color: #fff;
}

/* Pseudo-element Selector *,
p::first-line {
    color: #fff;
}

/* Attribute Selector */
input[type="text"] {
    color: #fff;
}
```

Simple Selectors

Simple selectors refer to elements based on an identifier, such as type, class, or ID.

ID (prefix #) is a unique value assigned to an HTML element. It cannot be shared by other elements (hence unique), making it easy to isolate and target.

Class (prefix .) is a category value assigned to zero or more elements. It can be shared, and all categorised elements will be targeted by this type of selector.

```css
/* Simple Element Selector */
body {
    color: #00ff00;
}

/* Simple ID Selector */
#myUniqueID {
    color: #ff0000;
}

/* Simple Class Selector */
.myClass {
    color: #0000ff;
}
```

CSS Colour

Colours can be modified with styles.
Common colour styles are:

- Color (foreground colour, like text)
- Background-color
- Border-color

```
p {
    color: #fff;
    background-color: #ff5555;
    border-color: #0000ff;
}
```

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

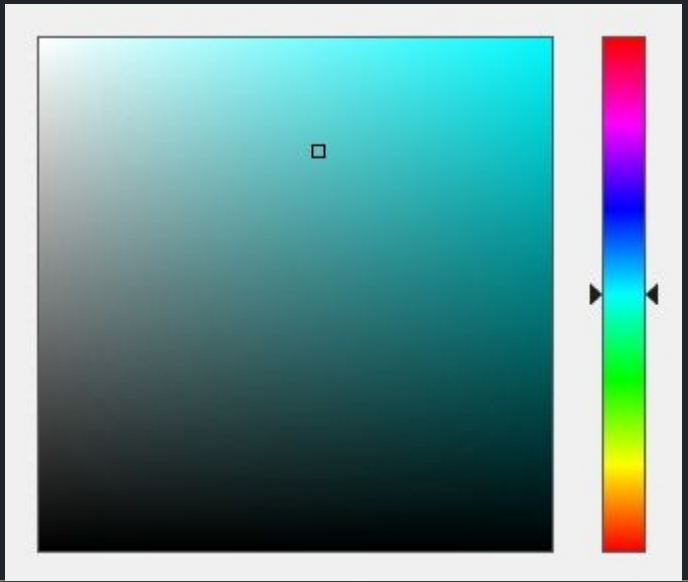
CSS
Colour values

Colours can be represented with the following formats:

- Colour name
- Hex value shorthand
