Cascading Style Sheet (CSS)

Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation of a document written in HTML.

Types of CSS

- Inline CSS

- Internal CSS

- External CSS

Inline CSS

styling properties of a tag inserted using style attribute.

Example:

<div style=”background-color: red;”>

<p> This has a red background </p>

</div>

Internal CSS

styling properties written inside style tags using css selectors

css selec tors ??

defines the patterns to select elements to which a set of CSS rules are then applied along with their specificity.

Types of selectors

- type selector

- class selectors

- id selectors

Example:

<style>

h2{

border-width: 2 px;

border-radius: 10px;

background-color: green;

}

.paragraph{

background-color: blue;

border-width: 2px;

border-style: dashed;

padding: 10px;

color: red;

}

#special{

background-image: url(“image.png”);

border: 2px solid red;

font-size: 30px;

color: green;

}

</style>

<body>

<h2> This heading has green background</h2>

<p class=”paragraph”> This is a paragraph and this paragraph has red fonts</p>

<p id=”special”> This is an example with id… This has red border</p>

</body>

External CSS

CSS written in separate file and imported within head section of HTML document.

File must have .css extension.

Example:

<head>

<link rel=”stylesheet” href=”style.css”/>

</head>

<body>

<h2> This heading has green background</h2>

<p class=”paragraph”> This is a paragraph and this paragraph has red fonts</p>

<p id=”special”> This is an example with id… This has red border</p>

</body>

CSS file ::

h2{

border-width: 2 px;

border-radius: 10px;

background-color: green;

}

.paragraph{

background-color: blue;

border-width: 2px;

border-style: dashed;

padding: 10px;

color: red;

}

#special{

background-image: url(“image.png”);

border: 2px solid red;

font-size: 30px;

color: green;

}

Basic css properties:

color => rgb, hex, name, hsl

border =>

height, width => size

text => color, alignment, decoration, transform, spacing, text shadow

font=> family, size, weight, style

comments, quotes, address, emphasize, strong, sup, sub, article, section, font, span, hr, form, input, button, legend, caption

Form

label

input

select

textarea

button

fieldset

legend

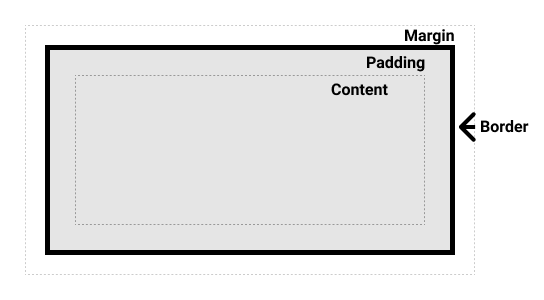
datalist

output

option

Header

CSS Box Model



The CSS box model as a whole applies to block boxes and defines how the different parts of a box — margin, border, padding, and content — work together to create a box that you can see on a page.

Parts of a box

- Content box: The area where your content is displayed; size it using properties like inline-size and block-size or width and height.

- Padding box: The padding sits around the content as white space; size it using padding and related properties.

- Border box: The border box wraps the content and any padding; size it using border and related properties.

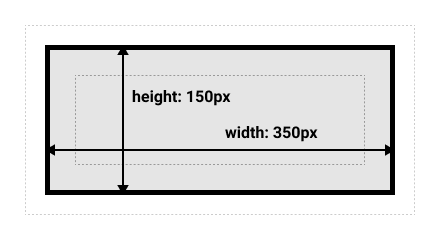
- Margin box: The margin is the outermost layer, wrapping the content, padding, and border as white-space between this box and other elements; size it using margin and related properties.

The alternative CSS box model

.box {

box-sizing: border-box;

}



CSS Flexbox

display: flex;

justify-content;

align-items;

flex-wrap;

flex-direction;

flex;

gap;

CSS Grid

display: grid;

grid-template-columns;

grid-template-rows;

gap;

justify-items;

justify-self;

grid-column: 1/ span 2;

grid-row: 1/ span 2;

Media Query

@media only screen and (max-width: 600px){

……….

}

**Javascript**

JavaScript is a powerful programming language that can add interactivity to a website.

**Advantages of JavaScript**

The merits of using JavaScript are −

Less server interaction − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.

Immediate feedback to the visitors − They don't have to wait for a page reload to see if they have forgotten to enter something.

Increased interactivity − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.

Richer interfaces − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

**Limitations of JavaScript**

Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.

JavaScript cannot be used for networking applications because there is no such support available.

