香港中文大學
The Chinese University of Hong Kong

CSS

CSCI2720 2022–23 Term 2

Building Web Applications

Dr. Chuck-jee Chau
chuckjee@cse.cuhk.edu.hk

OUTLINE

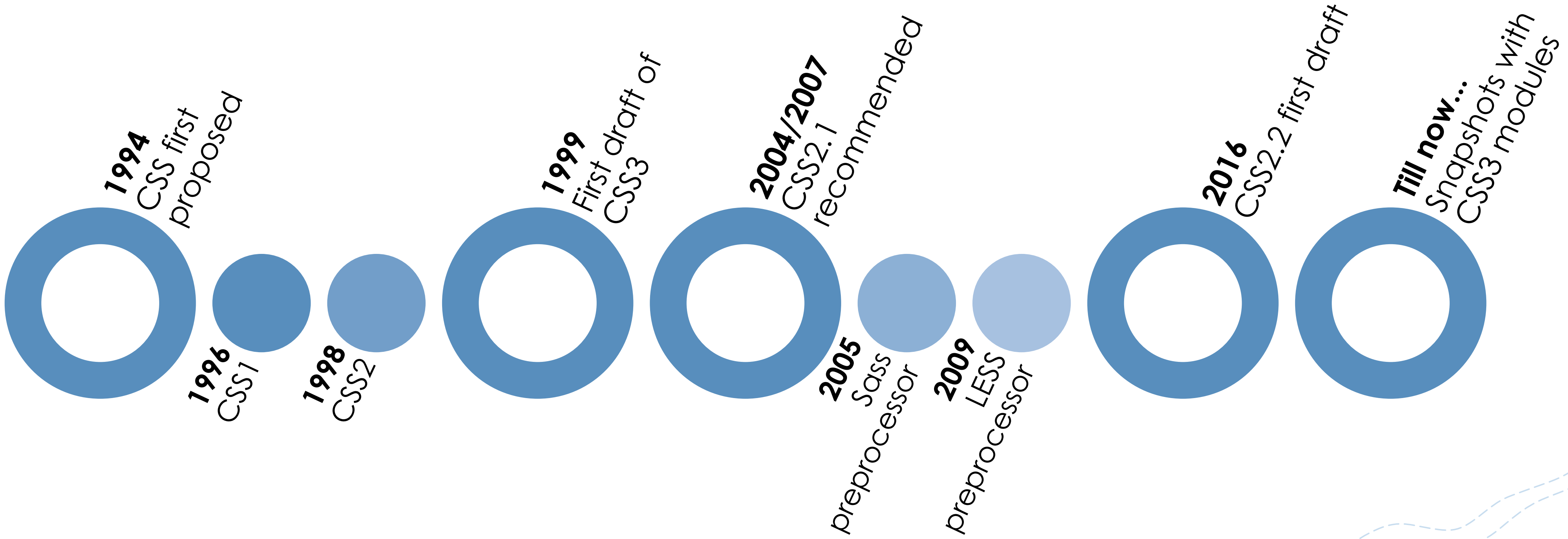
- CSS Basics
- Using CSS with HTML
- Inheritance and cascading
- Selectors and properties
- Inline vs. block-level elements
- Displaying and positioning
- The box model
- Responsive web design

CSS BASICS

- CSS – **C**ascading **S**tyle **S**heets
- Again, it is ***not a programming language***, but is for styling contents in HTML

```
t-weight: normal;

heading {
ding: 0px;

title a {
ding: 10px 15px;
th: 100%;
glish/images/bg_chevron_u

glish/images/bg_chevron_d

h {
e: nowrap;*/

t-child {

h {

h:first-child, table.cont
 {

ontact-table > thead > tr > th, table.contact-table >
, table.contact-table > tbody > tr > td {
der-top: none;

sc {
t-weight: normal;
```

A BRIEF HISTORY OF CSS


1994 CSS first proposed
1996 CSS1
1998 CSS2
1999 First draft of CSS3
2004/2007 CSS2.1 recommended
2005 Sass preprocessor
2009 LESS preprocessor
2016 CSS2.2 first draft
Till now… Snapshots with CSS3 modules

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* from *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

CSS SYNTAX

Selector

Declaration of property names with values

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

- Like HTML, CSS is generally ***not case-sensitive***
  - Except HTML attribute values, e.g., the value of `id="SomeName"`

