**Chapter 2 CSS**

**2.1 Introduction**

In Chapter [1](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#html), we introduced HTML—the data format for specifying the contents and structure of web pages. Using HTML alone, however, the web pages you create will share the same default single-column arrangement, default colors, default fonts, etc. You will almost never encounter “real” websites with the outdated appearance like the examples shown in Chapter [1](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#html). The main thing that differentiates modern-looking, customized web pages from simple ones is **CSS**—the data format for specifying style and appearance of web pages.

**2.2 What is CSS?**

[**Cascading Style Sheets (CSS)**](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) is a language for specifying the *style* and *appearance* of web pages. For example, CSS is used to specify:

* Fonts
* Colors
* Arrangement of elements on the page

The key to understanding how CSS works is to imagine that there is an *invisible box* around every HTML element (Figure [2.1](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:css-boxes)). CSS allows us to create rules that control the way that each individual box, and the contents of that box, is presented. CSS is run when it is linked to HTML code, while that HTML code is processed and visually presented by the browser.

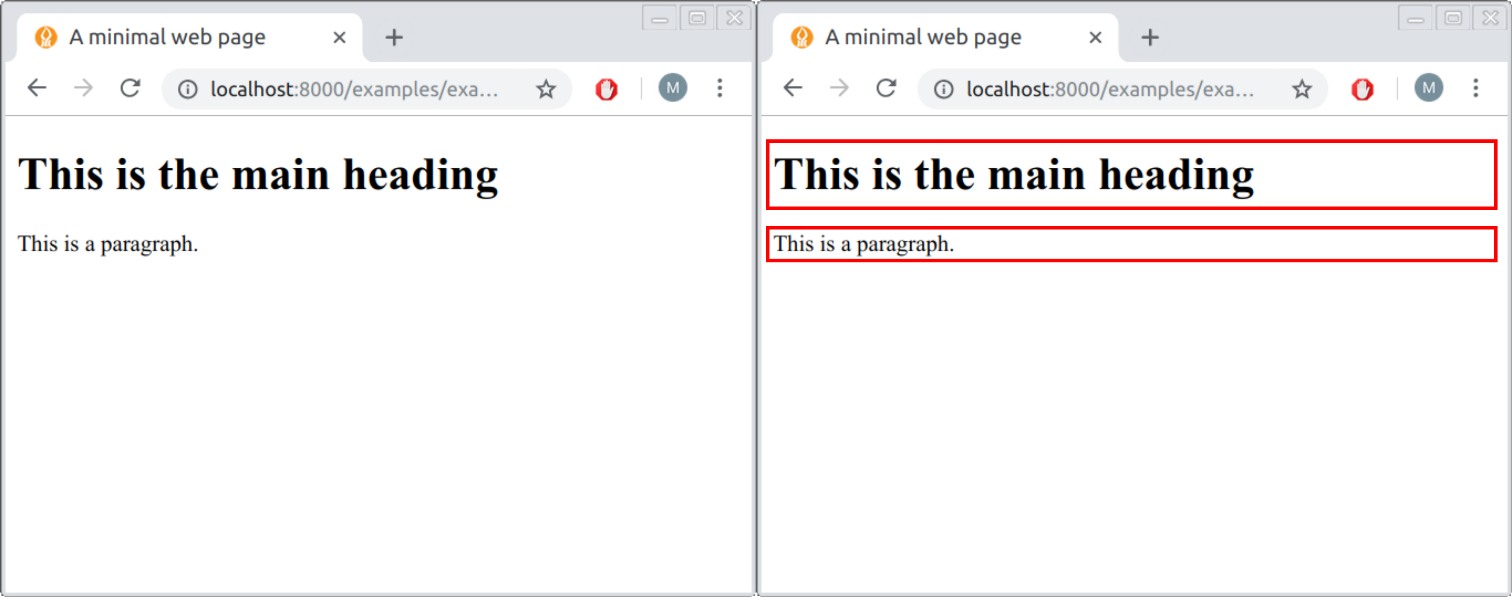


FIGURE 2.1: HTML elements considered as boxes

**2.3 CSS rules**

CSS works by associating styling **rules** with HTML elements. The CSS rules govern how the contents of specified elements should be displayed. A CSS rule contains two parts:

* A **selector**
* One or more **declarations**

The selector specifies which HTML elements the rule applies to. The declarations indicate how the elements referred to by the selector should be styled. Each declaration is split into two parts: a property *name* and a property *value*. Here is an example of a CSS rule:

a { **color**: white; }

This particular rule indicates that all <a> elements, i.e., links (Section [1.6.9.1](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#html-a)), will be displayed with white color. Let’s go over the structure of this rule:

* a is the **selector**, specifying that the rule applies to all <a> elements.
* Curly brackets {...} contain the **declarations**.
* In this case there is only one declaration, color: white;, which means that the affected elements will be colored in white. The declaration is composed of:
  + A property **name** (color) indicating the property that we want to change
  + A property **value** (white) specifying the settings we want to use for the chosen property
  + The property name and property value are separated with :
  + The declaration ends with ;

Here is another example:

a { **font-style**: italic; }

This time the rule indicates that all <a> elements should be shown in italics. The only difference from the previous example is that the declaration color: white; was replaced with the declaration font-style: italic;. Note the different components of the CSS rule, most importantly the selector, the property name, and the property value (Table [2.1](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#tab:css-rule-structure)).

| TABLE 2.1: CSS rule structure | |
| --- | --- |
| **Component** | **Example** |
| CSS rule | a { font-style: italic; } |
| Selector | **a** { font-style: italic; } |
| Declaration | a { **font-style: italic;** } |
| Property name | a { **font-style**: italic; } |
| Property value | a { font-style: **italic**; } |

**2.4 CSS selectors**

**2.4.1 Overview.**Within a CSS rule, the selector specifies which HTML elements will be affected by the rule. There are many different types of CSS selectors that allow you to target rules to a very specific selection of elements in an HTML document. Some CSS selectors are used to target specific HTML elements (Table [2.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#tab:css-selectors-specific)), while other selectors specify selector *combinations* (Table [2.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#tab:css-selector-combinations)).

| TABLE 2.2: CSS selectors targeting specific elements | | |
| --- | --- | --- |
| **Selector** | **Meaning** | **Example** |
| *Type Selector* | Matches element names | h1 {} targets all <h1> elements |
| *Class Selector* | Matches all elements whose class matches the one specified after the dot (.) | .note {} targets any element whose class attribute has a value of "note". p.note {} targets only <p> elements whose class attribute has a value of "note" |
| *ID Selector* | Matches an element whose id matches the one specified after the hash (#) | #introduction {} targets the element whose id attribute has the value of "introduction" |

| TABLE 2.3: CSS selector combinations | | |
| --- | --- | --- |
| **Selector** | **Meaning** | **Example** |
| *Multiple Selector* | Matches elements corresponding to any of the selectors | h1, h2, h3 {} targets the <h1>, <h2>, and <h3> elements |
| *Descendant Selector* | Matches an element that is a descendant of another element (not necessarily direct child) | p a {} targets all <a> elements that sit inside a <p> element, even if there are other elements between them |
| *Child Selector* | Matches an element that is a direct child of another element | li>a {} targets all <a> elements that are children of <li> elements |
| *Adjacent Sibling Selector* | Matches an element that is the nearest sibling of another | h1+p {} targets those <p> elements which come right after an <h1> element |
| *General Sibling Selector* | Matches an element that is a sibling of another | h1~p {} targets all <p> elements that follow, immediately or not, an <h1> element |

