香港中文大學
The Chinese University of Hong Kong

CSCI2720 - Building Web Applications

Lecture 4: CSS

Dr Colin Tsang

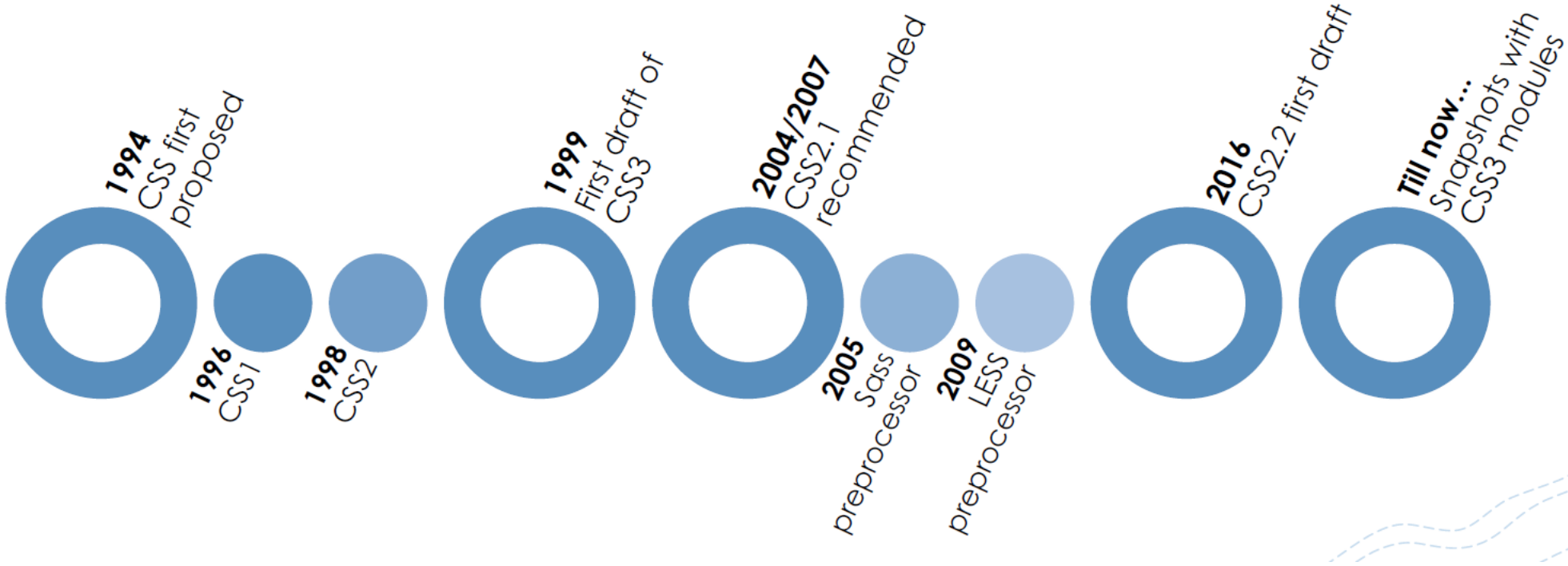
CSS Basics

- Basics
- Using CSS with HTML
- Inheritance and cascading
- Selectors and properties
- Inline vs. block-level elements
- Displaying and positioning
- The box model
- Responsive web design

CSS Basics

- CSS – Cascading Style Sheets
- It is not a programming language, but is for styling contents in HTML
- Designed to enable the separation of content and presentation