USING CSS IN HTML

- If the task is to change the behaviour of `<p>` in an HTML file, there are multiple ways
  - ***External stylesheet***: where a stylesheet file (`.css`) is linked
  - ***Internal stylesheet***: 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 colors

EXTERNAL STYLE SHEET

- Include an external style sheet using **`<link>`** in **`<head>`**

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

```css
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>`**

```
<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

```
<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 style > internal stylesheets > external stylesheets
  2. More ***specific ones*** override generic ones
  3. Naturally, ***later ones*** override earlier ones
  - Properties marked `!important` overrides everything else

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

More on pseudo selectors: *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

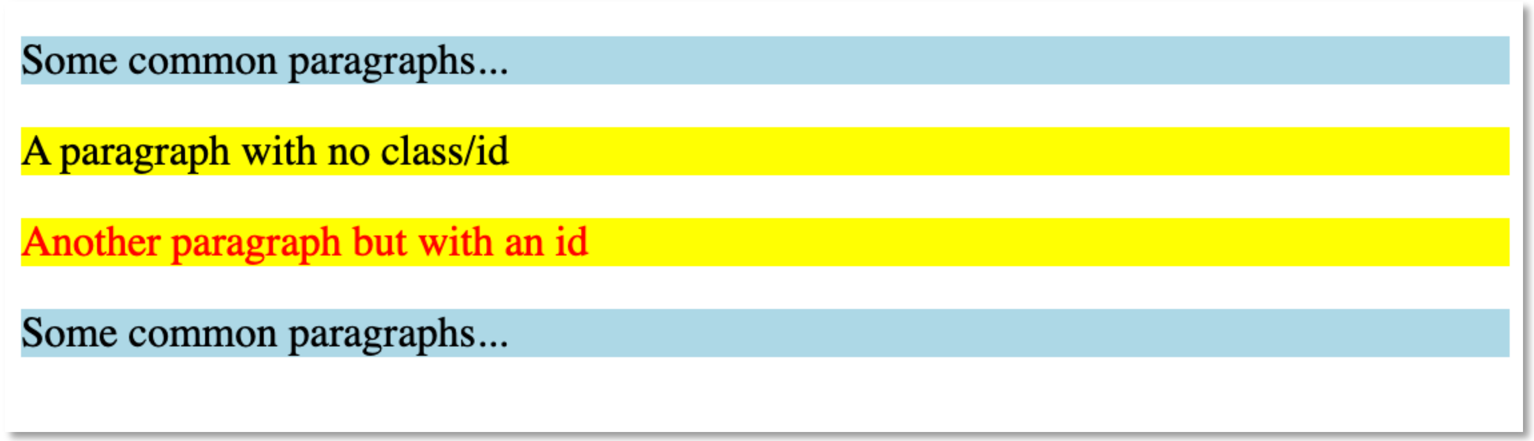
Pseudo-class selectors	Description
`a:hover`	Select all **`<a>`** elements that has the mouse cursor over it
`a:link`	Select all unvisited **`<a>`** elements

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>
```

https://codepen.io/chuckjee/pen/XWeyLwB

```
/* 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 common paragraphs...

A paragraph with no class/id

Another paragraph but with an id

Some common paragraphs...

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 the `@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*

COLORS

- A few different ways to represent colors in CSS
  - Color names, e.g., *white*, *black*, *green*, *lightgreen*, …
  - A combination of Red, Green, and Blue values
    - **`#rrggbb`**, where each of **`rr`**/**`gg`**/**`bb`** are hexadecimal values from 00 to ff, e.g., **`#000000`** is black, **`#ffff00`** is yellow
  - Other functions including **`rgb()`**, **`hsl()`**, …
- *Note: mind the spelling must be **color** but not colour*
- Try and pick colors: [*https://www.w3schools.com/colors/colors_picker.asp*](https://www.w3schools.com/colors/colors_picker.asp)

INLINE VS. BLOCK-LEVEL ELEMENTS IN HTML

- There are different kinds of HTML elements
  - **`p`**, **`h1`**, **`table`**, **`blockquote`**, **`li`**, … ➔ ***block-level*** elements
  - **`i`**, **`a`**, **`img`**, **`small`**, … ➔ ***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

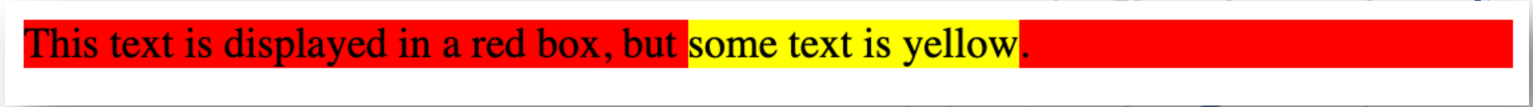
INLINE VS. BLOCK-LEVEL ELEMENTS

- Two special generic HTML elements usually used for applying CSS styles
  - `<div>` is ***block-level***, which is often used as a container for layouts
  - `<span>` 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 height and width
  - See: *http://learnlayout.com/inline-block.html*

https://codepen.io/chuckjee/pen/KKgJwZj