Throughout this book, we will use the following most commonly used selectors:

* Type selector, possibly combined in a multiple selector (h1, h2, h3 {})
* Class selector (.note {})
* ID selector (#introduction {})
* Descendant selector (p a {})

More details on these selectors are given in the following Sections [2.4.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#type-selector)–[2.4.5](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#descendant-selector).

**2.4.2 Type selector.**The **type selector** applies to the element(s) of the specified type in the linked HTML code. In case there are more than one, the type names should be separated by a comma, in a combination known as a **multiple selector** (Table [2.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#tab:css-selector-combinations)). For example, the following rule applies to all <a> elements:

a { **color**: white; }

The following rule, however, applies to all <a> *and* all <p> elements:

p, a { **color**: white; }

**2.4.3 Class selector.**The **class selector** contains a full stop (.), and the part after the full stop specifies the name of a class (Section [1.7.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#html-class)). All elements that have a class attribute with the given value will be affected by the rule. For example, the following rule applies to all elements that have the attribute class="figure":

.figure {

**display**: block;

**margin-left**: auto;

**margin-right**: auto;

}

We can combine the class selector with the type selector. This means the rule will no longer apply to all elements with the appropriate class, but only to the HTML elements of a particular type *and* the appropriate class. For example, the following rule applies only to the <img> elements that have class="figure":

img.figure {

**display**: block;

**margin-left**: auto;

**margin-right**: auto;

}

**2.4.4 ID selector.**The **ID selector** contains a hash character (#) followed by an ID. The rule applies to the specific element that has the given id attribute. The ID selector can be used to control the appearance of exactly *one* element, since the id is unique (Section [1.7.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#id)). For example, the following rule applies to the specific element which has the attribute id="footer":

#footer { **font-style**: italic; }

**2.4.5 Descendant selector.**The **descendant selector**[8](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fn8) contains several element names, separated by spaces. This allows a CSS rule to be applied to an element only when the element is contained *inside* another type of element. For example, the following rule applies to all <a> elements that are within <p> elements:

p a { **color**: white; }

Note that, in the above example, the <a> element does not need to be immediately within the <p> element, unlike when using the **child selector** (Table [2.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#tab:css-selector-combinations)).

What is the difference between the p.important and p .important selectors?

**2.5 CSS conflicts**

It is possible for CSS rules to **conflict**, i.e., for there to be more than one rule affecting the same *property* in the same *element*. In case of conflict, generally a more [specific](https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity) rule will override a less specific one. For example, an ID selector will override a type selector, since the former is more specific. Given the following pair of CSS rules, a <p> element with id="intro" will be colored red according to the more specific rule, while all other <p> elements will be colored black:

p { **color**: black; }

#intro { **color**: red; }

If there are two rules having the same level of specificity, the rule that is specified *last* wins. For example, if two CSS files are linked in the header of an HTML document (see Section [2.7.4](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#external-css) below) and they both contain rules with the same selector, then the rule in the second file—the one processed later by the browser—will override the rule in the first file. As another example, consider the pair of CSS rules shown below. In this case, all <p> elements will be colored red—according to the *last* specified rule:

p { **color**: black; }

p { **color**: red; }

The next example is more complex, containing eight CSS rules with numerous conflicts:

\* { **font-family**: Arial, Verdana, sans-serif; }

h1 { **font-family**: "Courier New", monospace; }

i { **color**: green; }

i { **color**: red; }

b { **color**: pink; }

p b { **color**: blue; }

#intro { **font-size**: 100%; }

p { **font-size**: 75%; }

Ignore the properties for now, only consider the selectors. Based on the conflict-resolving guidelines described previously, here is how this particular set of rules will be processed by the browser:

* The h1 rule will override the \* rule, since h1 is more specific than \*.
  + Therefore, first-level headings will be shown in Courier New font rather than Arial.
* The second i selector will override the first one, since it is specified later.
  + Therefore, *italic* text will be shown in red rather than green.
* The p b rule will override the b rule, since p b is more specific than b.
  + Therefore, bold text inside a paragraph will be shown in blue rather than pink.
* The #intro rule will override the p rule, since #intro is more specific than p.
  + Therefore, the paragraph with id="intro" will be shown with larger font.

An example of a web page where the above CSS rules are applied (example-02-01.html) is shown in Figure [2.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-01).

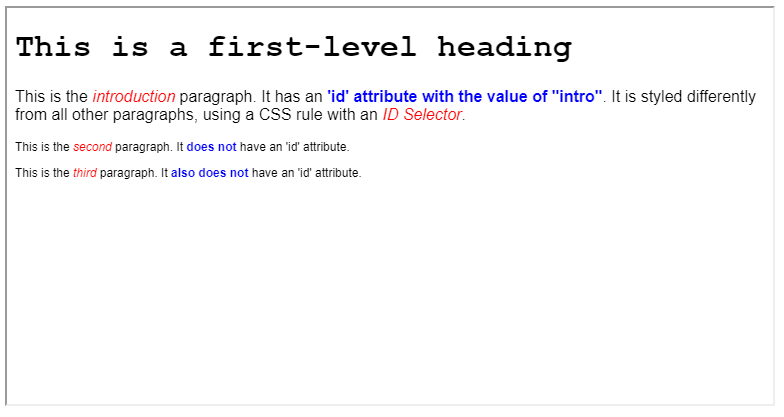


FIGURE 2.2: example-02-01.html [(Click to view this example on its own)](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\examples\example-02-01.html)

**2.6 CSS inheritance**

Some CSS property values are **inherited**, since it makes sense to specify them for an element as well as all of the element’s descendants. Other CSS properties are *not* inherited, since it makes sense to specify them just for the targeted element and not for its descendants.

For example, it makes sense for font-family and color to be inherited, as that makes it easy to set a document-wide *base font* and *base color* by setting the font-family and color on the <body> element. That way, all of the heading, paragraphs, and all other elements in the document—which are all descendants of the <body> element—are uniformly displayed in the same font and color. When necessary, you can override the fonts and/or colors on individual elements where needed, relying on the fact that a specific rule overrides a general one (Section [2.5](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#css-conflicts)). It would be really annoying to have to set the base font separately on every element.