JavaScript doesn't have any multi-threading or multiprocessor capabilities.

**Syntax**

<script language = "javascript" type = "text/javascript">

JavaScript code

</script>

**JavaScript has 8 Datatypes**

1. String – “ajkgsadjk789%”

2. Number – 1213456- 64bits --357623

3. Bigint- JavaScript BigInt is a new datatype (ES2020) that can be used to store integer values that are too big to be represented by a normal JavaScript Number.

4. Boolean- true or false

5. Undefined -

6. Null -

7. Symbol

8. Object- {

a: “Name”,

b: 3,

c: undefined,

}

**The Object Datatype**

The object data type can contain:

1. An object

2. An array – [“one”, “two”, “three”]- string-

[{a: “Name”,

b: 3,

c: undefined, },

{a: “Name2”,

b: 32,

c: undefined,

d: “ggg”}]

3. A date- Date( )

Arithmetic Operators => +, -, \*, /, %, \*\*, ++, --

Assignment Operators => =, +=, -=, \*=, /=, \*\*=, %=,

Comparison Operators => ==, ===, >, <, >=, <=, ?:

String Operators => +, >, <

Logical Operators => &&, ||, !=

Ternary Operators => ?:

Type Operators => typeof, instanceof

{

a: string;

b: number;

}

**Conditional Statements**

**if**

EXAMPLE

if (10 > 5) {

var outcome = "if block";

}

outcome;

OUTPUT

"if block"

Here’s what’s happening in the example above:

The keyword if tells JavaScript to start the conditional statement.

(10 > 5) is the condition to test, which in this case is true — 10 is greater than 5.

The part contained inside curly braces {} is the block of code to run.

Because the condition passes, the variable outcome is assigned the value "if block".

**Else**

EXAMPLE

if ("cat" === "dog") {

var outcome = "if block";

} else {

var outcome = "else block";

}

outcome;

OUTPUT

"else block"

In the example above, "cat" and "dog" are not equal, so the else block runs and the variable outcome gets the value "else block".

Else if

EXAMPLE

if (1st condition) {

var outcome = "if block";

} else if (2nd condition) {

var outcome = "else if block";

} else {

var outcome = "else block";

}

outcome;

OUTPUT

"else if block"

You can use multiple if else conditionals, but note that only the first else if block runs. JavaScript skips any remaining conditionals after it runs the first one that passes.

EXAMPLE

a=5

if (a>10) {

var outcome = "if block";

} else if (a>4) {

var outcome = "first else if block";

} else if (a>3) {

var outcome = "second else if block";

} else {

var outcome = "else block";

}

outcome;

OUTPUT

"first else if block"

**Looping Statements**

The JavaScript loops are used to iterate the piece of code using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

There are four types of loops in JavaScript.

for loop

while loop

do-while loop

**1) JavaScript For loop**

The JavaScript for loop iterates the elements for the fixed number of times. It should be used if number of iteration is known. The syntax of for loop is given below.

for (initialization; condition; increment)

{

code to be executed

}

<script>

for (i=1; i<=10; i++)

{

document.write(i + "<br/>")

}

</script>

Output:

1

2

3

4

5

**2) JavaScript while loop**

The JavaScript while loop iterates the elements for the infinite number of times. It should be used if number of iteration is not known. The syntax of while loop is given below.

while (condition)

{

code to be executed

}

<script>

var i=11;

while (i<=15)

{

document.write(i + "<br/>");

i++;

}

</script>

Output:

11

12

13

14

15

**3) JavaScript do while loop**

The JavaScript do while loop iterates the elements for the infinite number of times like while loop. But, code is executed at least once whether condition is true or false. The syntax of do while loop is given below.

do{

code to be executed

}while (condition);

<script>

var i=21;

do{

document.write(i + "<br/>");

i++;

}while (i<=25);

</script>

Output:

21

22

23

24

25

**FUNCTIONS**

JavaScript functions are reusable blocks of code that perform a specific task, taking some form of input and returning an output.

To define a function, you must use the function keyword, followed by a name, followed by parentheses ( ). Then you have to write the function logic between curly brackets { }

EXAMPLE

function addTwoNumbers(x, y) {

return x + y;

}

- “Function” is the keyword required to actually start declaring a function

- “addTwoNumbers” is the function’s name, which is customizable. Function names can contain letters, numbers, and certain special characters, just like variables.

- (x, y) are parameters, which are variable names for the inputs the function will accept. These parameters are also referred to as arguments.

