Day 7

What is HTML?

- Acronym for Hypertext Markup Language
- It is a language used to create web pages and web applications
- A web page is created by using HTML elements to markup (annotate) the page
  - HTML elements are tags enclosed in `<tag_name>`
  - The W3C defines a standard set of HTML elements
  - Eg `<h1>` The following is a heading `</h1>`
- Hypertext are links to other text, documents, images, etc
  - The links are known as hyperlinks
  - Allows a web page markup with HTML elements to link to other pages
- Browsers are used to render the documents
  - Actually rendering engines in the browser renders the web pages
    - Eg Chrome – Webkit, Firefox – Gecko
  - HTML documents are not WYSIWYG

HTML Document Structure

<head> Element

- The **<head>** element in a HTML document contains information on the document
  - Title, author, license, copyright, etc
  - External resources like stylesheet and JavaScripts
  - Meta information such as description, document keywords
  - Caching information
  - Instructions to search bots
- Some of the tags that can be used inside <head>
  - <title>
  - <meta>
  - <link>
  - <style> and <script>

Document Body

- **<body>** element contains the contents of the HTML document
- <h1> to <h6> defines headings
  - The headings are place between a pair <h1>
- <p> is used to make a group of statements as a paragraph
- Use
 to add a line break
  - HTML renderes do not honour line breaks in your document
  - Eg. If you have 2 sentences on 2 different lines, the HTML rendering engine will just display them one after another
- Multiple white spaces between words are treated as a single white space
- To draw a line, use the <hr/> tag. You typically draw a line between headings of different sections

Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Therefore play</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <h1>All work and no play makes Jack a dull boy</h1>
    <p>All work and no play makes Jack a dull boy.
```

LINEBREAK

```
  All work and no play
```

SPACE

```
  makes Jack a dull boy.
    </p>
    <hr/>
    <p>All work and no play makes Jack a dull boy. <br/>
    All work and no play makes Jack a dull boy.
    </p>
    <hr/>
    <h2>All work and no play makes Jack a dull boy</h2>
    <h3>All work and no play makes Jack a dull boy</h3>
    <h4>All work and no play makes Jack a dull boy</h4>
    <h5>All work and no play makes Jack a dull boy</h5>
    <h6>All work and no play makes Jack a dull boy</h6>
  </body>
</html>
```

Rendering engine will not honor these

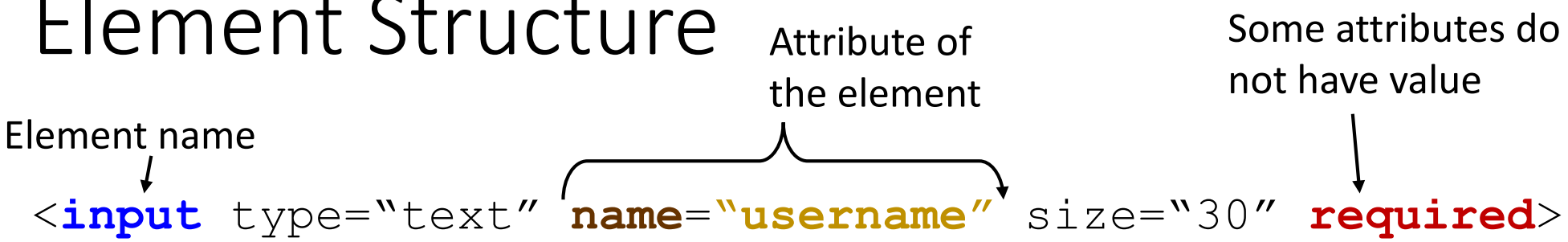
Text Formatting

- Format the text with the following tags
  - `<em>` - emphasize the text/bold it
  - `<i>` - italics
  - `<u>` - underline
  - `<del>` - strike through

```
Big <em>black bug<em> bleeds <del>blue</del> black blood
```

- Better to use CSS to format text, HTML to create the structure of the document

Element Structure



- Element name is the first word following the opening (**<**)
- An element can be followed by zero or more attributes
  - Attributes are key value pair
  - The value is quoted " "
  - Attributes are used to 'configure' the element
  - Some attributes are boolean attributes viz. they do not have value
- Some elements required a closing tag others don't
  - Requires closing so the effects of the tag do not 'leak'
    - Eg. `<p>...</p>`, `<div>...</div>`
  - Examples of tag that have no closing tags eg. `<br>`, `<input>`

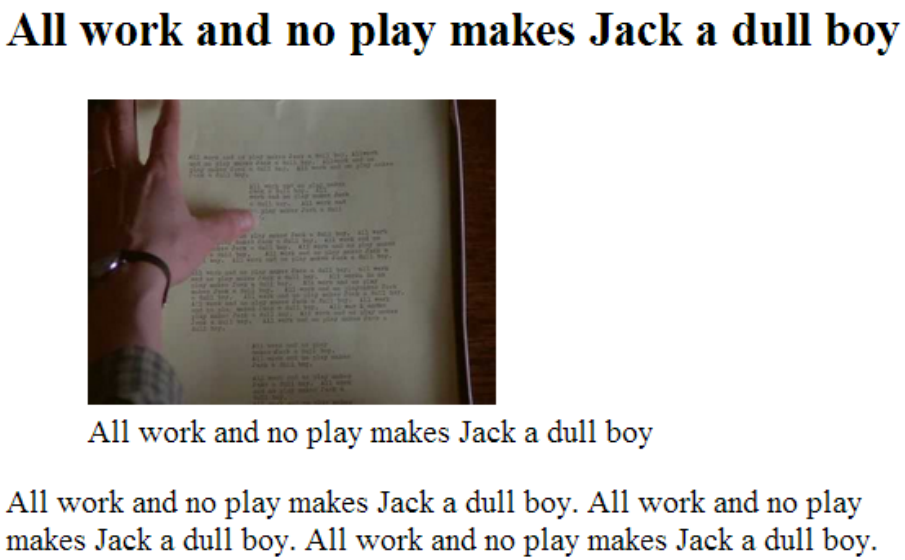
Ordered and Unordered List

- `<ol>` and `<ul>` creates ordered and unordered list respectively
- Each `<li>` within them creates an element of the list
- `type` attribute allows you to change the ‘bullet point’ eg to roman numeral
  - `type=“i”`
  - Not recommended. For styling use CSS

```
<ol type="i">
  <li>All work and no play makes
Jack a dull boy.</li>
   <li>All <b>work</b> and no play
makes Jack a
   <i>dull</i> boy.</li>
  <li>All <strong>work</strong> and
no play makes
   Jack a <em>dull</em> boy.</li>
</ol>
```

Images

- Images are displayed using the `<img>` tag
  - The `src` attribute to specify the location of the image
- The image can be scaled with **`height`** and **`width`** attribute
- Optionally you can enclose a image within a `<figure>` tag
  - A `<figcaption>` can be used inside a `<figure>` tag and is used to add a caption to the figure

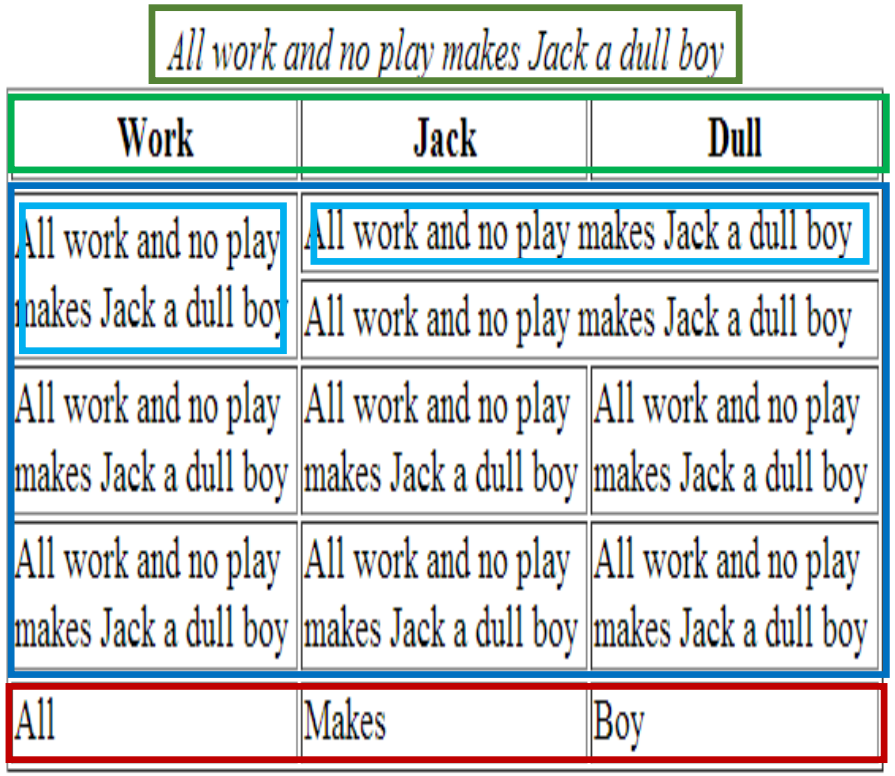