Contrariwise, it makes no sense for margin, padding, and border to be inherited. These CSS properties control the placement and border style of HTML elements (Section [2.8.4](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#boxes)). Imagine the styling mess that would occur if you set these properties on a container element and had them inherited by every single internal element! Nevertheless, in rare situations forcing inheritance is necessary. Forcing a property to inherit values from their “parent” element can be done by using the keyword inherit as the respective property value (see Section [2.9](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#hurricane-scale) for an example)[9](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fn9).

**2.7 Linking CSS to HTML**

**2.7.1 Methods for linking CSS.**How can we actually **link** our CSS code to an HTML document, to affect its presentation in the browser? There are three methods in which CSS code can be linked to an HTML document:

* **Inline** CSS (Section [2.7.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#inline-css))
* **Embedded** CSS (Section [2.7.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#embedded-css))
* **External** CSS (Section [2.7.4](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#external-css))

The following Sections [2.7.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#inline-css)–[2.7.4](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#external-css) shortly explain each of these methods.

**2.7.2 Inline CSS.**The simplest approach to associate CSS code with HTML elements is to include the CSS code within the style attribute of an HTML element. As mentioned in Section [1.7.4](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#style-2), style is one of three general attributes that can appear in every type of HTML element (along with id and class). The value of the style attribute contains CSS declarations applying to that element. This method is called **inline CSS**. For example, a paragraph that has the following start tag will be displayed with italic font, thanks to the inline CSS rule. Here, CSS is used to control the appearance of text within *this* paragraph only.

**<p** style="font-style:italic;"**>**

Note that, with inline CSS, the code consists of just declaration(s), with no selectors. Selectors are not needed, since inline CSS by definition only applies to a specific HTML element where the style attribute appears.

Using inline CSS is generally not recommended, and should be used sparingly, because it:

* Mixes CSS code with HTML code, making both of them harder to maintain
* Requires repeating CSS code in all elements where the same style needs to be applied

We will see one example where inline CSS comes in handy later on (Section [8.6](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\symbology-and-interactivity.html#adding-a-legend)).

**2.7.3 Embedded CSS.**It is also possible to include CSS code within a <style> element within the <head> element of HTML code (Section [1.6.4.1](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#style-1)). This approach is called **embedded CSS**. For example, here are the first few lines of an HTML document that uses embedded CSS:

<!DOCTYPE html>

**<html>**

**<head>**

**<style>**

p { **font-style**: italic; }

**</style>**

...

In this example, the appearance of text within *all* paragraphs in the document is controlled with a single CSS rule. Embedded CSS is a better approach than inline CSS, because with embedded CSS the rules do not need to be attached to each individual HTML element. However, embedded CSS is still not ideal, because:

* the separation between HTML and CSS is not complete—although all CSS code is in one place, it is still *embedded* in the HTML document; and
* consequently, any reuse of the CSS code with other HTML code requires making multiple copies of the same CSS code in the <head> of each page.

**2.7.4 External CSS.**The third approach, called **external CSS**, is writing CSS code in a separate .css file and referring to that CSS file from the HTML code. The CSS file is referred to using a <link> element within the <head> element (Section [1.6.4.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#link)). For example, here are the first few lines of an HTML document that uses external CSS:

<!DOCTYPE html>

**<html>**

**<head>**

**<link** rel="stylesheet" href="style.css"**>**

...

The above <link> element refers to a file with CSS code, in this case a file named style.css. The contents of the style.css file is plain CSS code, such as:

p { **font-style**: italic; }

The external CSS approach is the recommended way of using CSS, since it makes HTML and CSS code completely *separate*, and thus the CSS code is easier to maintain. Additionally, external CSS makes the code *reusable*, since the same CSS file can be linked to multiple HTML files. That way, for example, the same styling can be applied to all pages of a given website, while still keeping the CSS code in one place rather than repeating it. For example, the online version of this book (Section [0.7](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\index.html#online-version)) is composed of multiple HTML documents (one for each chapter), all of them linked to the same external CSS files and thus sharing the same style.

In this book, we will mostly use embedded CSS (Section [2.7.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#embedded-css)), since our code examples will be relatively short and since embedded CSS makes the presentation of the material easier: the entire source code, including HTML and CSS, can be viewed in one place in a single code file. The reader should keep in mind, however, that when working on larger projects keeping CSS in external files is the recommended way to go.

**2.8 CSS properties**

**2.8.1 Overview.**So far we have discussed the general structure of CSS code (Section [2.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#css-rules)), selector types (Section [2.4](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#css-selectors)), conflict and inheritance issues (Sections [2.5](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#css-conflicts)–[2.6](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#css-inheritance)), and ways to associate CSS code with an HTML document (Section [2.7](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#linking-css-to-html)). In this section, we will go over specific, commonly used CSS properties. The CSS properties we are going to meet can be divided into three groups, according to the styling aspect they influence:

* Color (Section [2.8.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#color))
* Text (Section [2.8.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#text))
* Boxes (Section [2.8.4](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#boxes))

Table [2.4](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#tab:css-properties) lists the specific CSS properties that we are going to cover. Note that there are many other CSS properties that we are not going to cover since they are less relevant to web mapping[10](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fn10).

| TABLE 2.4: Common CSS properties | | |
| --- | --- | --- |
| **Category** | **Property** | **Description** |
| Color | color: | Foreground color |
|  | background-color: | Background color |
| Text | font-family: | Font family |
|  | font-style: | normal or italic |
|  | font-weight: | normal or bold |
|  | font-size: | Font size |
|  | text-align: | Text alignment (e.g., center) |
| Boxes | width: | Box width |
|  | height: | Box height |
|  | border-width: | Border width |
|  | border-style: | Border style (e.g., dotted) |
|  | border-color: | Border color |
|  | border-radius: | Border radius |
|  | margin: | Margin size |
|  | padding: | Padding size |
|  | position: | Positioning method (e.g., fixed) |
|  | top: | Top offset |
|  | bottom: | Bottom offset |
|  | right: | Right offset |
|  | left: | Left offset |
|  | z-index: | Z-index |

**2.8.2 Color**

**2.8.2.1 color:, background-color:**The color: property controls the **foreground color**. The background-color: property controls the **background color**. For example, when referring to a <p> element, color: specifies the color of text inside the paragraph, while background-color: specifies the background color of the paragraph “box” (Figure [2.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-02)). The most common ways of specifying the color itself, i.e., the *values* of these properties, are summarized in Table [2.5](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#tab:css-colors) and described as follows.

| TABLE 2.5: Methods for specifying color in CSS | |
| --- | --- |
| **Method** | **Example** |
| RGB | rgb(255,255,0) |
| RGBA | rgba(255,255,0,1) |
| HEX | #ffff00 |
| Color name | yellow |

**RGB** values express colors in terms of how much red, green, and blue are used to make them up. The values can be integers (from 0 to 255) or percentages (0% to 100%). For example: rgb(255,255,0) means yellow, since red and green are set to maximum, while blue is set to zero; rgb(100%,100%,0%) is another way of specifying the same color. **RGBA** values are similar to RGB values, but with an extra channel, known as the alpha channel, representing transparency (0 = fully transparent, 1 = fully opaque). For example: rgba(0,0,0,0.5) means 50% transparent black, i.e., grey.