- Hex value longhand
- RGB channels
- HSL channels

RGB can be replaced with RGBA to access alpha channel (opacity)

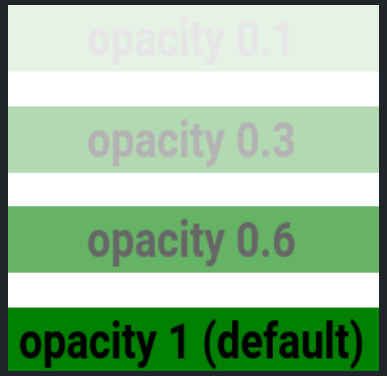
```
p {
    color: white;
    color: #fff;
    color: #ffffff;
    color: rgb(255, 255, 255);
    color: hsl(0, 100%, 100%)
}
```



CSS Background

Element backgrounds can be modified with:

- Background-color
- Background-image
- Background-repeat
- Background-attachment
- Background-position
- Background (inline shorthand)



CSS
Box Model

Every element is composed of four boxes:

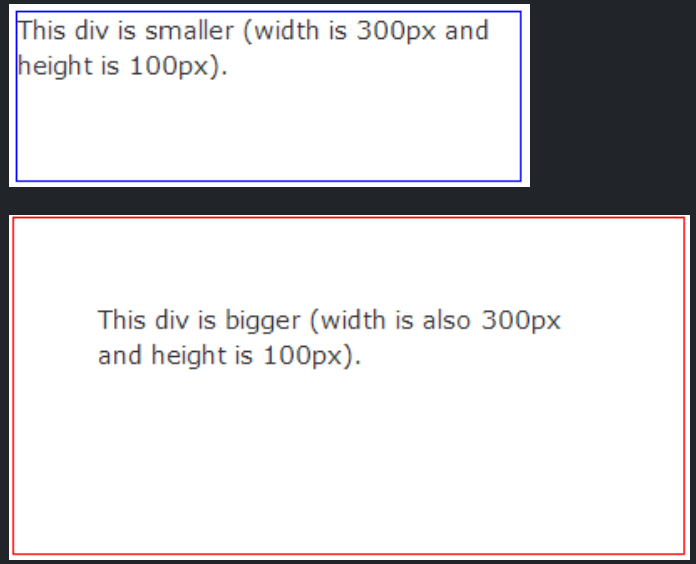
- Content: The contents of the element is represented with this box
- Padding: The gap between the content and the border
- Border: The ring that identifies the bounds of an element
- Margin: The gap between an element's bounds and the next element

CSS
Box Model

By default element width and height is calculated like this:

- Content width + padding + border = width
- Content height + padding + border = height

We can make this calculation easier in practise by using the boxing-sizing: border-box trick. This moves the border inside the element bounds, removing it from the calculation.



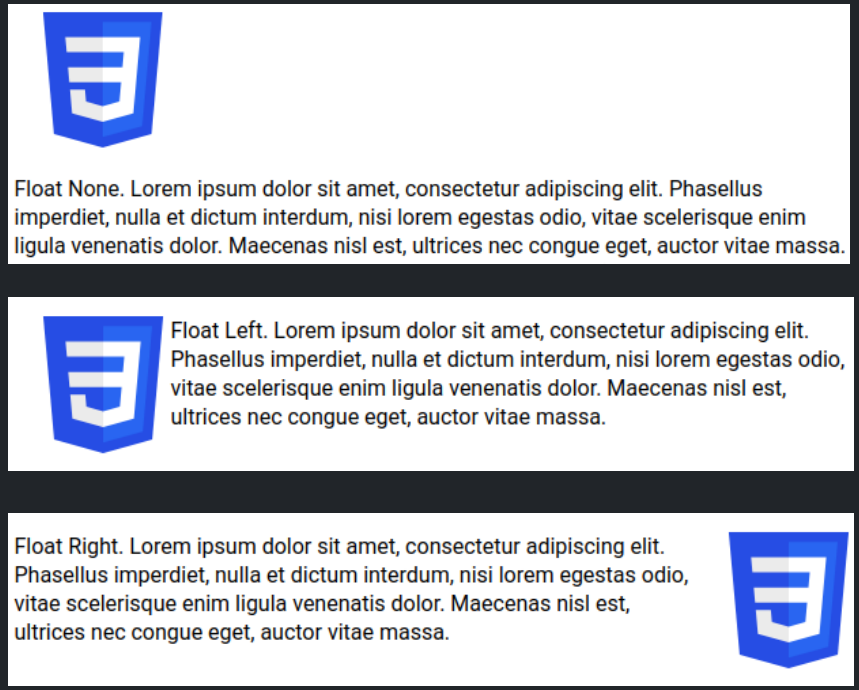

