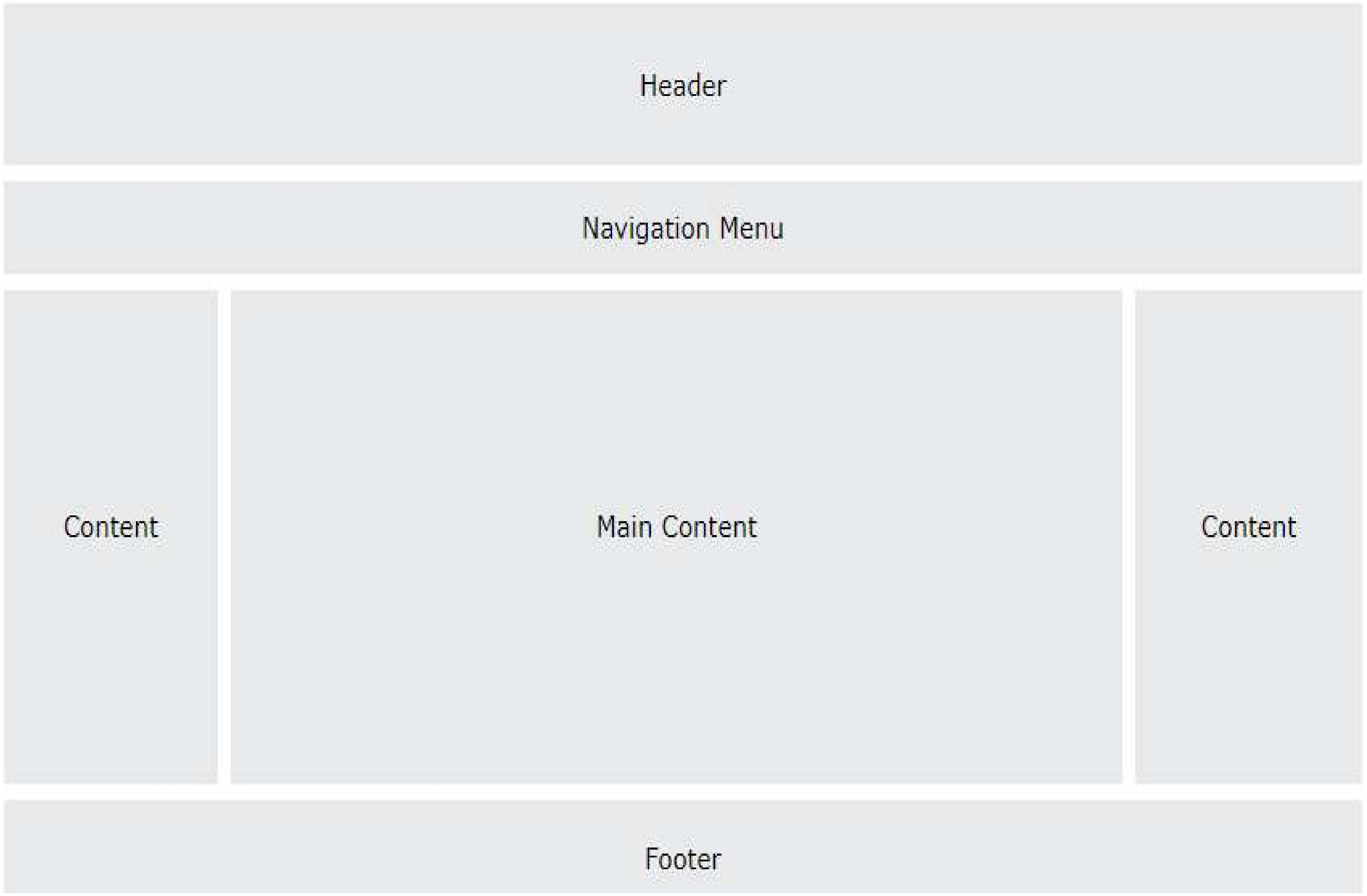
Lec- 6

- Introduction To CSS Layout


Header
Navigation Menu
Content
Main Content
Content
Footer

CSS page layout techniques allow us to take elements contained in a web page and control where they're positioned relative to the following factors :

- ➢ their **default position in normal layout flow.**
- ➢ the o**ther elements around them,** their **parent container**,

and the main **viewport/window**

The page layout techniques we'll be covering in more detail in this module are:

- •Normal flow
- •The display property
- •Flexbox
- •Grid
- •Floats
- •Positioning
- •Table layout
- •Multiple-column layout

Each technique has its uses,dengised si euqinhcet oN .**segatnavdasid dna ,segatnavda**
to be used in **isolation** a ni eb ll'uoy rof dengised si dohtem tuoyal hcae tahw gnidnatsrednu yB .
.ksat hcae rof etairporppa tsom si dohtem hcihw dnatsrednu ot noitisop doog

The display property

➢The main methods for achieving page layout in CSS all involve specifying values for the display property. This property allows us to change the default way so
mething displays. Everything in normal flow has a def ault value for display;
i.e., a default way that elements are set to behave .
For example, the fact that paragraphs in English display one below the other is because they are styled with display:

➢ block. If you create a link around some text inside a paragraph,
that link remains inline with the rest of the text, and doesn't break onto a new line.

➢ This is because the <a> element is display: inline

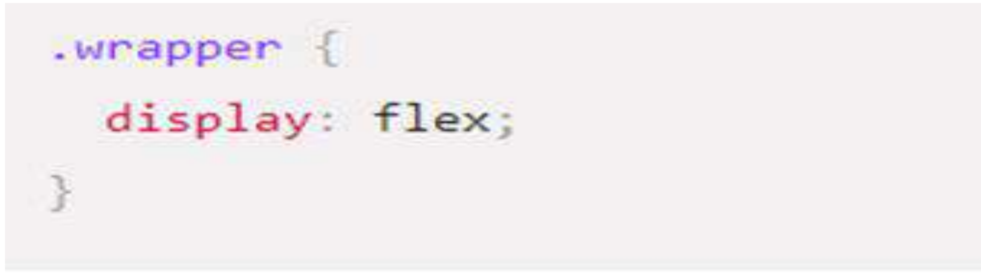
You can change this default display behavior. For example, the <li> element is display: ➢ block by default, meaning that list items display one below the other in our English document. If we were to change the display value to inline they would display next to each other, as words would do in a sentence. The fact that you can change the value of display for any element means that you can pick HTML elements for their semantic meaning without being concerned

