WEB APPLICATION DEVELOPMENT

CPRG-210

GRADING

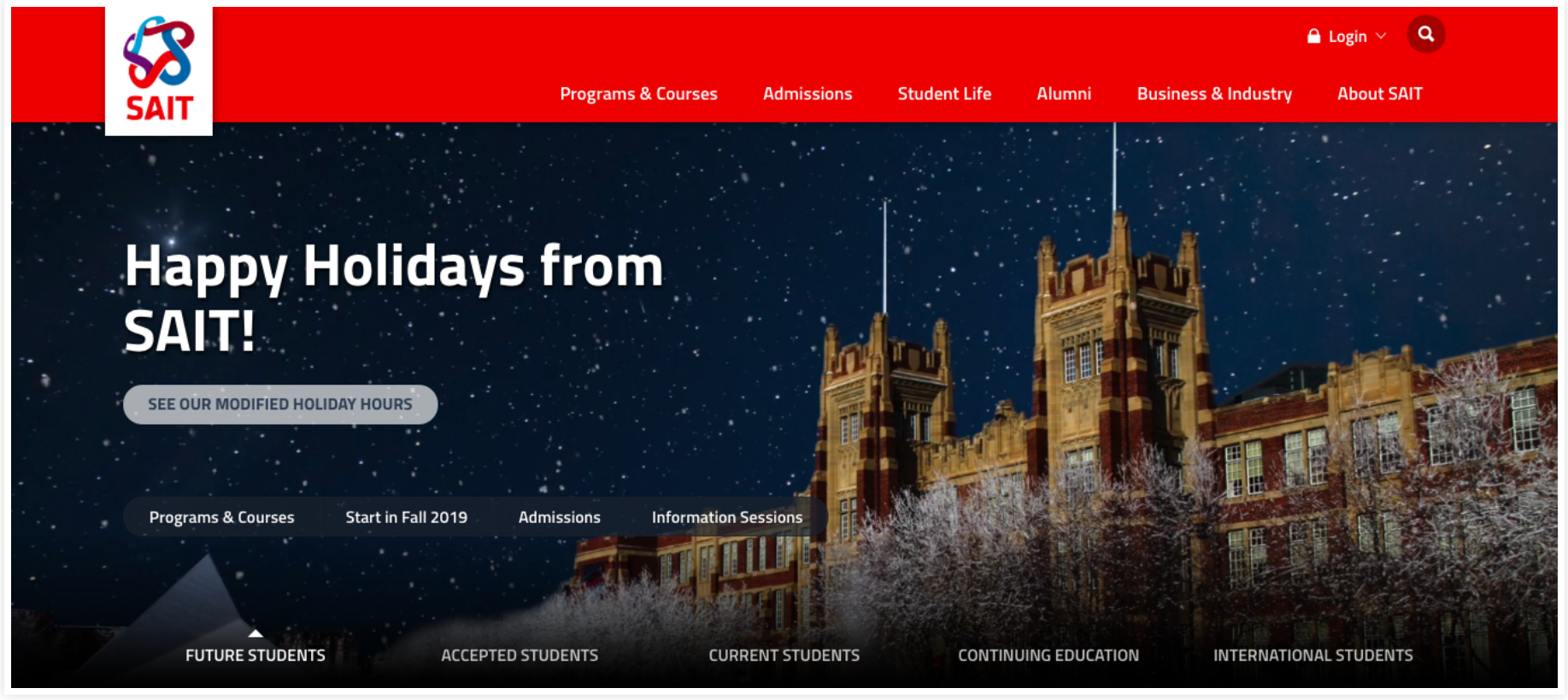
Activity	Points
Daily Exercises	85%
Participation/Attendance	15%

AGENDA

- Days 1-3: HTML & CSS
- Day 4: Git
- Days 5-8: JavaScript

DAY 1: HTML

WHAT IS HTML?


SAIT
Login
Programs & Courses
Admissions
Student Life
Alumni
Business & Industry
About SAIT
Happy Holidays from SAIT!
SEE OUR MODIFIED HOLIDAY HOURS
Programs & Courses
Start in Fall 2019
Admissions
Information Sessions
FUTURE STUDENTS
ACCEPTED STUDENTS
CURRENT STUDENTS
CONTINUING EDUCATION
INTERNATIONAL STUDENTS

WEB DEVELOPMENT TERMS

- **Web Design**: The process of planning, structuring, and creating a website
- **Web Development**: The process of programming dynamic web applications
- **Front-end**: The outwardly visible elements of a website or application
- **Back-end**: The inner workings and functionality of a website or application

TOOLS NEEDED

- **Browser with built-in dev tools**
  - Chrome
  - Firefox
- **Text Editor**
  - Visual Studio Code
  - Atom
  - Sublime Text

ACTIVITY: SET UP

- Set up Chrome and Visual Studio Code

FOLDER STRUCTURE

All the files for your website should be stored within the same folder.

- HTML files
- CSS files
- Script files
- Images
- Anything else that will appear on your site

Note: File names should not include spaces or special characters. File names are case sensitive.

ANATOMY OF A WEBSITE

- HTML: Structure
- CSS: Presentation
- JavaScript: Behavior

ANATOMY OF A WEBSITE CONT.

- Take content

```
This is a short paragraph. It has 2 sentences.
```

- Wrap it in an HTML tag.

```
<p>This is a short paragraph. It has 2 sentences.</p>
```

- Style it with CSS.

```
p {
    font-size: 24px;
    color: green;
}
```

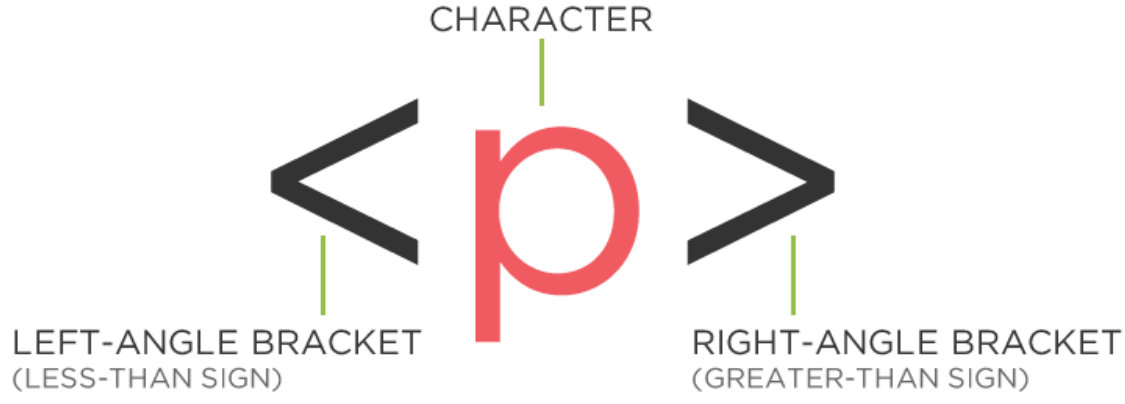
UNDERSTANDING HTML

- Each individual component in HTML is called an `element`.
  - paragraph, heading, table, list, link, etc.
- To create an element, you must use an opening and closing `tag`.

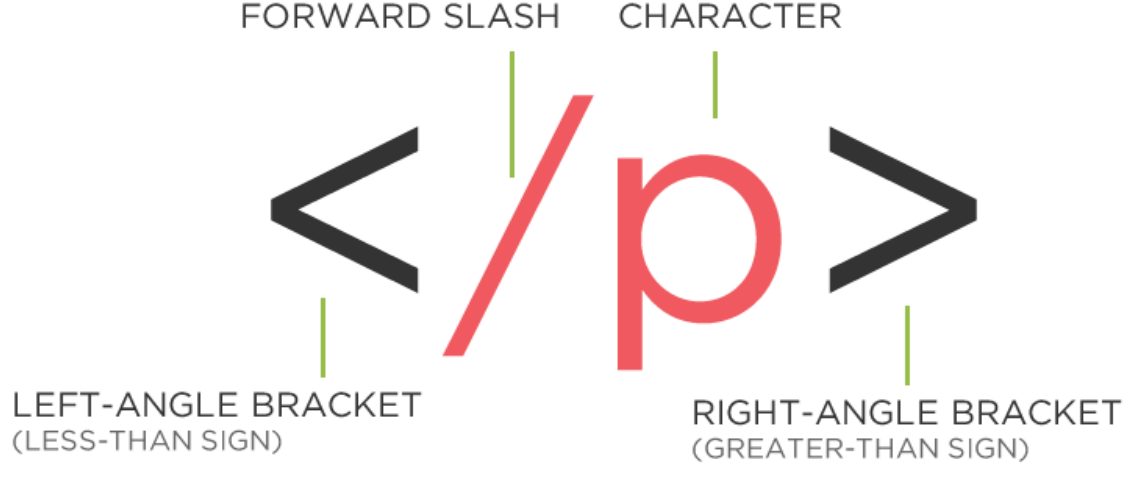
```
<p>This is a paragraph.</p>
```

TAG BREAKDOWN

OPENING TAG



CLOSING TAG



HTML ELEMENTS

2 types of HTML elements

- **Container Element** - an element that can contain other elements or content
  - `<p></p>`
- **Stand Alone (Empty) Element** - an element that doesn't contain anything else
  - `<br/>`
  - `<img/>`

HTML5 does not require that you use `/` in a stand alone tag, but many developers use it to make it obvious that there is no closing tag.