- “Return” is the keyword that exits the function and shares an optional value outside.

- The code that the function will execute must be put inside of curly brackets { }

EXAMPLE

function greetThePlanet() {

return "Hello world!";

}

greetThePlanet();

OUTPUT

"Hello world!"

Arguments and parameters

The values that are declared within a function when the function is called are known as an argument. The variables that are defined when the function is declared are known as parameters.

EXAMPLE

function square(number) {

return number \* number;

}

square(16);

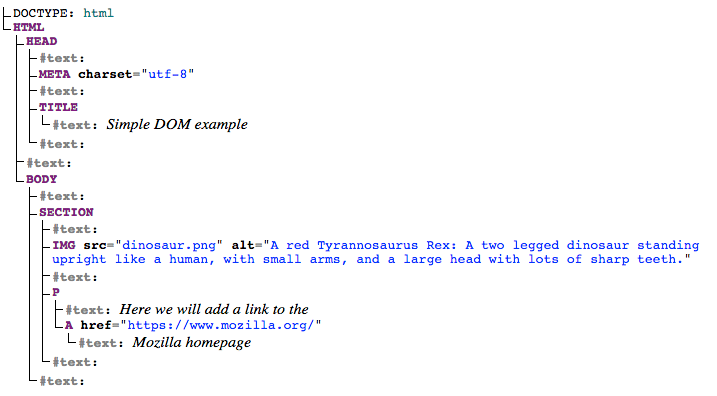
square(20);

OUTPUT

256

**What is the DOM?**

The Document Object Model (DOM) is a programming interface for web documents. It represents the structure of a document as a tree of objects, where each object corresponds to a part of the document, such as elements, attributes, and text. JavaScript can manipulate this tree structure, allowing developers to dynamically alter the content and appearance of a webpage.



**How to access DOM elements**

To manipulate the DOM, we need to access its elements. This is commonly done using the document object, which represents the entire HTML document. Let's look at a simple example:

// Accessing an element by its ID

const headerElement = document.getElementById('header');

<h1 id=’header’> This is header element</h1>

// Accessing elements by class name

const paragraphs = document.getElementsByClassName('paragraph');

<p class=”paragraph”> This is a paragraph</p>

// Accessing elements by tag name

const images = document.getElementsByTagName('img');

const headings = document.getElementsByTagName(‘h1’);

**How to modify element content**

Once we have access to an element, we can modify its content using the innerHTML property:

// Modifying the content of an element

headerElement.innerHTML = 'New Header Text';

**Events and Event Handling**

Events are actions or occurrences that happen in the browser, such as a user clicking a button or resizing the window. JavaScript allows us to handle these events and execute code in response. Event handling is a crucial aspect of creating interactive web pages.

**How to add event listeners**

To respond to events, we can use event listeners. These are functions that "listen" for a specific event on a particular element. Let's consider a button click event as an example:

// Accessing a button element

const myButton = document.getElementById('myButton');

// Adding a click event listener

myButton.addEventListener('click', function() {

alert('Button Clicked!');

});

**How to Manipulate Styles**

DOM manipulation also extends to modifying the styles of elements, allowing us to create visually appealing and dynamic web pages.

**How to change styles dynamically**

We can use the style property of an element to change its appearance. Let's take an example of changing the color of a paragraph when a button is clicked:

// Accessing a paragraph element

const myParagraph = document.getElementById('myParagraph');

// Accessing a button element

const colorButton = document.getElementById('colorButton');

// Adding a click event listener to the button

colorButton.addEventListener('click', function() {

// Changing the color style of the paragraph

myParagraph.style.color = 'blue';

**How to create new elements**

The createElement method is used to create a new HTML element. Let's create a new paragraph element and append (add) it to the body of the document:

// Creating a new paragraph element

const newParagraph = document.createElement('p');

// Setting the text content of the new paragraph

newParagraph.textContent = 'This is a new paragraph.';

// Appending the new paragraph to the body of the document

document.body.appendChild(newParagraph);

**How to modify attributes**

We can also modify the attributes of existing elements. Let's consider changing the source of an image dynamically:

// Accessing an image element

const myImage = document.getElementById('myImage');

<img id=”myImage” src=”images.jpg” alt=”attr”/>

// Changing the source attribute of the image

myImage.src = 'new-image.jpg';

**How to update form input values**

Let's consider a scenario where you want to update the value of a text input based on user interaction:

// Accessing a text input element

const myInput = document.getElementById('myInput');

// Adding an input event listener

myInput.addEventListener('input', function() {

// Updating a paragraph with the input value

document.getElementById('inputValue').textContent = myInput.value;

**Differences between Git and GitHub**

Git is a version control system that manages and keeps track of your code. GitHub, on the other hand, is a service that let you host, share, and manage your code files on the internet.