History of CSS



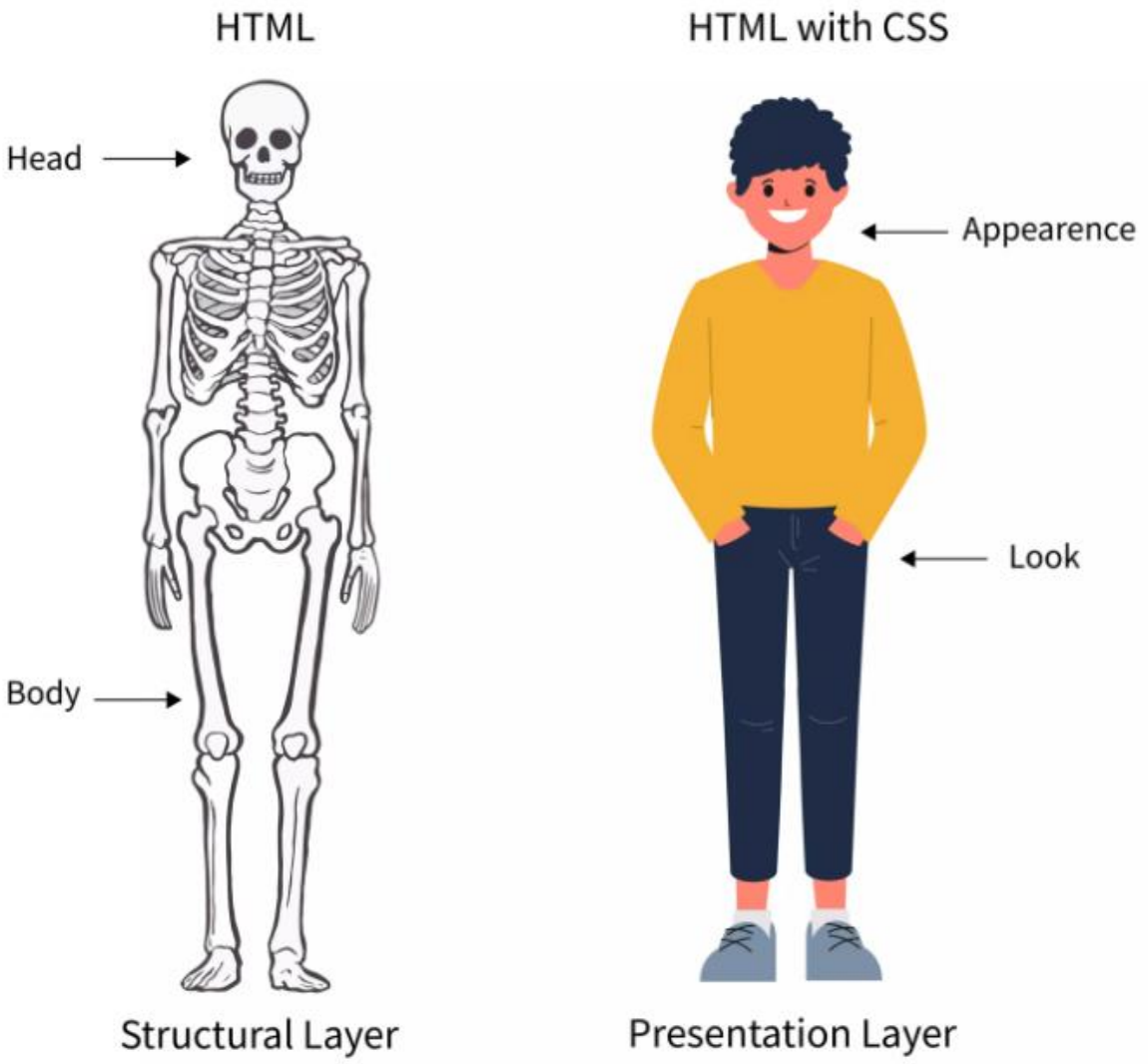


Image from: https://www.scaler.com/topics/difference-between-html-and-css/

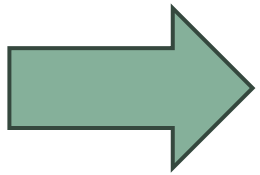
Why CSS?

- Every element in HTML that are presentable has a set of style properties that can be modified via CSS
  - e.g., *font-family*, *color*, *line-height* of *<p>*
- Separating design and contents:
  - Hopefully handled by different teams in development
  - Easily changing the skin of a web page
  - Sharing of the stylesheet among pages on the same site

Coding in CSS without Bootstrap

```html
<!DOCTYPE html>
<html>
<head>
  <title>CSS Example</title>
  <style>
    /* CSS styles */
    h1 {
      color: blue;
      font-size: 24px;
    }
    p {
      color: red;
      font-size: 16px;
    }
    .highlight {
      background-color: yellow;
    }
  </style>
</head>
<body>

  <h1>This is a heading</h1>

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

  <p class="highlight">This paragraph has a highlight class applied to it.</p>

</body>
</html>
```

. = class attribute



This is a heading

This is a paragraph.