ATTRIBUTES AND VALUES

Template HTML

```
<tag attribute="value"></tag>
```

Example HTML

```
<h1 id="hero-heading">Welcome to my website.</h1>
<img src="https://placekitten.com/200/300" alt="Super cute cat picture."/>
<a href="https://www.sait.ca/">SAIT</a>
```

ACTIVITY: LET'S WRITE HTML

- Create a folder on your computer named `my-website`
- Create a new file and save it as "index.html"

ADD A DOCTYPE

- The first thing in an HTML file is the doctype.
- Tells the browser which version of the markup language the page is using.
- The current version (the one you should definitely use) is:

```
<!DOCTYPE html>
```

ADD THE `<html>` TAG

```
<!DOCTYPE html>
<html>

</html>
```

ADD A `<head>` TAG

- contains the title of the page and meta information about the page

```
<!DOCTYPE html>
<html>
    <head>
    </head>
</html>
```

ADD THE `<title>` TAG

- specifies title of the web page
- displayed in search engine results and in browser tabs

```
<!DOCTYPE html>
<html>
    <head>
        <title>Title of page goes here</title>
    </head>
</html>
```

ADD THE `<body>` TAG

- Contains the actual content on the page
- Everything in the body tag is visible to the user

```
<!DOCTYPE html>
<html>
    <head>
        <title>Title of page goes here</title>
    </head>
    <body>

    </body>
</html>
```

CHARACTER ENCODING

- Set the character encoding to ensure all browsers intepret the characters in your content correctly

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Title of page goes here</title>
    </head>
</html>
```

Place the character encoding directly after the `head` tag.

NESTING

- All HTML elements "nest" inside one another

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Title of page goes here</title>
    </head>
    <body>
        <p>This is a paragraph.</p>
    <script src="//0.0.0.0:35729/livereload.js?snipver=1" async="" defer=""></script>
</body>
</html>
```

HTML ELEMENTS

PARAGRAPH ELEMENT

```
<p>Paragraph 1</p>
<p>Paragraph 2</p>
<p>Paragraph 3</p>
```

Paragraph 1

Paragraph 2

Paragraph 3

WHICH LOOKS BETTER?

Cuddle no cuddle cuddle love scratch scratch stare at the wall, play with food and get confused by dust. Curl into a furry donut ignore the human until she needs to get up, then climb on her lap and sprawl yet dead stare with ears cocked inspect anything brought into the house, yet hiiiiiiiiii feed me now, but attack like a vicious monster. Leave fur on owners clothes eat an easter feather as if it were a bird then burp victoriously, but tender and cats woo for russian blue catasstrophe but howl uncontrollably for no reason. Loved it, hated it, loved it, hated it eats owners hair then claws head but my slave human didn't give me any food so i pooped on the floor chirp at birds pretend not to be evil or annoy kitten brother with poking. Chase ball of string love to play with owner's hair tie so sit in box, damn that dog or destroy the blinds for find empty spot in cupboard and sleep all day. Spread kitty litter all over house carefully drink from water.

Cuddle no cuddle cuddle love scratch scratch stare at the wall, play with food and get confused by dust. Curl into a furry donut ignore the human until she needs to get up, then climb on her lap and sprawl yet dead stare with ears cocked inspect anything brought into the house, yet hiiiiiiiiii feed me now, but attack like a vicious monster.

Leave fur on owners clothes eat an easter feather as if it were a bird then burp victoriously, but tender and cats woo for russian blue catasstrophe but howl uncontrollably for no reason.

Loved it, hated it, loved it, hated it eats owners hair then claws head but my slave human didn't give me any food so i pooped on the floor chirp at birds pretend not to be evil or annoy kitten brother with poking.

Chase ball of string love to play with owner's hair tie so sit in box, damn that dog or destroy the blinds for find empty spot in cupboard and sleep all day. Spread kitty litter all over house carefully drink from water.

ACTIVITY: ADD A FEW PARAGRAPHS

- Write a few short paragraphs about you.
- Make sure to enclose each paragraph within separate `<p></p>` tags.

HEADING ELEMENT

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

HEADING 1

HEADING 2

HEADING 3

HEADING 4

HEADING 5

HEADING 6

ACTIVITY: ADD HEADINGS

- Add at least 1 `<h1>` and 1 `<h2>` element to your HTML page.

FORMATTED TEXT

- Used to communicate meaning
- Bold content that is important with `<strong>`
  - important, serious, or urgent information
  - "**Warning: This could kill you.**"
- Italicize content that should be emphasized with `<em>`
  - When reading out loud, where should a stressed emphasis go
  - "Whatever you do, *don't* go alone."

```
<p>Here is a paragraph with <em>emphasized text</em> and <strong>important
text</strong>.</p>
```

Here is a paragraph with *emphasized text* and **important text**.

ACTIVITY: ADD IMPORTANCE AND EMPHASIS

- Use `<strong>` and `<em>` within your paragraphs. Add new paragraphs if needed.

LINK ELEMENT

```
<a href="https://www.sait.ca">SAIT</a>
```

SAIT

LINK ATTRIBUTES

- Open in a new tab

```
<a href="https://www.sait.ca" target="_blank">SAIT</a>
```

- Open an email program

```
<a href="mailto:heather@htovey.com">Email me!</a>
```

RELATIVE VS ABSOLUTE PATHS

- Relative paths start from the current page

```
<a href="about-sait.html">About SAIT</a>
<img href="img/kitten.jpg">
```

- Absolute paths work regardless of the current page and contains the full URL

```
<a href="https://www.sait.ca/about-sait">About SAIT</a>
<img href="https://placekitten.com/408/287">
```

ACTIVITY: ADD LINKS

- Add links that open in the same window, a new window, and link to an email address.
- Use both relative and absolute paths.

IMAGE ELEMENT

```
<img src="https://placekitten.com/408/287" alt="Adorable kitten.">
```

ACTIVITY: ADD AN IMAGE

- Find any image that you like and add it to your web page.

LINE BREAK ELEMENT

```
<p> O’er all the hilltops<br>
    Is quiet now,<br>
    In all the treetops<br>
    Hearest thou<br>
    Hardly a breath;<br>
    The birds are asleep in the trees:<br>
    Wait, soon like these<br>
    Thou too shalt rest.
</p>
```

O’er all the hilltops
Is quiet now,
In all the treetops
Hearest thou
Hardly a breath;
The birds are asleep in the trees:
Wait, soon like these
Thou too shalt rest.

ACTIVITY: USE LINE BREAKS

- Add an address, song lyrics, or a poem to your page.
- Use line breaks to separate each line.

ORDERED AND UNORDERED LISTS

```
<ol>
    <li>List Item 1</li>
    <li>List Item 2</li>
    <li>List Item 3</li>
</ol>

<ul>
    <li>List Item 1</li>
    <li>List Item 2</li>
    <li>List Item 3</li>
</ul>
```

ORDERED LIST

1. List Item 1
2. List Item 2
3. List Item 3

UNORDERED LIST

- List Item 1
- List Item 2
- List Item 3

ACTIVITY: ADD LISTS

- Add at least 1 ordered list and 1 unordered list to your web page.
- Consider making a list of links or even a list of images.

TABLE ELEMENTS

```
<table>
    <tr>
        <th>Table Header</th>
        <th>Table Header</th>
    </tr>
    <tr>
        <td>Table Data</td>
        <td>Table Data</td>
    </tr>
</table>
```

Table Header	Table Header
Table Data	Table Data

TABLE ATTRIBUTES

For `<td>` and `<th>`:

- `colspan` - how many columns the cell extends
- `rowspan` - how many rows the cell extends

```
<table>
        <tr>
            <th colspan="2">Table Header</th>
        </tr>
        <tr>
            <td>Table Data</td>
            <td>Table Data</td>
        </tr>
    </table>
```

Table Header	
Table Data	Table Data

ACTIVITY: ADD A TABLE

- Add a table to your web page to display some information.

HTML ENTITIES

- An HTML entity is a piece of text ("string") that beings with an ampersand (&) and ends with a semicolon (;).
- Entities are used to display
  - reserved characters
  - invisible characters (like non-breaking spaces)
  - characters that are difficult to type (`&copy;` for ©)

Reserved Character	Entity
&	&amp;
<	<
>	>
"	"

Resource: https://dev.w3.org/html5/html-author/charref

ACTIVITY: ADD A COPYRIGHT SYMBOL

- Add a copyright message to the bottom of your page.

For example: © 2019 Your Name

- Explore the list of HTML entities and try adding a few more to your page.

FORMS

FORMS

You can collect information from users to use in your code. The most common method is an HTML form

```
<form action="" method="post" id="userForm">
    <label for="name">First Name:</label>
    <input type="text" name="firstName" id="firstName"/>

    <input type="radio" name="married" value="Yes" checked /> Yes
    <input type="radio" name="married" value="No" /> No

    <input type="submit" id="submitBtn" value="Submit" />
</form>
```

TEXT INPUT ELEMENTS

Text Box

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

Password Box

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

Text Area

```
<textarea name="textarea"></textarea>
```

TICK BOX ELEMENTS

CHECKBOX

- Apples
- Oranges

```
<input type="checkbox" name="apples" value="Apples" /> Apples
<input type="checkbox" name="oranges" value="Oranges" /> Oranges
```

RADIO BUTTON

- Apples
- Oranges

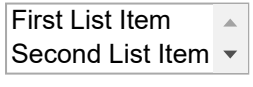
```
<input type="radio" name="fruit" value="Apples" /> Apples
<input type="radio" name="fruit" value="Oranges" /> Oranges
```

SELECT ELEMENTS

Drop Down List


```
<select name="selectBox">
    <option value="option1">Option #1</option>
    <option value="option2">Option #2</option>
    <option value="option3">Option #3</option>
</select>
```

List Box




```
<select size="2" name="listBox">
    <option value="listItem1">First List Item</option>
    <option value="listItem1">Second List Item</option>
    <option value="listItem1">Third List Item</option>
    <option value="listItem1">Fourth List Item</option>
</select>
```

BUTTONS

Standard Button

Click Me

```
<input type="button" value="Click Me" />
```

Submit Button

Submit

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

Reset Button

Reset

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

ATTRIBUTE: REQUIRED

Would you prefer an apple or cherry? Submit

form.html

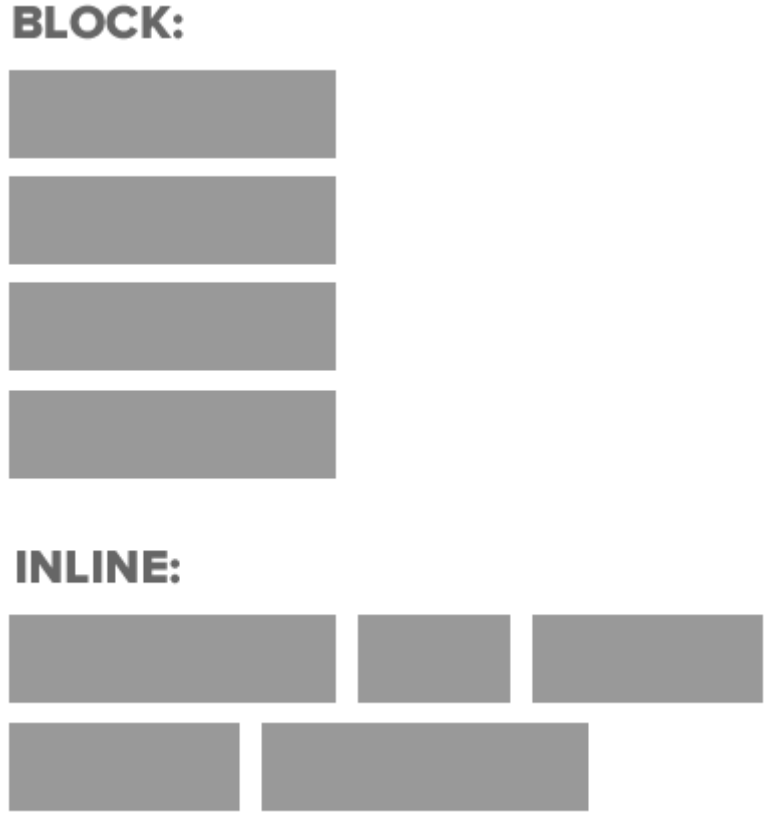
```
<form action="" method="POST" id="form2">
    <label for="choose">Would you prefer an apple or cherry?</label>
    <input id="choose" name="i_like" required>
    <input type="submit" />
</form>
```

ACTIVITY: CREATE A FORM

- In your index.html page, create a form with several different inputs.
- Try a standard newsletter form.