**HEX** codes are six-digit codes that represent the amount of red, green, and blue in a color, preceded by a hash # sign. Each pair of digits specifies a number between 0 and 255 in [hexadecimal](https://en.wikipedia.org/wiki/Web_colors#Hex_triplet) notation. For example, #ffff00 specifies yellow color, since in hexadecimal notation ff specifies 255 and 00 specifies 0. RGB(A) and HEX are, therefore, identical in their ability to convey \(256^3=16,777,216\) [different colors](https://en.wikipedia.org/wiki/RGB_color_model). The difference between the two methods is that RGB uses the *decimal* number notation while HEX uses the *hexadecimal* notation[11](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fn11).

Finally, there are several predefined **color names** which are recognized by browsers, such as black, white, red, green, blue, and yellow. It should be noted that using predefined colors such as red and blue is usually not ideal in terms of graphic design. RGB and HEX color specifications should be generally preferred, as they make it possible to make subtle color optimization and to use predefined, carefully chosen color scales, as we will see later on (Section [8.4.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\symbology-and-interactivity.html#towns-example)). The [*X11 Color Names*](https://en.wikipedia.org/wiki/Web_colors#X11_color_names) Wikipedia entry gives a list of predefined color names along with their RGB and HEX codes. It is useful to go over it, to get a feeling of the three color-specification systems[12](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fn12).

To summarize the four color-specification methods, the following CSS code specifies the color property for <h1> and <h2> elements, and the color as well as the background-color of the <p> elements on the page. Note how the code uses all four above-mentioned methods for specifying colors with CSS:

h1 {

**color**: DarkCyan; */\* Color name \*/*

}

h2 {

**color**: #ee3e80; */\* HEX \*/*

}

p {

**background-color**: rgba(255,255,0,0.8); */\* RGBA \*/*

**color**: rgb(100,100,90); */\* RGB \*/*

}

A small example, example-02-02.html, demonstrating the effect of these rules, is shown in Figure [2.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-02).

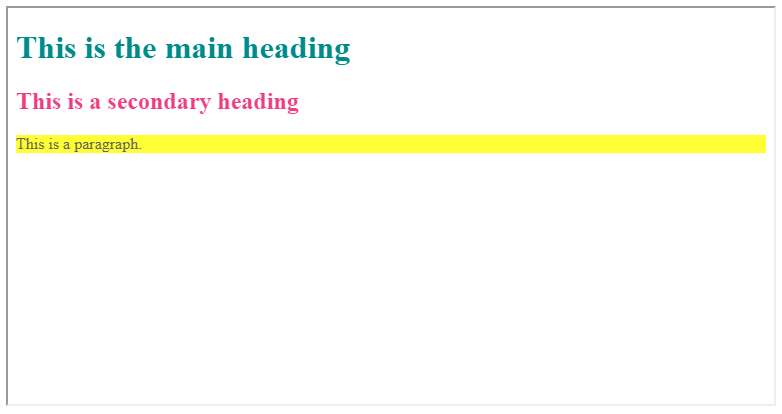


FIGURE 2.3: example-02-02.html [(Click to view this example on its own)](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\examples\example-02-02.html)

Note that we used CSS comments in the last example. CSS comments are written between the /\* and the \*/ symbols, like so:

*/\* This is a CSS comment \*/*

**2.8.3 Text**

**2.8.3.1 font-family:**The font-family: property controls the **font** for text within an element. The value of the font-family: property can be:

* A *specific* font family name, such as "Times New Roman", Arial or Courier
* A *generic* font family name, such as serif, sans-serif, or monospace

It is a good idea to specify several values, separated by commas, to indicate that they are alternatives. The browser will select the first font in the list that is installed or that can be downloaded. The [recommendation](https://www.w3.org/TR/CSS21/fonts.html#propdef-font-family), which we follow here, is to put quotes around font names that contain spaces, such as "Times New Roman" instead of Times New Roman.

The last option on the list is usually a generic font family name. In case the person viewing the result does not have any of the specific fonts on their computer, the browser will fall back to any available font from the generic family. For example:

.serif { **font-family**: Times, "Times New Roman", Georgia, serif; }

.sansserif { **font-family**: Verdana, Arial, Helvetica, sans-serif; }

.monospace { **font-family**: "Lucida Console", Courier, monospace; }

In this example, the first rule means that a Times, "Times New Roman", or Georgia font will be used if available (ordered from highest to lowest preference). Otherwise, the browser will choose any serif font that is available[13](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fn13).

**2.8.3.2 font-weight:, font-style:**These properties control the *appearance* of text. The font-weight: can be normal or bold. The font-style: can be normal or italic. The font-weight: and font-style: CSS properties are therefore alternatives of the <b> (Section [1.6.6.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#html-b)) and <i> (Section [1.6.6.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#html-i)) HTML elements for specifying italic or bold font, respectively.

**2.8.3.3 font-size:**The font-size: property determines *font size*. It can be specified in several ways:

* In **pixels**, for example: 12px means 12 pixels
* In **percentages**, relative to size in parent element, for example: 75% means 75% of parent element size
* In **ems**, relative to *font* size in parent element, for example: 1.3em means 1.3 times larger than parent element size

Note that pixels are *absolute* units, while percentages and ems are *relative* units[14](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fn14).

**2.8.3.4 text-align:**The text-align: property controls the **alignment** of text within an element, with possible values left, right, center, or justify.

The following example includes several of the above-mentioned CSS properties related to text:

body {

**font-family**: "Lucida Console", Courier, monospace;

}

h2 {

**font-family**: "Times New Roman", Times, serif;

**font-style**: italic;

}

p {

**font-size**: 19.2px;

**text-align**: center;

}

The example-02-03.html web page demonstrates the visual effects of the these CSS rules (Figure [2.4](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-03)).