Flexbox

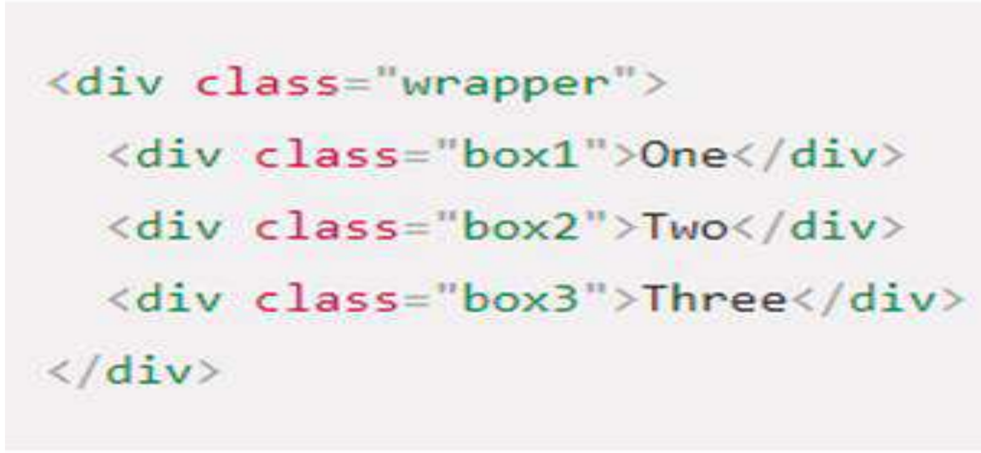
- Flexbox is the short name for the Flexible Box
Layout CSS module, designed to make it easy for us to
either as a row or as —lay things out in one dimension
a column. To use flexbox, you apply display: flex
- to the parent element of the elements you want to lay
out; all its direct children then become flex items. We
.can see this in a simple example

Setting display: flex

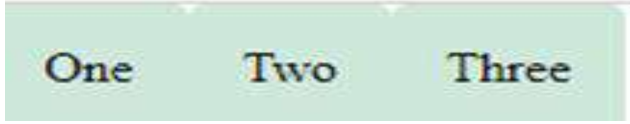
- The HTML markup below gives us a containing element
with a class of wrapper, inside of which
- are three <div> elements. By default these would display
as block elements, that is, below one another in our
.English language document



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



```html
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
</div>
```



Floats

➢Floating an element changes the behavior of that element and the block level elements that follow it in normal flow.

➢ The floated element is moved to the left or right and removed from normal flow , and the surrounding content floats around it.

➢The float property has four possible values:

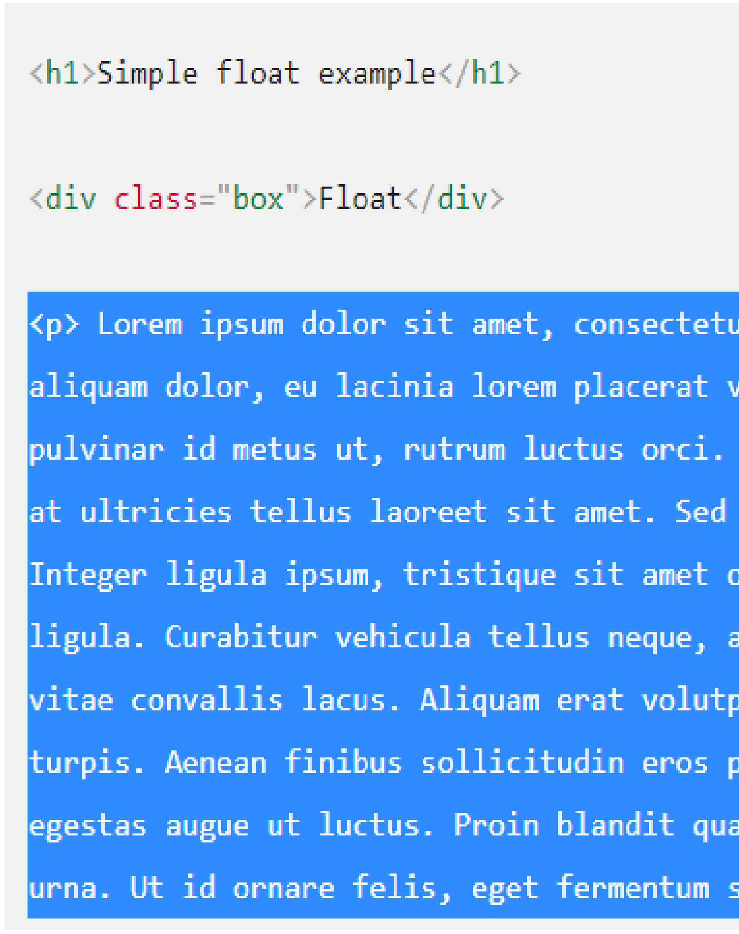
➢left + right —Floats the element to the left or right.

➢none —Specifies no floating at all. This is the default value.

➢inherit —Specifies that the value of the float property should be

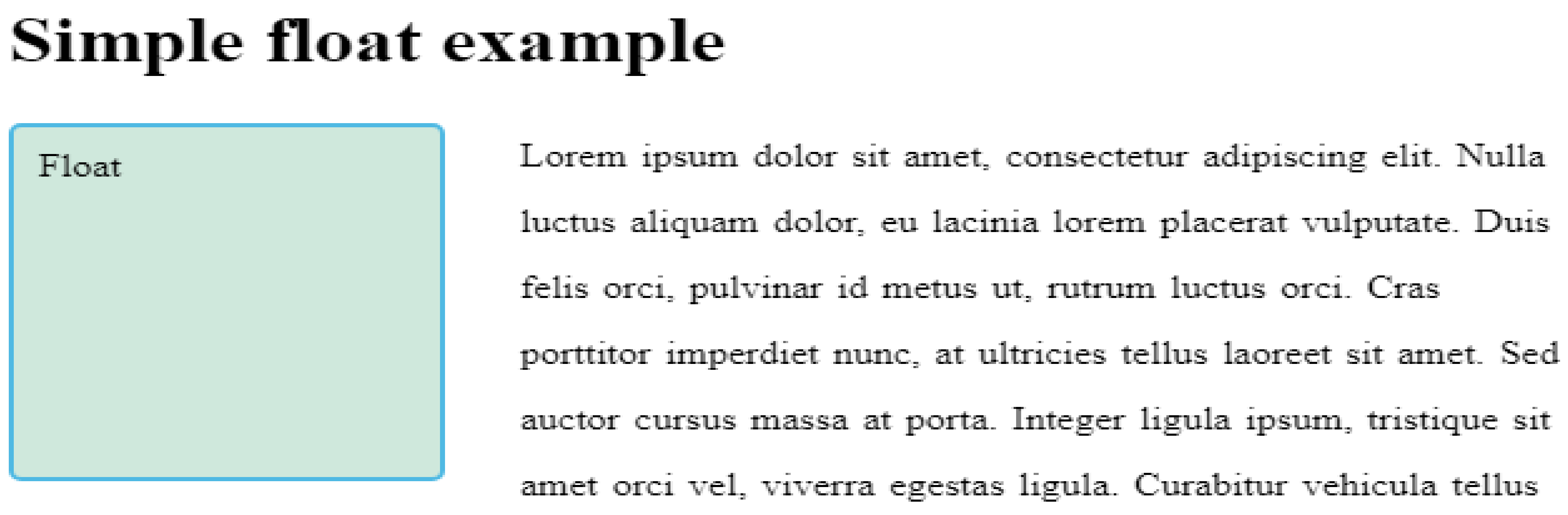
➢ inherited from the element's parent element.