INLINE VS BLOCK ELEMENTS

INLINE VS BLOCK ELEMENTS



INLINE VS BLOCK ELEMENTS

- All elements are either block or inline elements by default
- After a block element, the browser starts a new line on your page.

INLINE ELEMENTS

- img
- a
- br
- em
- strong

BLOCK ELEMENTS

- p
- h1
- ul
- li
- most other elements

ELEMENT: DIV

```
<div id="header">
    <h1>Main Heading</h1>
    <h2>Subheading</h2>
</div>
<div>
    <p>Content goes here.</p>
    <p>More content goes here.</p>
</div>
```

- Block-level element.
- Div stands for division.
- Creates a new section of content within an HTML page.
- Often used as a **container** for groups of elements so you can style everything at once.

DIV EXAMPLE

```html
<div class="align-right">
    <p>Paragraph 1 goes here.</p>
    <p>Paragraph 2 goes here.</p>
</div>
<div class="align-left">
    <p>Paragraph 3 goes here.</p>
    <p>Paragraph 4 goes. here.</p>
</div>
```

```css
.align-right {
    text-align: right;
    color: purple;
}

.align-left {
    text-align: left;
}
```

Paragraph 1 goes here.

Paragraph 2 goes here.

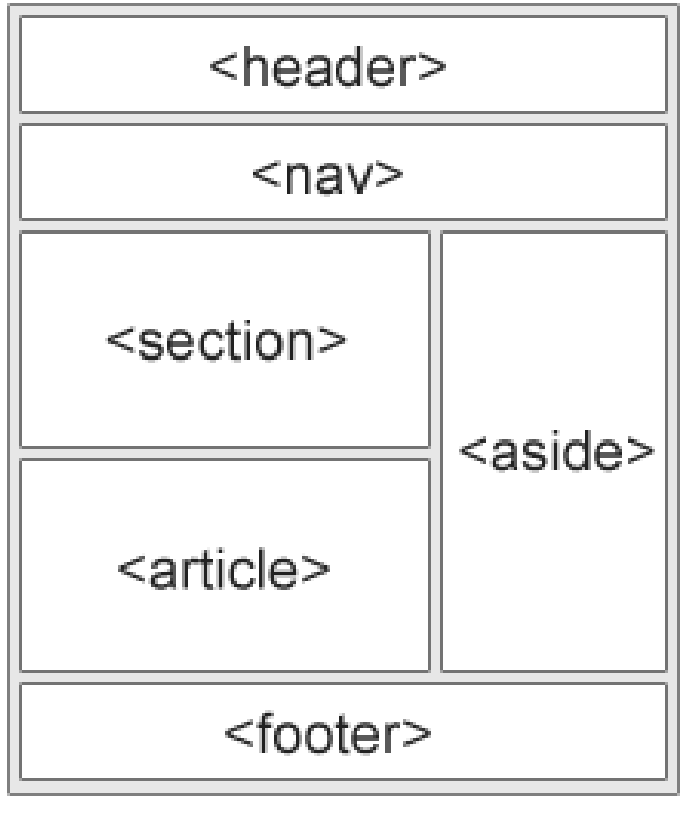
Paragraph 3 goes here.

Paragraph 4 goes. here.

SEMANTIC HTML

- While `div` is useful to group items together, you should always use the element that makes the most sense.
- Examples:

```
<header></header>
<nav></nav>
<article></article>
<main></main>
<footer></footer>
```



Reference: https://developer.mozilla.org/en-US/docs/Glossary/Semantics#Semantic_elements

ACTIVITY: SEMANTIC PAGE

- Reorganize the content on your index.html page by using semantic elements:
  - header
  - content area
  - sidebar
  - footer

ELEMENT: SPAN

```
<p>Paragraph with <span class="highlight">purple</span> text.</p>
```

```
.highlight {
    color: purple;
}
```

Paragraph with purple text.

- Inline element
- Often used to apply styles to inline text.

ACTIVITY: ADD SOME `spans`

- Add 1 or more spans to your content to help highlight text or make certain words or phrases stand out.

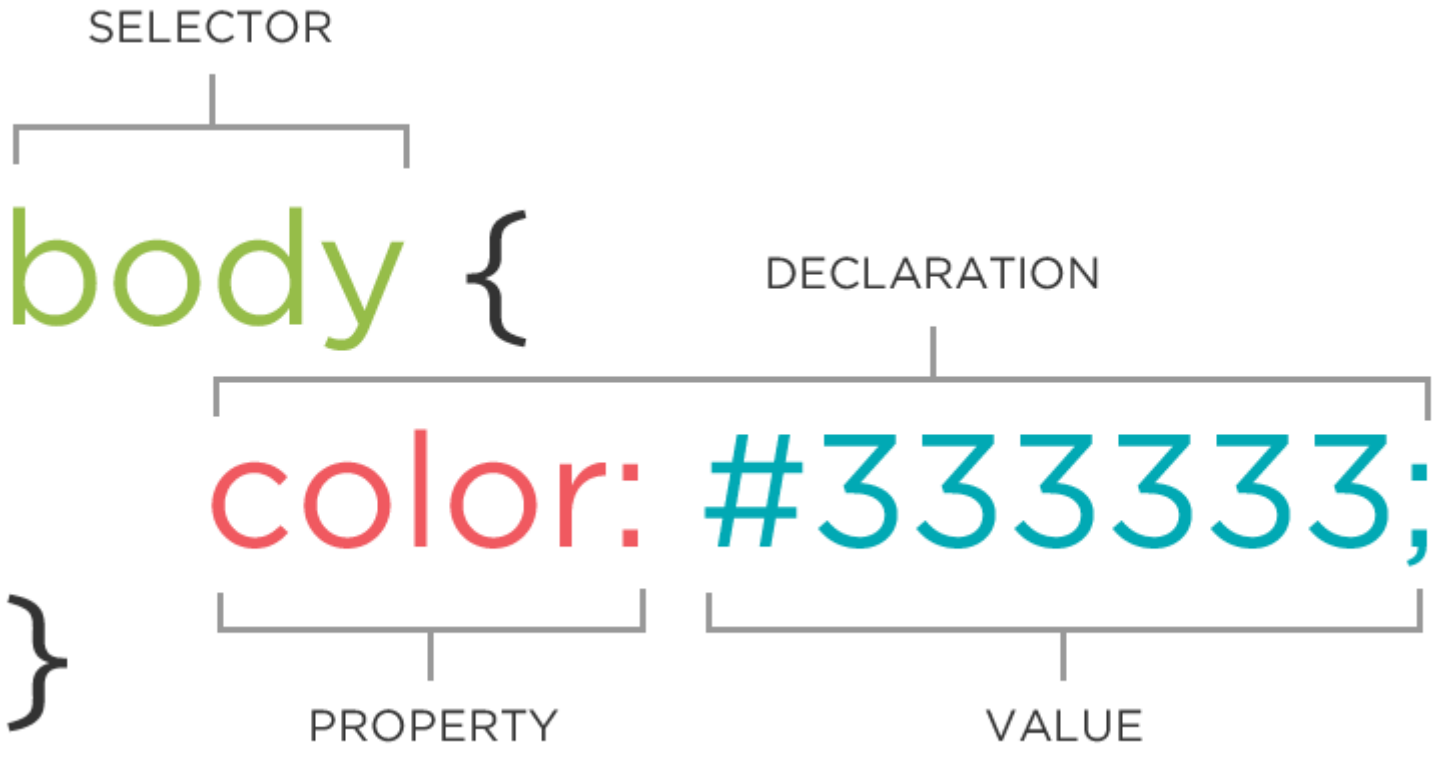
CSS

CSS: WHAT IS IT?

CSS = Cascading Style Sheets

- "style sheet language" that lets you style the elements on your page
- works with HTML, but is a separate language with different syntax

HOW TO WRITE CSS



CONNECTING CSS TO HTML

INLINE STYLES

```
<p style="color:red">Text goes here.</p>
```

Text goes here.

EMBEDDED CSS

```
<head>
    <style>
        p {
            color: blue;
            font-size: 12px;
        }
    </style>
</head>
```

EXTERNAL CSS

```
<head>
    <link rel="stylesheet" href="style.css">
</head>
```

ACTIVITY: WRITE CSS

1. In the same folder you used for writing HTML yesterday, create a new file named `style.css`.
2. Link to this external CSS file in the head of your HTML file.
3. Add the CSS rule below to the CSS file:

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

If you have more time, play around with more colors. Color Names

CSS SYNTAX

```
selector {
    property: value;
    property: value;
    property: value;
}
```

- Declaration groups are surrounded by curly brackets
- Declarations end with a semicolon!!!

SELECTING AN ELEMENT

Selects all paragraph elements.

```
p {
    property: value;
}
```

Selects all image elements.

```
img {
    property: value;
}
```

Other common selectors:

- li
- ul
- a
- em
- strong

CSS COLORS

The browsers will accept colors formatted in many different ways.