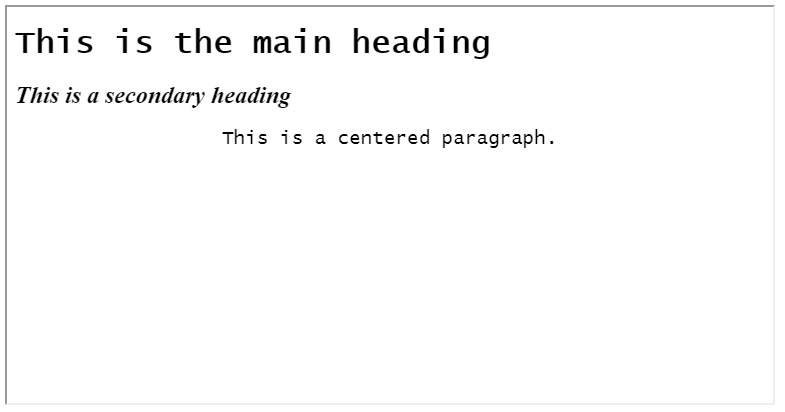


FIGURE 2.4: example-02-03.html [(Click to view this example on its own)](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\examples\example-02-03.html)

**2.8.4 Boxes**

**2.8.4.1 CSS box properties.**In the beginning of this chapter, we mentioned how CSS treats each HTML element as if it lives in its own box (Figure [2.1](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:css-boxes)). You can set several properties that affect the appearance and placement of these boxes. For example, using CSS it is possible to:

* Control the **dimensions** of boxes
* Create **borders** around boxes
* Set **margins** and **padding** for boxes
* Place the boxes in different **positions** on the page

The terms margin, border, and padding can be confusing[15](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fn15), so we will define them right from the start. Every box has three properties that can be adjusted to control its boundaries (Figure [2.5](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:css-box-model)):

* **Margin**—Margins sit outside the edge of the border. You can set the width of a margin to create a gap between the borders of two adjacent boxes. Margins are always transparent.
* **Border**—Every box has a border, even if it is not visible or is specified to be 0 pixels wide. The border separates the edge of one box from another.
* **Padding**—Padding is the space between the border of a box and any content contained within it. Adding padding can increase the readability of the contents.

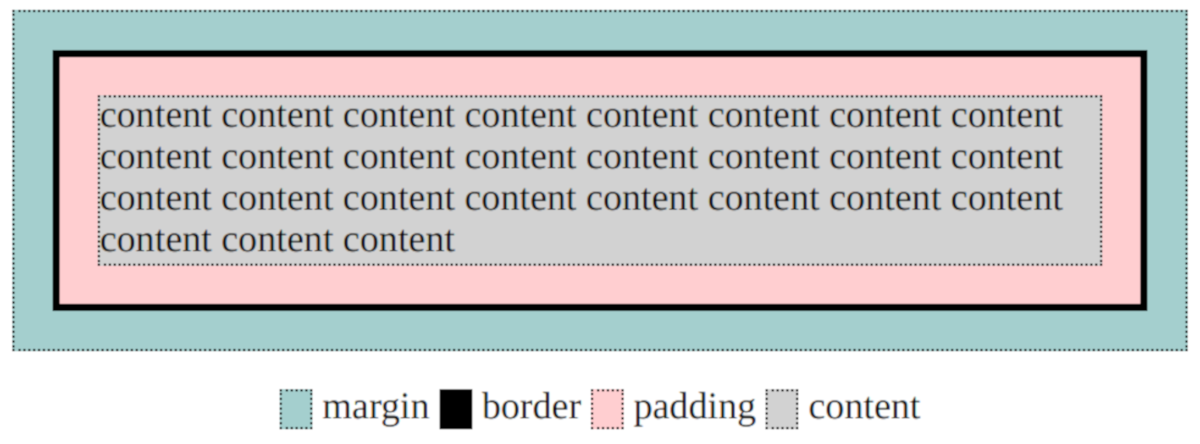


FIGURE 2.5: Margin, border, and padding around an HTML element

**2.8.4.2 width:, height:**By default, a box is sized just big enough to hold its contents. For example, a paragraph of text expands to fill the entire width of the page and uses as many lines as necessary (Section [1.6.5.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#html-p)), while an image has an *intrinsic size*—the number of pixels in each direction (Section [1.6.10.1](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#html-img)). The width: and height: properties provide means to override the default width and height and explicitly control the dimensions of an element.

Widths or heights can be specified either as *percentages* of the parent element, or as *absolute* values (such as in px units). For example, within a web page that is 800 pixels wide, to make a paragraph of text use half of the page width, we can use either of the following specifications:

p { **width**: 50%; }

p { **width**: 400px; }

It is important to note that element width: and height: determine the content size, so if we also add padding (Section [2.8.4.5](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#padding)) and borders (Section [2.8.4.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#border)) then the final size of the element will be larger than intended. There is a property named box-sizing:, which solves this problem by specifying the total size, including content, padding, and border.

Also note that—unlike width:—the height: property can only be specified with absolute units (such as px), and cannot be set with relative units (such as %)[16](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fn16).

The following HTML document example-02-04.html uses the width: and height: properties to create three <div> elements, each having 100px height and 20% width. The <div> elements have little content—just the numbers 1, 2 and 3—yet their entire area is visible because of the background color.

<!DOCTYPE html>

**<html>**

**<head>**

**<title>**CSS box size**</title>**

**<style>**

div {

**height**: 100px;

**width**: 20%;

**background-color**: powderblue;

}

**</style>**

**</head>**

**<body>**

**<p>**

The following three 'div' elements have heights of

**<b>**100px**</b>** and widths of **<b>**20%:**</b>**

**</p>**

**<div>**1**</div>**

**<div>**2**</div>**

**<div>**3**</div>**

**</body>**

**</html>**

The resulting page is shown in Figure [2.6](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-04).

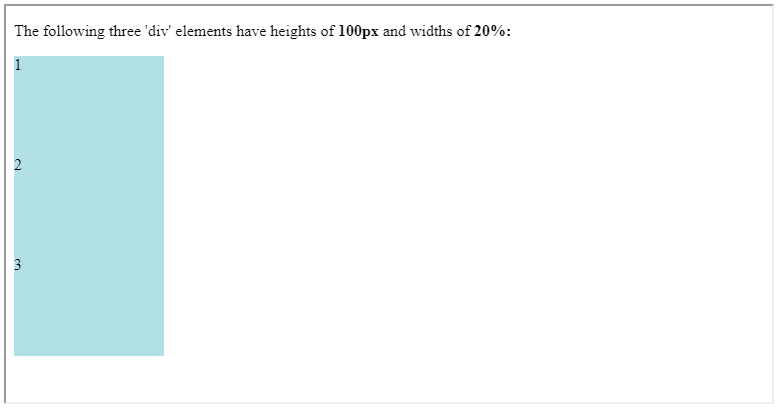


FIGURE 2.6: example-02-04.html [(Click to view this example on its own)](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\examples\example-02-04.html)

**2.8.4.3 border-width:, border-style:, border-color:, border-radius:**These properties control the appearance of **borders** (Figure [2.5](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:css-box-model)) around an element. The border-width: property sets border width. The border-style: property sets border style. Possible values for border-style: include solid, dotted, double and dashed. The border-color: property sets border color.