In the example below, we float a <div> left and give it a margin on ➢
the right to push
the surrounding text away from it. This gives us th ➢
e effect of text wrapped around the boxed element, and is most of ➢
.what you need to know about floats as used in modern web design



```
<h1>Simple float example</h1>

<div class="box">Float</div>

<p> Lorem ipsum dolor sit amet, consectetu
aliquam dolor, eu lacinia lorem placerat v
pulvinar id metus ut, rutrum luctus orci.
at ultricies tellus laoreet sit amet. Sed
Integer ligula ipsum, tristique sit amet c
ligula. Curabitur vehicula tellus neque, a
vitae convallis lacus. Aliquam erat volutp
turpis. Aenean finibus sollicitudin eros p
egestas augue ut luctus. Proin blandit qua
urna. Ut id ornare felis, eget fermentum s
```

```
.box {
    float: left;
    width: 150px;
    height: 150px;
    margin-right: 30px;
}
```



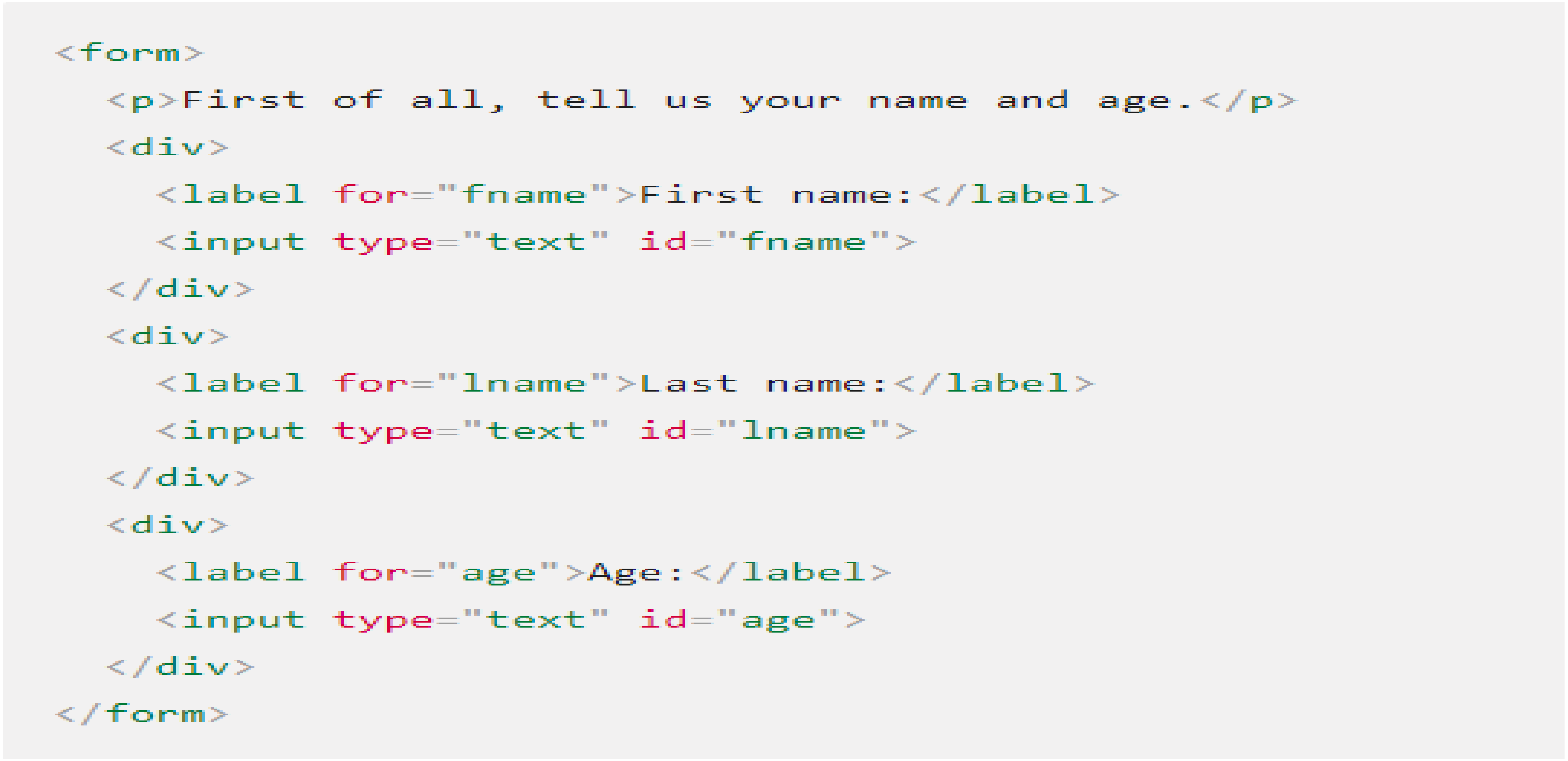
Simple float example

Float

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus

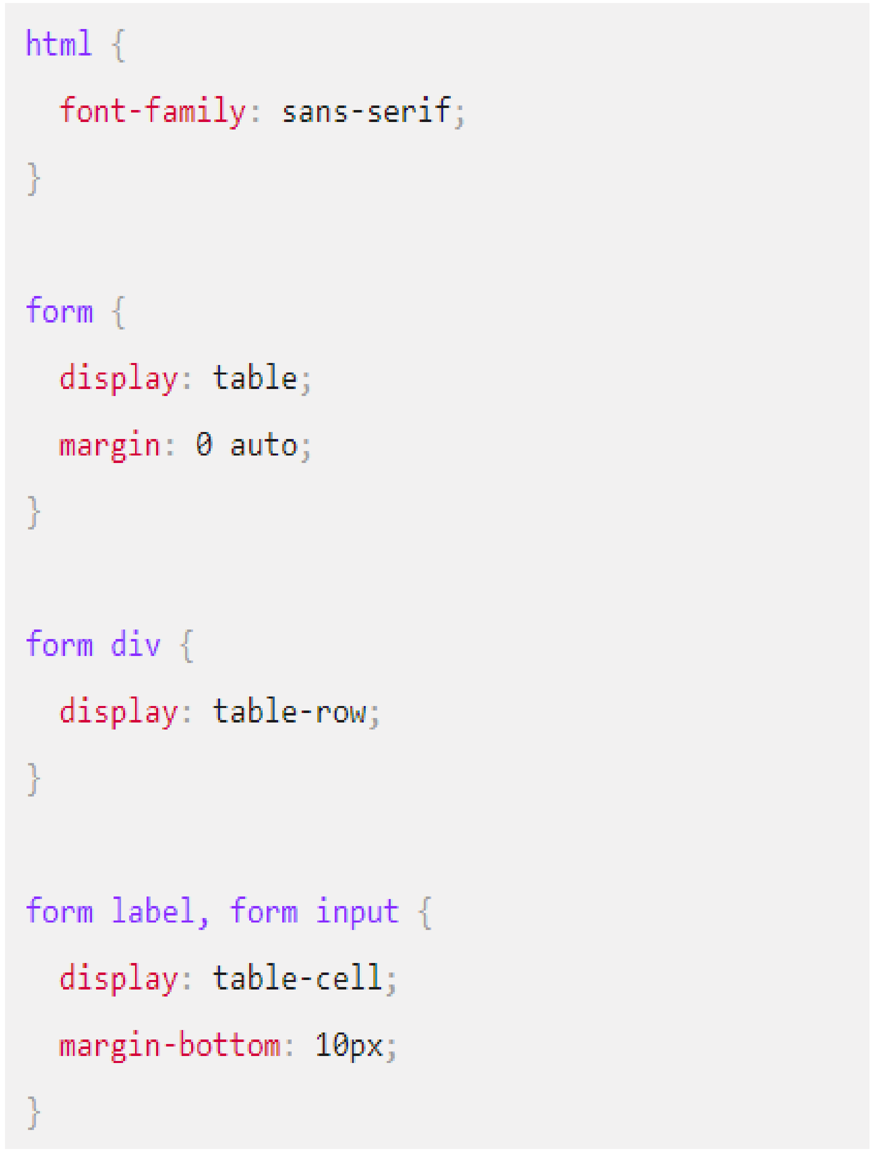
Table layout

- ➢HTML tables are fine for displaying tabular data, but many years ago —before even basic CSS was supported reliably across browsers web developers used to also use tables for entire web page layouts, putting their headers, footers, columns, etc .
into various table rows and columns.
This worked at the time, but it has many problems: table layouts are inflexible ,
- ➢very heavy on markup, difficult to debug, and semantically wrong (e.g., screen reader users have problems navigating table layouts).
- ➢The way that a table looks on a webpage when you use table markup is due to a set of CSS properties that define table layout .
- ➢These same properties can also be used to layout elements that aren't tables, a use which is sometimes described as "using CSS tables."
- ➢The example below shows one such use. It must be noted, using CSS tables for layout should be considered a legacy method at this point, for those situations where you have very old browsers that lack support for Flexbox or Grid.
- ➢Let's look at an example. First, some simple markup that creates an HTML form. Each input element has a label, and we've also included a caption inside a paragraph. Each label/input pair is wrapped in a <div> for layout purposes.



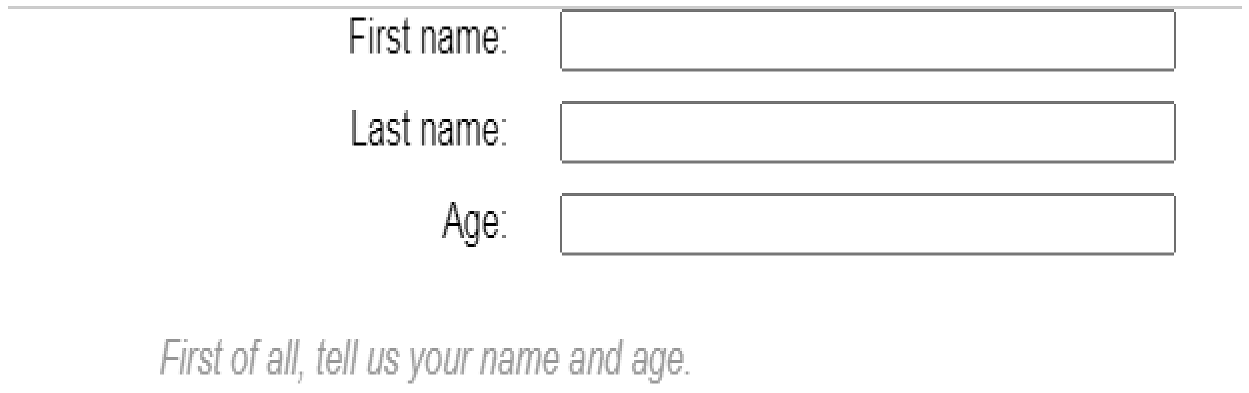