```
html {
    box-sizing: border-box;
}
*, *:before, *:after {
    box-sizing: inherit;
}
```

CSS Float

Similar to word-wrap in MS Office applications, **float** lets us adjust how an element interacts with the elements around it.

- Float: left;
- Float: right;
- Float: none;
- Float: inherit;

Float can let us flow elements horizontally if they fit in the available window space.

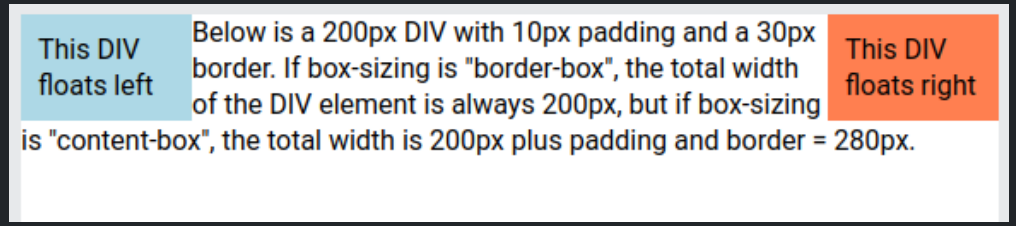


CSS
Float - Clear

Clear is a property that specifies whether or not an element should allow floated elements to interact with it.

- Clear: None;
- Clear: Left;
- Clear: Right;
- Clear: Both;
- Clear: Inherit

Combining **Float** and **Clear** provides a lot of flexibility controlling element layout in the box model.



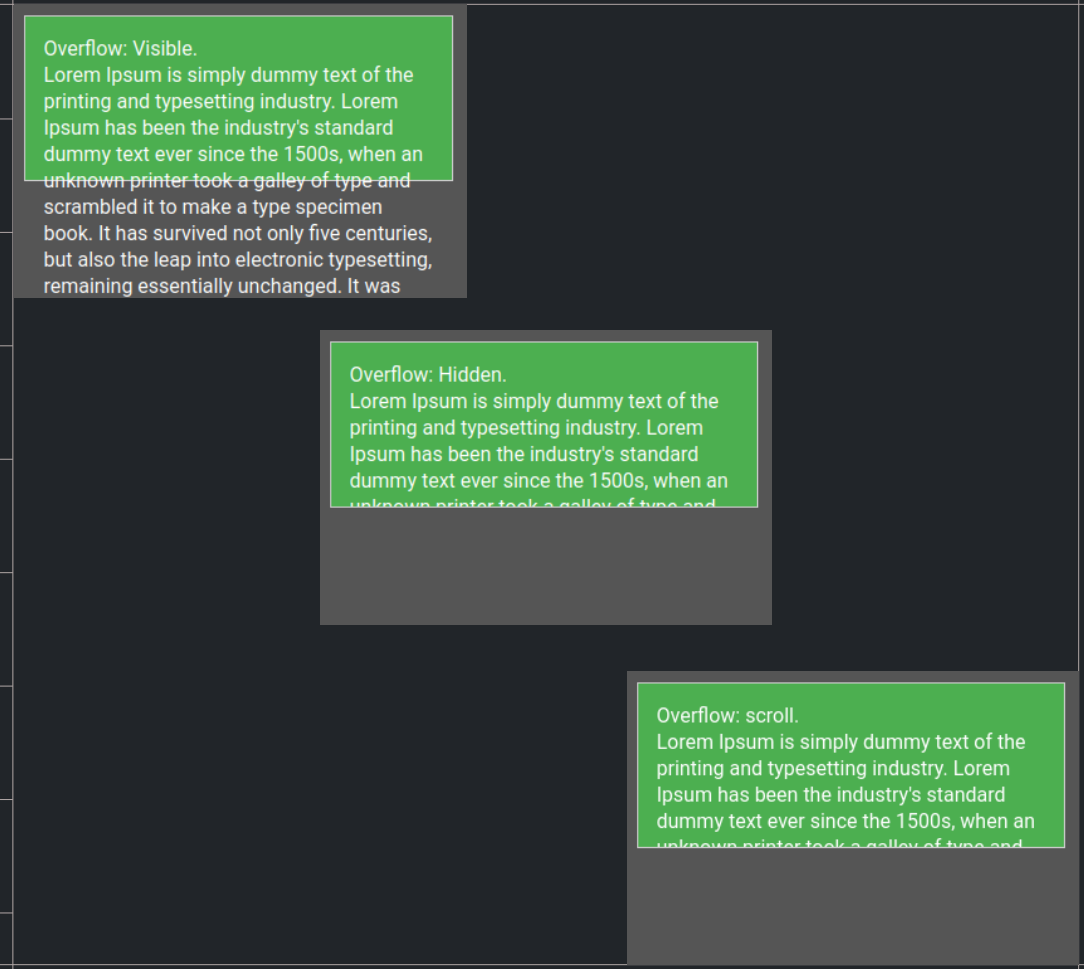

```css
#textDIV {
    height: 300px;
}

#blueDIV {
    background: lightblue;
    width: 80px;
    padding: 10px;
    float: left;
}

#redDIV {
    background: coral;
    width: 80px;
    padding: 10px;
    float: right;
}
```

CSS Overflow

The **overflow** property defines how overlapping or out-of-bounds container content is handled.

- **Overflow: visible**. Default behaviour, left content be as large as it wants to be
- **Overflow: hidden**. Clip the content to the bounds of the container
- **Overflow: scroll.** Hidden but with scrollbars to access clipped content.
- **Overflow: auto**. Same as scroll but only when neccessary.

Overflow only works for blocks with a specified height. Otherwise content will simply expand the content bounds vertically.



CSS Animations