This paragraph has a highlight class applied to it.

CSS Syntax

- Like HTML, CSS is generally not case-sensitive
  - Except HTML attribute values, e.g, the value of id="SomeName".
  - You must use the American spelling color……

Selector →

Declaration of property names with values

```
p {
  font-family: "Arial", "Helvetica", sans-serif;
  color: orange;
  line-height: 2em;
} /* a piece of comment, to be ignored by computer */
```

Using CSS in HTML

- If the task is to change the behaviour of <p> in an HTML file, there are multiple ways:
  - **External style sheet**: where a stylesheet file (.css) is linked
  - **Internal style sheet**: the styles are included in the HTML head
  - **Inline styles**: specifying the behaviour for a particular tag directly using a style attribute

- More commonly, CSS could be created or changed using *scripts* to increase interactivity, changing link colours.

External style sheet

- Include an external stylesheet using <link> in <head>
  - You can link a CSS file to multiple HTML file.

```
<head>
...
<link rel="stylesheet" href="style1.css" >
...
</head>
```

```
h1 { text-align: center; font-family: Arial; }
h2 { color: #440000;
     text-align: center;
     font-family: Arial Black, Arial, Helvetica;
}
```

style1.css

Internal style sheet

- Putting a <style> tag inside <head>
  - Can only affect the current HTML file.
  - Will override the external style sheet

```
<head>
...
<style>
  hr { color: sienna; }
  p { margin-left: 20px; }
  body { background-image: url("images/back40.gif"); }
</style>
...
</head>
```

Inline styles

- Set a style directly using a style attribute in the target tag
  - Will override external & internal style sheets.

```
<p style="color: sienna; margin-left: 20px;">
This is a paragraph
</p>
```

Inheritance and cascading

- A child inherits (copies) the parent's properties if unspecified
- The idea of cascading reflects priority of CSS rules:
  1. Overriding importance: **inline** > **internal** > **external**
  2. More *specific ones* override generic ones
  3. Later ones override earlier ones

  - Properties marked *!important* overrides everything else

    If it's in external file, it still override everything.

    It has to mark after every property (not p/h1).

    eg. font-size: 24px !important

Element and pseudo-element selectors

Element selectors	Description
`p`	Select all `<p>` elements
`h1, h2`	Select all `<h1>` and `<h2>` elements
`*`	Select all elements
`p a`	Select all `<a>` elements that is a child of a `<p>` element

Pseudo-element selectors	Description
`p:nth-child(3)`	Select all the `<p>` elements that are the 3rd child
`p::first-letter`	Select the first letters of all `<p>` elements

- See: https://blog.bitsrc.io/css-pseudo-selectors-you-never-knew-existed-b5c0ddaa8116