The border-width:, border-style: and border-color: properties affect all four borders. There are also specific properties that affect only the top, left, right, or bottom border of an element. These can be used by adding -top-, -left-, -right-, or -bottom- in the property name, respectively. For example, the following CSS rule can be used to produce horizontal lines at the top of all paragraphs on a web page:

p {

**border-top-width**: 1px;

**border-top-style**: solid;

}

The border-radius: property can be used to make the border corners appear *rounded* (see Section [2.10.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#example-map-description-box-position) for an example).

**2.8.4.4 margin:**The margin: property controls the **margin**—how much space is allocated around the *outside* of the element (Figure [2.5](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:css-box-model)), that is, between the given element and its neighboring elements. The size of margins can be expressed using various units of length, same as in the width: and height: properties (Section [2.8.4.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#css-width-height)).

The margin: property affects all four margins. Just like with borders (Section [2.8.4.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#border)), there are also properties for controlling individual margins: margin-top:, margin-bottom:, margin-right:, and margin-left:.

**2.8.4.5 padding:**The padding: property controls box **padding**—how much space is allocated between the border of the element and its contents (Figure [2.5](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:css-box-model)). Values are specified the same way as for margins. Similarly, there are specific properties (padding-top:, etc.), for controlling the padding on each side of the element.

* Make your own copy of example-02-04.html (Figure [2.6](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-04)) and modify the following aspects:
  + Add more text inside the <div> elements.
  + Add CSS rules to show a colored border around each <div> element.
  + Add CSS rules to separate the <div> elements from each other (using margins) and to separate their text from the border (using padding).

**2.8.4.6 position:, top:, bottom:, right:, left:**The position: property specifies the **positioning method** for an element. Possible values for the position: property include:

* static (the default)
* relative
* absolute
* fixed

In **normal** flow (position: static), each block-level element sits below the previous one. Since this is the default way in which browsers treat HTML elements, you do not need a CSS property to indicate that elements should appear in normal flow, but the syntax would be:

position***: static***;

**Relative** positioning with position: relative moves an element in relation to where it would have been in normal flow. For example, you can move it 10px lower than it would have been in normal flow or 20% to the right. With relative positioning, the offset properties, top: or bottom: and/or left: or right:, are used to indicate how far to move the element from where it would have been in normal flow. To move the box up or down, you can use the bottom: or top: offset properties, respectively. To move the box left of right, you can use the right: or left: offset properties, respectively. The values of the box offset properties are usually given in pixels (10px), percentages (30%), or ems (8em).

**Absolute** positioning with position: absolute places the element relatively to the nearest “positioned” ancestor. A “positioned” element is one whose position is anything except position: static. However, if an absolute positioned element has no positioned ancestors, it uses the document <body>. For example, we can specify that a <div> is located 10px from the top and 10px from the left of its containing <div>. An absolutely positioned element moves along with page scrolling, i.e., does not stay in the same place on screen.

**Fixed** positioning with position: fixed is relative to the viewport. As implied by the name *fixed*, this means the element always stays in the same place even if the page is scrolled.

It is important to note that both position: absolute and the position: fixed positioned elements are *taken out* of normal flow. This means the elements “hover” above the page, while other elements shift to fill the place they previously occupied. This also means that these elements can obstruct the view of other elements.

The example-02-05.html web page (Figure [2.7](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-05)) has three <div> elements, styled to have specific dimensions, background color, and borders, using the CSS code shown below:

div {

**height**: 100px;

**width**: 150px;

**background-color**: powderblue;

**padding**: 20px;

**margin**: 5px;

**border-width**: 1px;

**border-style**: solid;

}

There are two additional styling rules, #one and #two (see below), affecting just the first and second <div> elements, respectively. The first <div> has a relative position, thus shifted from its normal flow by 30px from bottom and by 40px from left. Note that though the element is shifted, its original position is maintained in normal flow, rather than filled up with other content (Figure [2.7](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-05)).

#one {

**position**: relative;

**bottom**: 30px;

**left**: 40px;

}

The second <div> element has a fixed position, placing it relative to the viewport. Specifically, the second <div> is placed 30px from the bottom of the viewport and 40px from the left of the viewport. Note that the second <div> obstructs some of the text shown in the <p> elements, as well as the third <div>. Both of these outcomes are because absolutely positioned elements go out of the ordinary flow. Therefore the second <div> is not affected by underlying text, as if it is “hovering” above the text. The place in the normal flow is then cleared for other content to take place, such as the third <div> (Figure [2.7](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-05)).

#two {

**position**: fixed;

**bottom**: 30px;

**left**: 40px;

}

The resulting page is shown in Figure [2.7](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-05).

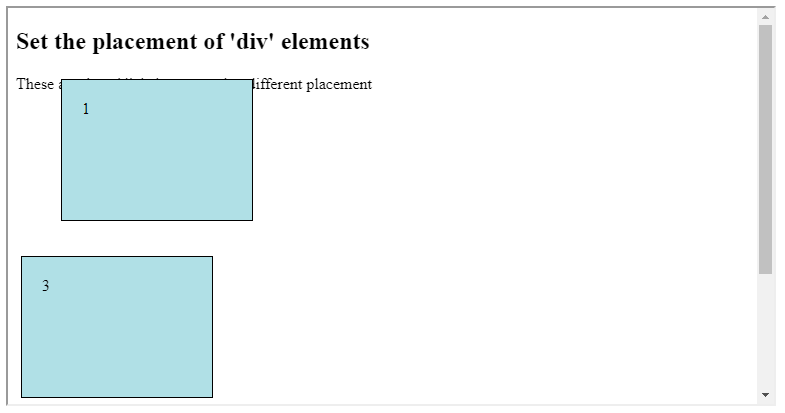


FIGURE 2.7: example-02-05.html [(Click to view this example on its own)](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\examples\example-02-05.html)

* Create your own copy of example-02-05.html.
* Delete the #one rule from the source code.
* Open the new version in the browser to observe how the first <div> goes back to its “natural” position (compared to Figure [2.7](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-05)).

**2.8.4.7 z-index:**As demonstrated in example-02-05.html, moving elements out of normal flow can lead to *overlap* between boxes (Figure [2.7](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-05)). The z-index: property allows you to control which box appears on top.

The z-index value can be specified with any integer (positive, zero, or negative). An element with greater z-index value is always in front of an element with a lower z-value. Note that the z-index only works on *positioned* elements, i.e., those positioned with position: relative, position: absolute, or position: fixed[17](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fn17).