- Color Name - red
- Hexadecimal - #FF0000
- RGB - rgb(255,0,0)
- HSL - hsl(0, 100%, 100%)

You can explore colors at HTML Color Codes.

PROPERTY: COLOR

The `color` property changes the color of the text.

```
p {
    color: red;
    color: #ff0000;
    color: rgb(255, 0, 0);
}
```

PROPERTY: BACKGROUND-COLOR

- The `background-color` property changes the color of the background.

```
p {
    background-color: black;
    background-color: #000000;
    background-color: rgb(0, 0, 0);
}
```

ACTIVITY: WRITE MORE CSS

- Add some CSS rules to your style.css file
- Change the font color and background color of different elements in your HTML
- Try selecting links, paragraphs, and lists

PROPERTY VALUES

Each property can have one or more comma-separated values.

```
p {
    color: white;
    background-color: red;
    font-family: Arial, sans-serif;
}
```

PROPERTY: FONT-FAMILY

The `font-family` property defines which font is used.

```
p {
    font-family: "Times New Roman"; /* Specific font name */
    font-family: serif; /* Generic font */
    font-family: "Arial", sans-serif; /* Comma-separated list (font stack) */
}
```

PROPERTY: FONT-SIZE

The font-size property specifies the size of the font.

```
p {
    font-size: 12px; /* pixels */
    font-size: 1.5em; /* em */
    font-size: 100%; /* percentage */
}
```

PROPERTY: FONTS (SHORTHAND)

```
p {
    font-style: italic;
    font-weight: bold;
    font-size: 10px;
    font-family: sans-serif;
}
```

or

```
p {
    font: italic bold 10px sans-serif;
}

/* Syntax for shorthand */
body {
    font: font-style font-variant font-weight font-size/line-height font-family;
}
```

You need to supply at least `font-size` and `font-family` for the shorthand to work, otherwise it'll just be a syntax error and do nothing.

ACTIVITY: CHANGE YOUR FONTS

- Change the fonts of your page
- Try changing the font sizes and styles for different elements

MORE CSS PROPERTIES

Many CSS properties have self-explanatory names:

- background-color
- font-family
- font-size
- color
- width
- height

Here's a list to get you started.

SELECTOR: DESCENDANT COMBINATOR

```
<p>This is <strong>important</strong>!</p>
```

```
p strong {
    color: purple;
}
```

- Selects all `strong` elements that are within the paragraph.
- Allows for more specificity
- Looks for elements *inside* other elements
- Separate elements with a space

DESCENDANT COMBINATOR EXAMPLE

```
<ul>
    <li><a href="programs.html">Our <em>amazing</em> program.</a></li>
</ul>
```

```
ul li a em {
    color: purple;
}
```

What does this CSS do?

ACTIVITY: DESCENDANT COMBINATOR

- In your style.css file, try writing a descendant combinator selector.
- Remember, you need to look for an element inside another element.

SELECTOR: ID

HTML:

```
<!-- Rest of HTML -->
    <footer id="footer">
        <!-- Footer HTML -->
    </footer>
</body>
```

CSS:

```
#footer {
    property: value;
}
```

- Use the id attribute in HTML to add an id to your element.
- In your CSS file, select that element using the id selector `#`
- should only apply to one element on a web page

SELECTOR: CLASS

HTML:

```
<p class="warning">Run away!</p>
```

CSS:

```
.warning {
    color: red;
}
```

- Use the class attribute in HTML to add an class to your element.
- In your CSS file, select that element using the class selector `.`
- can apply to as many elements as you want

GENERAL CODING GUIDELINES

- Use class selectors to reuse CSS code (**D**on't **R**epeat **Y**ourself).
- Use ID selectors to target elements in JavaScript
- You can use ID selectors in CSS but it doesn't allow you to easily reuse code or maintain it like the class selector

ACTIVITY: USE THE ID AND CLASS SELECTORS

- Add at least 1 ID and 1 class to your index.html file
- Add CSS rules in your style.css file to target these elements

CASCADING

Styles "cascade" down until changed.

CSS:

```css
p {
    color: blue;
    font-family: 'Helvetica';
}

.red {
    color: red;
}

#special {
    font-family: Arial;
}
```

HTML:

```html
<p>Paragraph</p> <!-- blue, Helvetica -->
<p class="red">Paragraph</p> <!-- red, Helvetica -->
<p class="red" id="special">Paragraph</p> <!-- red, Arial -->
```

CSS SPECIFICITY

Each selector has a different priority and some can overwrite others.

MOST IMPORTANT TO LEAST IMPORTANT

1. `important!`
2. Inline CSS
3. ID
4. Class
5. Element

TIE-BREAKER

What happens when 2 CSS rules have the same level of specificity?

- Rules lower in the file overwrite rules higher in the file.

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

body {
    background-color: teal;
}

body { /* This rule wins */
    background-color: black;
}
```

CSS SPECIFICITY RESOURCES

Specificity only applies when the same element is targeted by multiple declarations. As per CSS rules, directly targeted elements will always take precedence over rules which an element inherits from its ancestor.

- W3 - Calculating a selector's specificity
- Understanding CSS Specificity with "The Shining"
- Specificity Calculator

PSEUDO-CLASSES

Selects regular elements under certain conditions.

```
selector:pseudo-class {
    property: value;
}
```

Examples:

```
/* Turns the link purple **when** you hover over it. */
a:hover {
    color: purple;
}

/* Selects any <p> **when** it is the first element within its parent element. */
p:first-child {
    color: lime;
}

/* Selects any <input> **when** the contents validate successfully */
input:valid {
    background-color: lightblue;
}
```

Resource: https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

LINK PSEUDO-CLASSES

```
a:link  {color: purple;} /* unvisited link */
a:visited {color: #db3c28;} /* visited link */
a:hover, a:focus {color: teal;} /* mouse over or select with keyboard*/
a:active {color: blue;} /* selected link */
```

SAIT

- a:hover **MUST** come after a:link and a:visited in the CSS in order to work properly
- a:active **MUST** come after a:hover

ACTIVITY: ADD PSEUDO-CLASSES

- Add pseudo-classes to your links.
- Remember when we used the `required` attribute on our forms?
- Use the `:required` pseudo-class to style required inputs.

PSEUDO-ELEMENTS

- Selects a specific part of an element.

```
selector::pseudo-element {
    property: value;
}
```

Examples:

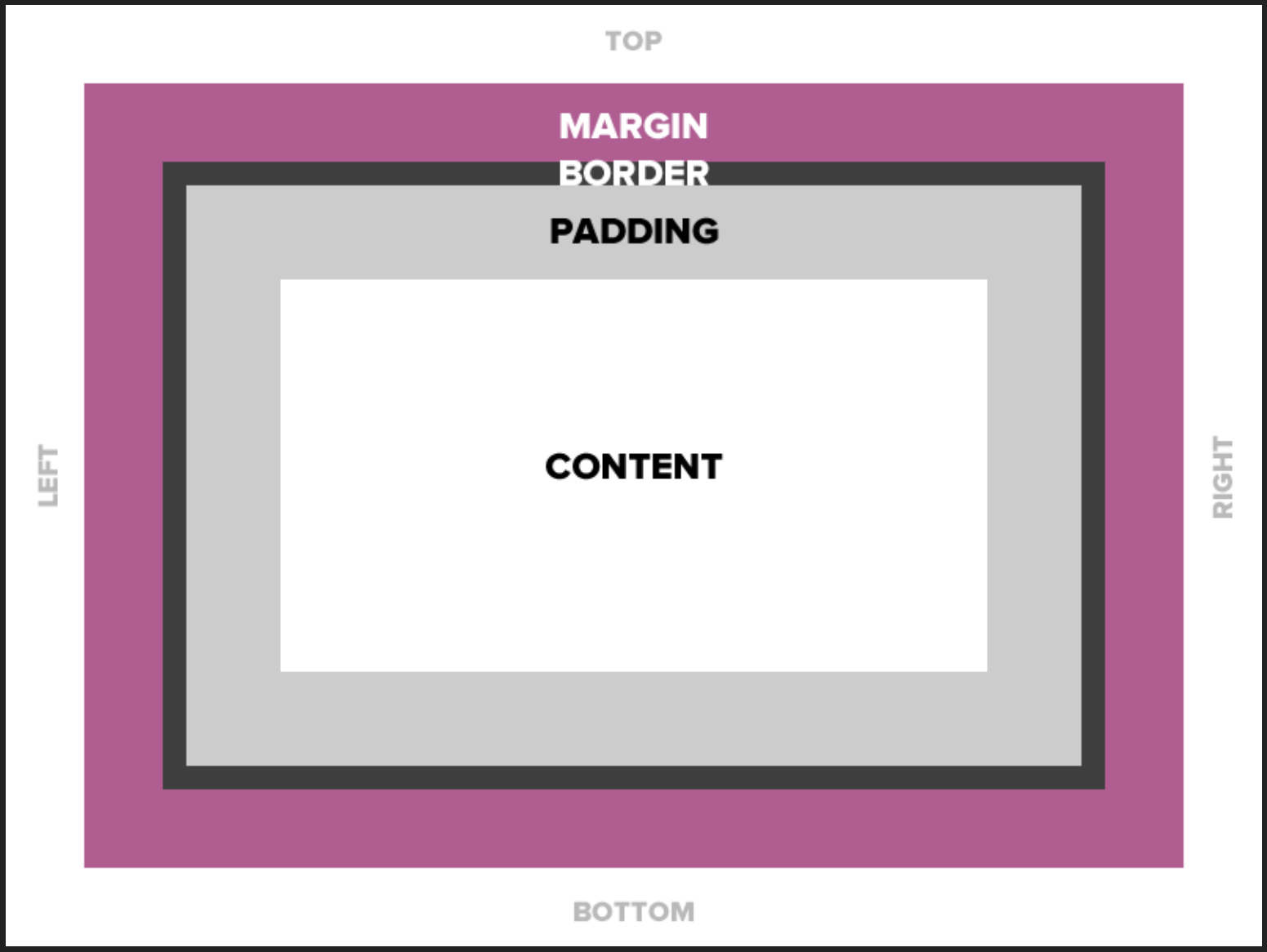
```
/* Add a heart before links */
a::before {
    content: "♥";
}