ID and class/pseudo-class element selectors

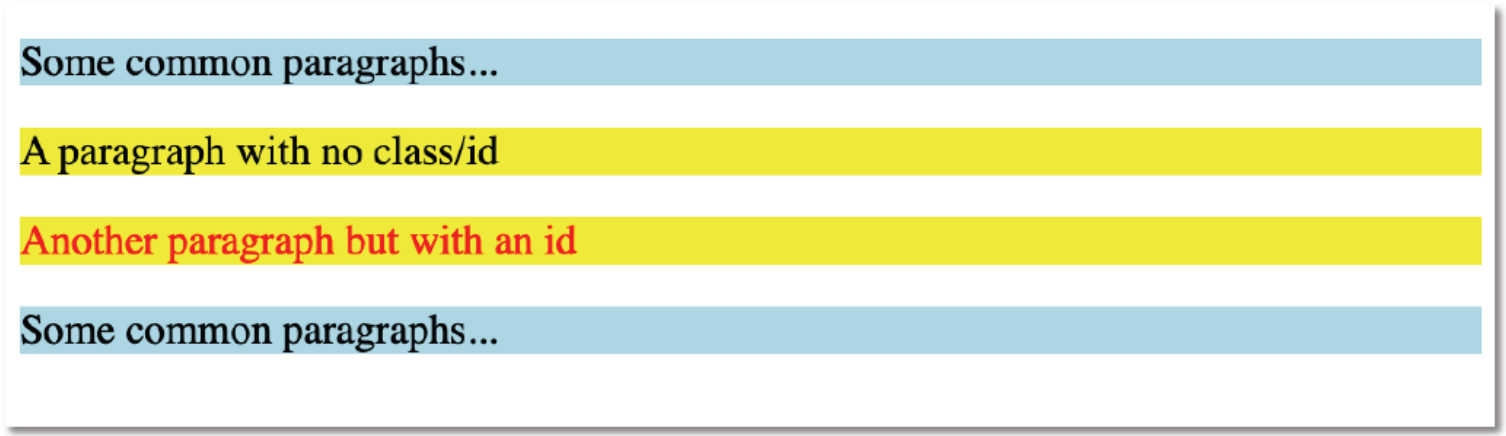
ID and class selectors	Description
`#example`	Select the only HTML element having attribute `id="example"` *Note: the* `id` *value should be unique in the document*
`.new`	Select all HTML elements having attribute `class="new"`
`p.new`	Select all `<p>` elements having attribute `class="new"`
`p a`	Select all `<a>` elements that is a child of a `<p>` element

Pseudo-class selectors	Description
`a:hover`	Select all `<a>` elements that has the mouse cursor over it
`a:link`	Select all unvisited `<a>` elements

An example of ID and class

```
<p class="lightblue">Some
common paragraphs...</p>
<p>A paragraph with no
class/id</p>
<p id="new">Another paragraph
but with an id</p>
<p class="lightblue">Some
common paragraphs...</p>
```

```
/* any p element */
p { background: yellow; }
/* p of class "lightblue", more specific */
p.lightblue { background: lightblue; }
/* any element of id "new" */
#new { color: red; }
```



Some useful properties

- There are way too many properties you can set in CSS stylesheets
- Learn the useful properties and their possible values, and then look up new ones when needed
  - Text: *font-family*, *font-size*, *font-weight*, *color*,…
  - Layout: *text-spacing*, *line-height*, *text-align*,…

- Want more? Read: https://css-tricks.com/lets-look-50-interesting-css-properties-values/

Fonts

- Besides using installed fonts on the user's computer, you can also use web fonts with *@font-face* selector

- There are popular online font repositories that you can use the fonts freely (under certain licenses)
  - e.g., https://fonts.google.com/

Length units

- **px**
  - One dot on screen (pixel)
- **em**
  - Relative to current font size
- **rem**
  - Relative to the root element font size
- **%**
  - Size of the same property of the parent
- **vh**
  - 1% of the viewport (browser screen) height
- **vw**
  - 1% of the viewport width

- You can also use printed units like cm or in, yet results could be unexpected.
- See: https://engageinteractive.co.uk/blog/em-vs-rem-vs-px

Colours

- A few different ways to represent colours in CSS:
  - Colour names, e.g., white, black, green,…
  - A combination of Red, Green, and Blue values: #rrggbb, where each of rr/gg/bb are hexadecimal values from 0 to ff.
  - Other function including rgb(), hsl(),…

- Mind the spelling must be *color* but not colour.

- See: https://www.w3schools.com/colors/colors_picker.asp

Inline vs block-level elements in HTML

- There are different kinds of HTML elements
  - *p*, *h,1*, *table*, *li*, … = block-level elements
  - *i*, *a*, *img*, … = inline elements

- How do they differ?
  - Block-level elements occupy the *full width* and *enforces to start at a new line*.
  - Inline elements *can start anywhere*, and it *cannot be set a width and height*.