**2.9 Hurricane scale example**

**2.9.1 HTML contents.**In this section and the next one, we will summarize what we learned about CSS in two practical examples. In the first example, we create a page that visualizes the hurricane scale based on [wind speed](https://en.wikipedia.org/wiki/Saffir–Simpson_scale), going from a tropical depression (\(\leq 17\, \frac{m}{s}\)) to a level-five hurricane (\(\geq 70\, \frac{m}{s}\)). We begin with a plain HTML code of example-02-06.html, without any styling:

<!DOCTYPE html>

**<html>**

**<head>**

**<title>**Hurricane scale**</title>**

**</head>**

**<body>**

**<h1>**Hurricane Scale**</h1>**

**<p** id="five"**>**Five &ge;70 m/s**</p>**

**<p** id="four"**>**Four 58&ndash;70 m/s**</p>**

**<p** id="three"**>**Three 50&ndash;58 m/s**</p>**

**<p** id="two"**>**Two 43&ndash;49 m/s**</p>**

**<p** id="one"**>**One 33&ndash;42 m/s**</p>**

**<p** id="ts"**>**Tropical storm 18&ndash;32 m/s**</p>**

**<p** id="td"**>**Tropical depression &le;17 m/s**</p>**

**<p>**

**<a** href="https://en.wikipedia.org/wiki/

Saffir%E2%80%93Simpson\_scale"**>**(Data Source)

**</a>**

**</p>**

**</body>**

**</html>**

This page has one <h1> heading and seven <p> paragraphs representing the various wind-speed classes. Note that &ge;, &ndash; and &le; are special ways to specify the \(\geq\), – (en dash) and \(\leq\) characters[18](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fn18), respectively, using HTML code. The result does not look very impressive (Figure [2.8](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-06)).

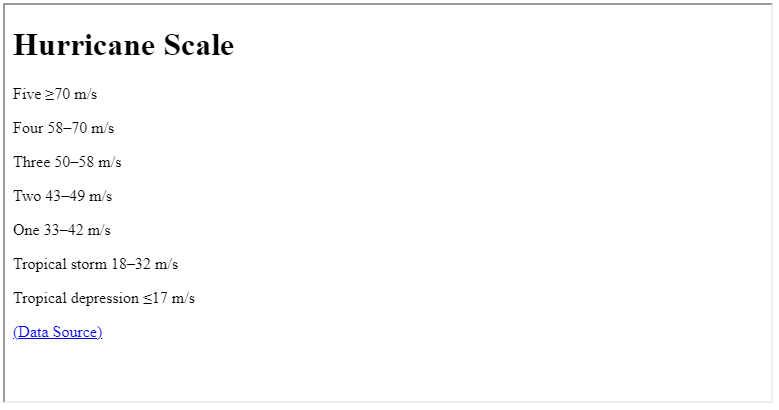


FIGURE 2.8: example-02-06.html [(Click to view this example on its own)](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\examples\example-02-06.html)

**2.9.2 CSS styling.**We can improve this web page by adding graduated colors to the hurricane classes. Here is CSS code that we can use to make the page look more interesting:

body {

**background-color**: silver;

**padding**: 20px;

**font-family**: Arial, Verdana, sans-serif;

}

h1 {

**background-color**: rgba(255,255,255,0.5);

**color**: #64645A;

**padding**: inherit;

}

p {

**padding**: 5px;

**margin**: 0px;

}

#five { **background-color**: #ff6060; }

#four { **background-color**: #ff8f20; }

#three { **background-color**: #ffc140; }

#two { **background-color**: #ffe775; }

#one { **background-color**: #ffffcc; }

#ts { **background-color**: #00faf4; }

#td { **background-color**: #5ebaff; }

* Edit a copy of example-02-06.html and link the previous CSS code to the HTML document. You can use either of the last two methods presented in Section [2.7](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#linking-css-to-html):
  + Embedded CSS—Code goes in the <style> element within the <head> element of the HTML document (Section [2.7.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#embedded-css))
  + External CSS—Code goes in a separate .css file, then linked using the <link> element within the <head> element of the HTML document (Section [2.7.4](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#external-css))
* Remove or modify some of the rules to examine their effect.

The hurricane-scale page, after applying the CSS rules, is given in example-02-07.html (Figure [2.9](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-07)).

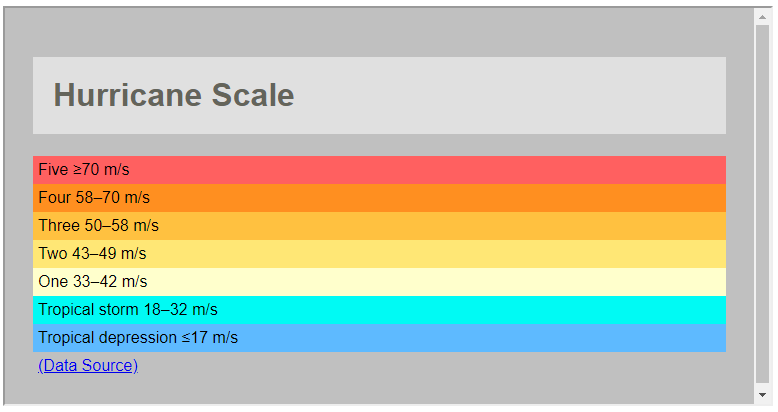


FIGURE 2.9: example-02-07.html [(Click to view this example on its own)](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\examples\example-02-07.html)

**2.10 Map description example**

**2.10.1 HTML contents.**In our second practical CSS example, we are going to create styled title and description panels, which can be used to describe a web map in cases when the map fills the entire screen. You can see the final result we aim at in Figure [2.13](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-11).

We start with an HTML document that has the following content in its <body> element:

**<h1** id="title"**>**Leaflet Quick Start Guide**</h1>**

**<div** id="description"**>**

**<h2>**About this map**</h2>**

**<p>**This is the final result of the

**<a** href="https://leafletjs.com/examples/quick-start/"

target="\_blank"**>**Leaflet Quick Start Guide**</a>**.

The map demonstrates Leaflet basics, such as setting up a Leaflet map,

adding markers, polylines and popups.**</p>**

**<p>**The placement and styling of the title and description boxes is

done using CSS. Check out the source code of this page to see how.**</p>**

**<p>**Map authored by Michael Dorman**</p>**

**</div>**

The HTML code contains:

* An <h1> element, which will comprise the map *title*
* A <div> element, which will comprise the map *description*

The map *title* in the <h1> element is just text, saying Leaflet Quick Start Guide. The map *description* in the <div> element is composed of a smaller title (<h2>) and three paragraphs (<p>). Using the <div> element we bind the entire description into a group (Section [1.6.12.2](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#html-div)) with a specific ID, id="description". The ID will be used to set the description style with CSS. Similarly, the <h1> element is marked with id="title".

The way our HTML page appears—before any CSS styling is applied—is shown in Figure [2.10](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-08). This is quite different from the final result (Figure [2.13](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-11)), but hang on. We will get there using CSS styling.