```html
<form>
  <p>First of all, tell us your name and age.</p>
  <div>
    <label for="fname">First name:</label>
    <input type="text" id="fname">
  </div>
  <div>
    <label for="lname">Last name:</label>
    <input type="text" id="lname">
  </div>
  <div>
    <label for="age">Age:</label>
    <input type="text" id="age">
  </div>
</form>
```

- As for the CSS, most of it's fairly ordinary except for the uses of the display property.
- The <form>, <div>s, and <label>s and <input>s have been told to
- display like a table, table rows, and table cells respectively .
- Basically, they'll act like HTML table markup, causing the labels and inputs to line up nicely
- by default. All we then have to do is add a bit of sizing, margin, etc., to make everything look
- a bit nicer and we're done.
- You'll notice that the caption paragraph has been given display: table-caption ,;
- which makes it act like a table <caption>, and caption-side: bottom; to tell the caption to
- sit on the bottom of the table for styling purposes, even though the markup
- is before the <input> elements in the source. This allows for a nice bit of flexibility.



```css
html {
  font-family: sans-serif;
}

form {
  display: table;
  margin: 0 auto;
}

form div {
  display: table-row;
}

form label, form input {
  display: table-cell;
  margin-bottom: 10px;
}
```

```
form label {
  width: 200px;
  padding-right: 5%;
  text-align: right;
}

form input {
  width: 300px;
}

form p {
  display: table-caption;
  caption-side: bottom;
  width: 300px;
  color: #999;
  font-style: italic;
}
```



First name:

Last name:

Age:

First of all, tell us your name and age.

Grids

CSS Grid Layout is a two-dimensional layout system for the web. It lets you lay content out in rows and columns ,
and has many features that make building complex layouts straightforward. This article will give you all you need
to know to get started with page layout, then test your grid skills before moving on.

Floats

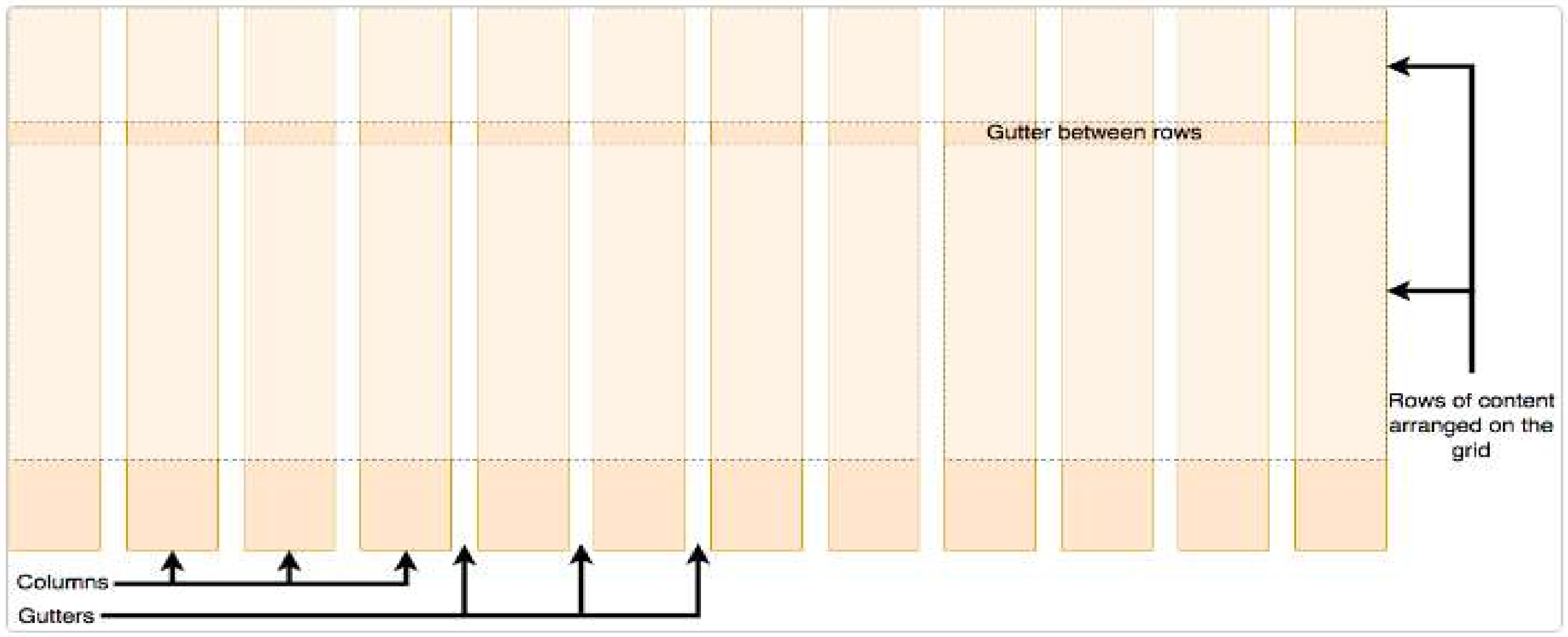
Originally for floating images inside blocks of text, the float eht fo eno emaceb ytreporp eht htiW .segapbew no stuoyal nmuloc elptilum gntiaerc rof sloot desu ylnommoc tsom won sah ti dirG dna xobxelF fo tnevda
returned to its original purpose, as this article explains.

Positioning

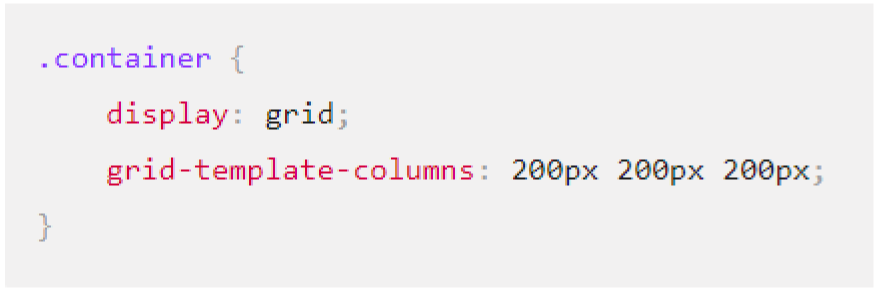
Positioning allows you to take elements out of the normal document layout flow and make them behave differently,
for example, by sitting on top of one another, or by always remaining in the same place inside the browser viewport .
This article explains the different position.meht esu ot woh dna seulav

What is grid layout?

- A grid is a collection of horizontal and vertical lines creating a pattern against which we can line up our design elements. They help us to create layouts in which our elements won't jump around or change width as we move from page to page, providing greater consistency on our websites.
- A grid will typically have **columns**, **rows**, and then gaps between each row and column. The gaps are commonly referred to as **gutters**.

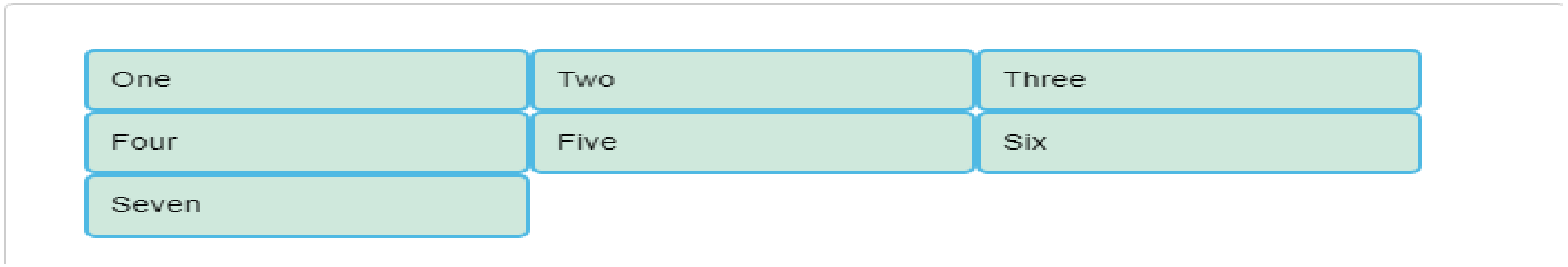


To see something that looks more grid-like, we'll need to add some columns to the grid. Let's add three 200-pixel columns. You can use any length unit or percentage to create these column tracks.



```
.container {
    display: grid;
    grid-template-columns: 200px 200px 200px;
}
```

Add the second declaration to your CSS rule, then reload the page. You should see that the items have rearranged themselves such that there's one in each cell of the grid.



Positioning allows you to take elements out of normal document flow and make them behave differently,
 for example, by sitting on top of one another or by always remaining in the same place inside the browser viewport.
 This article explains the different position.meht esu ot woh dna seulav

Static positioning

Static positioning is the default that every element gets. It just means “put the element into its normal position in the document flow —nothing special to see here".
To see this (and get your example set up for future sections) first add a class fo positioned ni <p> dnoces eht ot :LMTH eht

```
<p class="positioned"> ... </p>
```

Now add the following rule to the bottom of your CSS:

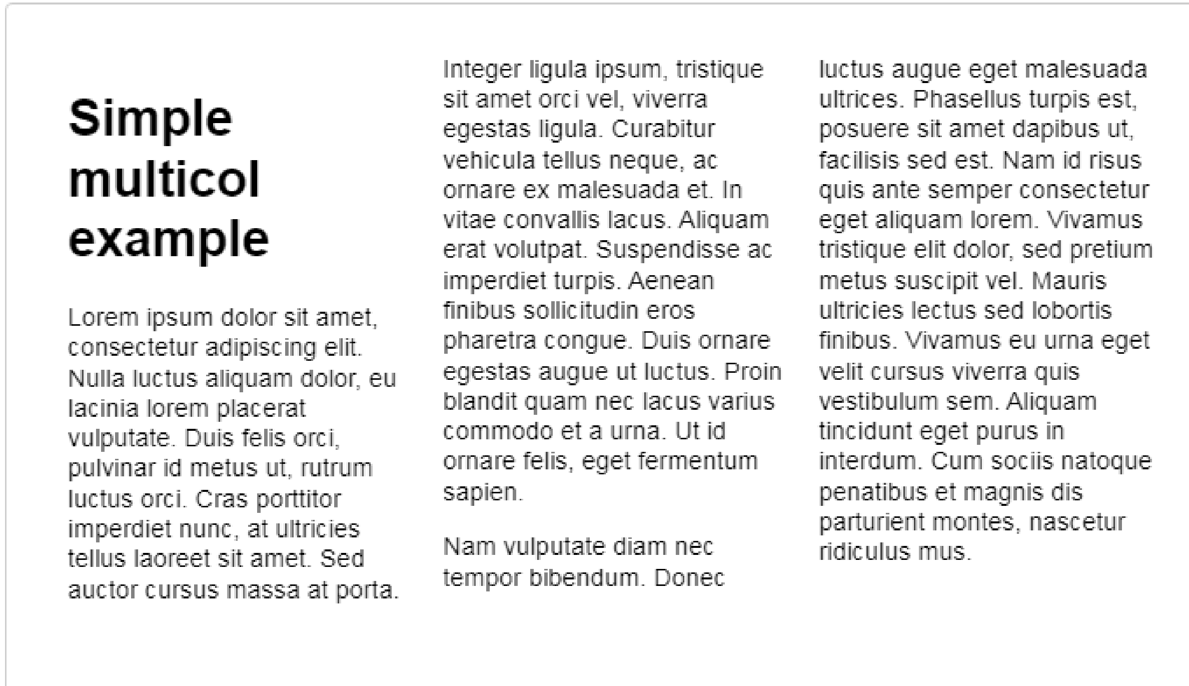
```
positioned { position: static; background: yellow }.
```

If you save and refresh, you'll see no difference at all, except for the updated background color of the paragraph. This is fine —as we said before, static positioning is the default behavior!

Our starting point file contains some very simple HTML: a wrapper with a class of container, s dna gnidaeh a si hcihw fo edisni
ome paragraphs.
The<div> with a class of container will become our multicol container. We enable multicol by using one of two properties :
column-count ro column-width
The column-count sa rebmun a sekat ytreporp fI .snmuloc fo rebmun taht setaerc dna eulav sti
uoy
add the following CSS to your stylesheet and reload the page, you'll get three columns:

```
.container {
  column-count: 3;
}
```

The columns that you create have flexible widths — the browser works out how much spac
to assign each column.



Simple multicol example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat vulputate. Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem. Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus. Vivamus eu urna eget velit cursus viverra quis vestibulum sem. Aliquam tincidunt eget purus in interdum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Setting column-width

Change your CSS to use column-width:swollof sa

```
.container { column-width: 200px; }
```

The browser will now give you as many columns as it can of the size that you specify; any remaining space is then shared between the existing columns.

This means that you won't get exactly the width that you specify unless your container is exactly divisible by that width.

Styling the columns

.The columns created by multicol cannot be styled individually

There's no way to make one column bigger than other columns or to change the background or text color of a single column. You have two opportunities

:to change the way that columns display

.Changing the size of the gap between columns using the column-gap

.Adding a rule between columns with column-rule

.Using your example above, change the size of the gap by adding a column-gap property

.the property accepts any length unit —You can play around with different values

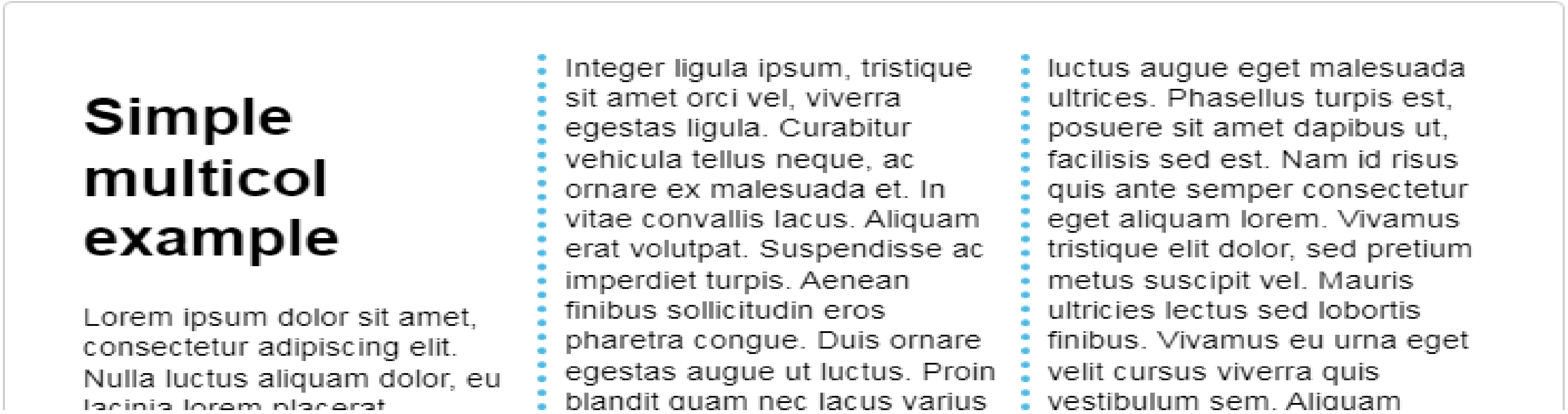
.Now add a rule between the columns with column-rule

,In a similar way to the border property that you encountered in previous lessons

,column-rule is a shorthand for column-rule-color, column-rule-style, and column-rule-width

```
.container {
  column-count: 3;
  column-gap: 20px;
  column-rule: 4px dotted rgb(79, 185, 227);
}
```

Try adding rules of different styles and colors.



Simple multicol example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla luctus aliquam dolor, eu lacinia lorem placerat

Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius

luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem. Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus. Vivamus eu urna eget velit cursus viverra quis vestibulum sem. Aliquam

Lec- 7

JavaScript1 Language Fundamental

JavaScript was initially created to *"make webpages alive"*.

- JavaScript is a high-level programming language
- Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web
- allow client-side script to interact with the user and make dynamic pages.
- All the modern browsers come with built-in support for JavaScript.

<u>In-browser JavaScript is able to:</u>

- Add new HTML to the page, change the existing content.
- modify styles.
- React to user actions, run on mouse clicks, pointer movements, key presses.
- download and upload files
- Get and set cookies, ask questions to the visitor, show messages.
- Remember the data on the client-side
- You can validate user input before sending the page off to the server

JavaScript - Syntax

- JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>**.
- JavaScript ignores spaces, tabs, and newlines
- Semicolons are **Optional**
- Case-sensitive language.

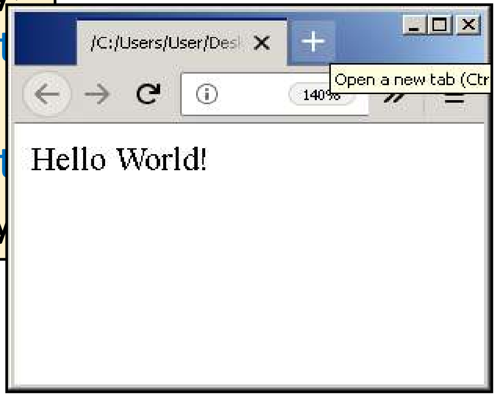
Note: Old JavaScript examples may use a type attribute:
<script type="text/javascript"> ... </script>
The type attribute is not required. JavaScript is the default scripting language in HTML.

Placement in HTML File

- Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.
- Scripts can also be placed in external files

JS: Hello World

```
<body>
<script
document.write("Hello World!
</script
</body
```



```
Hello World!
```

The document .write is a javascript command telling the browser that what follows within the parentheses is to be written into the document.

* JavaScript accepts both double and single quotes.

Values

- **Numbers**
  - Integers (whole numbers) e.g. 0, 1, 2, 3, 4
  - Floats (fractional numbers) e.g. 3.14
- **Strings**
  - `"A string of characters"`
  - `'This is also a string, but in single quotes'`
  - `"Escape quotes with backslash like this: \" ";`
- **Booleans**
  - `true`
  - `false`

Numeric operators

- Addition: 5 + 3
- Subtraction: 7 – 6
- Multiplication: 5.3 * 2.7
- Division: 20 / 4
- Modulo: 8 % 2
- Increment: 5++
- Decrement: 2--

Try evaluating some numeric expressions using the console

String operators

- Concatenation

```
// Evaluates to "this is a concatenated string"
"this is " + "a concatenated string"
```

- What happens if you add a number to a string?

```
"Here is a string with a number " + 7
```

Try it in the console!

Variables

Store a value with a name for reference elsewhere in the program

Declaration:

```
var myVariable;
```

Assignment statements:

```
myVariable = true;
```

Declaration and initial assignment:

```
var x = 0;
var myString = "This is a string";
```

Assignment shorthands

Shorthands for variable assignment include:

```
+=
-=
*=
/=
%=
```

```
x += 3
```

is equivalent to

```
x = x + 3
```

Experiment with variable declarations and assignment statements using the console

Statements

- Optionally separated by semi-colons
- Use curly braces { } to group statements into blocks

```
var radius = 3;
var circumference = 2 * Math.PI * radius;
console.log("result is " + circumference)
```

Comments

```
// This is a comment until the end of this line only
var aVariable = "Not a comment";

/*
*   This is  a comment spanning several lines,
*   until the star slash
*/
// Use comments to disable/enable statements
// var anotherVariable = "Disabled code";
```

Functions

- A block of statements that can be named and called
- Can take parameters e.g. radius
- Can perform an action or return a result (or both!)

```
function calculateCircumference (radius) {
  var circumference = 2 * Math.PI * radius;
  return circumference;
}

// The function is called elsewhere in the program, we
  pass in the value 3 for the radius
var myCircumference = calculateCircumference(3);
```

Built-in libraries

- Math.PI is a constant (a variable that never changes value) from the built-in Math library. Some additional Math fu
  - `Math.round(4.7)`
  - `Math.sqrt(9)`
  - `Math.max(1,5)`
  - `Math.min(6,7)`
  - `Math.floor(5.6)`
  - `Math.random()`
- `console.log()` is a built-in function (in Chrome) that prints values to the console

Experiment with these functions using the console

Comparison operators

- Expressions based on comparison operators evaluate to a boolean:
  - Equal:
    - 3.5 == 2 // (evaluates to false)
  - Not equal:
    - "aString" != "anotherString" // (true)
  - Greater than / (or equal):
    - 6 > 6 // (false)
    - 6 >= 6 // (true)
  - Less than / (or equal):
    - 5 < 3 // (false)
    - 5 <= 3 // (false)

Boolean operators

- Combine boolean expressions using logical operators:
  - AND
    `&&`
  - OR
    `||`
  - NOT
    `!`

Conditional statements

Implement alternative behaviours based on conditions

```
if (temperature < 20) {
  console.log("It is cold");
} else if (temperature >= 20 && temperature < 29) {
  console.log("It is warm");
} else {
  console.log("it is hot");
}
```

Loops

While loop

```
var maxLimit = 20, counter = 0, value = 0;
while (value != 6 && counter < maxLimit) {
  // ensure variables in loop condition can change
}
```

For loop

```
for (var i = 0; i < 10; i++){
  // print 0,1,2,3,4,5,6,7,8.9
  console.log(i);
}
```

Update the dice game to keep rolling until the result is 6.

Display the number of rolls it took to win.

Sample solution: http://jsfiddle.net/AnnaGerber/epaAs/2/

Arrays

An ordered list of values

```
var myArray = [1,6,10];
var anotherArray =
["first value", 5, "third value", false];

// Access values – indexed from 0
myArray[0] // 1
mnotherArray[2] // "third value"
```

What is the DOM?

- ***"The Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.“***
- **The DOM defines a standard for accessing documents.**

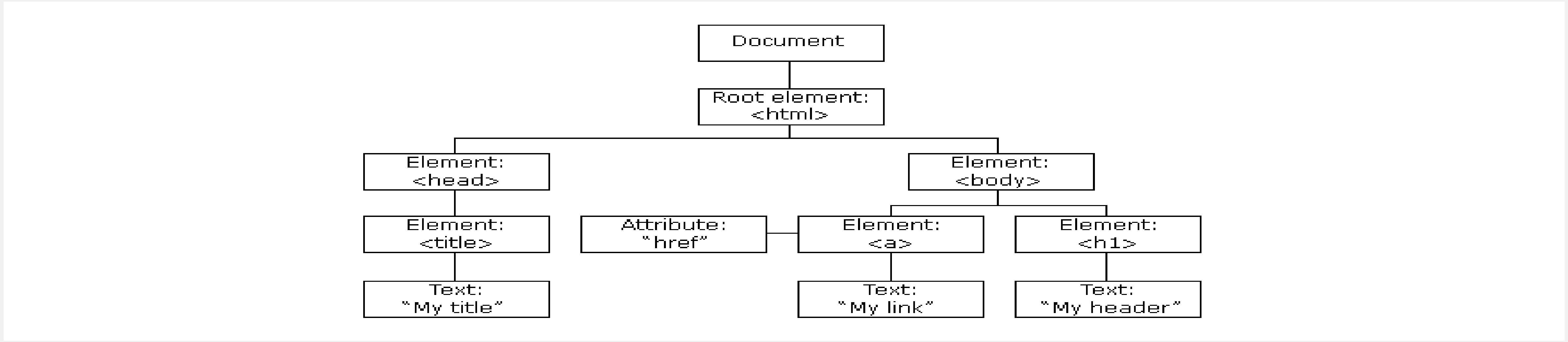
What is the HTML DOM?

- **The HTML DOM is a standard object model and programming interface for HTML. It defines:**
- **The HTML elements as objects**
- **The properties of all HTML elements**
- **The methods to access all HTML elements**
- **The events for all HTML elements**
- **In other words: The HTML DOM is a standard for how to get, access change, add, or delete HTML elements.**

HTML DOM (Document Object Model)

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The **HTML DOM** model is constructed as a tree of **Objects**:
- The HTML DOM Tree of Objects

The HTML DOM Tree of Objects



JavaScript - HTML DOM Methods

- **HTML DOM methods are actions you can perform (on HTML Elements).**
- **HTML DOM properties are values (of HTML Elements) that you can set or change.**
- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as **objects**.
- The programming interface is the properties and methods of each object.
- **A property is a value that you can get or set (like changing the content of an HTML element).**
- **A method is an action you can do (like add or deleting an HTML element).**

The following example changes the content (the innerHTML) of the <p> element with id="demo":

```
<html>
<body>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
</body>
</html>
```

JavaScript Can Change HTML element

- If you want to access/change any element in an HTML page, first you should find the specific element.

Method	Description
document.getElementById(id)	Find an element by element id
document.getElementsByTagName(name)	Find elements by tag name
document.getElementsByClassName(name)	Find elements by class name

Changing HTML Elements

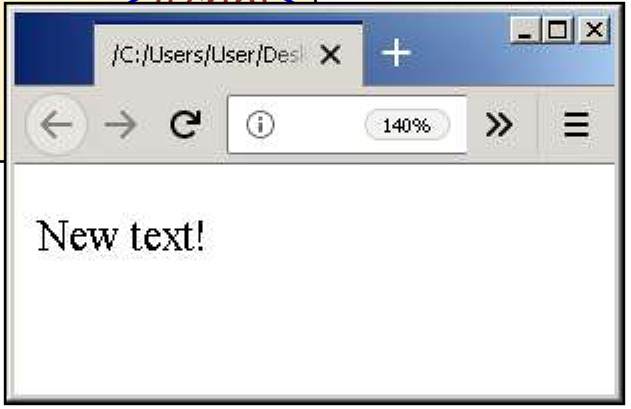
- If you want to access/change any element in an HTML page, first you should find the specific element.

Method	Description
element.innerHTML = new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute(attribute, value)	Change the attribute value of an HTML element
element.style.property = new style	Change the style of an HTML element

Example of Changing HTML Content (by ID)

- The easiest way to modify the content of an HTML element is by using the **innerHTML** property.
- This example changes the content of a <p> element:

```
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "New text!";
</script>
```



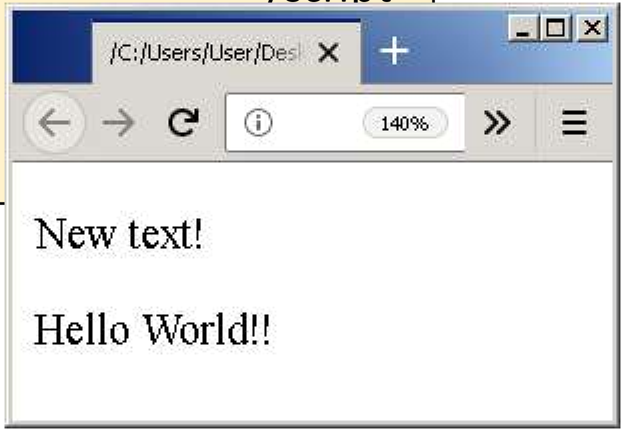
New text!

A JavaScript changes the content (innerHTML) of that element to "New text!"

Example of Changing HTML Content (by Tag)

- This example changes the content of a <p> element by Tag's name

```
<body>

<p>Hello World!!</p>
<p>Hello World!!</p>

<script>
document.getElementsByTagName("p")[0].innerHTML = "New text!";
</script>
```



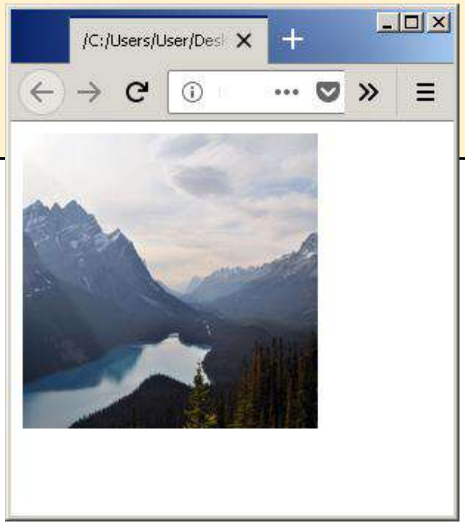
New text!

Hello World!!

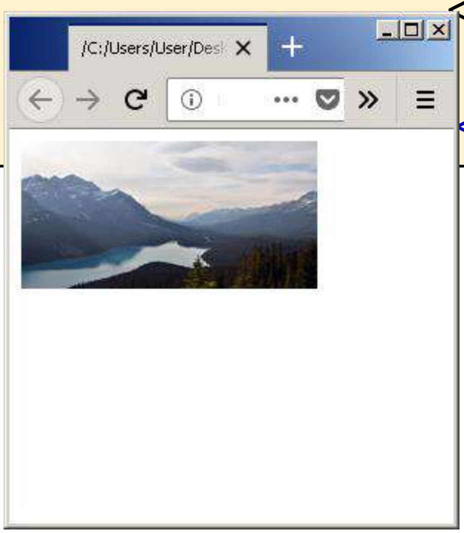
The Tag can be accessed by index numbers. The index starts at 0.

- **Example of Changing the Value of an Attribute**
- This example changes an HTML image by changing the attribute height of an <img> tag

```
<body>

<img id="myImage" src="img_1.jpg" width="200px" height="200">

<script>
document.getElementById("myImage").height = "100";
</script>

</body>
```



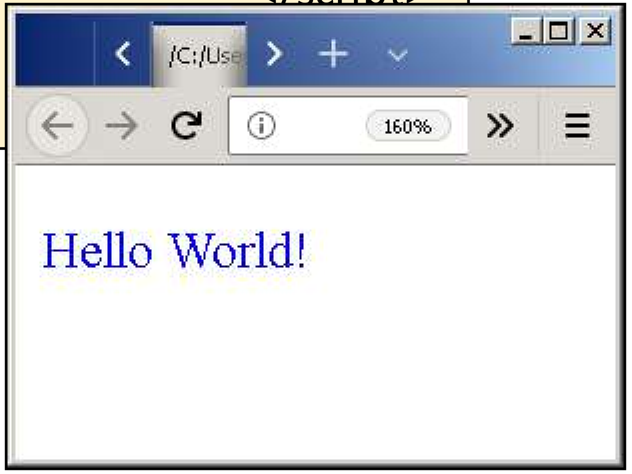
before changing the height



after changing the height

- **Example of Changing the style of HTML elements**
- The following example changes the style of a <p> element:

```
<body>

<p id="p2" style="color:red">Hello World!</p>

<script>
document.getElementById("p2").style.color = "blue";
</script>
```



Hello World!

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- **JavaScript can change all the HTML elements in the page**
- **JavaScript can change all the HTML attributes in the page**
- **JavaScript can change all the CSS styles in the page**
- **JavaScript can remove existing HTML elements and attributes**
- **JavaScript can add new HTML elements and attributes**
- **JavaScript can react to all existing HTML events in the page**
- **JavaScript can create new HTML events in the page**

In the example above, getElementById is a method, while innerHTML is a property.

The getElementById Method

- The most common way to access an HTML element is to use the id of the element.
- The easiest way to get the content of an element is by using the innerHTML property.
- The innerHTML property is useful for getting or replacing the content of HTML elements.
- The innerHTML property can be used to get or change any HTML element, including <html> and <body>.
- The document object represents your web page.
- If you want to access any element in an HTML page, you always start with accessing the document object.
-

Finding HTML Elements by CSS Selectors

- If you want to find all HTML elements that match a specified CSS selector (id, class names, types, attributes, values of attributes, etc), use the querySelectorAll() method.

- This example returns a list of all <p> elements with class="intro".

Example

- var x = document.querySelectorAll("p.intro");
-

Finding HTML Elements by HTML Object Collections

- This example finds the form element with id="frm1", in the forms collection, and displays all element values:
- Example
-
```
var x = document.forms["frm1"];
var text = "";
var i;
for (i = 0; i < x.length; i++) {
  text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

Changing HTML Style

To change the style of an HTML element, use this syntax:

- document.getElementById(*id*).style.*property* = *new style*
- The following example changes the style of a <p> element:
- Example
-
```
<html>
<body>

<p id="p2">Hello World!</p>
<script>
document.getElementById("p2").style.color = "blue";
</script>
<p>The paragraph above was changed by a script.</p>
</body>
</html>
```