Inline VS. Block-level elements

- Two special generic HTML elements usually used for applying CSS styles
  - * * is *block-level*, which is often used as a container for layouts
  - ** is *inline*, which is often used for enclosing a group of text for markup

- The inline-block: an inline-level block element, but you can apply hight and width

- See: https://learnlayout.com/inline-block.html

Displaying

- An element that is block-level can be changed to inline, and vice versa.
  - This is especially useful for laying out elements while keeping their semantic meanings
    - e.g., keeping a list of links in <nav>
  - *Graceful degradation*: a design principle focuses on ensuring a web app remains functional and usable even with certain features are not supported.
    - The list is displayed with browser defaults without CSS

```
<nav>
<ul>
  <li>Link A</li>
  <li>Link B</li>
  <li>Link C</li>
</ul>
</nav>
```

```
nav li {
  display: inline;
  background: yellow;
  margin: 5px;
}
```

Displaying

- There are many interesting option for displaying an element, besides inline, block, and inline-block
  - See: https://www.w3schools.com/cssref/pr_class_display.php

- To NOT display an element, you may set…
  - *Display: none;*
    - The element occupies no space at all
  - *Visibility: hidden;*
    - The element still takes the space

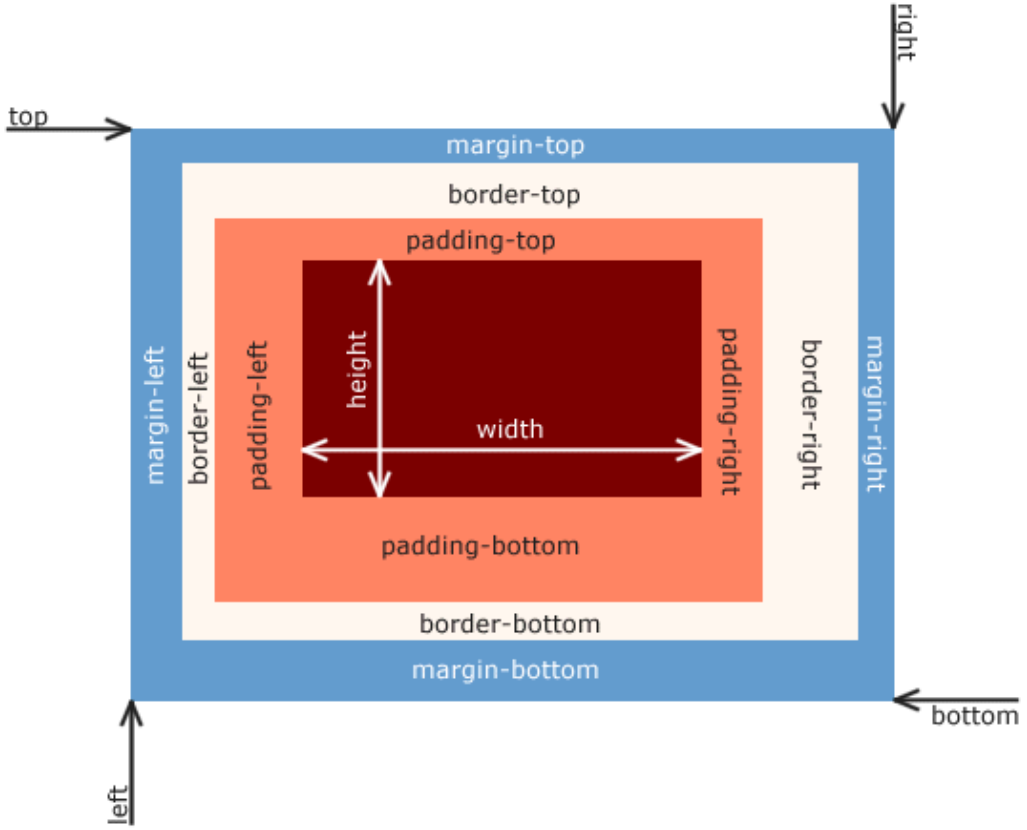
Positioning

- By default, all HTML elements has a static position
- Four other possibilities
  - *Absolute*: define the top-left using top and left properties Relative to the container of the element
  - *Fixed*: positioned relative to the browser window Relative to the window upper left corner
  - *Relative*: relative to original static position Compare to its original position
  - *Sticky*: position becomes fixed at a certain scroll position, often used for navigation bar or site title bar

