Code.Hub

The first Hub for Developers

Ztoupis Konstantinos

HTML - CSS

Code.Learn Program:

React

What is the WWW?

- A distributed document delivery system
- Uses a client-server model
- Main presentation language is HTML

Client-Server Model

Two processes (possibly networked):

- The client
  - Sends requests to the server
  - Blocks until reply is received
- The server
  - Processes requests from clients
  - Never blocks
  - Can reply to several clients simultaneously

HTML Background

- Many “markup” languages in the past
- SGML: Standard Generalized Markup Language
    - HTML (Hypertext Markup Language) based on SGML
- XML (eXtensible Markup Language) “replaces” SGML
    - XHTML is replacing HTML

HTML

- Hypertext Markup Language
- Intended to be maximally portable
  - Logical markup
  - Graceful degradation of presentation

HTML Documents

- HTML documents are text documents
  - We use simple ASCII text files
  - Html file extensions: .html or .htm
- You can create html documents using:
  - Notepad in Windows and TextEdit (MAC OS X)
- You can also use HTML Editors

Markup Languages

- Markup:
  - Embedded codes in documents
  - Codes are called `tags`
  - Codes
    - Describe the structure documents
    - Include instructions for processing
- Markup language:
  - Computer language for describing syntax of tags
  - May be used with other tools to specify rendering

Logical Markup

- Describes parts of document
- Does not specify how to render
- Presentation is client’s `decision`
- When client cannot present then there is graceful degradation

HTML

- is a fairly simple language made up of elements
- can be applied to pieces of text to give them different meaning in a document (Is it a paragraph? Is it a bulleted list? Is it part of a table?)
- structure a document into logical sections (Does it have a header? Three columns of content? A navigation menu?)
- embed content such as images and videos into a page

CSS

Cascading Stylesheets or CSS is the first technology you should start learning after HTML. While HTML is used to define the structure and semantics of your content, CSS is used to style it and lay it out. For example, you can use CSS to alter the font, color, size, and spacing of your content, split it into multiple columns, or add animations and other decorative features.

BOOTSTRAP

is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with a responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

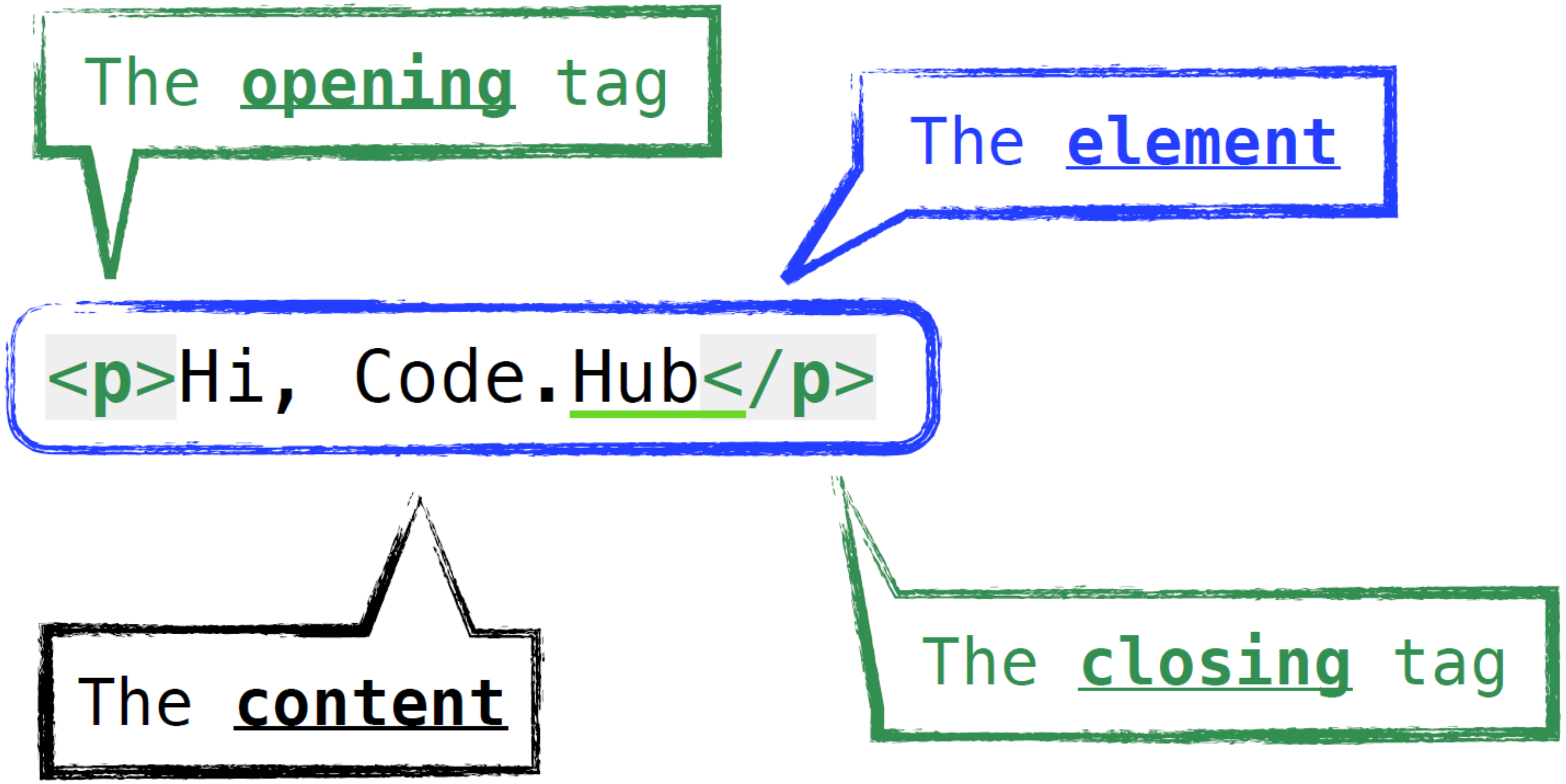
JAVASCRIPT

is a programming language that allows you to implement complex things on web pages. Every time a web page does more than just sit there and display static information for you to look at - displaying timely content updates, or interactive maps, or animated 2D/3D graphics, or scrolling video, and so on - you can bet that JavaScript is probably involved.

HTML element

Anatomy of an HTML element

The paragraph element



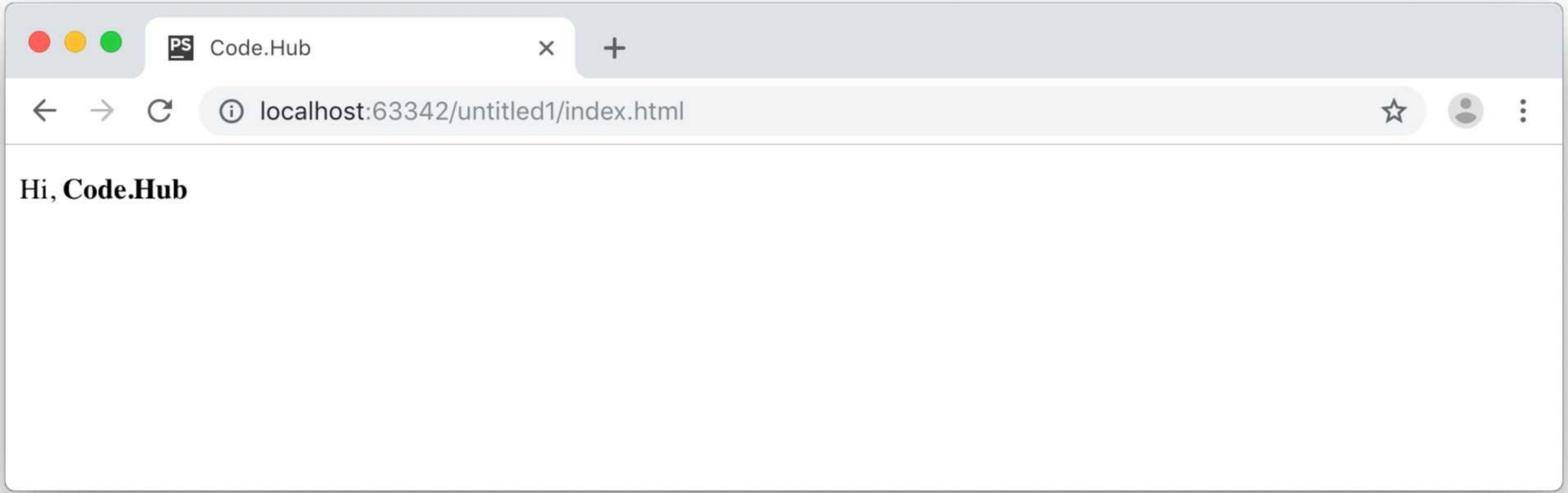
Code.Hub

HTML nested elements

A nested HTML element

```
<p>
    Hi, <strong>Code.Hub</strong>
</p>
```

HTML Document



Code.Hub

localhost:63342/untitled1/index.html

Hi, **Code.Hub**

Code.Hub

Common HTML Elements

```html
<!—Comment—>
<h1>Heading 1</h1> <h2>Heading 2</h2>
<h3>Heading 3</h3> <h4>Heading 4</h4>
<h5>Heading 5</h5> <h6>Heading 6</h6>
<p>Paragraph</p>
<a href="http://www.google.com"
  title="Title of the link">Click here</a>
<span>Plain inline text</span>
<strong>Bold</strong> or <b>Bold</b>
<i>Italics</i>
<em>Emphasis</em>
<br>
<hr>
```

Common HTML Elements

```html
<div>

  <h6>Heading6</h6>

  <p>Paragraph</p>

</div>

<button>Click me!</button>

<iframe src="https://www.w3schools.com"></iframe>

<img src="profile.jpg" alt="John Doe" height="42" width="42">

<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4>

  <source src="movie.ogg" type="video/ogg">

    Your browser does not support the video
</video>
```

Code.Hub

LISTS

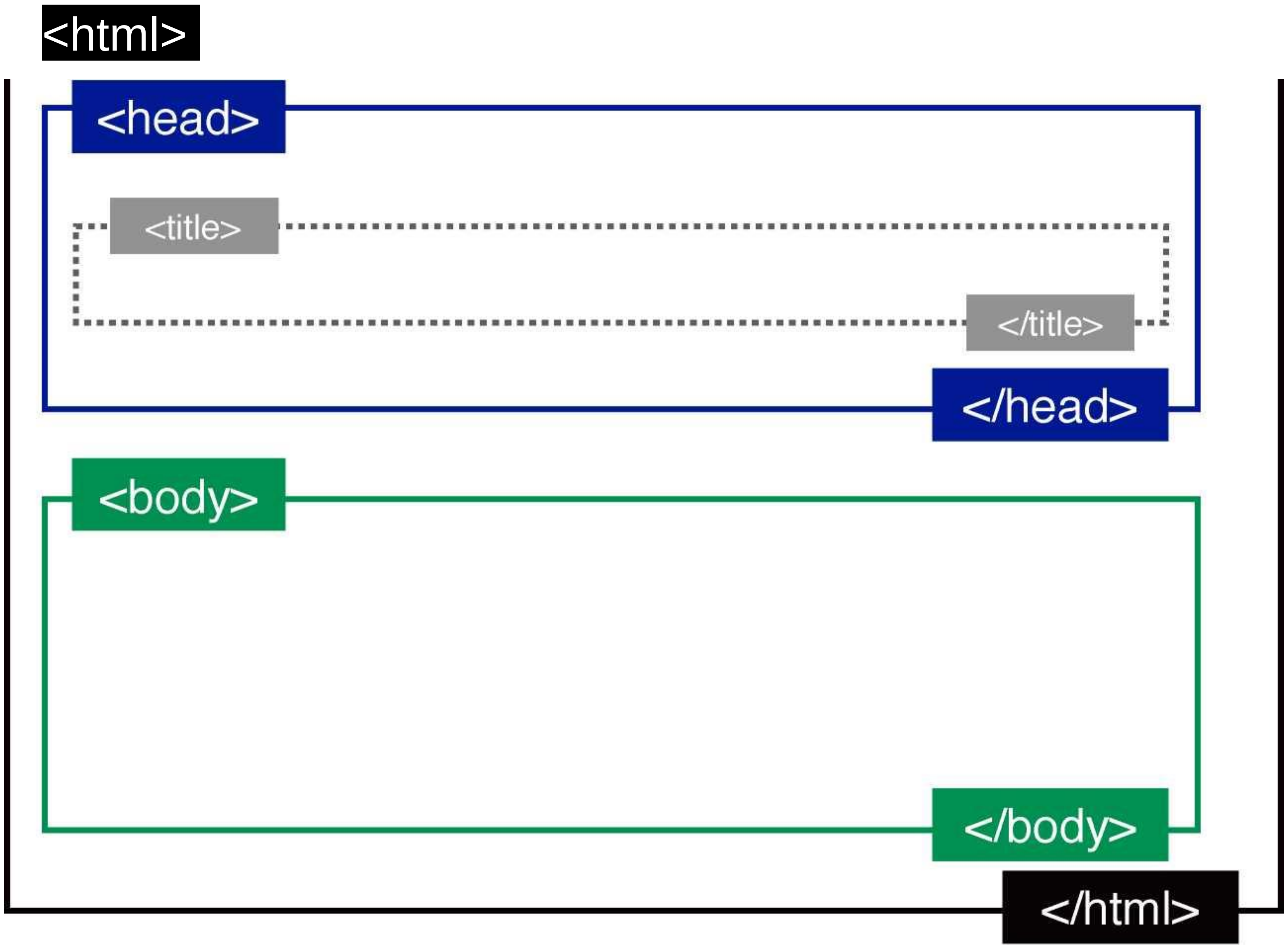
```html
<ul>
  <li>Bullet list</li>
  <li>Bullet list</li>
  <li>Bullet list</li>
</ul>
```

```html
<ol>
  <li>Ordered list with numbers</li>
  <li>Ordered list with numbers</li>
  <li>Ordered list with numbers</li>
</ol>
```

HTML Basics

- 3 Parts HTML document
  - DOCTYPE
    - What DTD are you using
  - Head
    - Meta information
    - Only <title> is required
  - Body
    - Text to render

Contents of an HTML Document



Code.Hub

HTML Basics

- Tags
  - Elements
  - Attributes
- Entities
  - <,>,&,‘ ’
  - Ö,ð,÷,©, etc.
  - See example at CS4173 website
- Comments

Code.Hub

HTML Document

HTML
Document
(index.html)

```
<!DOCTYPE html>
  <html>

  <body>

    <!—this is a paragraph element —>

    <p>Hi, Code.Hub</p>

  </body>

</html>
```

Code.Hub

HTML <!DOCTYPE> Declaration

```
<!DOCTYPE html>
```

- It’s is not an HTML tag
- Indicates what version of HTML the page is written in it goes before the <html> tag
- The latest version is HTML5
- HTML elements/tags supported in HTML5 are listed here

Code.Hub

<html>

```
<!DOCTYPE html>
<html lang="en">
</html>
```

- Tells the browser that it is an HTML document
- Represents the root element of the document
- The container for all other HTML elements

Code.Hub

<head>

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Homepage</title>
  </head>
</html>
```

Provides general information (metadata) about the document, including its title and links to its scripts and style sheets

<head>

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">

    <title>Homepage</title>
    <link rel="stylesheet" type="text/css" href="./styles.css"/>
    <script src="./app.js"></script>
    </head>
</html>
```

Elements that can be used inside:
<title>, <base>, <link>, <style>, <meta>, <script>, <noscript>

Links are self-closed tags

Code.Hub

HTML <meta> tag (viewport)

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="./app.css">
</head>

<body>

     <!-this is a paragraph element-> <p>Hi, Code.Hub</p>

</body>

</html>
```

Code.Hub

<body>

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8M>
  <title>Homepage</title>
</head>
<body>

</body>
</html>
```

Code.Hub

Elements

- Block-level and in-line elements
  - TABLE, P, BLOCKQUOTE, DIV, etc.
  - CODE, Q, H1, SPAN, etc.

- Grouping Elements
  - DIV
  - SPAN

- One-part elements
  - BR, etc

Basic Tags

- Text display:
  - <em>, <strong>
- Structure:
  - <h1>, <h2>, <h3>
  - <p>
  - <ul>, <ol>, <blockquote>
- Attributes:
  - Align, text, bgcolor, etc.

Code.Hub

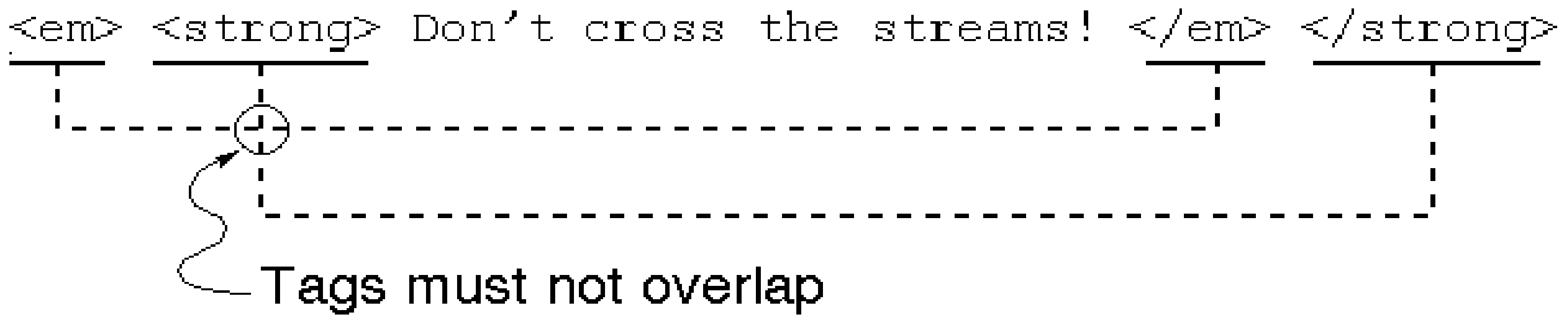
Text Formatting Tags

- <b> **Bold Face** </b>
- <i> *Italics* </i>
- <u> Underline </u>
- <p> New Paragraph </p>
- <br> Next Line

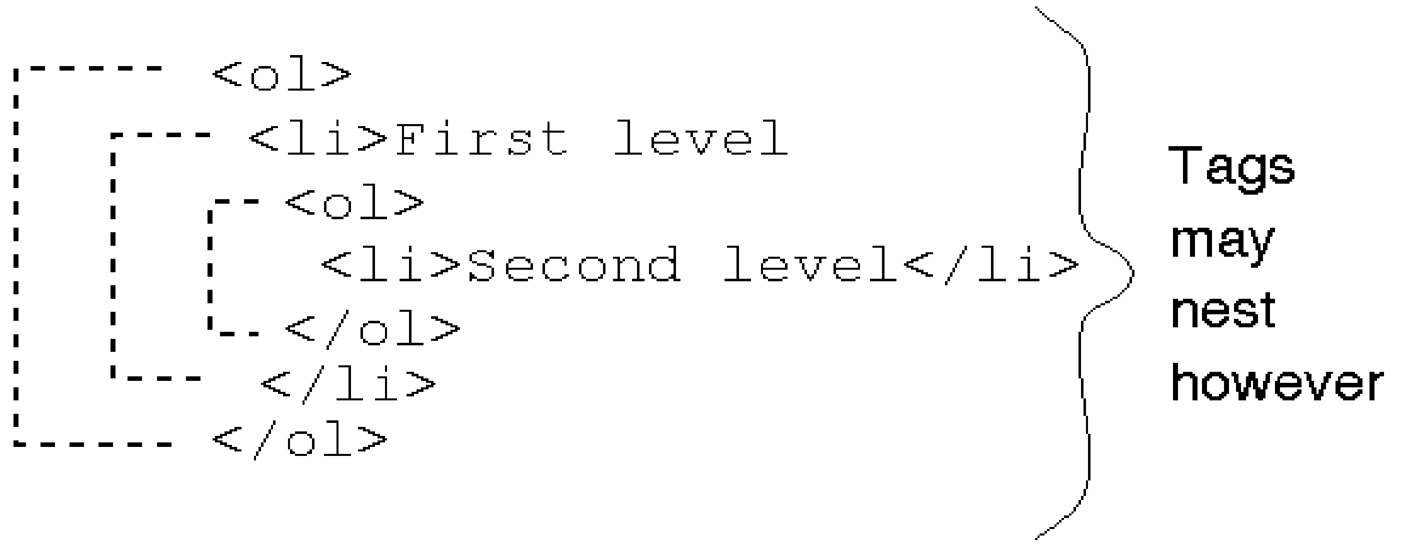
Tags

- Tags may be *nested*
- Tags may not overlap
- What happens when a browser doesn't recognize a tag?

Overlap versus Nesting



```
<em> <strong> Don't cross the streams! </em> </strong>
```



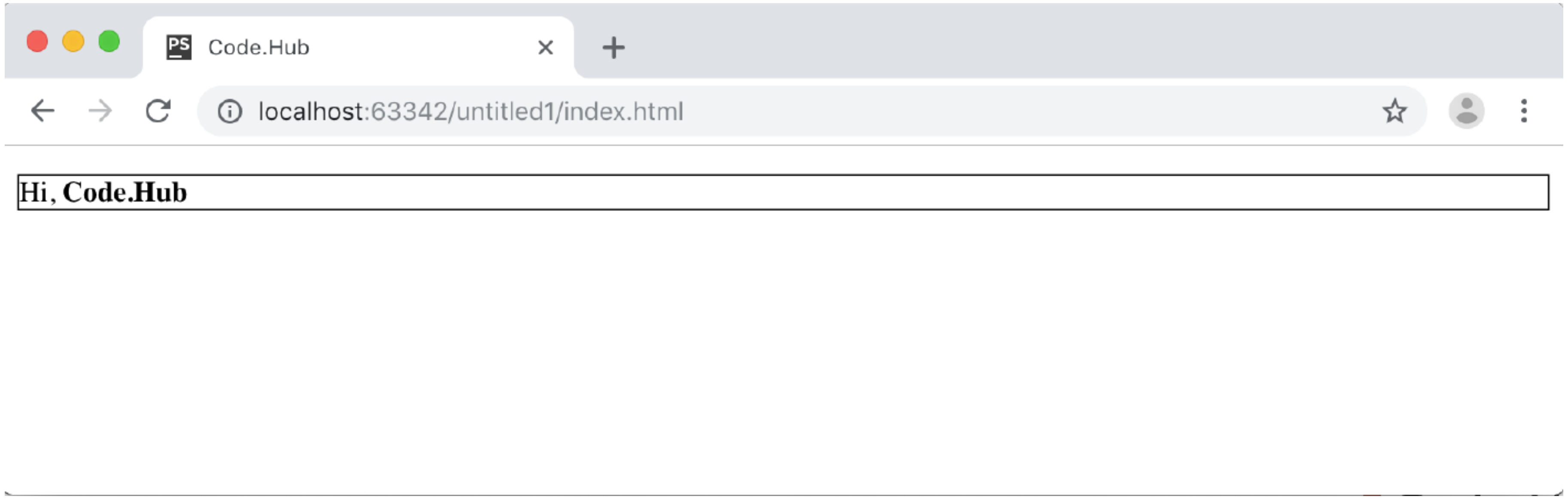
```
<ol>
 <li>First level
  <ol>
   <li>Second level</li>
  </ol>
 </li>
</ol>
```

Links

- 'A' element
  - HREF
  - NAME
  - CLASS
  - REL
  - TYPE
  - TITLE
  - ID
  - STYLE
  - Anchor Text
  - TABINDEX

HTML attributes

```
<p style="border: 1px solid black;">
  Hi, <strong>Code.Hub</strong>
</p>
```



Hi, **Code.Hub**

Code.Hub

Global attributes

```html
<section id="video">
  <h1 class="red">Video title</h1>
  <button title="Music video">Play</button>
</section>
```

Code.Hub

data-* attributes

```html
<section data-video-id="2459">
  <video src="./video.mp4"></video>
  <button title="Music video">Play</button>
</section>
```

- The data-* attributes gives us the ability to embed custom data attributes on all HTML elements.
- Usually, the stored (custom) data can be used in the page's JavaScript to create a more engaging user experience.
- The data-* attributes consist of two parts:
  - The attribute name should not contain any uppercase letters, and must be at least one character long after the prefix "data-"
  - The attribute value can be any string

Element specific attributes

```html
<img src="image.jpg" width="500" height="600">
```

- All HTML elements can have attributes
- Attributes are always specified in the start tag
- Attributes usually come in name/value pairs like: **name="value"**

Forms

- What are forms?
    - An HTML form is an area of the document that allows users to enter information into fields

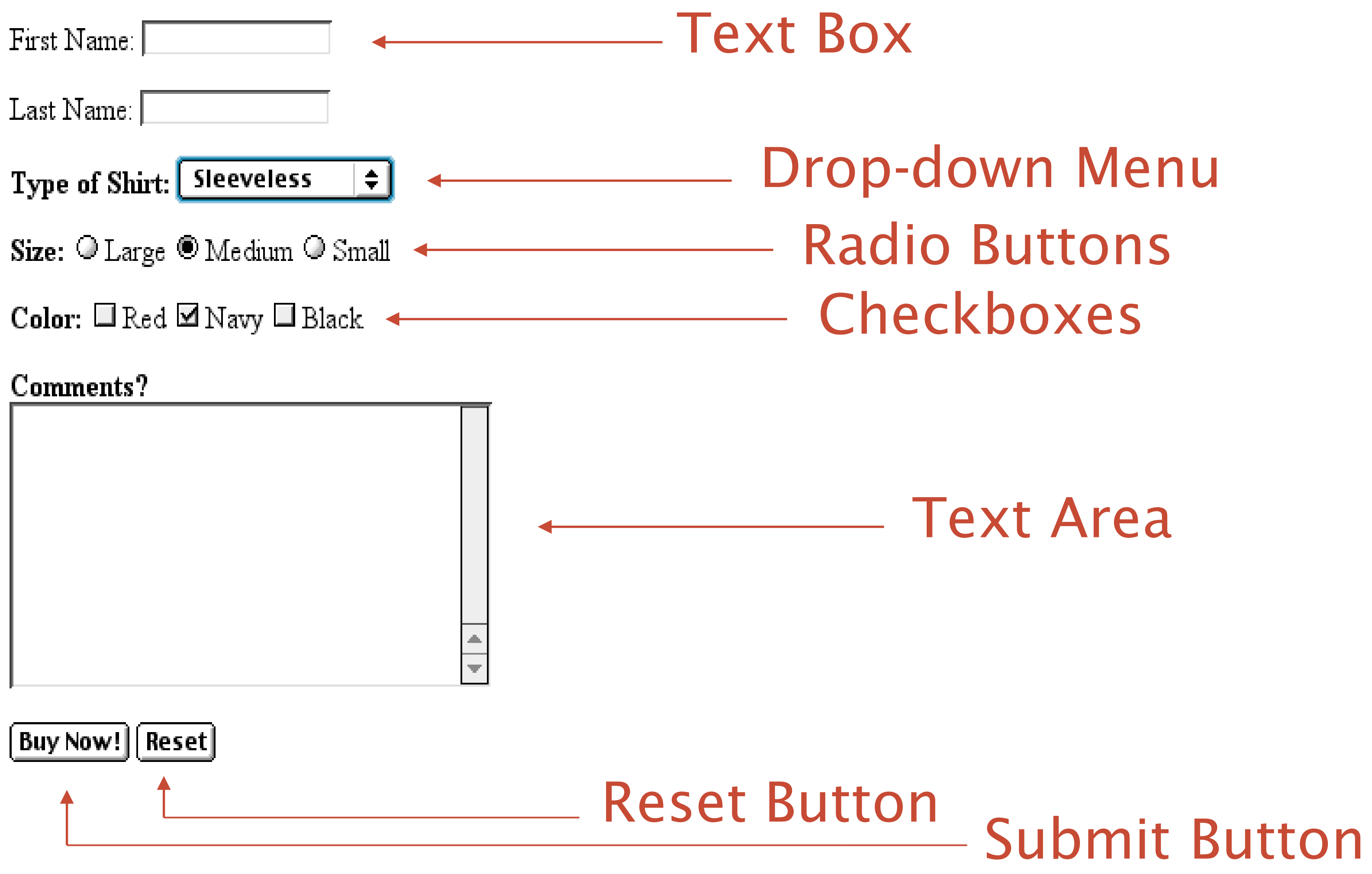
    - A form may be used to collect personal information, opinions in polls, user preferences and other kinds of information

Forms

- There are two basic components of a Web form: the shell, the part that the user fills out, and the script which processes the information

- HTML tags are used to create the form shell. Using HTML you can create text boxes, radio buttons, checkboxes, drop-down menus, and more...

Example: Form



First Name:

Last Name:

Type of Shirt: Sleeveless

Size: Large [✓ Medium] Small

Color: Red [✓ Navy] Black

Comments?

The Form Shell

A form shell has three important parts:

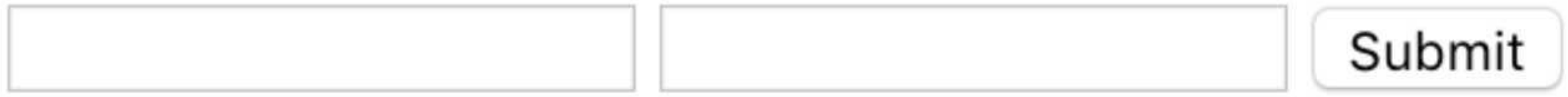
- the <form> tag, which includes the address of the script which will process the form
- the form elements, like text boxes and radio buttons
- the submit button which triggers the script to send the entered information to the server

Forms

```
<form id="contact">
  <input type="text" name="firstname" value="">
  <input type="text" name="lastname" value="">
  <input type="submit" value="Submit">
</form>
```

Simple form which will send a GET request to the server

Request URL: (......)/index.html?firstname=John&lastname=Doe



HTTP Requests

HTTP is a protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes.

A client (e.g. browser) opens a connection to make a request, then waiting until it receives a response.

HTTP Requests

Common HTTP Methods:

- *GET* is used to send data to a server
- *POST* is used to **create/update** a resource
- *PATCH* is used to partially **update** a resource
- *PUT* is used to **update** a resource
- *DELETE* is used to **delete** a resource

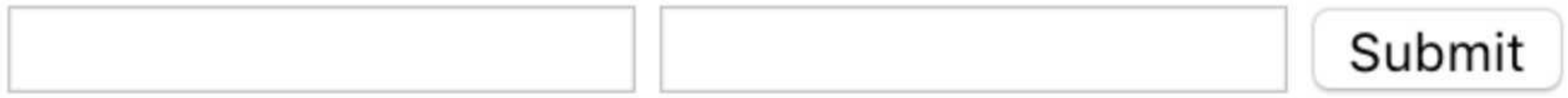
Code.Hub

Forms

```html
<form id="contact" action="/submit" method="get" >
  <input type="text" name="firstname" value="">
  <input type="text" name="lastname" value="">
  <input type="submit" value="Submit">
</form>
```

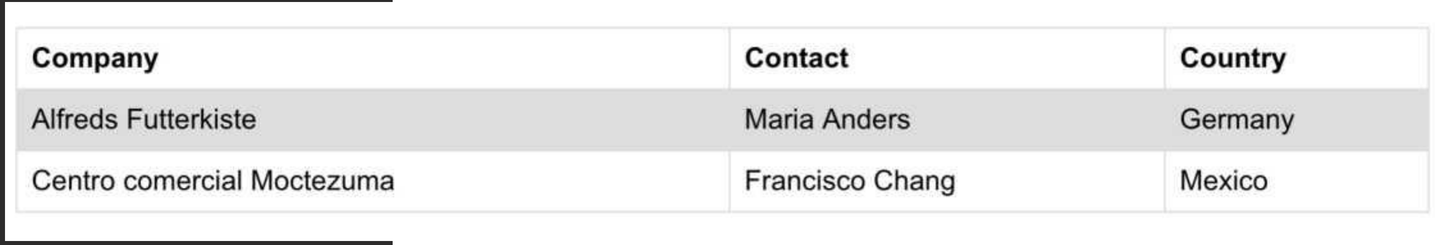
Simple form which will send a GET request

Request URL: (......)/submit?firstname=John&lastname=Doe



Tables

```
<table>
   <tr>
      <th>Company</th>
      <th>Contact</th>
      <th>Country</th>
   </tr>
   <tr>
      <td>Alfreds Futterkiste</td>
      <td>Maria Anders</td>
      <td>Germany</td>
   </tr>
   <tr>
      <td>Centro comercial Moctezuma</td>
      <td>Francisco Chang</td>
      <td>Mexico</td>
   </tr>
</table>
```



Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico

Code.Hub

Layout semantic elements

```
<header>
  <nav>
    <ul>
      <li>Home</ul>
      <li>Blog</ul>
      <li>Contact</li>
    </ul>
  </nav>
</header>
```

```
<section>
  <div></div>
</section>
```

```
<aside>
  <div></div>
</aside>
```

```
<main>
  <article></article>
</main>
```

```
<footer>
  <div></div>
</footer>
```

Code.Hub

Media/Objects

```html
<img src="./image.jpg" alt="An image title">

<video src="./video.mp4"></video>

<iframe src="https://www.w3schools.com"></iframe>
```

Comments

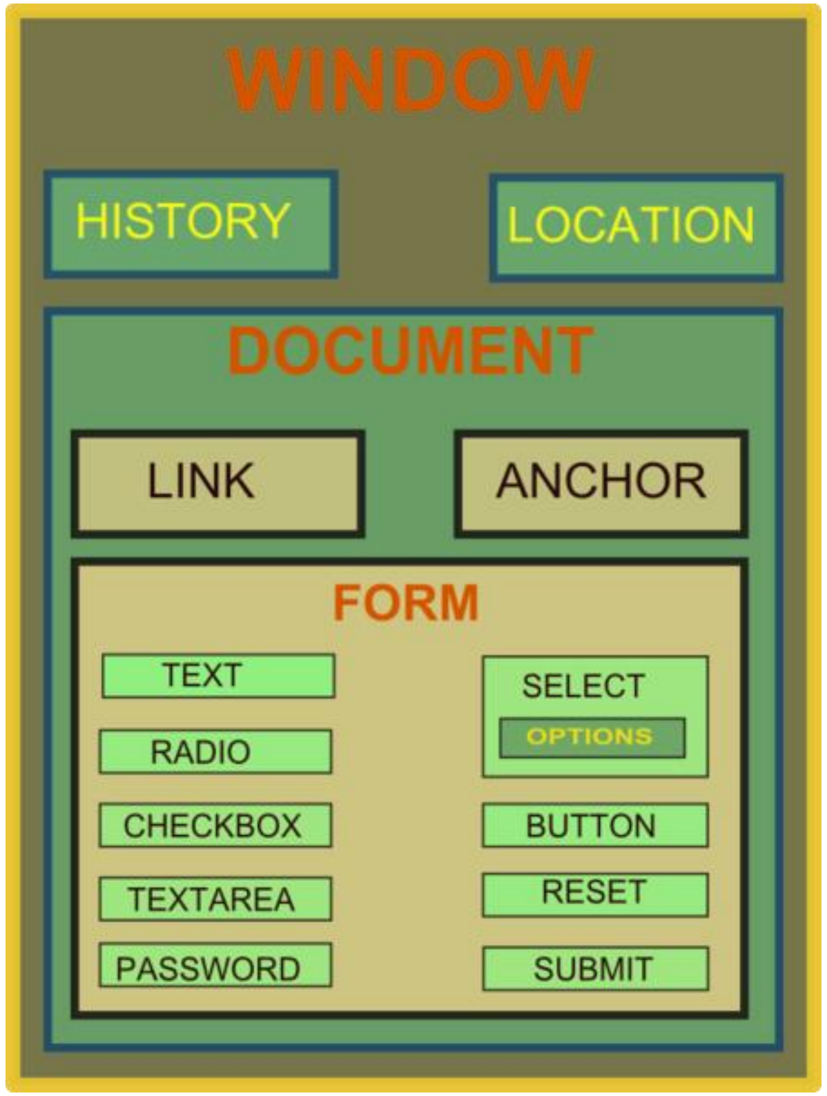
- Browser will NOT display text in between
  - <!-- This is a comment   -->

  - <!-- This is another
-                  comment
  -                -->
- I. E. uses the following tag as a comment:
  - <comment>   this a comment   </comment>

Document Object Model (DOM)

- An model for describing HTML documents (and XML documents)
    - A standard (ok, standards)
    - Independent of browser, language
    - A common set of properties/methods to access everything in a web document
- APIs in JavaScript, for Java, etc.

DOM


WINDOW
HISTORY
LOCATION
DOCUMENT
LINK
ANCHOR
FORM
TEXT
RADIO
CHECKBOX
TEXTAREA
PASSWORD
SELECT
OPTIONS
BUTTON
RESET
SUBMIT

What are Cascading Style Sheets?

- Stands for **C**ascading **S**tyle **S**heets
- Defines the visual style and appearance of our web pages
- Describes how HTML elements are to be displayed on screen, paper, or in other media
- **Saves a lot of work.** It can control the layout of multiple web pages all at once
- Uses selectors, properties and values to achieve its purpose

HISTORY OF CSS

- HTML was NEVER intended to contain tags for formatting a web page! It was created to **describe the content** of a web page.
- When tags like <font>, and color attributes were added to the HTML 3.2, it started a *nightmare* for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.
- CSS removed the style formatting from the HTML page!
- The style definitions are normally saved in external .css files: you can change the look of an entire website by changing just one file!

What are Cascading Style Sheets?

- Why the term “cascading”? In CSS, multiple styles can be applied to a particular document (usually a web page or XML file). The browser will interpret these styles in a top-down (*Cascading)* fashion:
  - Style rules set up site-wide are overridden by styles located within individual pages
  - Style rules located within individual pages are overridden by styles inside an individual tag
  - In addition, the end user can set up styles in the browser that will override the author’s styles

What are Cascading Style Sheets?

All matching rules for a particular selector will be applied, except where they conflict with each other. If rules are in conflict, the last rule to be declared is applied.

Why learn CSS?

- Control
- Customization
- Do something unique
- You want to understand why something in your web site doesn’t look right in browser, and fix it

Pros and Cons of Using CSS

- Pros
  - Greater designer control of the appearance of the page
  - Easier management of site-wide changes
  - Greater accessibility to web sites by non-graphical browsers and web-page-reading software

- Cons
  - Different browsers may interpret Style Sheets in different ways
  - Some styles may not be seen at all on some browsers

How to use CSS

- Connecting CSS to HTML
- Linked stylesheets
- Embedded stylesheets
- Inline style declarations

Code.Hub

Linked Stylesheets

```
<link rel="stylesheet" type="text/css" href="mystyle.css">
```

- The preferred way. An external .css file contains your CSS code, and can be referenced by multiple HTML documents. Maximizes reusability
- HTML document can reference multiple .css files, and take on styling from each of them
- <link> goes in the <head> element

Code.Hub

Embedded Stylesheets

```
<style><!--
/*your CSS here */
-->
<style>
```

- Not very reusable, since the CSS rules can only be utilized by the HTML document it is embedded within.
- Useful in the event that you need to make sure the HTML file stays with its CSS markup.
- Devious “css injection” possibilities… (don’t allow <style> tag in web forums, or users will do bad things to your site.)
- <style> is w3c-valid only in <head> but most browsers will accept it in <body>

Inline style attribute

```
style="declaration1; declaration2; "
```

```
<p style="font-size:10pt; color:red;"
```

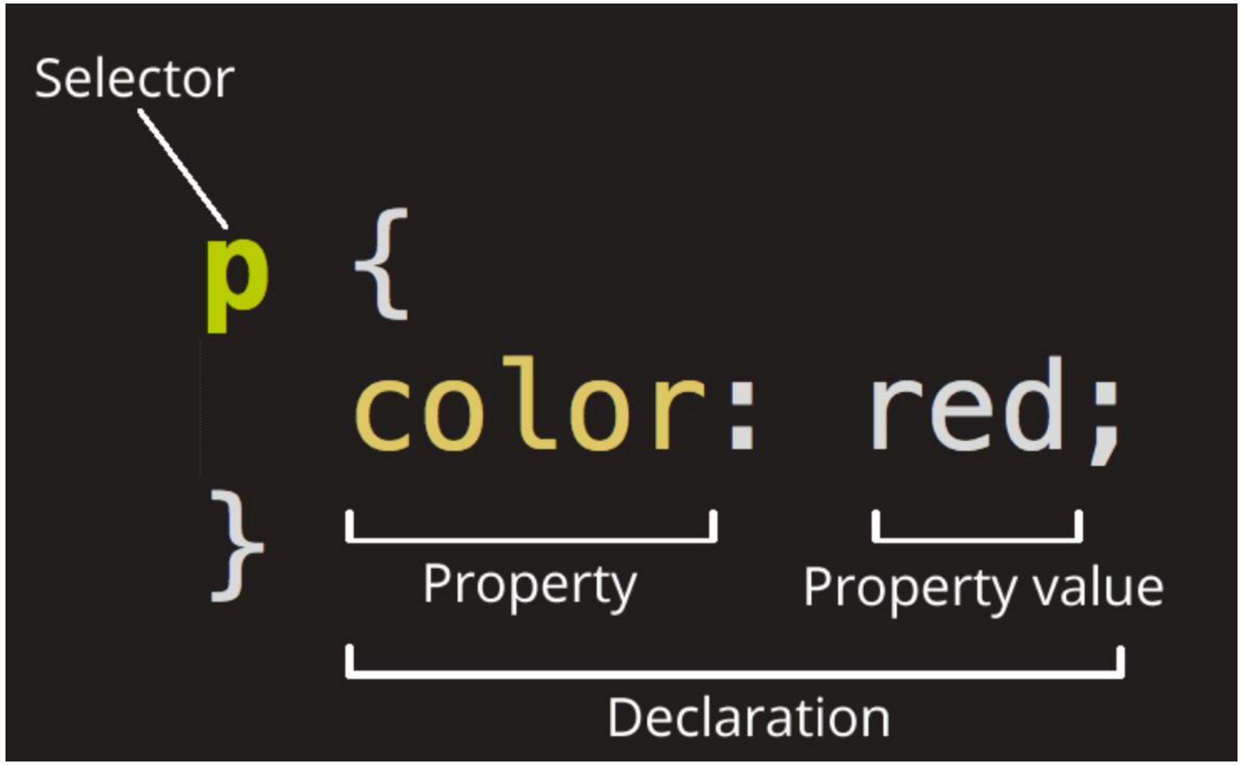
- Nearly any HTML element can take a style attribute
- Inline styles are more specific; that is, they will override rules declared in external or embedded stylesheets.
- Inline styles are good for ad hoc or temporary solutions, but at the expense of the re-usability of the HTML/content. They're fine for posting comments to discussion boards.

CSS document

```css
/* styles.css */


body {
   background: white;
   color: blue;
}
```

Code.Hub

Anatomy of a CSS ruleset



```
p {
color: red;
}
```

Code.Hub

Why take a class?

- Advanced CSS can be pretty hard
- Complex CSS can be hard to work with
- Dealing with browser inconsistencies can be a pain
- Keeping up to date with CSS as it develops takes effort as well as patience

Selector Strings

- Single element type:

```
p {
  font-size: smaller;
  letter-spacing: 1em;
}
```

- Single element type:

```
p, span, a {
  color: red;
}
```

- All element types:

```
* { font-weight: bold; }
```

- Specific elements by id:

```
#video-19 {
  background: blue;
}
```

Code.Hub

Selector Strings

- Elements belonging to a style class:

```
#p4, .takeNote {font-style: italic;}
```

  - Referencing a style class in HTML:

```
<span class="takeNote special cool">
```

- Elements of a certain type and class:

```
#span.special {font-style: x-large;}
```

Selector Strings

- Source anchor elements:

```css
a:link {color: black;}
a:visited {color: yellow;}
a:hover {color: green;}
a:active {color: red;}
```

- Element types that are descendents:

```css
ul lo li {letter-spacing: 1em;}
```

Different types of selectors

```
p { color: blue; }

#video-19 { background: blue; }

.video-wrapper { background: blue; }

[data-video-player], [src] { background: blue; }

* { font-weight: bold; }
a:hover { color: purple; }

div:first-child { color: purple; }

div:nth-child(3) { color: purple; }

div:last-child { color: purple; }
```

- Element selector (sometimes called a tag or type selector)
- ID selector
- Class selector
- Attribute selector
- Pseudo-class selector

Code.Hub

Combining selectors

```css
/* All descendants */
form input[type=text] {
  background-color: yellow;
}
/* Immediately following selector */
div + .video-player {
  background-color: yellow;
}
/* Direct descendants, children */
#video-89 > .video-container {
  background-color: yellow;
}
```

Code.Hub

CSS Specificity

```
/* Start at 0,
add 1000 for style attribute,
add 100 for each ID,
add 10 for each attribute, class or pseudo-class,

add 1 for each element name or pseudo-element. */

h1 {
  color: blue;
}

#title h1 {
  color: red;
}

/*
  <div id="title">
    <h1 style="color: green">Heading</h1>
  </div>
*/
```

Code.Hub

CSS Rule Cascade

Specificity of selector: more specific selectors will override more general ones. Pseudo-elements and pseudo-classes are counted as normal elements and classes, respectively.

CSS Rule Cascade

Specificity:

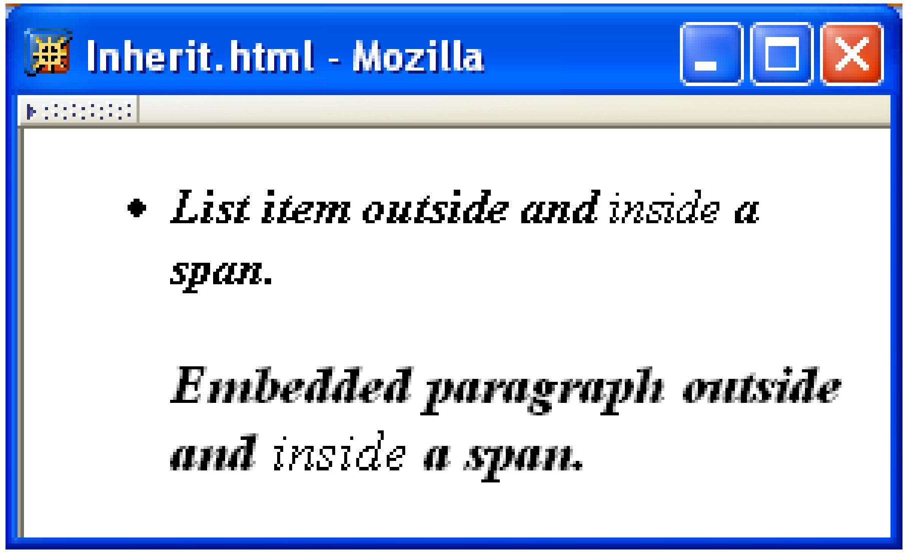
1. style attribute
2. rule with selector:
    1. ID
    2. class/pseudo-class
    3. descendant/element type
    4. Universal
3. HTML attribute

CSS Inheritance

- What if no style declaration applies to a property of an element?
- Generally, the property value is inherited from the nearest ancestor element that has a value for the property
- If no ancestor has a value (or the property does not inherit) then CSS defines an initial value that is used

CSS Inheritance

```css
body {font-weight: bold;}
li {font-style: italic;}
p {font-size: larger;}
span {font-weight: normal;}
```



- *List item outside and* inside *a span.*

  Embedded paragraph outside and inside *a span.*

```html
<body>
  <ul>
    <li>
      List item outside and <span>inside</span> a span.
      <p>
        Embedded paragraph outside and <span>inside</span> a span
      </p>
    <li>
  </ul>
</body>
```

Fonts and text

```
<head>
    <meta charset="utf-8">
    <title>My test page</title>

    <link rel="stylesheet" type="text/css"
href="https://fonts.googleapis.com/css?family=Open+Sans">

    <link rel="stylesheet" type="text/css" href="./styles.css">
    <script src="./app.js"></script>
</head>
```

Font

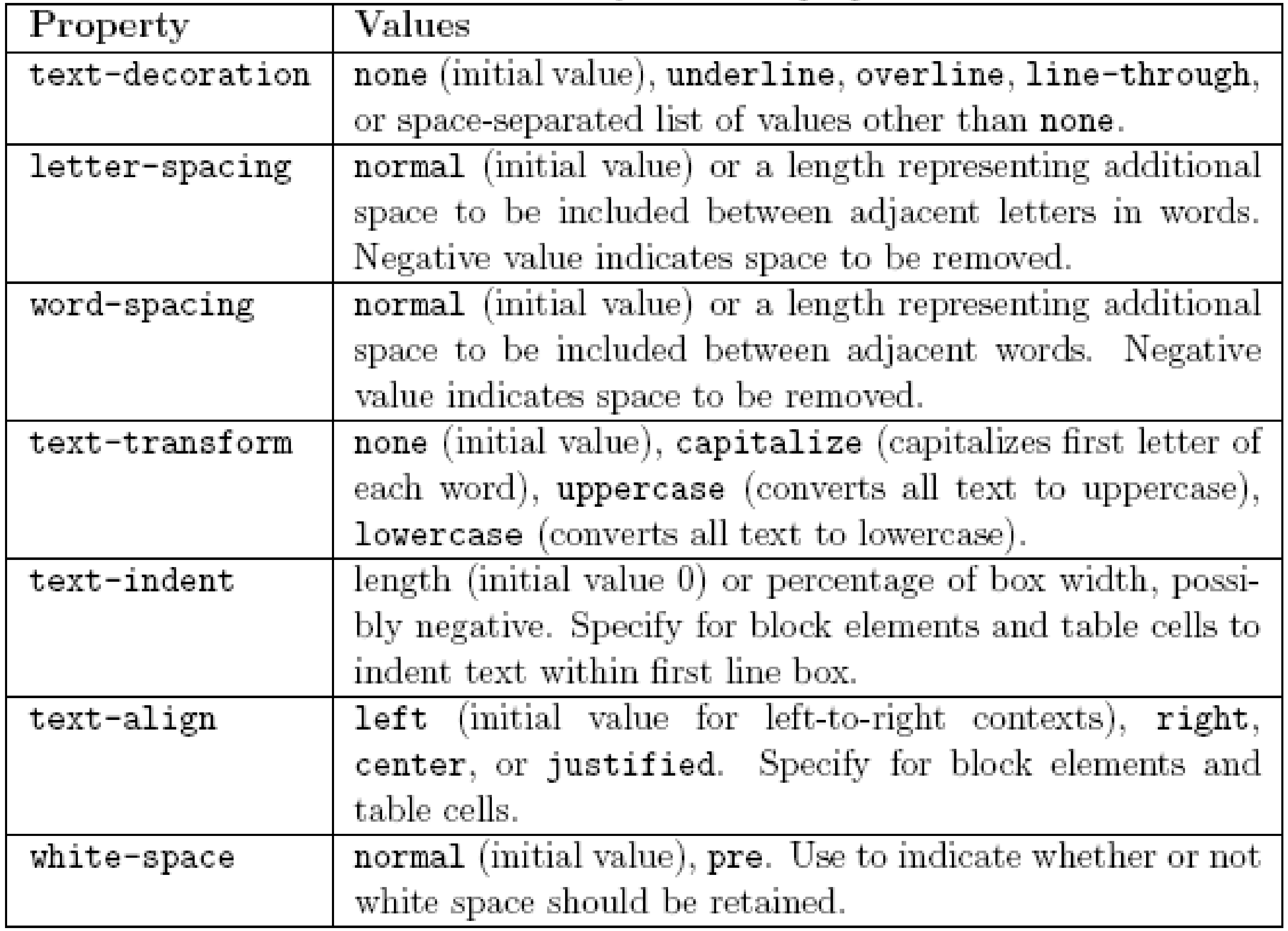
```css
html {
   font-size: 10px;
   font-family: "Open Sans", sans-serif;
   line-height: 1.4;

   font-style: italic;
   font-weight: bold;
}
```

CSS Text properties

```
p {
    color: blue;
    text-align: center;
    text-decoration: underline;
    text-transform: uppercase;
}
```

CSS Text Formatting

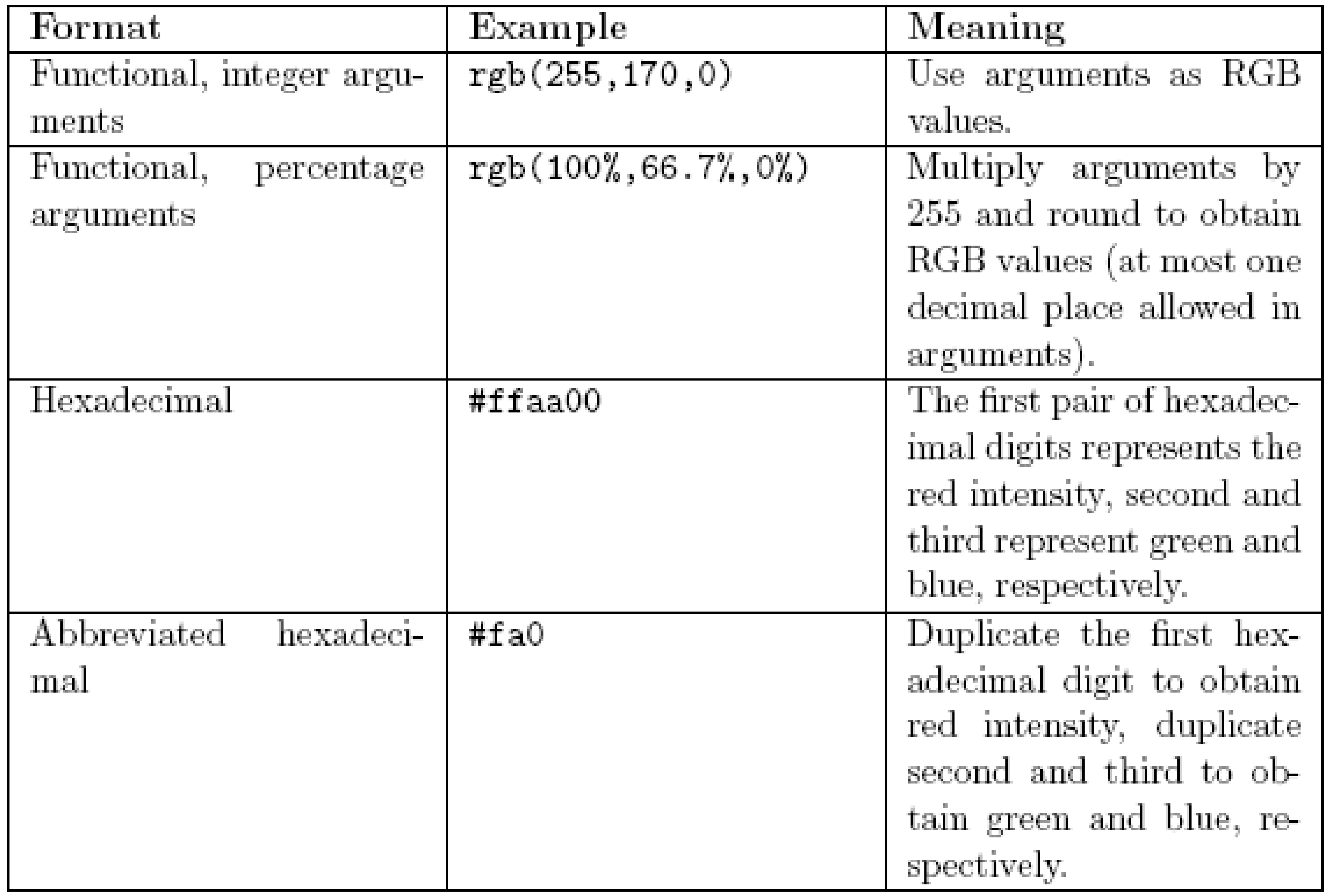
TABLE 3.6: Primary CSS text properties.



Property	Values
`text-decoration`	`none` (initial value), `underline`, `overline`, `line-through`, or space-separated list of values other than `none`.
`letter-spacing`	`normal` (initial value) or a length representing additional space to be included between adjacent letters in words. Negative value indicates space to be removed.
`word-spacing`	`normal` (initial value) or a length representing additional space to be included between adjacent words. Negative value indicates space to be removed.
`text-transform`	`none` (initial value), `capitalize` (capitalizes first letter of each word), `uppercase` (converts all text to uppercase), `lowercase` (converts all text to lowercase).
`text-indent`	length (initial value 0) or percentage of box width, possibly negative. Specify for block elements and table cells to indent text within first line box.
`text-align`	`left` (initial value for left-to-right contexts), `right`, `center`, or `justified`. Specify for block elements and table cells.
`white-space`	`normal` (initial value), `pre`. Use to indicate whether or not white space should be retained.

CSS Text Color

TABLE 3.7: Alternative formats for specifying numeric color values.



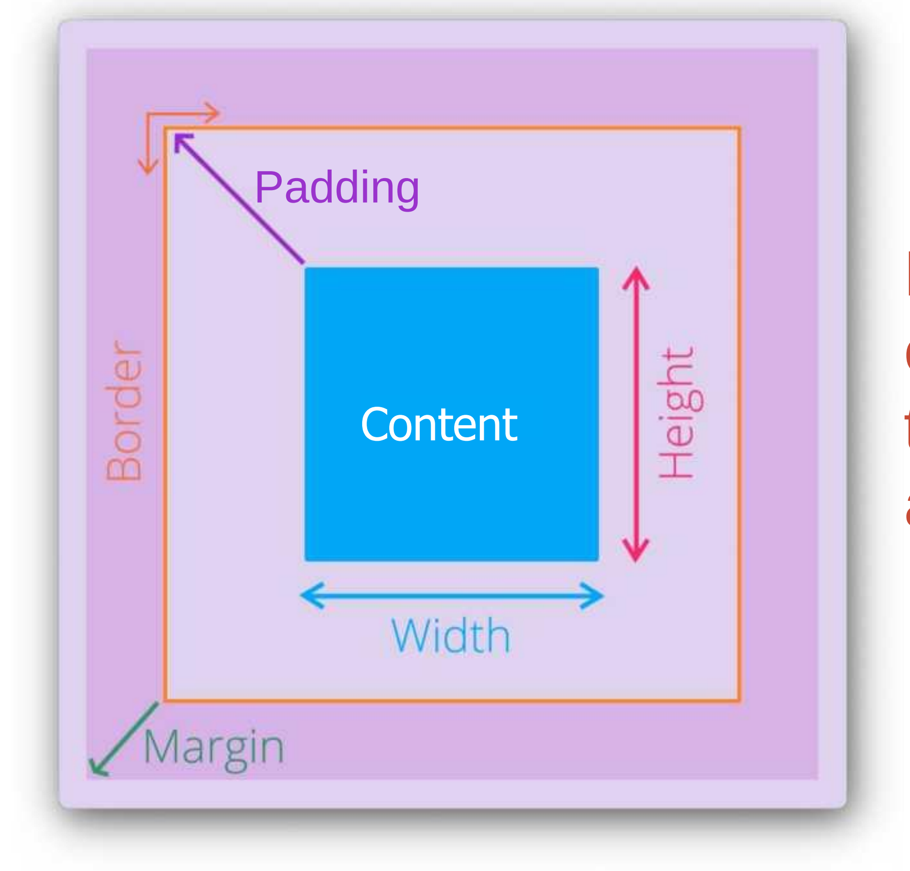
Format	Example	Meaning
Functional, integer arguments	`rgb(255,170,0)`	Use arguments as RGB values.
Functional, percentage arguments	`rgb(100%,66.7%,0%)`	Multiply arguments by 255 and round to obtain RGB values (at most one decimal place allowed in arguments).
Hexadecimal	`#ffaa00`	The first pair of hexadecimal digits represents the red intensity, second and third represent green and blue, respectively.
Abbreviated hexadecimal	`#fa0`	Duplicate the first hexadecimal digit to obtain red intensity, duplicate second and third to obtain green and blue, respectively.

CSS distance units

Absolute	
2px	pixels
1mm	millimeters
2cm	centimeters
0.2in	inches
3pt	printer point 1/72 inch

Relative	
2em	2 times the element's current font size
3rem	3 times the root element's current font size

Structure of CSS Box Model



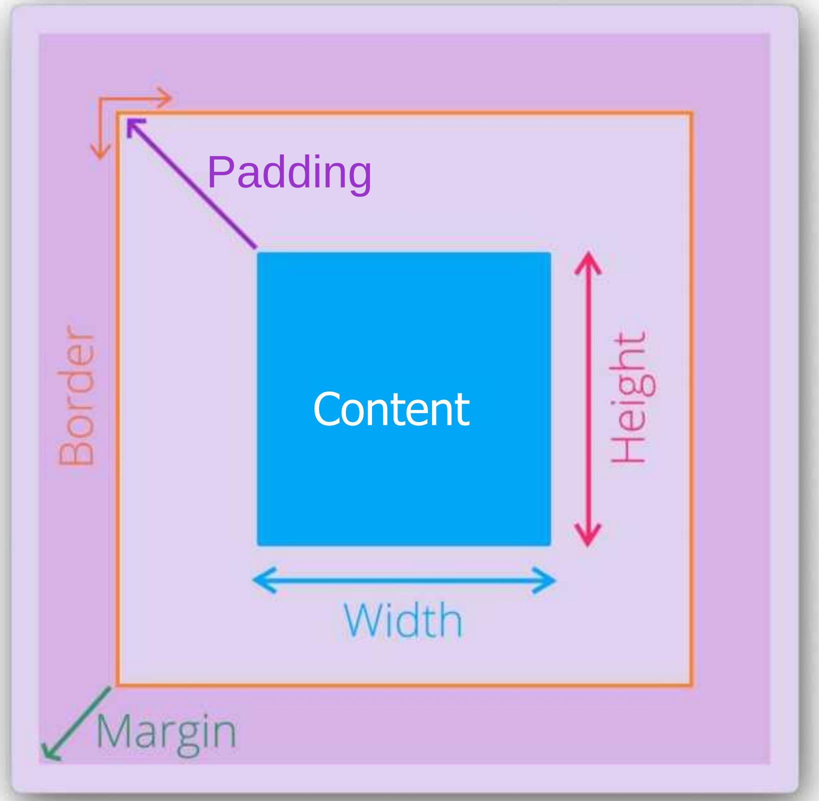
Every piece of content on a web page, or in other words, every HTML element is formed by a rectangle. Each of these rectangles has 5 main aspects that it's built around

Size Properties

width: Override element defaults
height
padding-top
padding-right
padding-bottom
padding-left
margin-top
margin-right
margin-bottom
margin-left

border-bottom-color
border-bottom-style
border-bottom-width
border-left-color
border-left-style
border-left-width
border-right-color
border-right-style
border-right-width

Inline vs block level elements



For inline elements, width and height gets ignored

Code.Hub

CSS Background properties

```css
background: white;
background: rgb(255, 255, 255);
background: #FFFFFF;
background: black url("./image.png") no-repeat center-right;
background-image: url("./image.png");
background-position: center right;
background-size: cover;
background-color: red;
```

CSS Border properties

```
border: 1px solid #0D3349;
border-width: 1px;
border-style: solid;
border-color: #0D3349;


border-top: none;
border-left: none;
border-width: 0 4px 5px 1px;

border-radius: 30px;
```

CSS Width/Height properties

```
width: 100px;
width: auto;
width: 80%;

width: 100vm; /* Relative to 1% of the width of the viewport */
width: 100vmin; /* Relative to 1% of viewport’s smaller dimension (width
or height) */
width: 100vmax; /* Relative to 1% of viewport’s larger dimension (width
or height) */
height: 100px;
height: auto;
height: 80%;

height: 100vh; /* Relative to 1% of the height of the viewport */
```

Position property

Position property	
position: static;	(default) - Position in document flow
position: relative;	Position relative to default position via top, right, bottom, and left properties
position: fixed;	Position to a fixed location on the screen via top, right, bottom, and left properties
position: absolute;	Position relative to ancestor absolute element via top, right, bottom, and left properties

Fixed position (0,0) is top left corner

Code.Hub

Element visibility control properties

Element visibility control properties	
display: none;	Element is not displayed and takes no space in layout
display: inline;	Element is treated as an inline element
display: block;	Element is treated as a block element
display: flex;	Element is treated as a flex container
display: grid;	Element is treated as a grid container

visibility: hidden;	Element is hidden but space still allocated
visibility: visible;	Element is normally displayed

CSS Alignment Cases

```
div {
    width: 100px;
    height: 200px;
    margin: 0 auto;
}

div {
  width: 100px;
  height: 200px;
  float: right;
}
```

CSS Alignment Cases

```css
div {
  width: 100px;
  height 200px;
  float: left;
}

div {
  display: inline-block;
  vertical-align: middle;
}
```

Flexbox and Grid layout

- display: flex; (Flexbox)
- display: grid; (Grid) new layout methods
    - Items flex to fill additional space and shrink to fit into smaller spaces
    - Useful for web app layout:
        - Divide up the available space equally among a bunch of elements
        - Align of different sizes easily
        - Key to handling different window and display sizes

Flexbox and Grid layout

- Flexbox - Layout one dimension (row or column) of elements
- Grid - Layout in two dimensions (rows and columns) of elements

Code.Hub

How to use Bootstrap? (CDN)

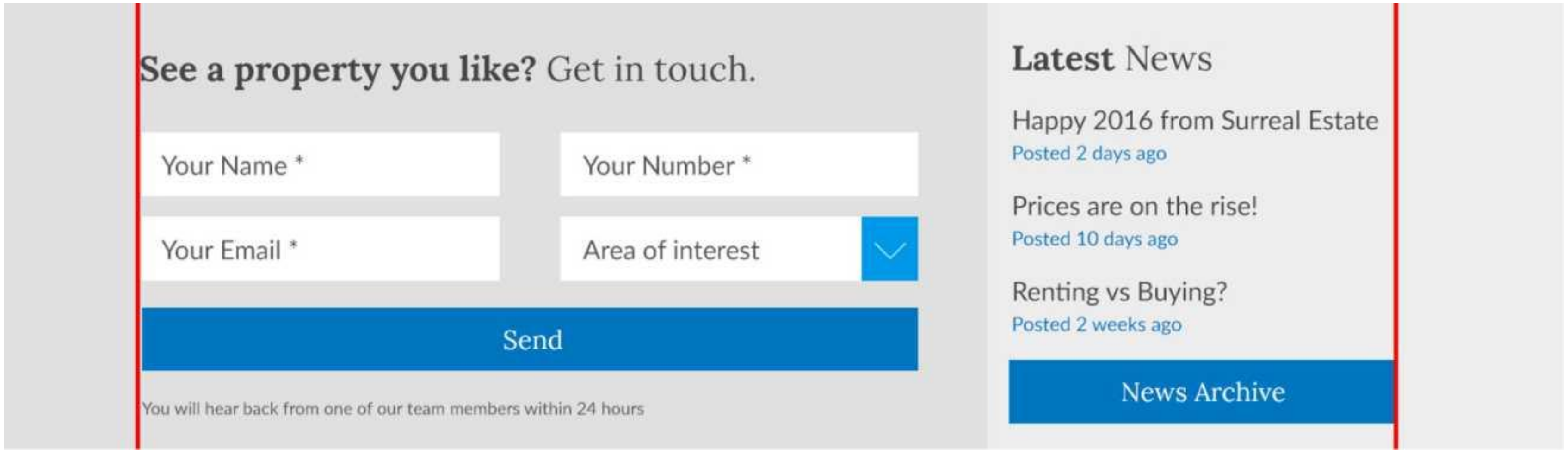
```
<head>
   <!–CDN–>
   <link rel="stylesheet" href="https://
maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/
bootstrap.min.css">
</head>
<body>
  <script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/
4.3.1/js/bootstrap.min.js"></sc ript>
</body>
```

How to use Bootstrap? (locally)

```html
<head>
    <!-locally–>
    <link href="./css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
   <script src="./js/bootstrap.min.js"></script> </body>
```

Container

Bootstrap requires a containing element to wrap site contents and house our grid system



See a property you like? Get in touch.

Your Name *

Your Number *

Your Email *

Area of interest

Send

You will hear back from one of our team members within 24 hours

Latest News

Happy 2016 from Surreal Estate
Posted 2 days ago

Prices are on the rise!
Posted 10 days ago

Renting vs Buying?
Posted 2 weeks ago

News Archive

Container

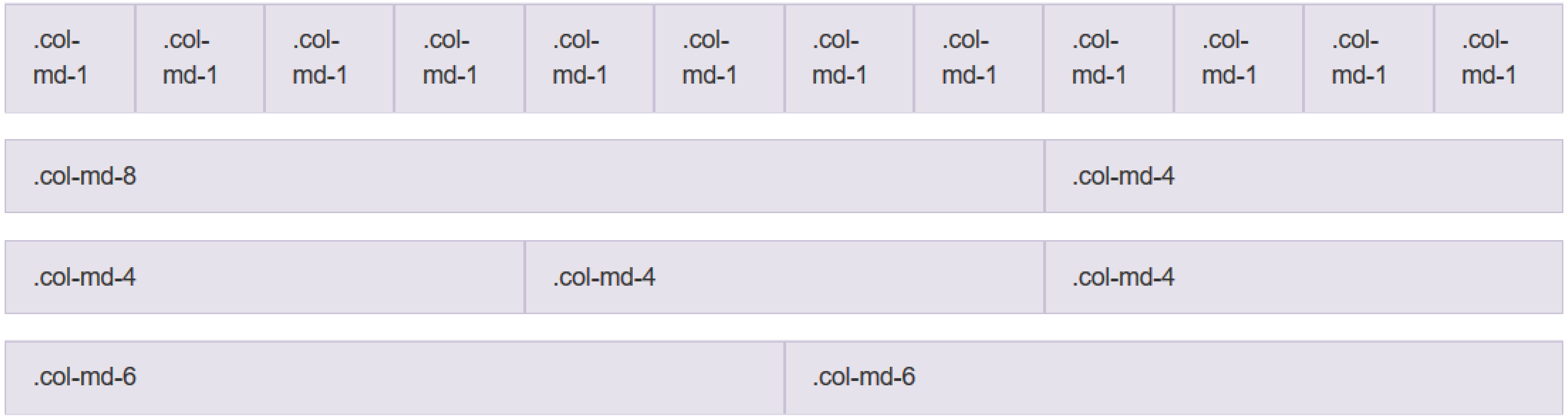
Bootstrap requires a containing element to wrap site contents and house our grid system

```html
<div class="container">…</div>

<div class="container-fluid">…</div>
```

Code.Hub

Grid System

Bootstrap includes a responsive, mobile first fluid grid system that appropriately scales up to 12 columns.



Code.Hub

Grid System

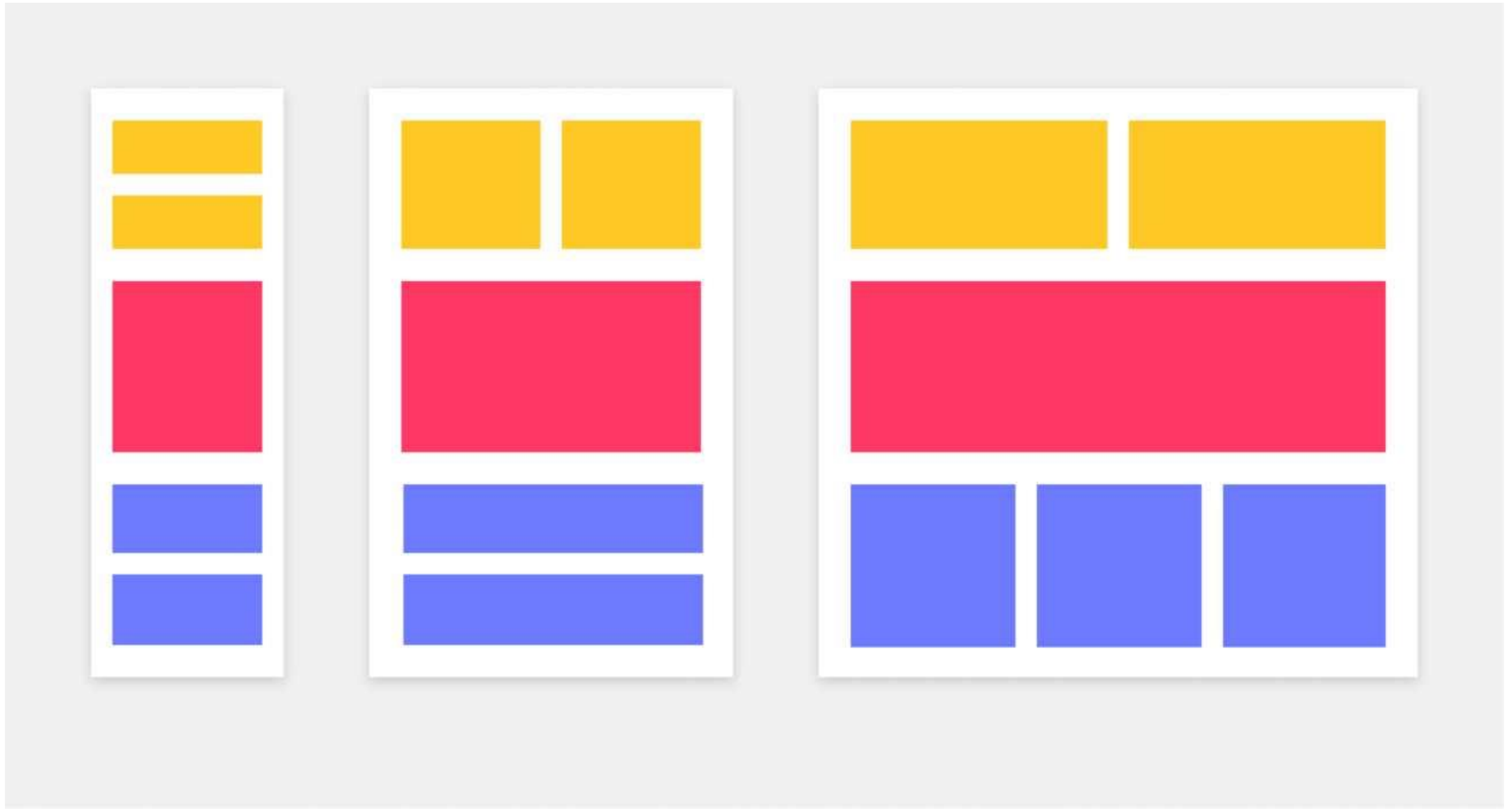
```
<div class="container">
    <div class="row">
      <div class="col-lg-5"></div>
      <div class="col-lg-3"></div>
      <div class="col-lg-4"></div>
    </div>
</div>
```

- Bootstrap includes a responsive, mobile first fluid grid system that appropriately scales up to 12 columns

- Use rows to create horizontal groups of columns

- Grid columns are created by specifying the number of twelve available columns you wish to span

Responsive Grid System



Grid System - Viewports

```
<div class="container">
   <div class="row">
      <div class="col-12 col-sm-6 col-md-8"> Mobile
         12 columns,
         Tablet 6 columns,
         Desktop 8 columns,
      </div>
   </div>
</div>
```

User appropriate column prefix to change the column size for specific viewports (Mobile, Tablet, Desktop)

Code.Hub

Grid System - Class prefix

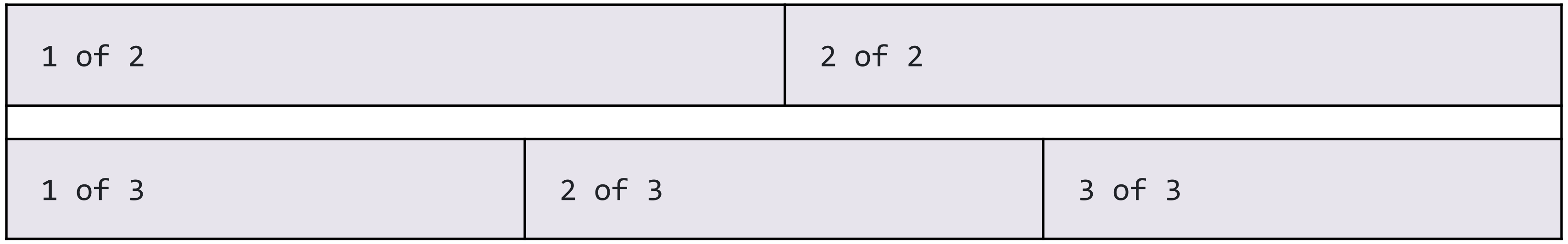
	Extra small	Small	Medium	Large	Extra large
	<576px	>576px	>768px	>992px	>1200px
Max container width	`None (auto)`	540px	720px	960px	1140px
Class prefix	. col-	.col-sm-	.col-md-	.col-lg-	.col-xl-

Code.Hub

Auto-layout columns

```html
<div class="row">
  <div class="col-12 col-sm">
    1 of 2
  </div>
  <div class="col-12 col-sm">
    2 of 2
  </div>
</div>
```

```html
<div class="row">
  <div class="col">
     1 of 3
  </div>
  <div class="col">
    2 of 3
  </div>
  <div class="col">
    3 of 3
  </div>
</div>
```



Nesting columns

```
<div class="container">
    <div class="row">
        <div class="col-12 col-sm-6 col-md-8">

            <div class="row">
                <div class="col-12 col-sm-6 col-md-8">

                </div>
            </div>

        </div>
    </div>
</div>
```

Code.Hub

Text format

```
<p class="text-left">Left aligned text.</p>
<p class="text-center">Center aligned text.</p>
<p class="text-right">Right aligned text.</p>
<p class="text-justify">Justified text.</p>
<p class="text-nowrap">No wrap text.</p>
<p class="text-lowercase">Lowercased text.</p>
<p class="text-uppercase">Uppercased text.</p>
<p class="text-capitalize">Capitalized text.</p>
```

Code.Hub

Text color styles

Muted: This text is grayed out.

Primary: Please read the instructions carefully before proceeding.

Success: Your message has been sent successfully.

Info: You must agree with the terms and conditions to complete the sign up process.

Warning: There was a problem with your network connection.

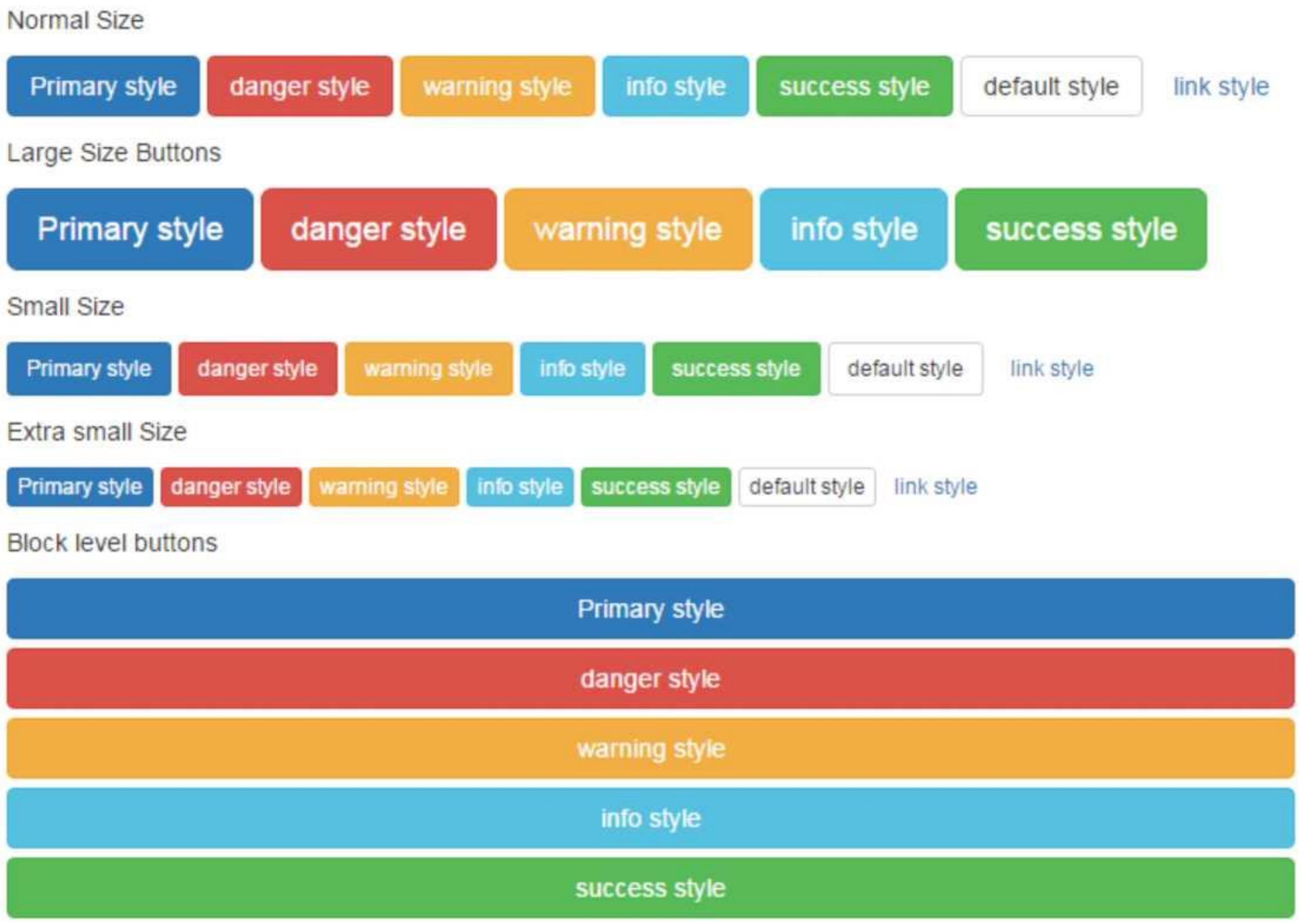
Danger: An error has been occurred while submitting your data.

Text color styles

```html
<p class="text-muted">...</p>
<p class="text-primary">...</p>
<p class="text-success">...</p>
<p class="text-info">...</p>
<p class="text-warning">...</p>
<p class="text-danger">...</p>
```

Code.Hub

Buttons



Code.Hub

Buttons styles

```
<!– Standard button –>
<button type="button" class="btn btn-default">Default</button>
<!– Provides extra visual weight and identifies the primary action in a
set of buttons –>
<button type="button" class="btn btn-primary btn-lg">Primary</button>
<!– Indicates a successful or positive action –>
<button type="button" class="btn btn-success">Success</button>
```

Buttons styles

```
<!– Contextual button for informational alert messages –>
<button type="button" class="btn btn-info">Info</button>

<!– Indicates caution should be taken with this action –>
<button type="button" class="btn btn-warning">Warning</button>

<!– Indicates a dangerous or potentially negative action –>
<button type="button" class="btn btn-danger">Danger</button>

<!– Deemphasize a button by making it look like a link while maintaining
button behavior –>
<button type="button" class="btn btn-link">Link</button>
```

Code.Hub

Buttons sizes

```
<p>
  <button class="btn btn-primary btn-lg">Large button</button>
  <button class="btn btn-default btn-lg">Large button</button>
</p>
<p>
  <button class="btn btn-primary">Default button</button>
  <button class="btn btn-default">Default button</button>
</p>
<p>
  <button class="btn btn-primary btn-sm">Small button</button>
  <button class="btn btn-default btn-sm">Small button</button>
</p>
<p>
  <button class="btn btn-primary btn-xs">Extra small button</button>
  <button class="btn btn-default btn-xs">Extra small button</button>
</p>
```

Forms

```
<form action="./sign-in">
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1"
  placeholder="Email">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control"
  id="exampleInputPassword1" placeholder="password">
  </div>
  <button type="submit" class="btn btn-primary btn-lg">Submit</button>
</form>
```