FIGURE 2.10: example-02-08.html [(Click to view this example on its own)](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\examples\example-02-08.html)

**2.10.2 CSS styling: box position.**Moving on to styling. Assuming our web map is going to fill the entire screen, we would like to position the title and description in boxes “above” the map. As discussed in Section [2.8.4.6](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#position), this means we need the boxes to exit normal flow, using position: absolute or position: fixed. To achieve the modified positioning, as well as specify other aesthetic properties of the boxes (discussed below), we add the following CSS rules in example-02-09.html:

#title {

**position**: fixed;

**top**: 10px;

**left**: 55px;

**margin-top**: 0;

**padding**: 10px 15px;

**background-color**: rgba(255, 255, 255, 0.5);

**border**: 1px solid grey;

**border-radius**: 3px;

}

#description {

**position**: fixed;

**bottom**: 20px;

**left**: 10px;

**width**: 280px;

**padding**: 0px 15px;

**background-color**: rgba(255, 255, 255, 0.7);

**border**: 1px solid grey;

**border-radius**: 3px;

}

In the above CSS code, we are using several concepts:

* The position: fixed property along with top: and left: or bottom: and left:, to put the title and description into a specific position on the page
* The margin-top:, to “disable” the default margin around <h1> elements, so the the box isn’t pushed further down than set with top:
* The padding: property, to make sure there is enough space around the text
* The border: and border-radius:, to make the box border visible and having rounded corners
* The background-color: property to make the background color semi-transparent white

There are also two CSS *shortcuts*, which we have not met up until now. CSS has many types of shortcuts of this sort, and we will encounter several other ones in later chapters:

* Using border: 1px solid grey; as a shortcut for border-width: 1px, border-style: solid and border-color: grey all in one declaration
* Using padding: 10px 15px; as a shortcut for padding-top: 10px, padding-bottom: 10px, padding-right: 15px and padding-left: 15px in one declaration

The result after applying the above CSS rules is shown in Figure [2.11](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-09). As you can see, the title and map description are now in separate bordered boxes, positioned in the specified locations inside the viewport.

FIGURE 2.11: example-02-09.html [(Click to view this example on its own)](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\examples\example-02-09.html)

**2.10.3 CSS styling: final touches.**There are more CSS properties that can be modified to make the boxes even nicer. For example, we can use *specific fonts* (free ones, loaded from a Google server) and change the *link style* when hovering with the mouse. We are not going to learn about these properties, but you are welcome to explore them in example-02-10.html, which gives the slightly modified result shown in Figure [2.12](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fig:example-02-10).

FIGURE 2.12: example-02-10.html [(Click to view this example on its own)](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\examples\example-02-10.html)

**2.10.4 Adding map in the background**

Finally, we are missing the web map itself which the title and description refer to. The final version example-02-11.html adds a web map in the background. If you inspect its source code, you will see that one of the things that has been added in the code is the following <script> element (Section [1.6.4.3](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\html.html#html-script)) in the end of the <body>:

**<script** src="map.js"**></script>**

This <script> element loads a JavaScript code from a separate file named map.js. Another thing that was added in the HTML code is a <div> element that has id="map". The script acts on the <div> element that has id="map", replacing it with an interactive map. The content of the script file map.js is not important for now, as we will learn all about it in Chapter [6](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\leaflet.html#leaflet).

Note that another addition in example-02-11.html is that both the map title and the map description are associated with the following z-index declaration, to make sure they are shown *above* the map (Section [2.8.4.7](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#z-index)):

z-index***: 800;***

FIGURE 2.13: example-02-11.html [(Click to view this example on its own)](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\examples\example-02-11.html)

**2.11 Exercise**

* Think about the data and concept that you would like to use for a web-mapping project of your own.
* Modify the content of the map description from the last example to describe your planned map.

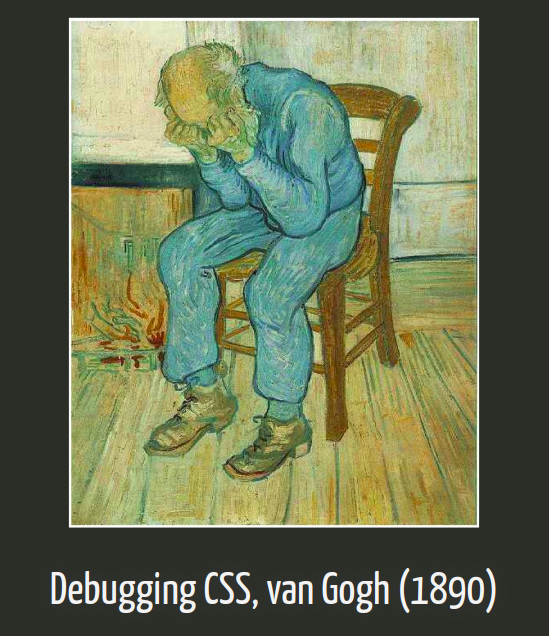


FIGURE 2.14 - Debugging CSS [(Source)](https://slides.yihui.name/2018-blogdown-rstudio-conf-Yihui-Xie.html#33)

1. The term *descendant* refers to an HTML element that is contained within another given HTML element, i.e., “downstream” in the HTML document hierarchy.[↩︎](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fnref8)
2. A more detailed description of CSS conflict resolution and inheritance can be found in the *Cascade and Inheritance* article (<https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Cascade_and_inheritance>) by Mozilla.[↩︎](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fnref9)
3. The *Introduction to CSS* (<https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS>) tutorial and the *CSS Reference* (<https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>) by Mozilla can be referred to for more information on CSS.[↩︎](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fnref10)
4. An interactive demonstration of the HEX and RGB notations can be found in the *Colors HEX* reference (<https://www.w3schools.com/colors/colors_hexadecimal.asp>) by W3Schools.[↩︎](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fnref11)
5. The *Color Reference* (<https://developer.mozilla.org/en-US/docs/Web/CSS/color_value>) by Mozilla gives more examples of specifying color in CSS.[↩︎](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fnref12)
6. More details on the font-family: property can be found in the *font-family* reference (<https://developer.mozilla.org/en-US/docs/Web/CSS/font-family>) by Mozilla.[↩︎](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fnref13)
7. For more information on units of size in CSS, see the *CSS values and units* reference (<https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Values_and_units>) by Mozilla.[↩︎](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fnref14)
8. For a good discussion of the CSS box model, see also the *CSS Positioning* article (<http://www.brainjar.com/css/positioning/>).[↩︎](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fnref15)
9. <https://stackoverflow.com/questions/5657964/css-why-doesn-t-percentage-height-work>[↩︎](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fnref16)
10. More information and examples on positioning can be found in the *Position* article (<https://developer.mozilla.org/en-US/docs/Web/CSS/position>) by Mozilla.[↩︎](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fnref17)
11. See the *UTF-8 Mathematical Operators* (<https://www.w3schools.com/charsets/ref_utf_math.asp>) and *UTF-8 General Punctuation* (<https://www.w3schools.com/charsets/ref_utf_punctuation.asp>) reference for these and other examples[↩︎](file:///C:\Users\User\Downloads\Telegram%20Desktop\Web_Gis\Web_Gis\js\css.html#fnref18)