- See: https://www.w3schools.com/cssref/playit.php?filename=playcss_position

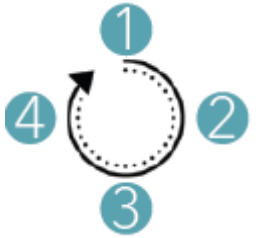
The box model

- All HTML elements are considered boxes, and they can occupy some space according to these properties:
  - **Height/width**: the content area
  - **Max-height, min-height, max-width, min-width**: the limits when resizing window
  - **Padding**: internal space, taking background color from contents
  - **Border**: lines surrounding the box
  - **Margin**: external space, taking background color of parent element



The box model

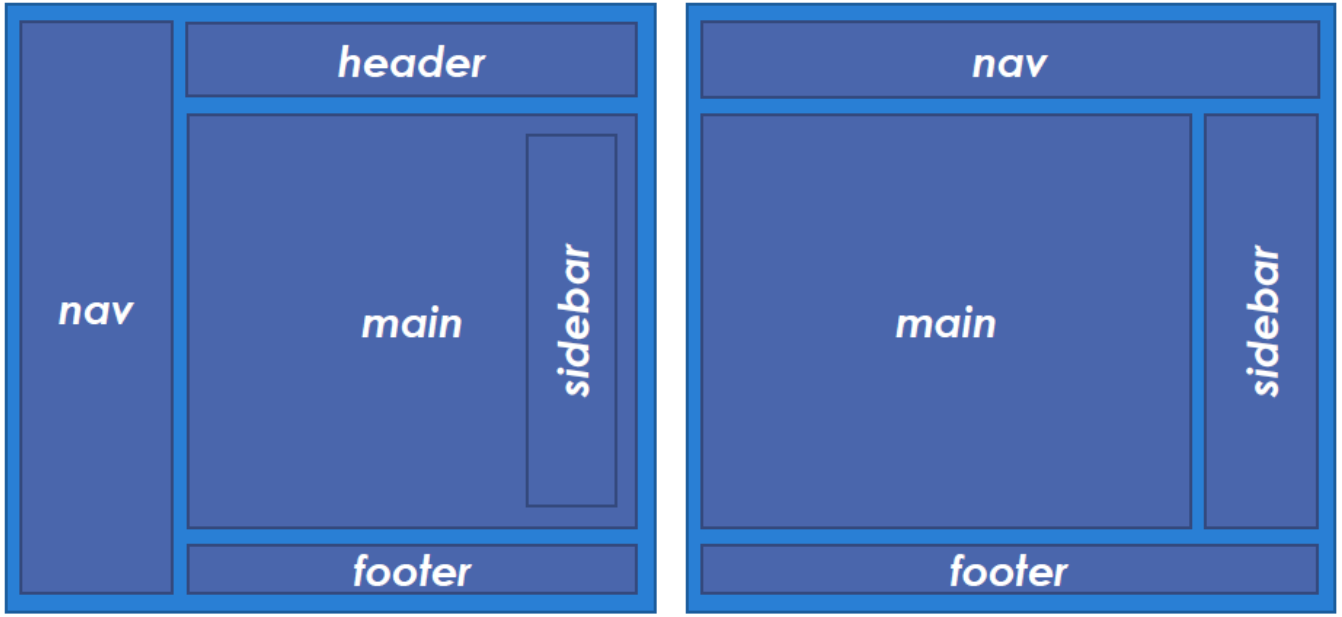
- The margins or paddings are often specified with a shorthand in this order:
  - Top, right, bottom, left.



`padding: 1px 2px 3px 4px;` `margin: 4px 3px 2px 1px;`	top, right, bottom, left padding/margin being set accordingly
`padding: 5px 0px;` `margin: 10px;`	Only two values: top/bottom, left/right Only one value: all sides
`padding-left: 5em;` `margin-top: 2em;`	Also possible to set values independently
`margin: 0 auto;` `border: 2px dotted red;`	To align a box in middle, use auto x-margin