```
<div id="redbox">This text is
displayed in a red box, but <span
class="yellowtext">some text is
yellow</span>.</div>
```

```
#redbox { background: red; }
.yellowtext { background: yellow; }
```


This text is displayed in a red box, but some text is yellow.

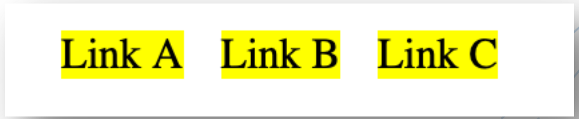
DISPLAYING

- 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***": allowing a barely useful website when enhancements (e.g., styles) are not available
    - The list is displayed with browser defaults without CSS

https://codepen.io/chuckjee/pen/YzGBPvY

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

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


Link A Link B Link C

DISPLAYING

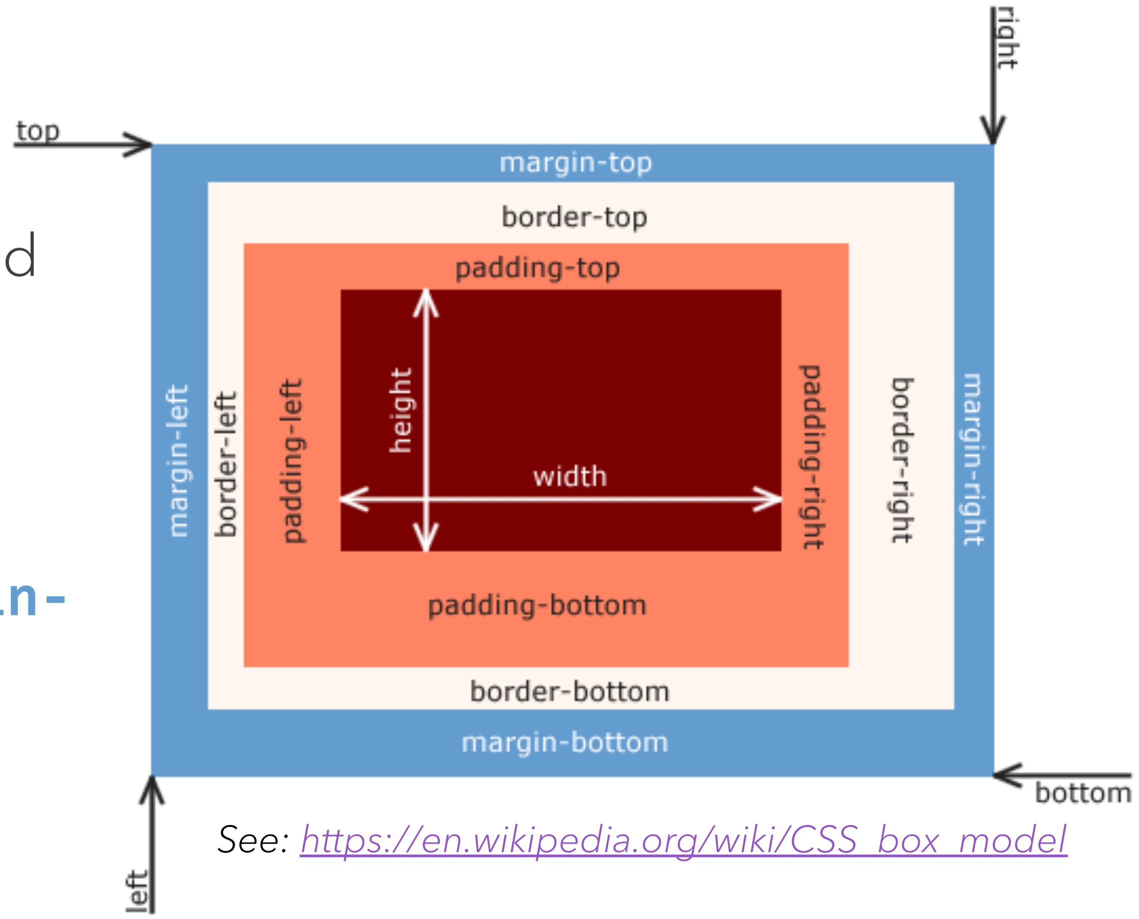
- There are many interesting option for displaying an element, besides inline, block, and inline-block
    - See: *https://www.w3schools.com/cssref/pr_class_display.asp*
- To NOT display an element, you may set…
    - `display: none;`
      → the element occupies no space at all
    - `visibility: hidden;`
      → 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