```
<h2>All work and no play makes Jack a
dull boy</h2>
  <p><figure>
      <img src="http://bit.ly/lIBAAL"
height="150"/>
      <figcaption>All work and no play
makes Jack a dull boy</figcaption>
    </figure>
    All work and no play makes Jack a
dull boy. All work and no play makes
Jack a dull boy. All work and no play
makes Jack a dull boy.
  </p>
```

Table

- Tables are created with **<html>** table
- The contents of a table can be divided into 3 portions
  - The header and footer enclosed in a `<thead>` and `<tfoot>` respectively. Optional
    - Table headers are defined with `<th>`
  - The body enclosed in a `<tbody>`
- Every table row is enclosed in a **<tr>** element
- Table columns are denoted by **<td>**
  - Placed inside a `<tr>`
  - If a column occupies more than one column, then use the `colspan` attribute to specify the number of columns that a column should fill to its right
  - If a column occupies more than on row, then use the `rowspan` attribute to specify the number of rows that a column should fill downwards
- A **<caption>** creates a caption for the table. Optional
  - Caption is placed inside a `<table>` element

Example



All work and no play makes Jack a dull boy

Work	Jack	Dull
All work and no play makes Jack a dull boy	All work and no play makes Jack a dull boy	
	All work and no play makes Jack a dull boy	
All work and no play makes Jack a dull boy	All work and no play makes Jack a dull boy	All work and no play makes Jack a dull boy
All work and no play makes Jack a dull boy	All work and no play makes Jack a dull boy	All work and no play makes Jack a dull boy
All	Makes	Boy

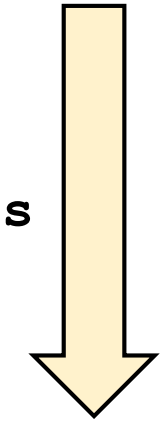
```
<table border="1">
  <caption><i>All work and no play makes Jack a dull boy</i></caption>
  <thead>
    <tr>
      <th>Work</th>
      <th>Jack</th>
      <th>Dull</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td rowspan="2">All work and no play makes Jack a dull boy</td>
      <td colspan="2">All work and no play makes Jack a dull boy</td>
    </tr>
    <tr>
      <td colspan="2">All work and no play makes Jack a dull boy</td>
    </tr>
    ...
  </tbody>
  <tfoot>
    <tr>
      <td>All</td>
      <td>Makes</td>
      <td>Boy</td>
    </tr>
  </tfoot>
</table>
```

Forms

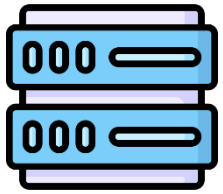
- Forms are use to collection information for user
- Information are collected in form fields eg text box, check boxes, etc.
- To process information the information the form needs to know
  - HTTP method to use
  - Action to send the information to
- A submit button is required to send the information in the to the backend to be processed

```
<form method="POST" action="/address">
  <label for="name">Name</label>
  <input type="text" id="name" name="name">
  <label for="addr">Address</label>
  <input type="text" id="addr" name="addr">
  <button type="submit">Add</button>
</form>
```

POST /address



Collect name and address
These information will be send to the server for processing



Form Field

- Form fields name their fields using the **`name`** attribute

```
<input type="text" name="username"/>
```

  - This allows the backend application to retrieve the data
- For certain elements, if you set a common name is taken to mean that you want single selection
  - Eg multiple `radio` type elements with name same name
- A form is submitted with the submit button
  - An input element with the attribute `type` set to `submit`
  - If the `type` is set to `reset`, then the button will clear the form, viz reset back to its original state

```
<button type="reset">Clear</button>
<button type="submit">Submit</button>
```

Form Fields

- Text field
  - Different types of fields
  - `text`, `email`, `search`, `tel`, `url`

```
<input type="text"/>
```

- Password

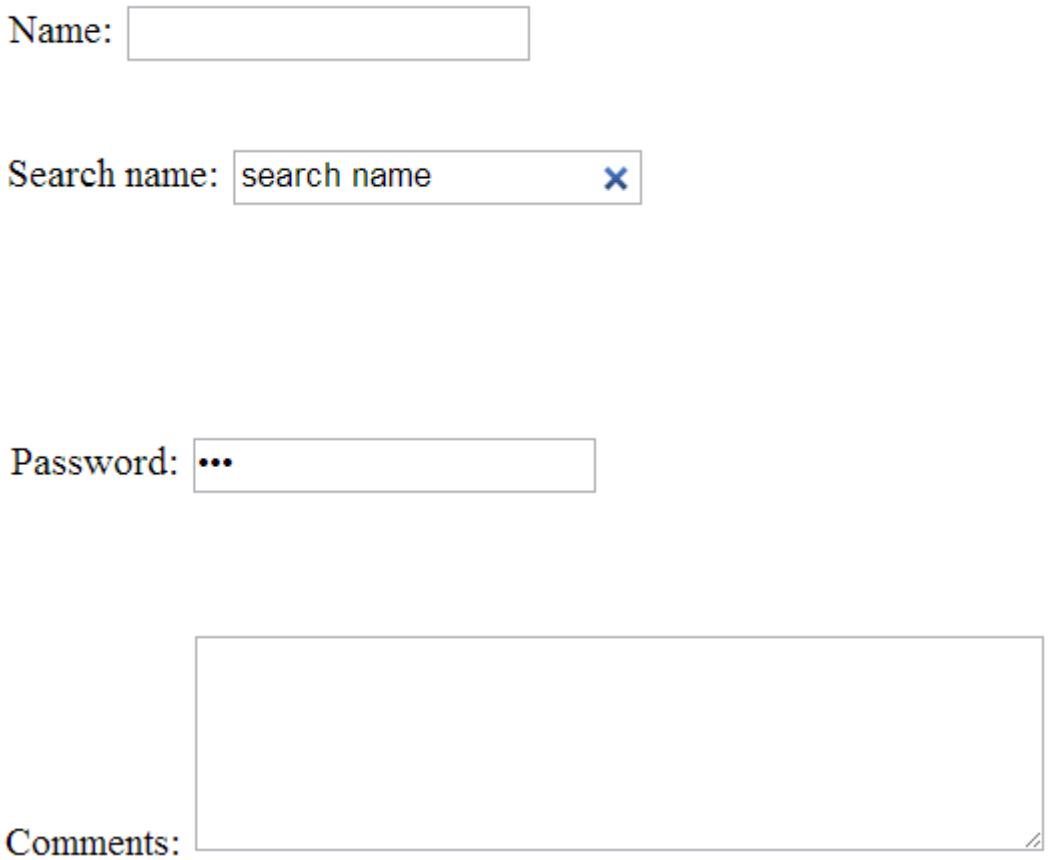
```
<input type="password"/>
```

- Hidden

```
<input type="hidden"value="id123"/>
```

- Text area

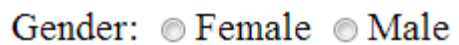
```
<textarea cols="50" rows="5"/>
```



Form Fields

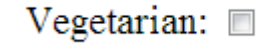
- Select one radio button
  - Set the same name for the field

```
<input type="radio" value="f"name="gender"/>
<input type="radio" value="m" name="gender"/>
```



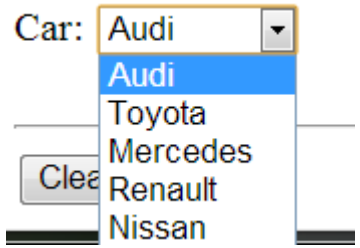
- Check box