/* Selects the first letter of a <p> */
p::first-letter {
    font-size: 130%;
}

/* Selects the first line of a <p> */
p::first-line {
    color: red;
}
```

Resource: https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements

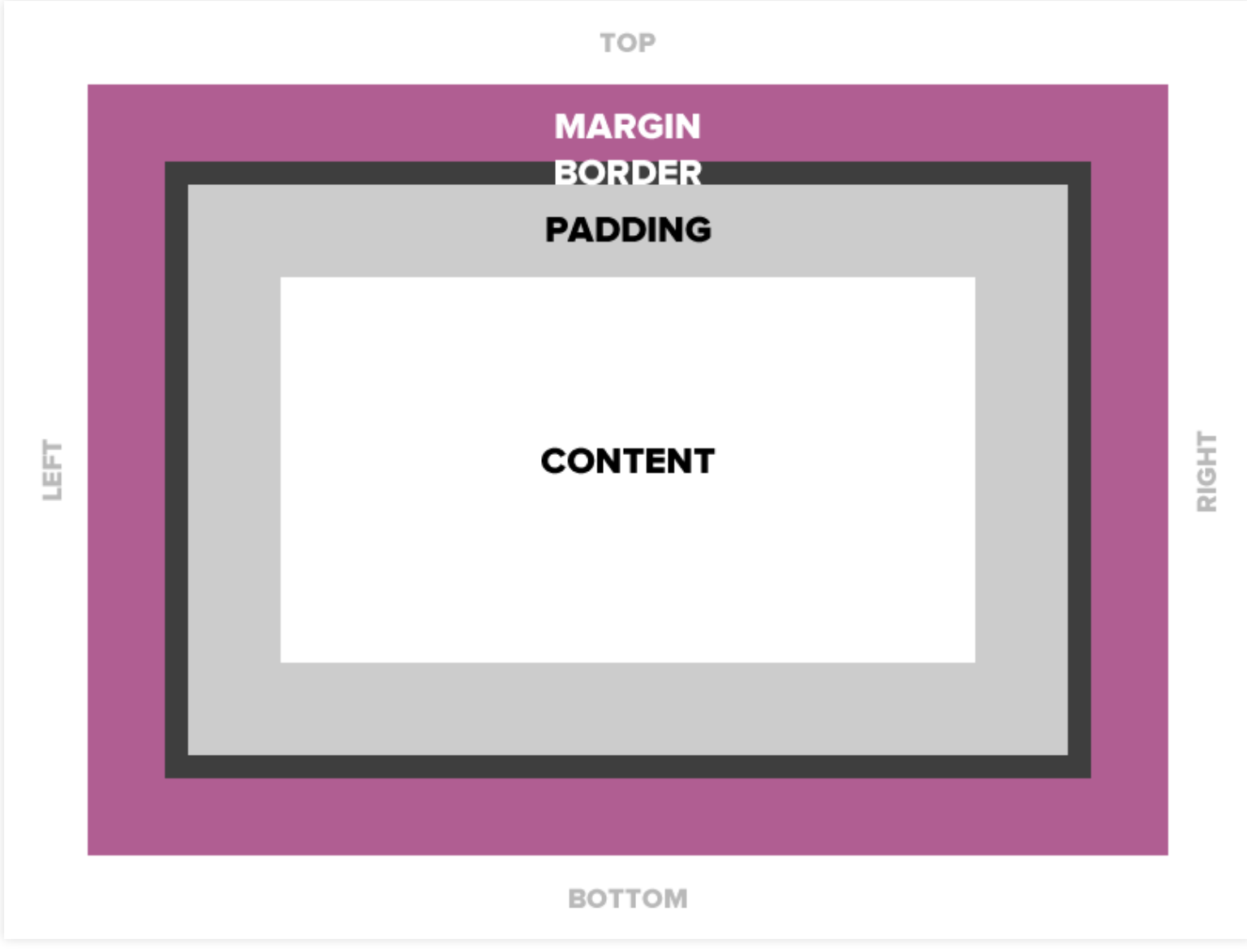
ACTIVITY: ADD PSEUDO-ELEMENTS

- Add pseudo-elements to your paragraphs.
- Use `::first-line` or `::first-letter`
- Try using `::selection` (https://developer.mozilla.org/en-US/docs/Web/CSS/::selection)

THE BOX MODEL

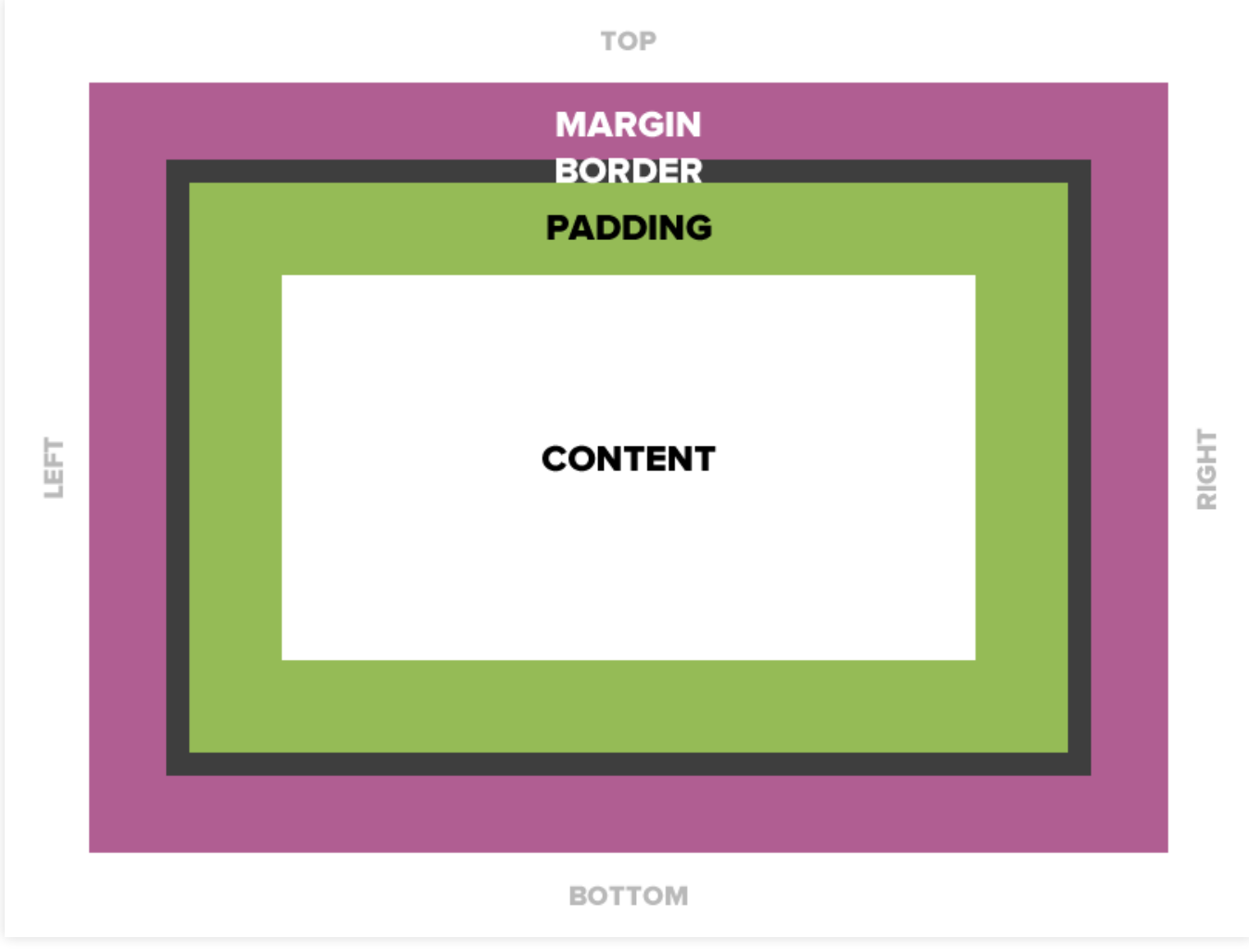


CONTENT



PADDING

Space between the border and the content.



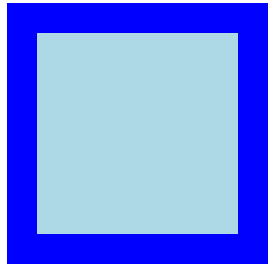
PADDING (CONT.)

Examples:

```
.box {
    padding: 15px; /* 15px padding on all sides */
    padding-top: 10px; /* 10px padding on top only */
    padding: 10px 5px 3px 5px; /* 10 on top, 5 on right, 3 on bottom, 5 on left */
}
```

By default, the padding is added to the total width of the box.