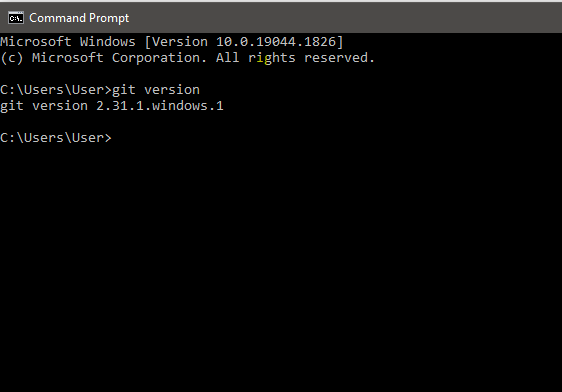
GitHub uses Git underneath, and lets you manage your Git repositories or folders easily on its platform.

So Git is the actual version control system and GitHub is the platform where you host your code.

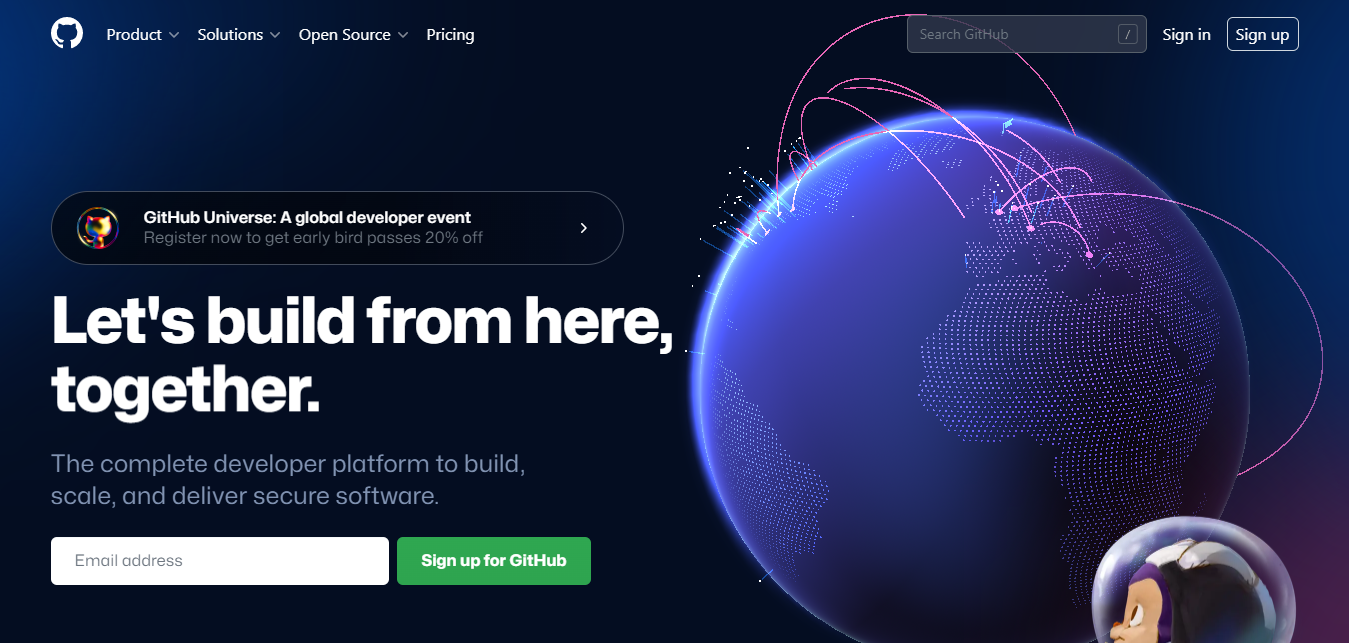
**How to Start Using Git and GitHub**

Step 1 – Install Git

Git comes preinstalled in some Macs and Linux-based systems, but you can always check if you have Git installed in your machine by typing git version in your terminal. You can use the Command Prompt to do this.



<https://git-scm.com/download>  
  
**Step 2 – Create a GitHub Account.**



To create an account on GitHub, you will be asked for some personal information like name, confirm your email, set a username and password, and your account should be set up in minutes.