```
<input type="checkbox" name="vegan">
```



- Combo box

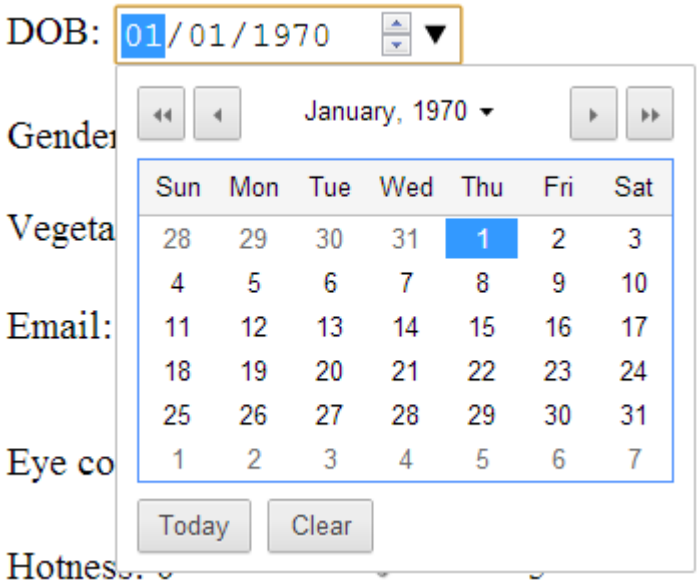
```
<select name="make">
  <option value="audi">
      Audi</option>
  <option value="toyota">
      Toyota</option>
  ...
</select>
```



Form Fields

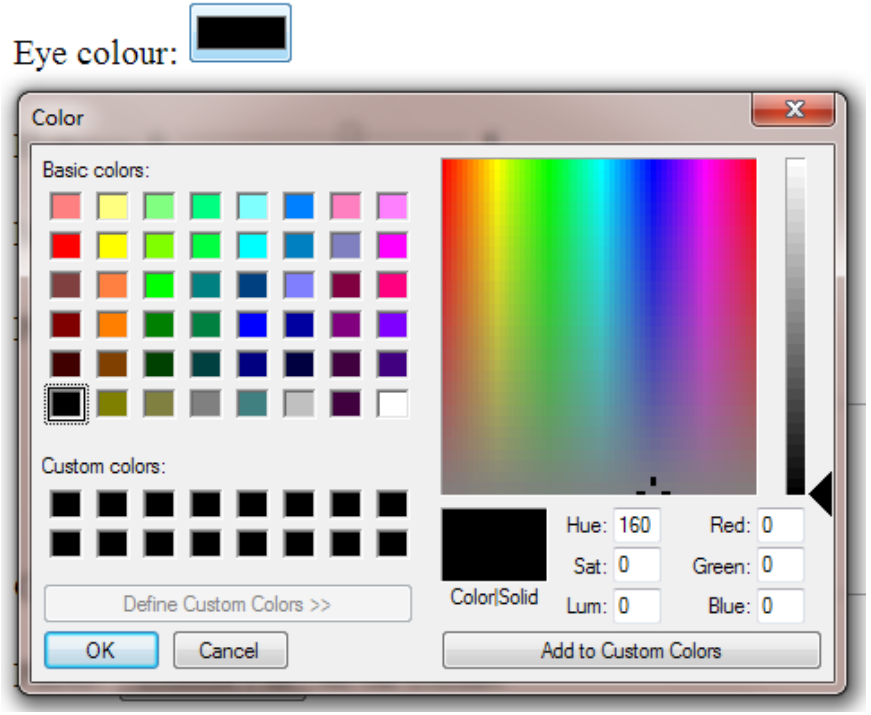
- Date

```
<input type="date" name="dob">
```



- Color

```
<input type="color" color="eye">
```

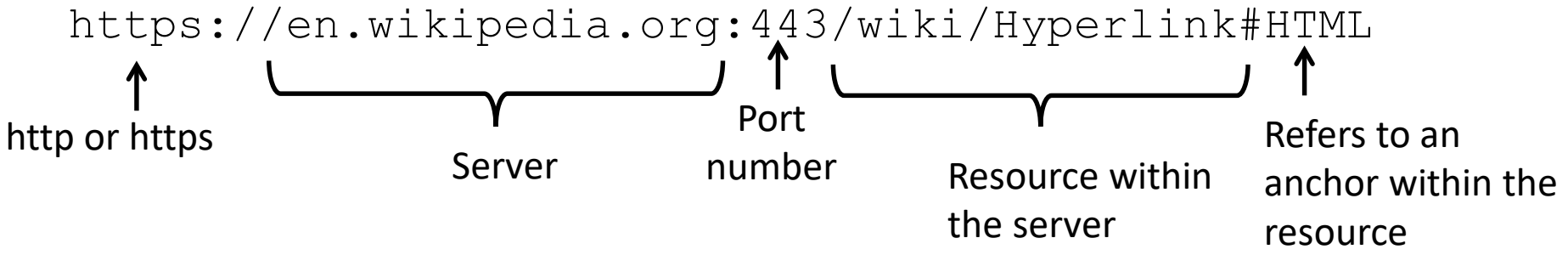


Block and Inline Elements

- Block elements causes a line break when rendering
  - Eg. `<p>All work</p><p>and no play</p>` will introduce a line break between the phrase “All work” and “and no play”
  - Other examples of block elements include all the headers, list, `<blockquote>`, table, etc
- Inline elements does not introduce a line break
  - Eg. All `<em>work</em> and <ins>no play</ins>` format the words
  - Some examples of inline elements include `<strong>`, `<b>`, `<sup>` and `<sub>`
- All HTML elements belongs to one of these 2 categories
  - Can be changed with CSS

Links

- The **<a>** introduces a hyperlink to an external (outside of the current document) resource
- The URL is specified in the **href** attribute of the **a** tag
  - The URL may refer to another to another document, email address, phone number, etc.
- The http scheme is a specific form of URL that references a resource located in a server that communicates using the HTTP protocol



References

- External resource
  - Includes server
- Another resource from the same server of the containing document
  - Starts with a / means ‘absolute’ viz. resolve the document from the root of the server
  - Without the / means from the same ‘directory’ as the current document
- Some where in the current document
  - Starts with a #

news.html

```
<html>
  ...
  <body>
    <h1>Latest News</h1>
      <p>Jump to
        <a href="#sports">Sports</a>
      </p>
      <div id="top">
        <h2>Top News</h2>
          ...
          <a href="/topnews.html">More</a>
      </div>
      <div id="sports">
        <h2>Sports</h2>
          ...
          <a href="http://sports.com">More</a>
      </div>
  </body>
</html>
```

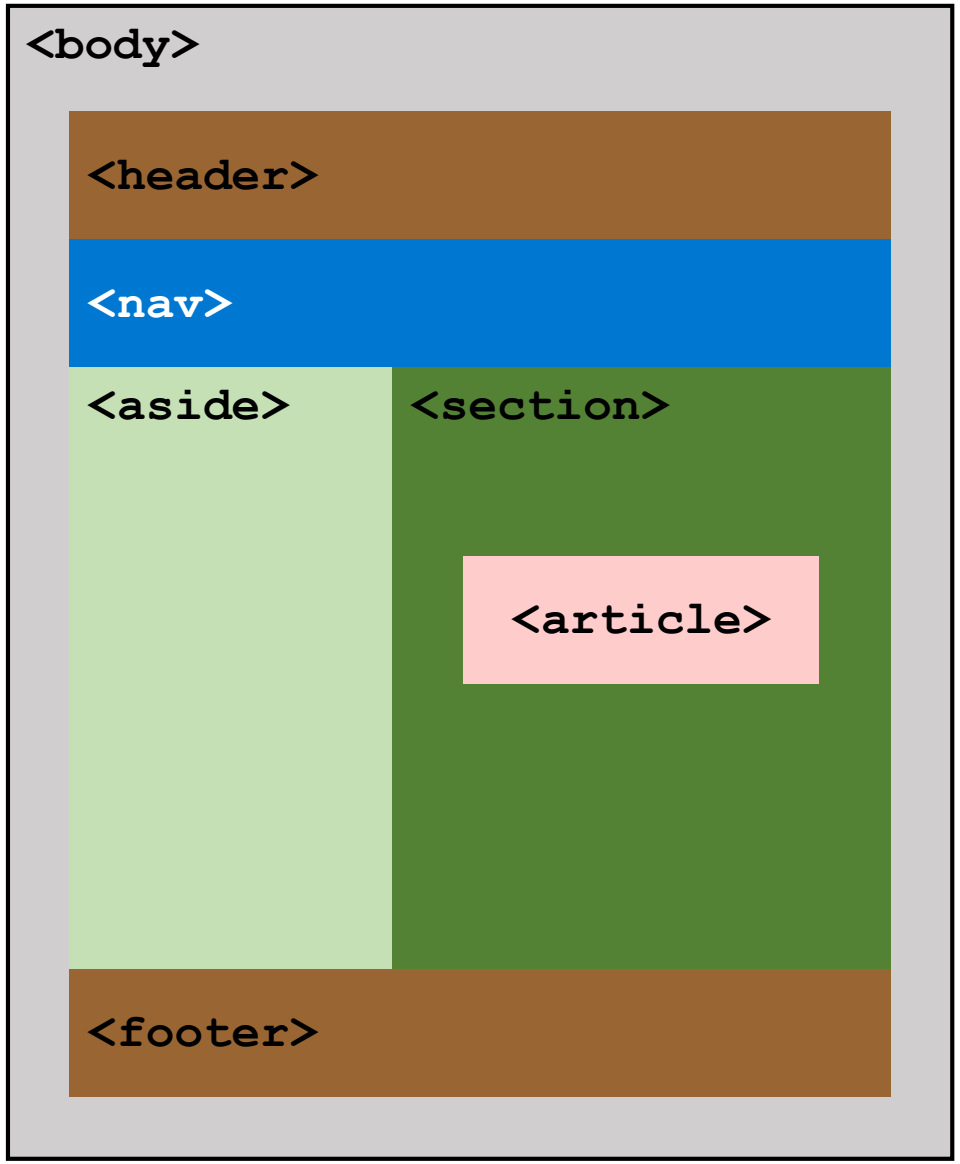
References the `id=sports` within this document

References a document on another web server

References another document on the same server which loaded the current document

HTML Structural Tags

- Semantic tags to define the different parts of a document
- `<header>` - heading, typically contains `<h1>` - `<h6>`
- `<nav>` - navigation elements eg. menu item
- `<section>` - one or more logical sections of the document eg sports, politics
- `<article>` - represents a complete self contained composition in the document eg. a news article
- `<aside>` - contains content which is related to the main content eg. related news
- `<footer>` - for copyright, logos, etc



data Attribute

- Custom data attributes allows you to insert custom data as attribute into any HTML element
- The attribute name must start with **data-**

```
<h1 data-title="The Shining">The Shining</h1>
<li data-favourite-food="noodles">Noodles</li>
```

- The data attributes acts as storage for private data
  - These attributes are not displayed/rendered by the browser
  - Enrich the attribute with additional non standard attributes
- Typically used by JavaScript application

Styling HTML Documents

- Older HTML documents places styles directly in the document
  - Eg. `<body bgcolor="#FFB400">`
  - Eg. `<font size="2" color="red"><em>...`
  - Eg. `<tr bgcolor="pink">`
- Mixing presentation with the structure of the document
  - Tedious to change eg. change way we display
  - Difficult to have a consistent look and feel
    - Especially if the you have lots of authors working on it

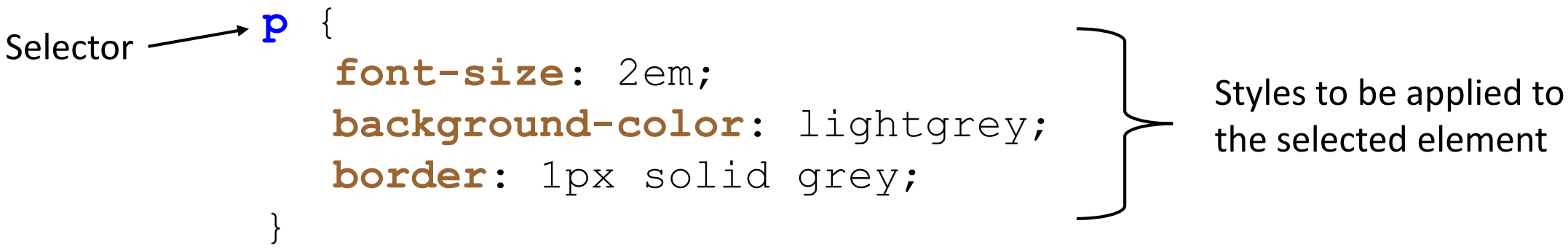
What are Stylesheets?

- Separate document use to describes the formatting/styles of a content document
  - Eg Word, HTML document
- Instead of placing formatting information inside the content document
  - Styles are set of formatting properties
    - Eg. Font type, character size, colour,
- The stylesheet applies the style to the content
  - Eg. All `<h1>` text should should be Arial, size 34, red and bold
- The result is that
  - Documents becomes consistently formatted
  - Documents formatting is flexible, apply a stylesheet before printing

CSS Rule and Selector

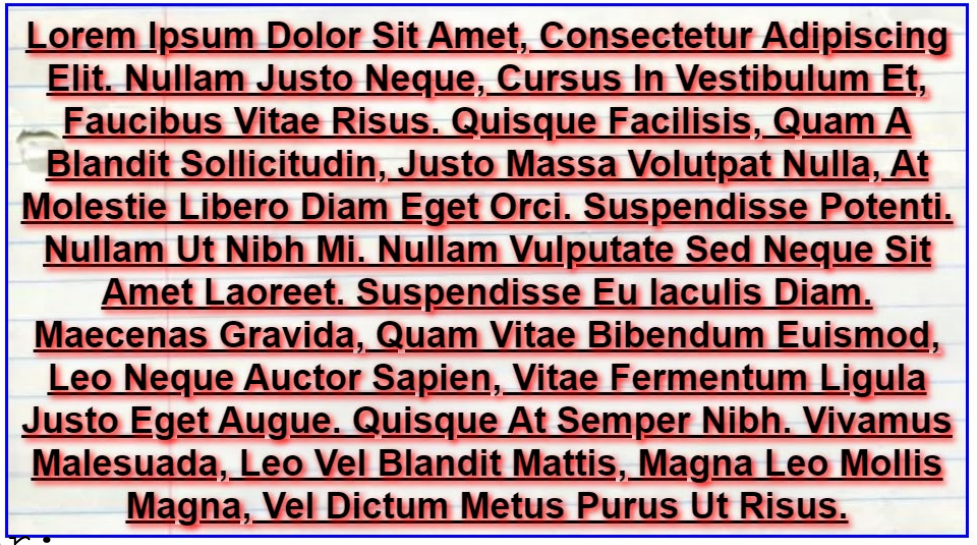
- A rule describe the style to be applied to HTML elements

- Rules are grouped in { }
- A selector selector selects elements in the document which the rules are applied to



Common CSS Styling Attributes

- **Font** - `font-size`, `font-weight`, `font-family`
- **Text** - `text-align`, `text-shadow`, `text-transform`, `text-decoration`
- **Border** - `border-width`, `border-style`, `border-color`
- **Background** - `background-image`,, `background-color`




```
font-size: 1.5em;
font-weight: bolder;
font-family: Arial, Times;
text-align: center;
text-shadow: 2px 2px 4px #ff0000;
text-transform: capitalize;
text-decoration: underline;
border-width: 1px;
border-color: blue;
border-style: solid;
background-image: url("https://...sxBKWg.jpg");
```

Loading Stylesheet

Place the mystyles.css
in the HTTP server

- The browser applies a default set of styles to a document if no stylesheet are loaded by a HTML document
- Stylesheets are loaded in the <head> with the following

```
<link rel="stylesheet" href="mystyles.css">
```

Place the mystyles.css
in the HTTP server

The relationship of the link with
the document is stylesheet

- They can also be applied to specific elements with the style attribute
  - Inline style

```
<p style="font-size: 2em; background-color: lightgrey;">
   lorem ipsum...
</p>
```

Selectors

- Selectors are patterns that CSS use to match elements in a document
- 3 basic type of selectors
  - Element selector
  - Class selector
  - Identifier selector
- Special selector `*` or `root` which refers to all the elements in the document

Element Selector

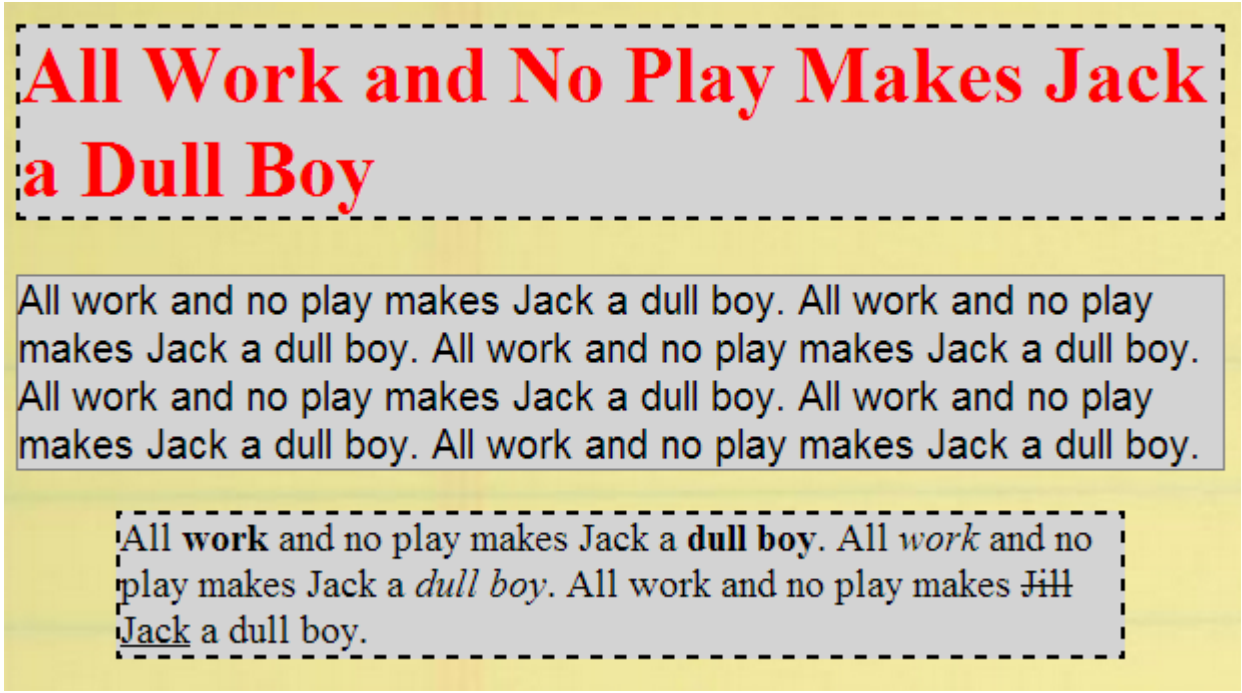
- Element selectors are use to match HTML element in the document

```
body {
  background-image: url(‘paper.jpg');
}

p {
  background-color: lightgray;
  border: 1px solid gray;
  font-family: arial;
}

h1, h2 {
  font-family: "roman times";
}

h1 {
  border-bottom: 1px solid black;
}

h2 {
  font-style: italic;
}
```

Class Selector

- A class allows you to define a style and reuse it on different elements
- Class name starts with a dot (.)

```
h1 {
  color: red;
}
.graybox {
  background-color: lightgray;
  border: 2px dashed black;
}

<h1 class="graybox">All Work and No Play Makes
Jack a Dull Boy</h1>
  <p>All work and no play makes Jack a dull boy.
...</p>

  <blockquote cite="http://bit.ly/dcObQ8"
      class="graybox">
    All <strong>work</strong> and no play makes
Jack a <b>dull boy</b>.
      All <i>work</i> and no play makes Jack a
<em>dull boy</em>.
      All work and no play makes <del>Jill</del>
<ins>Jack</ins> a dull boy.
```



Identifier Selector

```
#start {
  font-size: 1.3em;
  font-style: italic;
}

<p id="start">
All work and no play makes
Jack a dull boy. ...
</p>
```

All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy.

- A identifier selector applies the selector to a named element
- You can christen an element with the `id` attribute
  - id should be unique
- An identifier selector begins with a hash (**#**)

Class and Identifier Selector

- A identifier refers to a specific element, typically with special significance
  - So not a candidate for reuse
  - Eg The first paragraph in the `<article>`
- A class defines a general style and is reused multiple times in a document
  - Styles can be combined; class names are separated by a space
  - Eg. Graying the background of headers, odd rows in list and tables
- An element may have both a **`class`** and an **`id`** attribute

Units

- Many CSS properties require units eg. width, margin, font-size
- CSS provides 2 types of units: absolute and relative
- Absolute units are fixed sized eg. 16px, 1in
  - `px` - pixel
  - pt - point
- Relative units are units that are relative to something eg. 30% of the screen size
  - `%` - percentage eg 30% of the current size
  - `em` - relative to the font-size of the element eg 2em means 2 times the font size of the element
  - `rem` - relative to the font size of the root element
  - `vw`, `vh` - width and height of the viewport

Grouping Elements

- **** and ** ** are grouping element
  - They can be considered neutral element as they are not rendered by the browser
  - Use to group elements so that they can be manipulated or styled
- **** is an inlined element
- ** ** is a block element

```
All <span class="emphasize">work and no play</span> makes Jack a dull boy
```

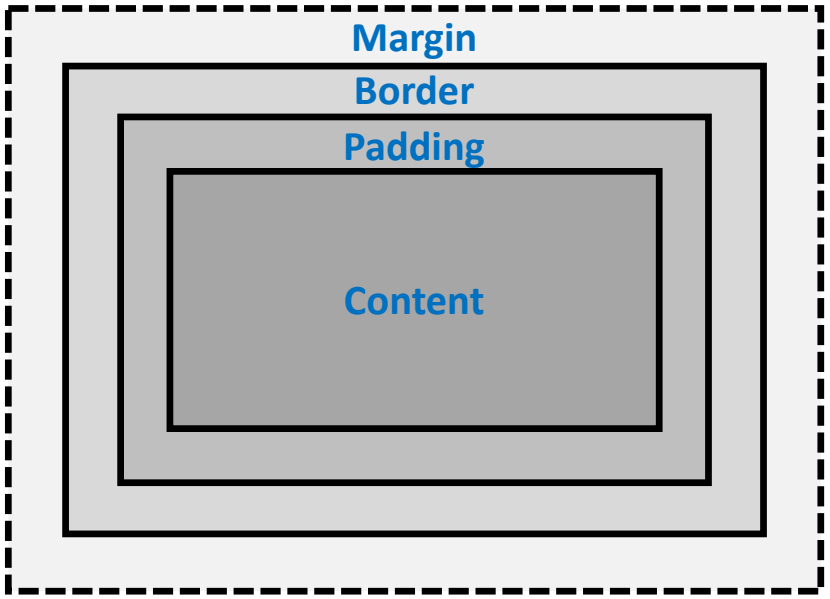
All **work and no play** makes Jack a dull boy

```
.emphasize {
  color: red;
  font-weight: bold;
  font-size: 1.5em;
}

.boxit {
  width: 100px;
  height: 100px;
  background-color: yellow;
}
```

All work and no play makes Jack a dull boy

```
<div class="emphasize boxit">All work and no play makes Jack a dull boy</div>
```

CSS Box Model

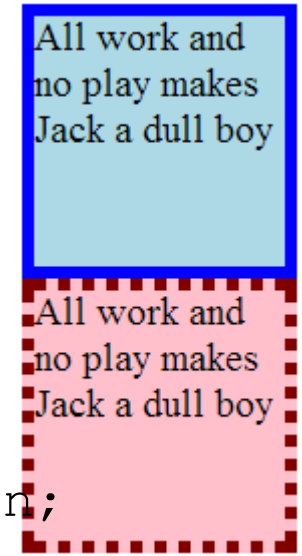


- CSS views every element as a box
  - Every element is wrapped in a box
- The box has the following properties
  - margin – the area around the border. This is **`transparent`**
  - **`border`** – the region around the content. You can set the line style, thickness, colour, etc. of the border
  - **`padding`** – the area between the content and the border. The padding takes the colour of the CSS background property
  - **`content`** – where your content goes. This can be text, images, forms, etc.

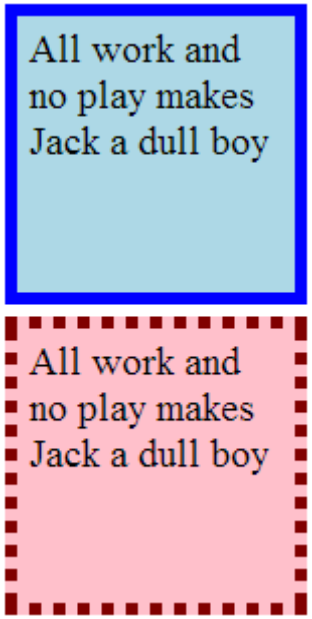
Example

```
<div class="box0">All work and no play makes Jack a dull boy</div>
<div class="box1">All work and no play makes Jack a dull boy</div>
```

```
.box0 {
  background: lightblue;
  border: 5px solid blue;
}

.box1 {
  background: pink;
  border: 5px dotted maroon;
}
```




```
.box0 {
  background-color: lightblue;
  border: 5px solid blue;
  margin: 5px;
  padding: 5px;
}

.box1 {
  background-color: pink;
  border: 5px dotted maroon;
  margin: 5px;
  padding: 5px;
}
```

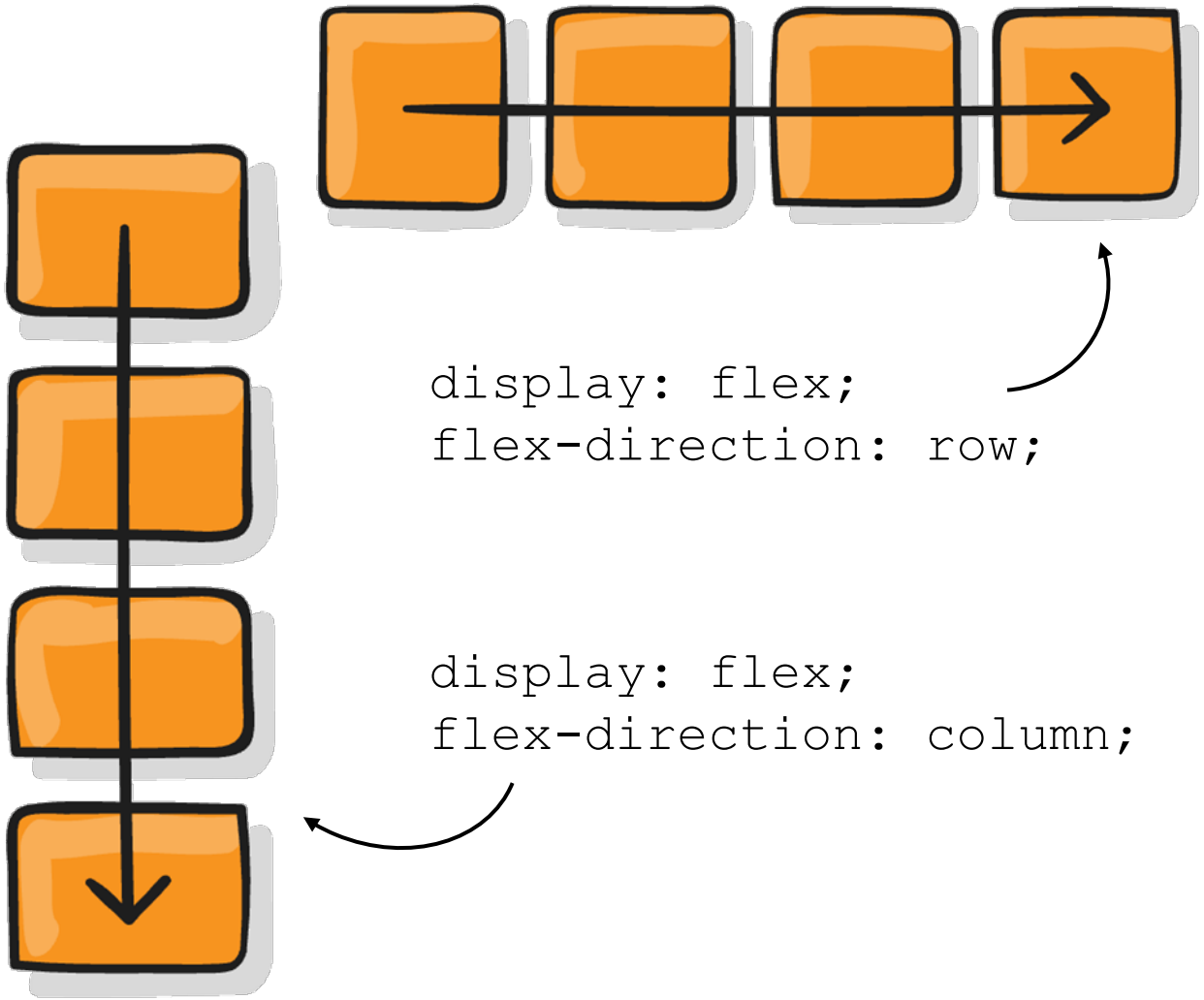


Positioning Elements with Flexbox

- The `display` attribute controls how an attribute is layout eg as block or inline
  - Eg. to change `<li>` from block to inline

```
li { display: inline; }
```

- Flexbox layout is a modern responsive layout model
- Elements can be layout by
  - rows followed by columns
  - columns followed by rows



Images from https://css-tricks.com/snippets/css/a-guide-to-flexbox/#basics-and-terminology

Example

Flexlayout layout the elements inside a container

```
<div class="fruits">
   <img src="/images/fruits/acorn_squash.png">
   <img src="/images/fruits/blueberries.png">
   <img src="/images/fruits/corn.png">
</div>
```

Row is the default direction

```
.fruits {
   display: flex;
}
```



Main axis is row
secondary axis is column

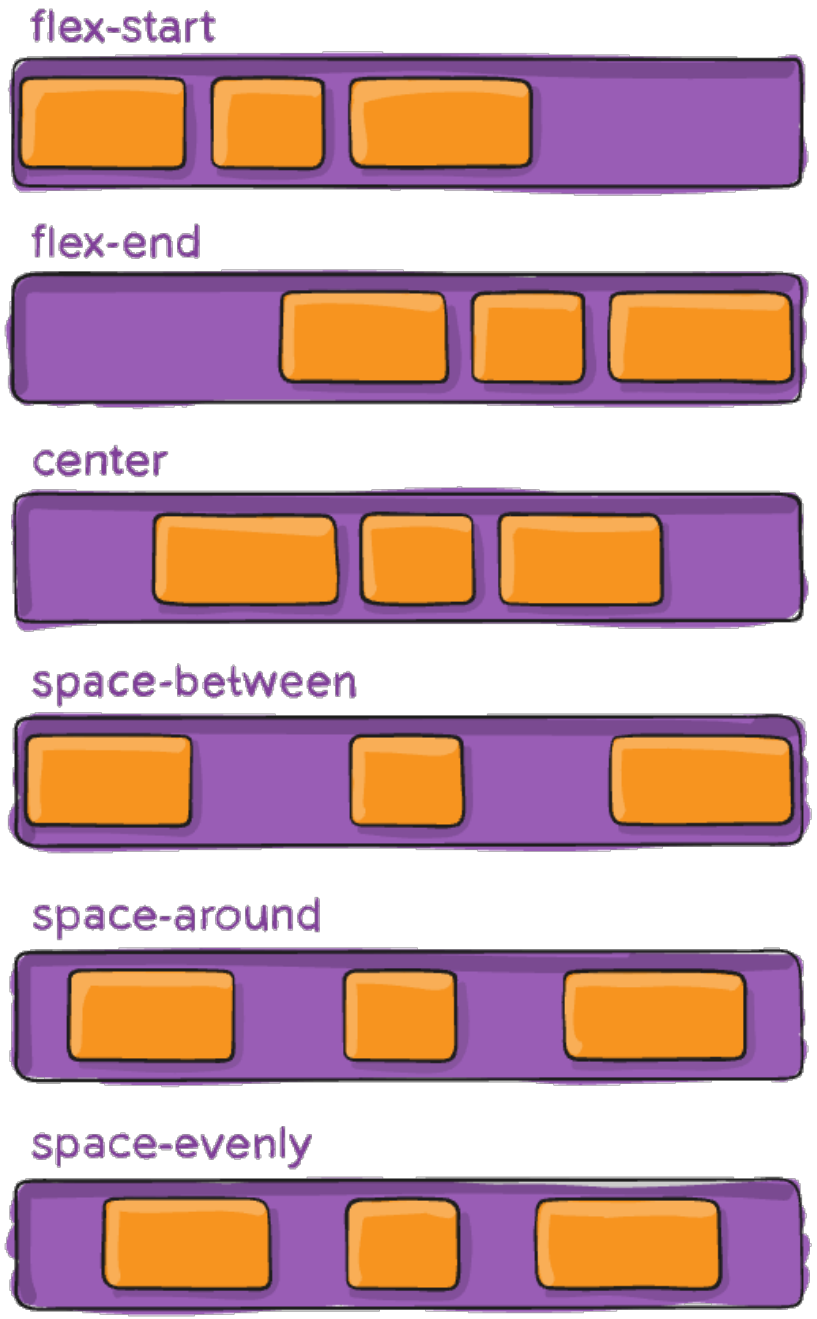
```
.fruits {
   display: flex;
   flex-direction: column;
}
```

Main axis is column
secondary axis is row



Spacing within the Main Axis

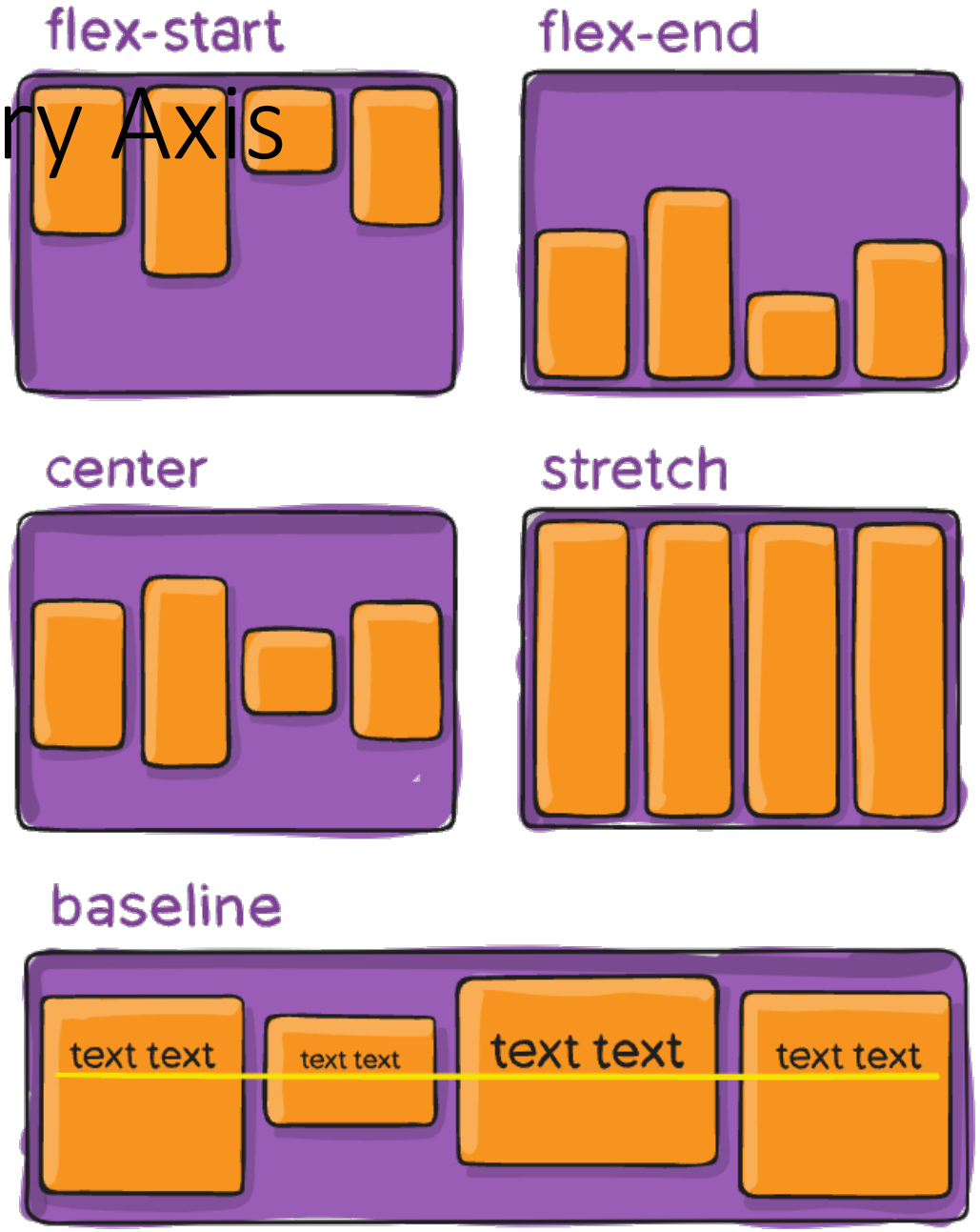
- **`justify-content`** position elements within the main axis
- By default the container will use the smallest possible size to fit all elements
  - Might not see if the effect if elements do not have room to ‘move’
  - Eg. set the size of the container to be larger than the elements



Images from https://css-tricks.com/snippets/css/a-guide-to-flexbox/#basics-and-terminology

Spacing within the Secondary Axis

- **`align-items`** position elements within the secondary axis



Images from https://css-tricks.com/snippets/css/a-guide-to-flexbox/#basics-and-terminology

Example

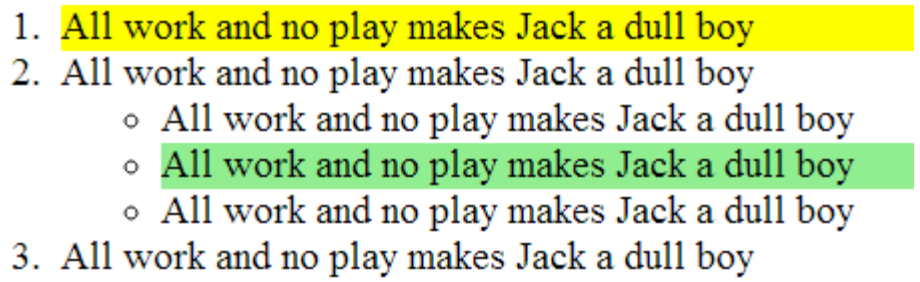
```
<div class="fruit-item">
   <h1>Apple</h1>
   <div> <img src="/images/fruits/apple.png"> </div>
</div>
```

```
.fruit-item {
   display: flex;
}
```


```
.fruit-item {
   display: flex;
   align-items: center;
}
```

Combining Selectors

- You can combine element, class and identity selector to make the selector more specific
  - Instead of just any element, or any class




```
li.olfirst {
  background: yellow;
}

li#ulsecond {
  background: lightgreen;
}
```

```
<ol>
  <li class="olfirst">All work and no play makes Jack
a dull boy</li>
  <li>All work and no play makes Jack a dull boy
    <ul>
      <li>All work and no play makes Jack a dull
boy</li>
      <li id="ulsecond">All work and no play makes
Jack a dull boy</li>
      <li>All work and no play makes Jack a dull
boy</li>
    </ul>
  </li>
  <li>All work and no play makes Jack a dull boy</li>
</ol>
```

Combining Selectors

- Two or more selectors separated by space
  - Defines a “path” to the final element in the list
  - The later element in the path are descendants of the earlier elements

1. All work and no play makes Jack a dull boy
2. All work and no play makes Jack a dull boy
   - All work and no play makes Jack a dull boy
   - All work and no play makes Jack a dull boy
   - All work and no play makes Jack a dull boy
3. All work and no play makes Jack a dull boy

```
ol li {
  background: lightgreen;
}
```

```
<ol>
  <li class="olfirst">All work and no play makes
Jack a dull boy</li>
  <li>All work and no play makes Jack a dull boy
    <ul>
      <li>All work and no play makes Jack a dull
boy</li>
      <li id="ulsecond">All work and no play makes
Jack a dull boy</li>
      <li>All work and no play makes Jack a dull
boy</li>
    </ul>
  </li>
  <li>All work and no play makes Jack a dull
boy</li>
</ol>
```

Combining Selectors

- Use the **>** operator for direct descendent of
  - Eg `p > table` – table is a direct descendent of viz. the selector is targeting a table inside a paragraph

1. All work and no play makes Jack a dull boy
2. All work and no play makes Jack a dull boy
   - All work and no play makes Jack a dull boy
   - All work and no play makes Jack a dull boy
   - All work and no play makes Jack a dull boy
3. All work and no play makes Jack a dull boy

```
ul > li {
  background: lightgreen;
}
```

```
<ol>
  <li class="olfirst">All work and no play makes Jack a dull
boy</li>
  <li>All work and no play makes Jack a dull boy
    <ul>
      <li>All work and no play makes Jack a dull boy</li>
      <li id="ulsecond">All work and no play makes Jack a
dull boy</li>
      <li>All work and no play makes Jack a dull boy</li>
    </ul>
  </li>
  <li>All work and no play makes Jack a dull boy</li>
</ol>
```

Pseudo Class

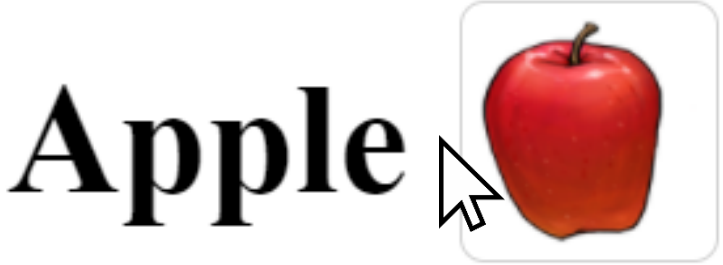
- A pseudo-class is a special keyword added to a selector that specifies the state of the element
  - Eg. when a mouse is over a link `<a>`
- Pseudo class allows us to dynamically style a document when an user is interacting with it
  - Eg. Highlight a word as link when the mouse is over it
- A pseudo-class is a selector that is followed by a **`:state_name`**
  - Eg `a:hover` – where a is the `<a>` and hover is the name of the pseudo-class
- List of pseudo classes
  - https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

Example

```
<div class="fruit-item">
    <h1>Apple</h1>
    <div> <img src="/images/fruits/apple.png"> </div>
</div>
```

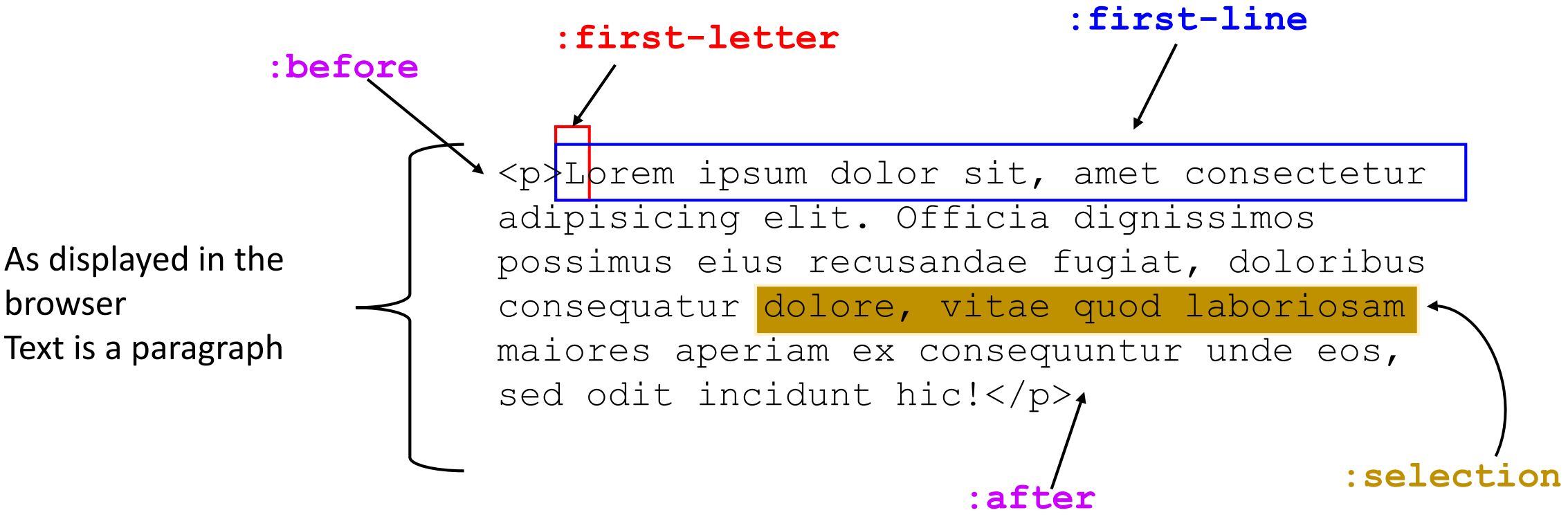
```
.fruit-item {
    display: flex;
    align-items: center;
    color: lightgrey;
}

.fruit-item img {
    filter: grayscale(100%);
}

.fruit-item:hover {
    color: black;
}

.fruit-item:hover img {
    filter: grayscale(0%);
}
```

Style to applied to these elements when the pointer hover over this class



Pseudo Element

- A pseudo-element is a special keyword added to a selector that allow you to apply styles to specific part of the selected element
  - Eg. first letter of a word paragraph



Example

```
<p> orem ipsum dolor sit,
...</p>
```

OREM IPSUM DOLOR SIT,
amet consectetur adipisicing elit. Officia dignissimos possimus eius recusandae fugiat, doloribus consequatur dolore, vitae quod laboriosam maiores aperiam ex consequuntur unde eos, sed odit incidunt hic!

```
p:first-line {
    text-transform: uppercase;
    font-size: 1.5em;
    font-weight: bolder;
    color: red;
}
p:before {
    content: url("/images/lawliet.png");
}
```

CSS function to load a resource

Transitions

- Transition allows you to perform animations on elements
- Transition works by defining the value of a property at 2 points in time
  - The current value and a future value
  - Eg. current background is blue; future is red
- Transition is typically started by an external event eg. hover

Transitions

- Transitions are controlled by the following properties
  - `transition-property` – the name of the property that we are applying the transition to
    - Eg. background
  - `transition-duration` – how long will a transition run
    - Eg. how long will it take a to change a background from blue to red
  - `transition-timing-function` – determine the pace of the transition
    - `linear` – constant
    - `ease-in` – start transition slowly, finish off at full speed
    - `ease-out` – start transition at full speed, finish off slowly
    - `ease-in-out` – starts and end slowly, full speed in the middle
  - `transition-delay` – how long before a transition starts

Example

```
<div class="box">GIDDY</div>
```


```
.box {
  text-align: center;
  vertical-align: middle;
  width: 100px;
  height: 100px;
  background: lightblue;
  border: 2px solid blue;
  transition-property: background border-radius border-color;
  transition-duration: 2s;
  transition-timing-function: linear;
}
.box:hover {
  border-radius: 120px;
  background: pink;
  border-color: maroon;
}
```

From values – to values

Transition properties

Transition when mouse hover over element

Example

```
Name: <input class="glow float" type="text" size="30"/>
```

```
.glow {
  border: 1px black dotted;
  background: lightgray;
  transition-property: border-color background border-radius;
  transition-duration: 250ms;
  transition-timing-function: ease-out;
}
.glow:focus {
  border: 1px blue solid;
  background: lightblue;
  border-radius: 10px;
}
.float {
  box-shadow: 0px 0px 0px darkgrey;
  transition-property: box-shadow;
  transition-duration: 250ms;
  transition-timing-function: ease-out;
}
.float:hover {
  box-shadow: 10px 10px 5px darkgrey;
}
```