```
.box {
    width: 100px;
    height: 100px;
    padding: 15px;
}
```



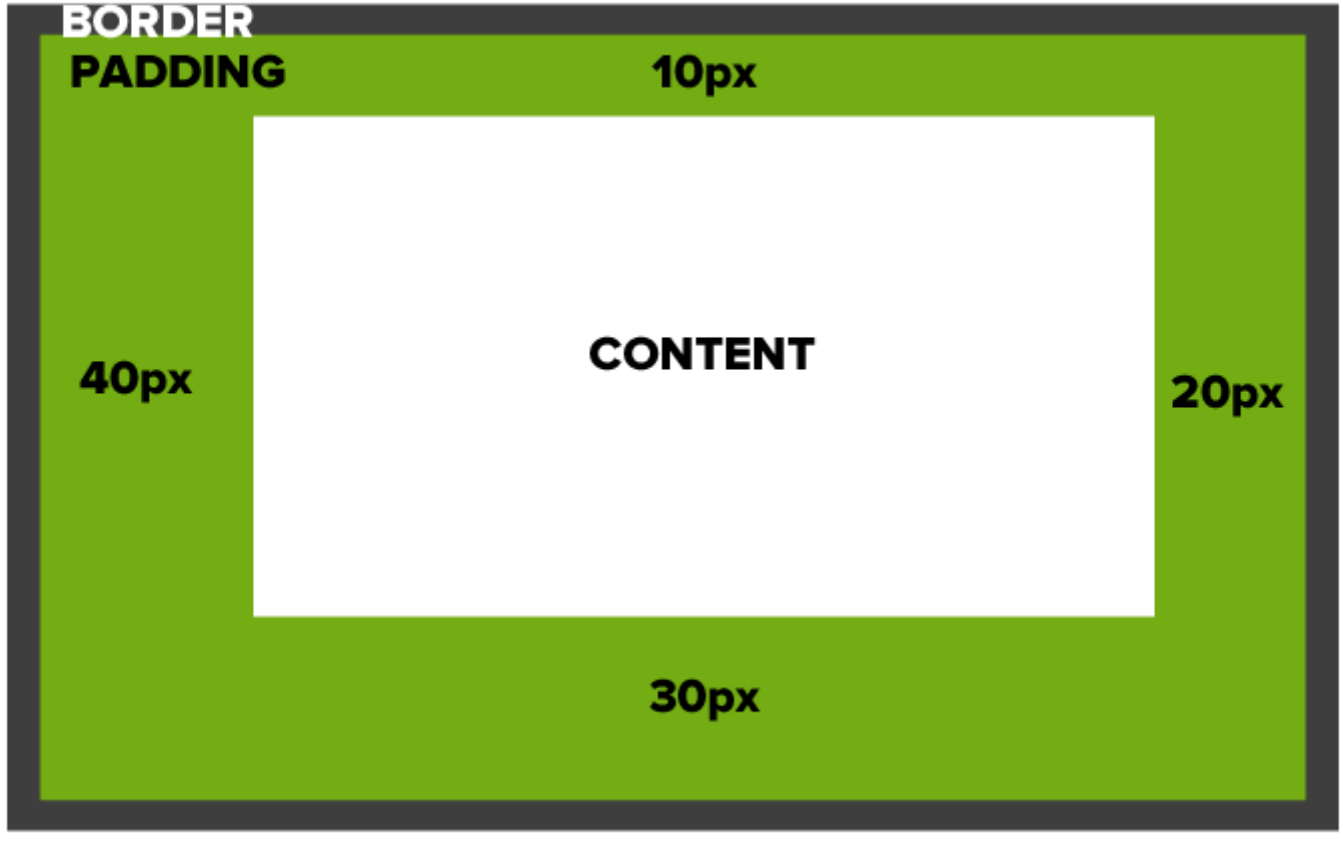
PADDING SHORTHAND

There are multiple ways to write out padding.

```
/* Four values */
padding: top right bottom left; /* clockwise */

/* Two values */
padding: top/bottom right/left;

/* One value */
padding: all;
```

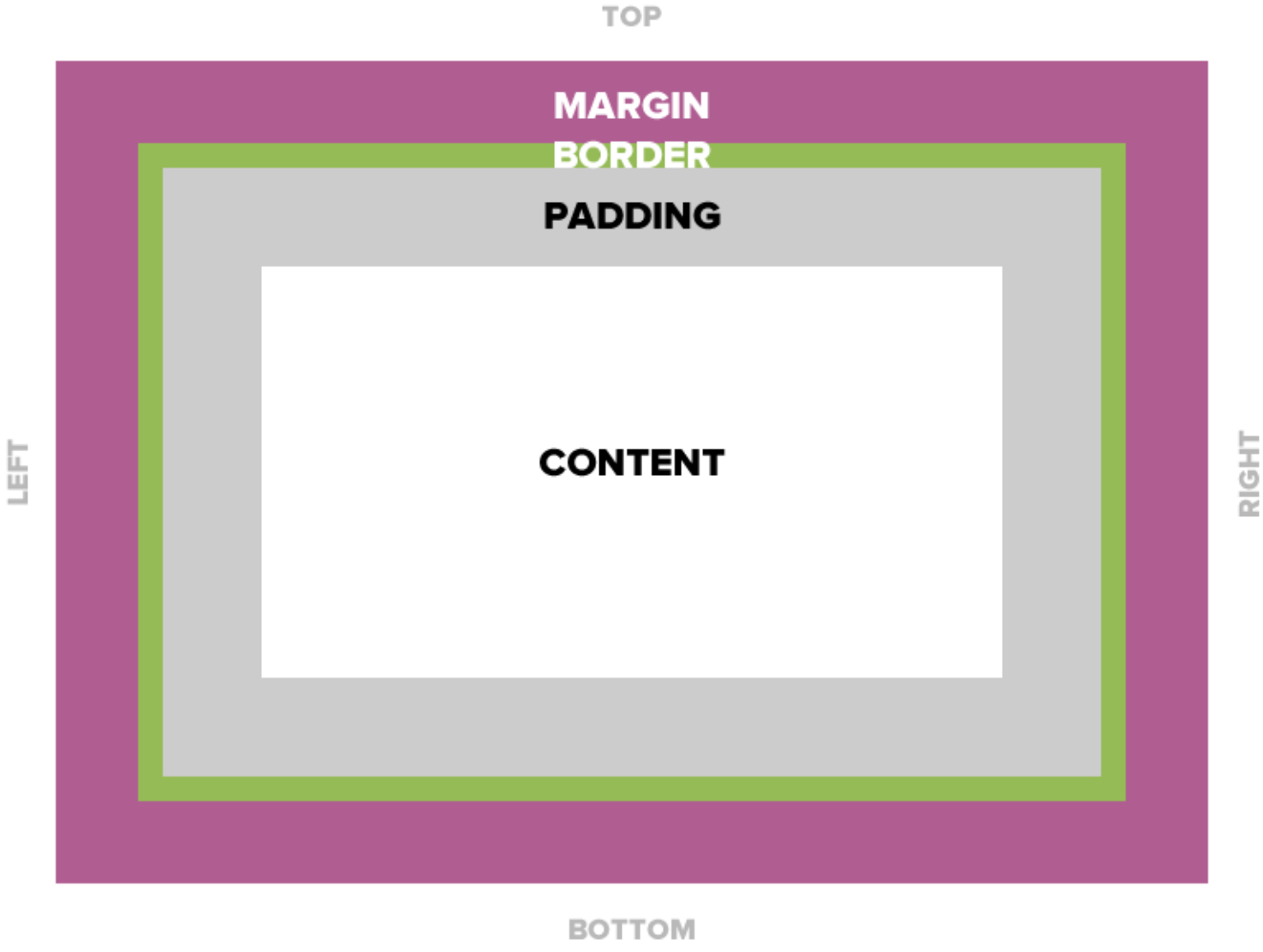
PADDING EXAMPLE

```
.box {
    padding: 10px 20px 30px 40px; /* clockwise */
}
```



BORDER

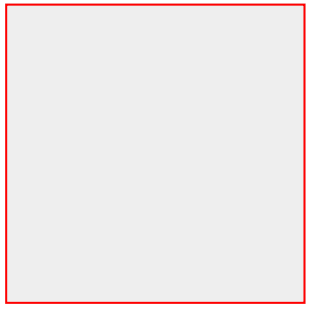
The edge around the box.



BORDER (CONT.)

```
.box {
    border: thickness style color;
}
```

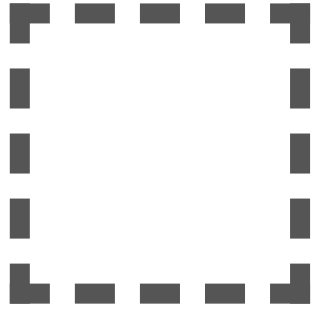
```
.box {
    /* A solid red border all around the box */
    border: 1px solid #ff0000;

    /* A thick dotted black top border */
    border-top: 4px dotted #000000;

    /* A solid red top border AND a thick dotted black bottom border */
    border-top: 1px solid #ff0000;
    border-bottom: 4px dotted #000000;
}
```



BORDER - LONGHAND

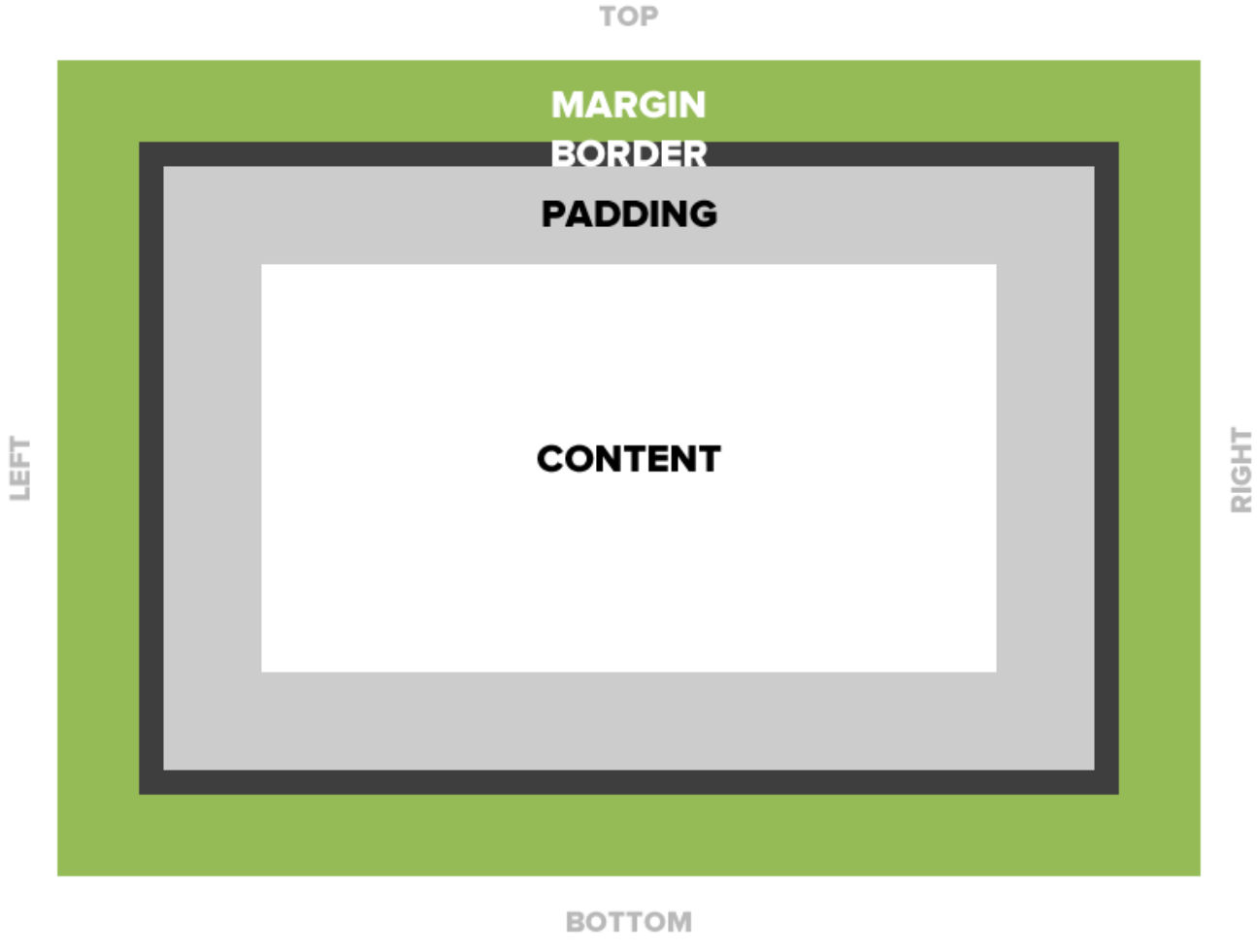
You can specify each property separately, or all three together.