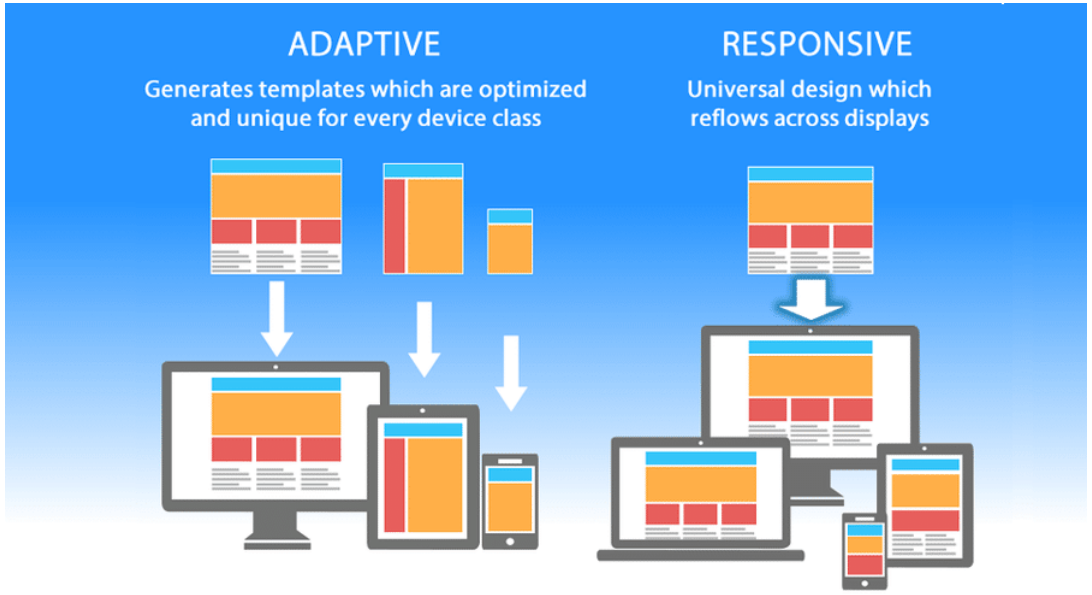
Preparing layouts

- Laying out in CSS are just arranging boxes
- It’s a matter of imagination



Responsive web design

- People are using all kinds of devices to visit your page, perhaps on a mobile phone, or with a huge screen at home.
- Responsive web design (RWD) ensures the web pages to *render well* depending on the screen size *with one design*.
- See: https://kinsta.com/blog/responsive-web-design/



Responsive web design

- Size contents to the viewport
  - Set viewport width to device screen width and zoom at 100%
  - `<meta name="viewport" content="width=device-width, initial-scale=1">`
  - Latest browser do it by default

  - Avoid large fixed-width elements, or make assumption on view port size
  - *Mobile-first* design
  - scale/rearrange elements using CSS *@media* queries (*not required in this course*).

Responsive handling of images

- Images can be scaled with parent size

```
img {
  max-width: 100%;
  height: auto;  /* keeping aspect ratio */
}
```

- The picture elements can load different images basing on screen size

```
<picture>
  <source srcset="cuhk-small.jpg" media="(max-width: 500px)">
  <source srcset="cuhk.jpg">
  <img src="cuhk.jpg"> <!-- backward compatibility -->
</picture>
```

CSS transforms, transitions, and animations

- 2D and 3D transforms
  - *Translate()*
  - *Rotate()*
  - *Skew()*

- Transition: to specify a different hover behaviour
- Animations: specify different behaviours for keyframes

- See: https://learn.shayhowe.com/advanced-html-css/transitions-animations/

CSS preprocessors

- For easier and more efficient web design
- More organized and cleaner code
- Simplified work with variables, special selectors, etc.
- Source code to be complied into regular CSS


```
$font-stack:    Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

Compile Sass into CSS

```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

CSS GURUS

- CSS is very powerful to dramatically alter the appearance of a web page. There are simply too much that can be done.
  - Even rendering a “game”: CSS only monument valley
  - https://codepen.io/miocene/pen/NWRWQpX

- You don’t need to learn everything.

- Know the syntax and learn reading the documentations.

Further readings

- w3schools CSS tutorial:
- https://www.w3schools.com/css/

- MDN introduction to CSS
- https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps