Chapter-6 Cascading Style Sheets

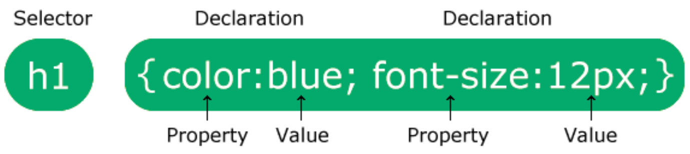
CSS:

What is CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

Cascading Style Sheet(CSS) is used to set the style in web pages that contain HTML elements. It sets the background color, font-size, font-family, color, … etc property of elements on a web page.

CSS Syntax



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

Example

In this example all <p> elements will be center-aligned, with a red text color:

```
p {
    color: red;
    text-align: center;
}
```

Example Explained

- `p` is a selector in CSS (it points to the HTML element you want to style: <p>).
- `color` is a property, and `red` is the property value
- `text-align` is a property, and `center` is the property value

CSS Selectors

A CSS selector selects the HTML element(s) you want to style.

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)

- Attribute selectors (select elements based on an attribute or attribute value)

The CSS element Selector

The element selector selects HTML elements based on the element name.

Example

Here, all <p> elements on the page will be center-aligned, with a red text color:

```
p {
    text-align: center;
    color: red;
}
```

The CSS id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example

The CSS rule below will be applied to the HTML element with id="para1":

```
#para1 {
    text-align: center;
    color: red;
}
```

The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

Example

In this example all HTML elements with class="center" will be red and center-aligned:

```
.center {
   text-align: center;
```

```
    color: red;
}
```

How To Add CSS

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Example

Inline styles are defined within the "style" attribute of the relevant element:

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

Example

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
   background-color: linen;
}

h1 {
   color: maroon;
   margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

External CSS

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

Example

External styles are defined within the <link> element, inside the <head> section of an HTML page:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

An external style sheet can be written in any text editor, and must be saved with a .css extension.

The external .css file should not contain any HTML tags.

Here is how the "mystyle.css" file looks:

"mystyle.css"

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

h1 {
    color: navy;
    margin-left: 20px;
}
```

Note: Do not add a space between the property value and the unit:
Incorrect (space): `margin-left: 20 px;`
Correct (nospace): `margin-left: 20px;`

CSS Basic Properties

Choosing the right font for your website is important!

Font Selection is Important

Choosing the right font has a huge impact on how the readers experience a website.

The right font can create a strong identity for your brand.

Using a font that is easy to read is important. The font adds value to your text. It is also important to choose the correct color and text size for the font.

Generic Font Families

In CSS there are five generic font families:

1. **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
2. **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
3. **Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
4. **Cursive** fonts imitate human handwriting.
5. **Fantasy** fonts are decorative/playful fonts.

Basic Font property:

font-family: “Arial”;

font-size: 20px/1rem/1em;

font-style: normal/*italic*/*oblique*;

font-weight: **bold/bolder**;

CSS Color and Background

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

CSS Color Names

In CSS, a color can be specified by using a predefined color name:

1. Tomato
2. Orange
3. DodgerBlue
4. MediumSeaGreen
5. Gray
6. SlateBlue
7. Violet
8. LightGray

CSS Background

The CSS background properties are used to add background effects for elements.

- `background-color`
- `background-image`
- `background-repeat`
- `background-attachment`
- `background-position`

- `background`

```
background-image: url("img_tree.png");
background-repeat: no-repeat;
background-position: right top;
background-attachment: fixed;
```

CSS background - Shorthand property

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.

Instead of writing:

```
body {
  background-color: #ffffff;
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

You can use the shorthand property `background`:

Example

Use the shorthand property to set the background properties in one declaration:

```
body {
  background: #ffffff url("img_tree.png") no-repeat right top;
}
```

CSS Border:

The CSS border properties allow you to specify the style, width, and color of an element's border.

The `border-style` property specifies what kind of border to display.

The following values are allowed:

- `dotted` -
- `dashed`
- `solid` - Defines a solid border
- `double` - Defines a double border
- `groove` - Defines a 3D grooved border. The effect depends on the border-color value
- `ridge` - Defines a 3D ridged border. The effect depends on the border-color value
- `inset` - Defines a 3D inset border. The effect depends on the border-color value
- `outset` - Defines a 3D outset border. The effect depends on the border-color value
- `none` - Defines no border
- `hidden` - Defines a hidden border

  The `border-style` property can have from one to four values (for the top border, right border, bottom border, and the left border).

The `border-width` property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick:

The `border-color` property is used to set the color of the four borders.

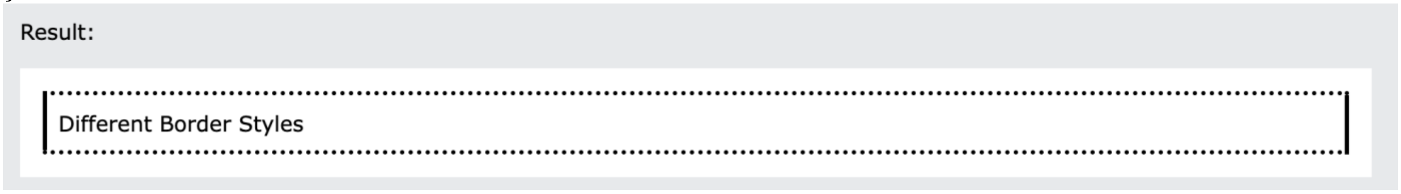
The color can be set by:

- name - specify a color name, like "red"
- HEX - specify a HEX value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- transparent

In CSS, there are also properties for specifying each of the borders (top, right, bottom, and left):

Example

```
p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
}
```


Result:

Different Border Styles

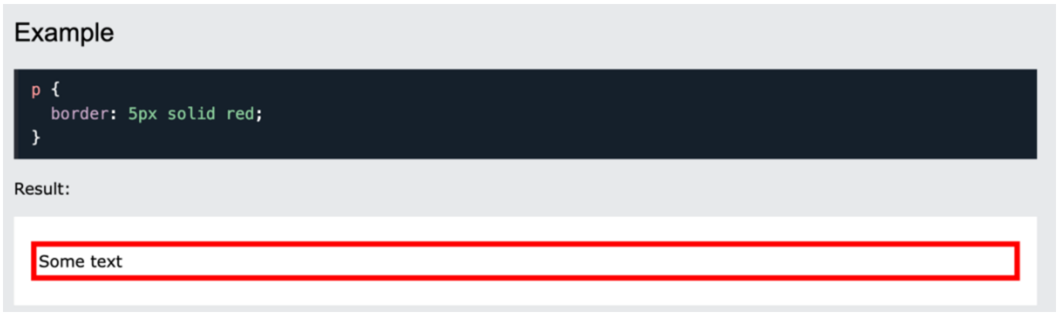
CSS Border - Shorthand Property

There are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The `border` property is a shorthand property for the following individual border properties:

- `border-width`
- `border-style` (required)
- `border-color`


Example

```
p {
  border: 5px solid red;
}
```

Result:

Some text

CSS Margins:

The CSS `margin` properties are used to create space around elements, outside of any defined borders. With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

Margin - Individual Sides

CSS has properties for specifying the margin for each side of an element:

- `margin-top`
- `margin-right`
- `margin-bottom`
- `margin-left`

All the margin properties can have the following values:

- auto - the browser calculates the margin
- *length* - specifies a margin in px, pt, cm, etc.
- *%* - specifies a margin in % of the width of the containing element
- inherit - specifies that the margin should be inherited from the parent element

Tip: Negative values are allowed.

Margin - Shorthand Property

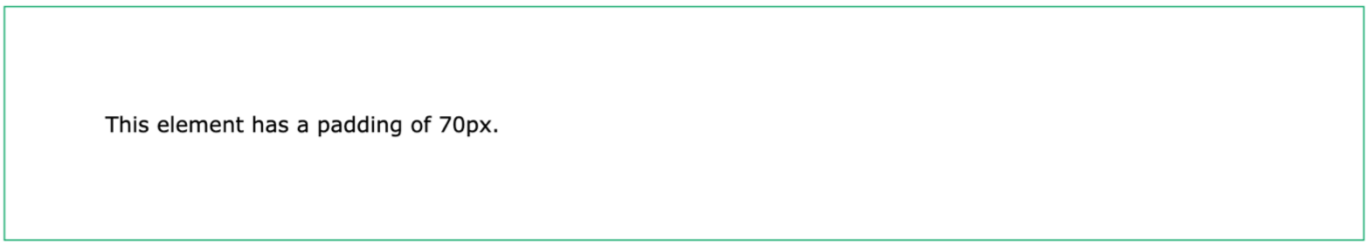
here is how it works:

If the `margin` property has four values:

- **margin: 25px 50px 75px 100px;**
  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px

CSS Padding:

The CSS `padding` properties are used to generate space around an element's content, inside of any defined borders.



This element has a padding of 70px.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

Padding - Individual Sides

CSS has properties for specifying the padding for each side of an element:

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

All the padding properties can have the following values:

- *length* - specifies a padding in px, pt, cm, etc.
- *%* - specifies a padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

Note: Negative values are not allowed.

Padding - Shorthand Property

here is how it works:

If the `padding` property has four values:

- **padding: 25px 50px 75px 100px;**
  - top padding is 25px
  - right padding is 50px
  - bottom padding is 75px
  - left padding is 100px

CSS Layout - The position Property

The position Property

The `position` property specifies the type of positioning method used for an element.

There are five different position values:

- `static`
- `relative`
- `fixed`
- `absolute`
- `sticky`

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the `position` property is set first. They also work differently depending on the position value.

position: static;

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page:

This <div> element has position: static;

Here is the CSS that is used:

Example

```
div.static {
  position: static;
  border: 3px solid green;
}
```

position: relative;

An element with `position: relative;` is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

This <div> element has position: relative;

Here is the CSS that is used:

Example

```
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
```

position: fixed;

An element with `position: fixed;` is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

Example

```
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
```

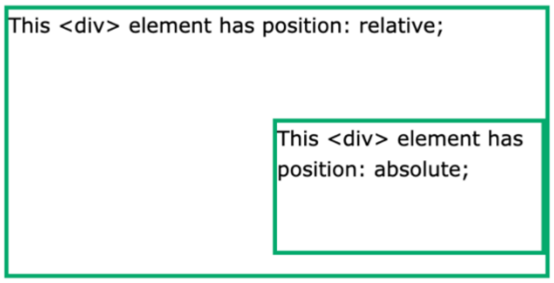
position: absolute;

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

Here is a simple example:


This <div> element has position: relative;
This <div> element has position: absolute;

Here is the CSS that is used:

```
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid green;
}

div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid green;
}
```

position: sticky;

An element with `position: sticky;` is positioned based on the user's scroll position.

A sticky element toggles between `relative` and `fixed`, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

Note: Internet Explorer does not support sticky positioning. Safari requires a -webkit- prefix (see example below). You must also specify at least one of `top`, `right`, `bottom` or `left` for sticky positioning to work.

In this example, the sticky element sticks to the top of the page (`top: 0`), when you reach its scroll position.

Example

```
div.sticky {
  position: sticky;
  top: 0;
  background-color: green;
  border: 2px solid green;
}
```

CSS Layout - Overflow

The CSS `overflow` property controls what happens to content that is too big to fit into an area.

CSS Overflow

The `overflow` property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The `overflow` property has the following values:

- `visible` - Default. The overflow is not clipped. The content renders outside the element's box
- `hidden` - The overflow is clipped, and the rest of the content will be invisible
- `scroll` - The overflow is clipped, and a scrollbar is added to see the rest of the content
- `auto` - Similar to `scroll`, but it adds scrollbars only when necessary

Note: The `overflow` property only works for block elements with a specified height.

Note: In OS X Lion (on Mac), scrollbars are hidden by default and only shown when being used (even though "overflow:scroll" is set).

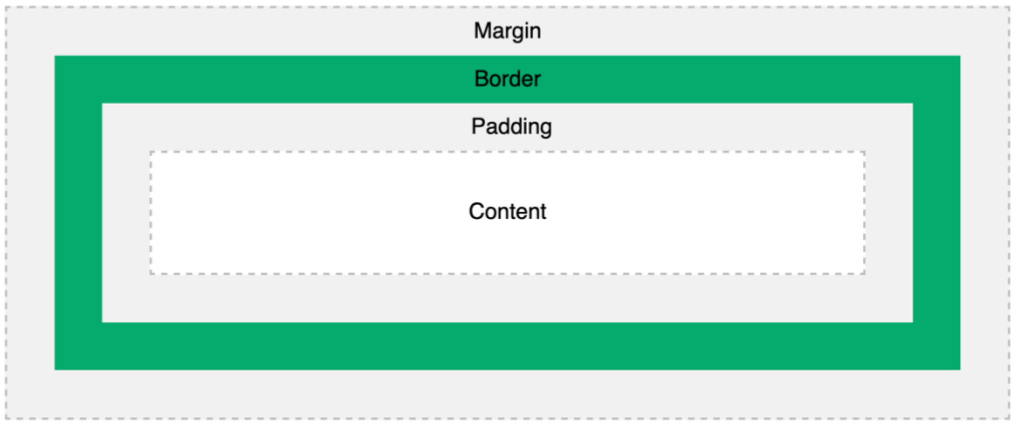
CSS Box Model

All HTML elements can be considered as boxes.

The CSS Box Model

In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



Explanation of the different parts:

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

CSS border-image Property

Definition and Usage

The `border-image` property allows you to specify an image to be used as the border around an element.

The border-image property is a shorthand property for:

- border-image-source
- border-image-slice
- border-image-width
- border-image-outset
- border-image-repeat

Omitted values are set to their default values.

Example

Specify an image as the border around an element:

```
#borderimg {
  border-image: url(border.png) 30 round;
}
```

CSS border-image-source Property

Definition and Usage

The `border-image-source` property specifies the path to the image to be used as a border (instead of the normal border around an element).

Tip: If the value is "none", or if the image cannot be displayed, the border styles will be used.

Tip: Also look at the border-image property (a shorthand property for setting all the border-image-* properties).

Example

Specify an image as the border around an element:

```
#borderimg {
  border-image-source: url(border.png);
}
```

CSS border-image-slice Property

Definition and Usage

The `border-image-slice` property specifies how to slice the image specified by border-image-source. The image is always sliced into nine sections: four corners, four edges and the middle.

The "middle" part is treated as fully transparent, unless the fill keyword is set.

Tip: Also look at the border-image property (a shorthand property for setting all the border-image-* properties).

Example

Specify how to slice the border image:

```
#borderimg {
  border-image-slice: 30%;
}
```

CSS border-image-width Property

Example

Specify the width of the border image:

```css
#borderimg {
  border-image-source: url(border.png);
  border-image-width: 10px;
}
```

Definition and Usage

The `border-image-width` property specifies the width of the border image.

Tip: Also look at the border-image property (a shorthand property for setting all the border-image-* properties).

CSS Text Overflow, Word Wrap, Line Breaking Rules, and Writing Modes

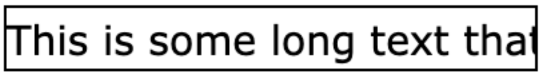
In this chapter you will learn about the following properties:

- `text-overflow`
- `word-wrap`
- `word-break`
- `writing-mode`

CSS Text Overflow

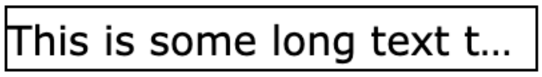
The CSS `text-overflow` property specifies how overflowed content that is not displayed should be signaled to the user.

It can be clipped:



This is some long text that

or it can be rendered as an ellipsis (...):



This is some long text t...

This is some long text that wi

Example

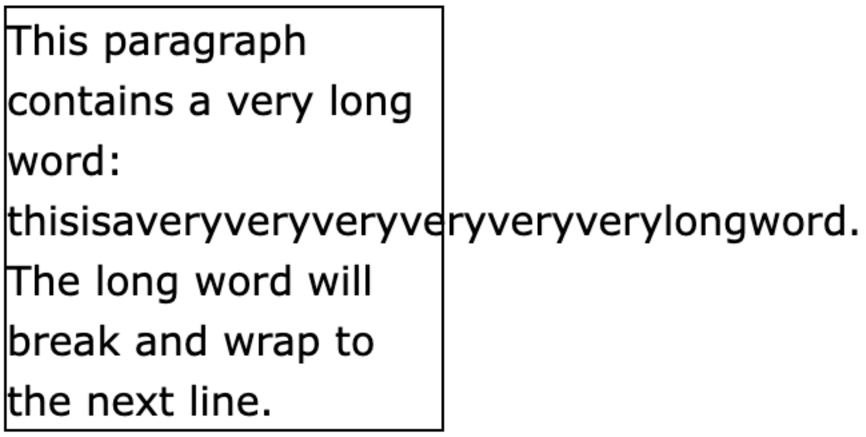
```
p.test1 {
  white-space: nowrap;
  width: 200px;
  border: 1px solid #000000;
  overflow: hidden;
  text-overflow: clip;
}

p.test2 {
  white-space: nowrap;
  width: 200px;
  border: 1px solid #000000;
  overflow: hidden;
  text-overflow: ellipsis;
}
```

CSS Word Wrapping

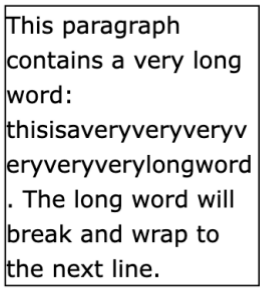
The CSS `word-wrap` property allows long words to be able to be broken and wrap onto the next line.

If a word is too long to fit within an area, it expands outside:



This paragraph contains a very long word: thisisaveryveryveryveryveryverylongword. The long word will break and wrap to the next line.

The word-wrap property allows you to force the text to wrap - even if it means splitting it in the middle of a word:



This paragraph contains a very long word: thisisaveryveryveryv eryveryverylongword . The long word will break and wrap to the next line.

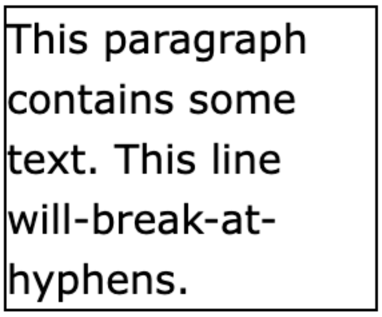
Example

Allow long words to be able to be broken and wrap onto the next line:

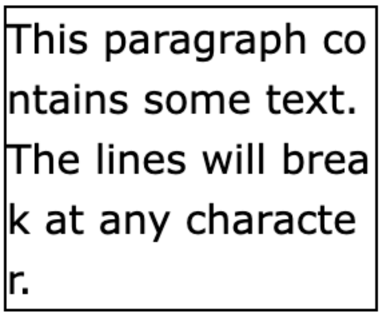
```
p {
  word-wrap: break-word;
}
```

CSS Word Breaking

The CSS `word-break` property specifies line breaking rules.



This paragraph
contains some
text. This line
will-break-at-
hyphens.



This paragraph co
ntains some text.
The lines will brea
k at any characte
r.

Example

Allow long words to be able to be broken and wrap onto the next line:

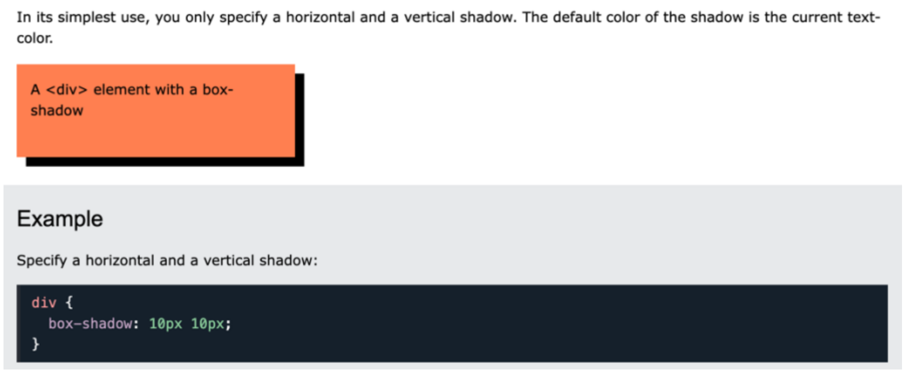
Example

```
p.test1 {
  word-break: keep-all;
}

p.test2 {
  word-break: break-all;
}
```

CSS Box Shadow

The CSS `box-shadow` property is used to apply one or more shadows to an element.



In its simplest use, you only specify a horizontal and a vertical shadow. The default color of the shadow is the current text-color.

A <div> element with a box-shadow

Example

Specify a horizontal and a vertical shadow:

```
div {
  box-shadow: 10px 10px;
}
```