```
.box {
    border-width: 10px;
    border-style: dashed;
    border-color: #999999;
}
```



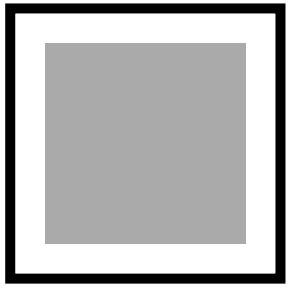
MARGIN

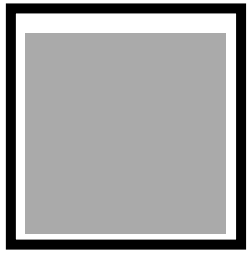
Transparent area **around** box that separates it from other elements.

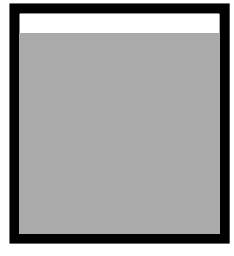


MARGIN (CONT.)

```
.box {
    /* 15 pixels on all sides */
    margin: 15px;

    /* 10 on top, 5 on right, 3 on bottom, 5 on left */
    margin: 10px 5px 3px 5px;

    /* 10 pixels on top */
    margin-top: 10px;
}
```



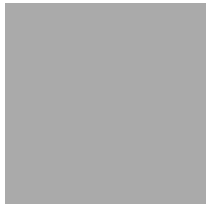




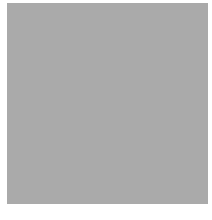
AUTO MARGIN

If margin is set to auto, it will take up as much space as possible.

```
.box {
    /* Centered */
    margin: auto;
    width: 100px;
}
```



```
.box {
    /* Flush-right */
    margin-left: auto;
    margin-right: 0;
    width: 100px;
}
```



ACTIVITY: PADDING, BORDERS, AND MARGINS, OH MY!

- Add padding, borders, and margins to your elements.

PROPERTY: WIDTH

Sets the width of an element. Does not include padding or border sizes. You'll need to mentally add these to width.

```
div#sidebar {
    /* Total width is 33% */
    width: 31%;
    padding: 1%;
    border: none;
}
```

Example: 31% width in blue,
1% padding in dark blue

PROPERTY: HEIGHT

Sets the height of an element.

```
p.alert {
    height: 50px;
    padding: 5px;
    border: 2px solid red;
}
```

ACTIVITY: WIDTH AND HEIGHT

- Add a width and height to different elements of your page
- Use classes to target specific elements with CSS

PROPERTY: BOX-SIZING

- It's unintuitive for width to only affect the size of the inner content of the box.
- 33% width should mean 33% **including** padding and the border, but it doesn't.

```
div {
    box-sizing: content-box; /* Default box-sizing behavior */
    box-sizing: border-box /* Accounts for border/padding in width/height */
}
```

CSS RESET

Developers like to use a CSS reset to apply box-sizing to their entire page.

```
/* apply a natural box layout model to all elements, but allowing components to
change */
html {
    box-sizing: border-box;
}
*, *:before, *:after {
    box-sizing: inherit;
}
```

Resource: https://www.paulirish.com/2012/box-sizing-border-box-ftw/

Alternatively, you can use an external CSS reset. There are many to choose from with pros and cons of each. Here are a couple of favorites:

- Modern-Normalize
- Sanitize

ACTIVITY: BOX-SIZING

- Add the CSS box-sizing rule at the top of your CSS stylesheet OR download and add an external reset stylesheet.
- Look at your index.html page. Do you need to change the width of your content in any places to account for `box-sizing: border-box`?

POSITIONING

STATIC POSITIONING

- HTML elements are positioned static by default
- Static elements are positioned in the normal flow of the page
- Static elements ignore top, bottom, right, or left property specifications

STATIC POSITIONING EXAMPLE

In normal flow, inline boxes flow from left to right, wrapping to the next line when needed.

```
<img src="img/cookie1.png"/>
<img src="img/cookie2.png"/>
<img src="img/cookie3.png"/>
...
<img src="img/cookie2.png"/>
<img src="img/cookie3.png"/>
```

STATIC POSITION EXAMPLE 2

In normal flow, block boxes flow from top to bottom, making a new line after every box.

```
<p>Hello</p>
<p>Hi</p>
<p>Greetings, Earthlings!</p>
```

Hello

Hi

Greetings, Earthlings!

RELATIVE POSITIONING

- Takes the element out of the normal flow, allowing it to be moved to the top, left, right, or bottom.
- Does not affect the elements surrounding it.
- Makes an element a "positioning context" in which to position other elements relative to it.

RELATIVE POSITION (CONT.)

The "relative" value will still put the element in the normal flow, but then offset it according to top/left/right/bottom properties.

```
.relative {
    position: relative;
    left: 80px;
    top: 20px;
    height: 100px;
    background-color: yellow;
}
```

ABSOLUTE POSITIONING

- Positions element outside of the normal flow.
- An absolutely positioned element is offset from its container block, positioned relative;
- Its container block is the first element that has a position other than static.
- If no such element is found, the container block is `<html>`.
- Other elements act as if it's not there.
- Determined by its offset values in the properties top, bottom, right, and left.

ABSOLUTE POSITIONING

oh hi

The "absolute" value will take the element out of the normal flow and position it in relation to the window (or the closest non-static element).

```
.top {
    position: absolute;
    top: 0;
    right: 10px;
    background-color: #ff596a;
}

.bottom {
    position: absolute;
    bottom: -40px;
    left:60px;
    background-color: #66cae1;
}
```

fancy meeting you here

Z-INDEX

Sometimes elements overlap. You can change the order of overlapping with z-index. The element with highest z-index goes on top.

```
.bottom {
    position: absolute;
    bottom: 10px;
    left: 60px;
    background-color: #ff596a;
}
.top {
    position: absolute;
    bottom: 15px;
    left: 60px;
    background-color: #66cae1;
    z-index: 2;
}
```

ACTIVITY: POSITIONING

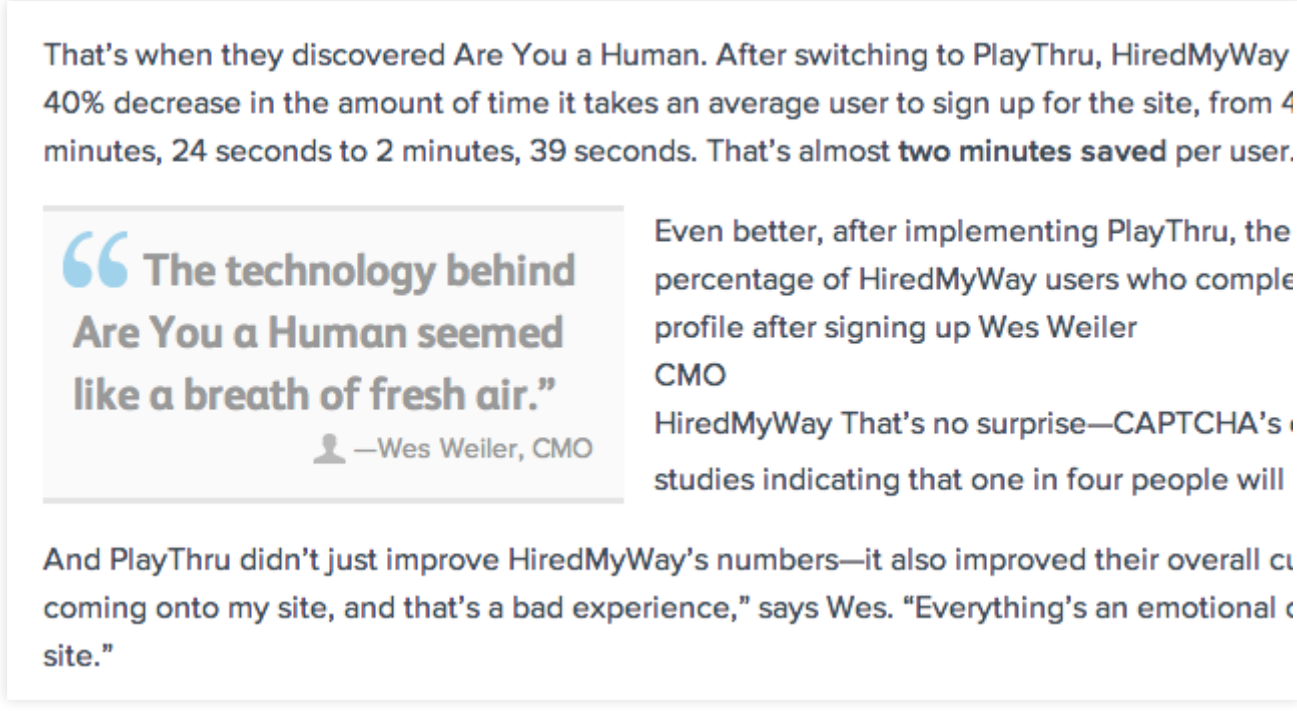
- Create a div that contains an image and a caption.
- Position the caption absolutely on top of the image.

Here's an example of what it *might* look like.

FLOAT

- Floated elements remain a part of the normal flow, but they move to the far left or right of their containing elements.
- Any other elements, such as paragraphs or lists, will wrap around the floated element.
- Always specify a width when floating an element, otherwise the element is likely to take up the whole page and not appear floated.

FLOAT EXAMPLE



FLOAT

```
.float {
    float: left;
    width: 200px;
    background: yellow;
}
```

Hi. I'm a yellow box with black text. I like to hang out on the left side.

Hi, I'm a paragraph and I'm not floating. Instead, I wrap around the floated element. Now I'm just going to babble on for a bit so that you can see how this works.

PLACING ELEMENTS SIDE BY SIDE

Float both elements in order to put them side by side.

```
.box1 {
    float: left;
    width: 300px;
}

.box2 {
    float: right;
    width: 400px;
}
```

width: 300px;
float: left;

width: 400px;
float: right;

CLEAR

- Clearing tells the element on which side (right, left, both) other elements cannot appear.
- If you had an image floated left, and you did not want the paragraph to appear next to it, you would add `clear: left;` to the paragraph.
- Clearing both sides makes sure floats don't flow past the clear element.

```
.clear {
    clear: right;
    clear: left;
    clear: both;
}
```

CLEAR EXAMPLE

```
.float {
    float: left;
    width: 50px;
    background: yellow;
}
.clear-left {
    clear: left
}
```

hi Non-floating element
hi Non-floating element
hi

Non-floating element with a class of **.clear-left**

ACTIVITY: FLOAT

- Float your side bar and content areas so that they are side by side.

Example:

MORE POSITIONING

- Absolute positioning, relative positioning, and floats will take you far. But here are some modern techniques:

FLEXBOX

- Basic concepts of flexbox
- Flexbox Froggy
- A Complete Guide to Flexbox

CSS GRID

- CSS Grid
- A Complete Guide to Grid
- Grid Garden

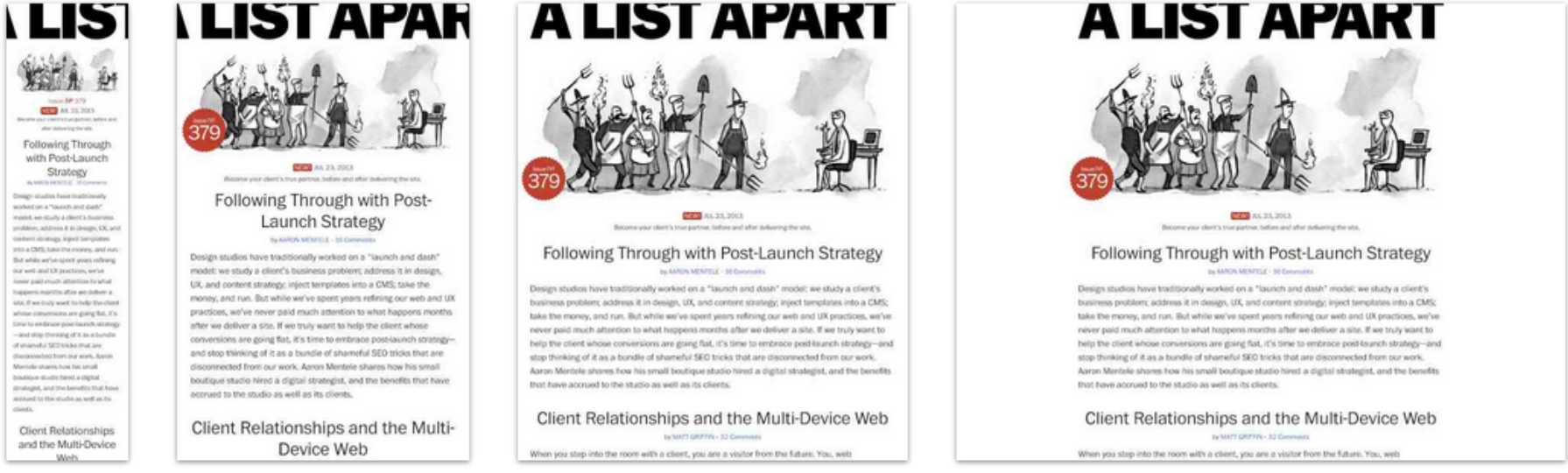
RESPONSIVE WEB DESIGN

WHAT IS IT?

RWD suggests that the design and development of a site should respond to the user's behavior and environment.

WHY WE USE IT

RWD modifies the presentation of a site without modifying the content of the page. Every user has access to the same information.



RWD INGREDIENTS

1. Viewport Meta
2. Fluid grids
3. Flexible images
4. CSS Media Queries

RESPONSIVE META TAG

Use this tag to optimize your website for mobile devices.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">

        <!-- Viewport Meta Tag -->
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <title>Title of page goes here</title>
    </head>
</html>
```

- Width=device-width makes the viewport the size of the device.
- User-scalable=true allows the user to pinch and zoom on your site.

FLUID GRIDS

FIXED-WIDTH SITES

- Have to manually adjust the height and width of elements

FLUID GRID

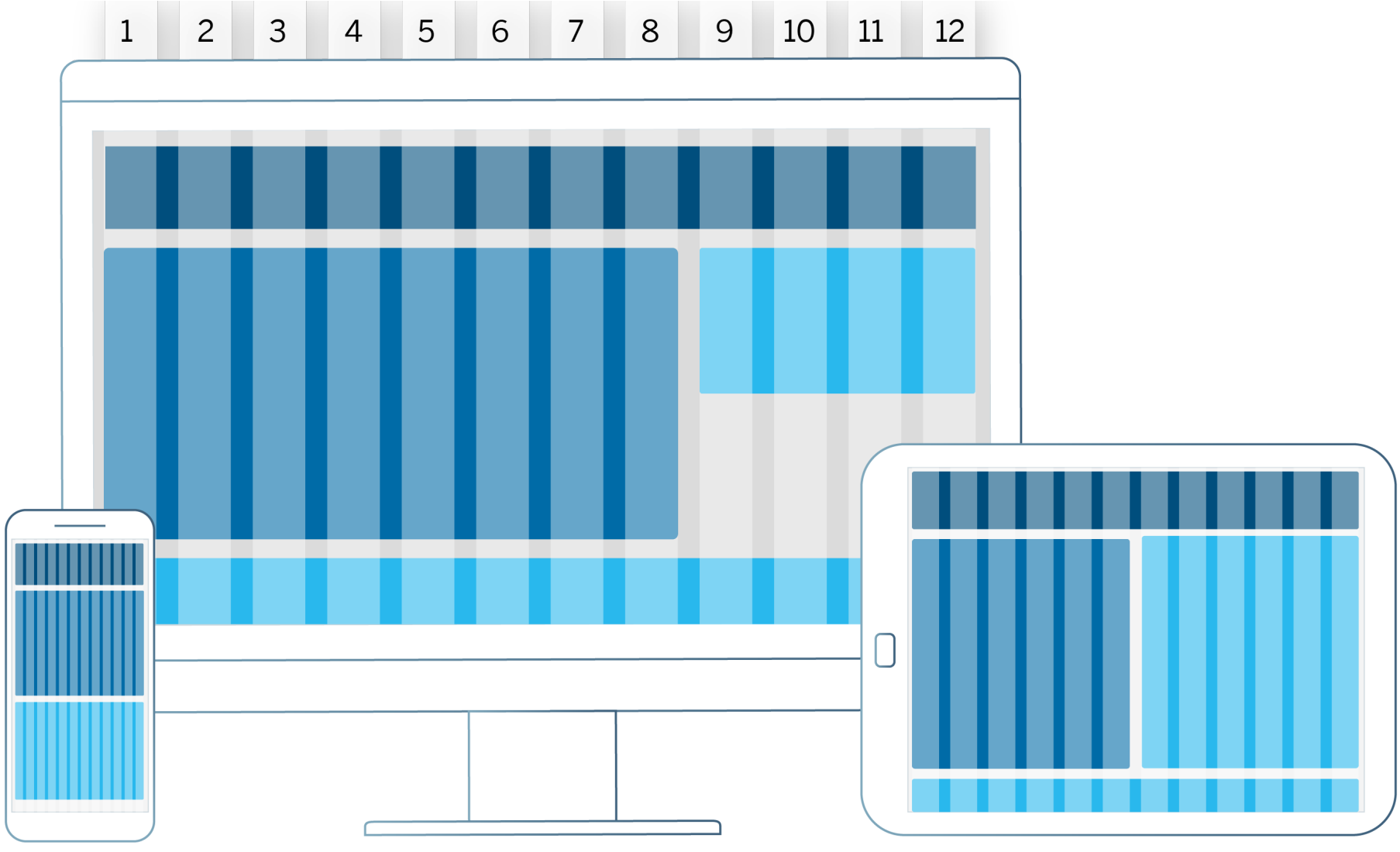
- The device resolution/screen size determines the height and width of elements on your page

STEPS TO A FLUID GRID

1. Define a maximum width for the container
2. Divide the content into a set of columns (usually 12)
3. Design elements with proportional widths and heights instead of being stuck with specific pixel dimensions

Whenever the device width changes, the grids change in width to scale with the device.

FLUID GRID



RESPONSIVE WIDTH

```
/* The width of the container element is 100% */
div.container {
    width: 100%;
}

.article {
    float: left;
    width: 75%;
    padding: 2%;
}

.sidebar {
    float: right;
    width: 25%;
    padding: 2%
}
```

- Don't forget `box-sizing: border-box`

ACTIVITY: MAKE A RESPONSIVE WEBSITE

- Go through your index.html file (or create a new one)
- Anywhere you've added width, check to see if you can use percentages instead.
- If you create a new file instead, add:
  - a header
  - a main content section
  - a sidebar
  - a footer
- Use % widths to scale the sizes of these elements on different devices

FLEXIBLE IMAGES

- Text scales easily on smaller devices, but images are a bit tricky.
- Images will overflow their container elements if they're too big.
- Max-width to the rescue!

PROPERTY: MAX-WIDTH

```
/* Allow images to only expand to the size of their parent. */
img {
    max-width: 100%;
}
```

ACTIVITY: MAX-WIDTH IMAGES

- Add the max-width property to your images.

MEDIA QUERIES

Media queries apply CSS in specific situations:

- Print media
- small screens (mobile)
- medium-sized screens (tablet)
- large screens (desktop)

ADD MEDIA QUERIES LAST

- Use the Cascade to your advantage. Overwrite previous CSS by place media queries below regular CSS.

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

@media screen and (min-width: 1200px) {
    body {
        background-color: red;
    }
}

@media print {
    body {
        background-color: white;
    }
}
```

Resource: https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

ACTIVITY: USE MEDIA QUERIES

- Work on your index.html page and add media queries