  - **`fixed`**: positioned relative to the browser window
  - **`relative`**: relative to original static 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.asp?filename=playcss_position*

THE BOX MODEL

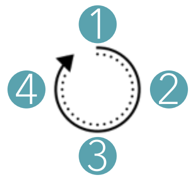
- All HTML elements are considered boxes, and they can occupy some space according to these properties
  - **`height`**/**`width`**: the content area
  - **`max-height`**, **`min-height`**, **`max-width`**, **`min-width`**: the limits when resizing window
  - **`padding`**: "*internal*" space, taking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



See: https://en.wikipedia.org/wiki/CSS_box_model

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

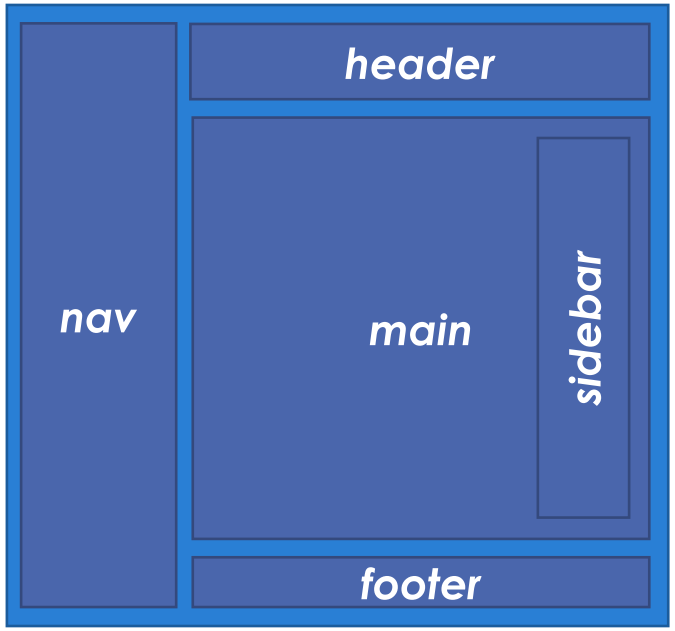
padding-left: 5em;
margin-top: 2em;
	Also possible to set values independently

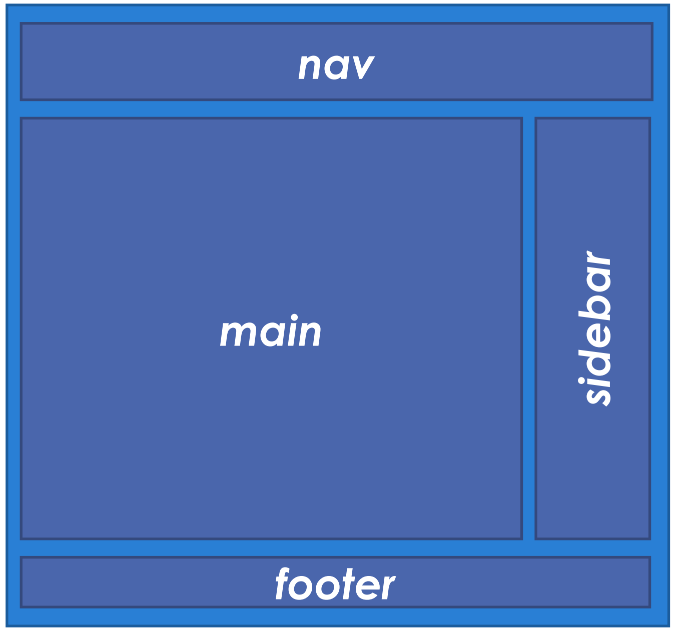
margin: 0 auto;
border: 2px dotted red;
	To align a box in middle, use auto x-margin

PREPARING LAYOUTS

- Laying out in CSS are just arranging boxes…
- It's a matter of ***imagination***!

…



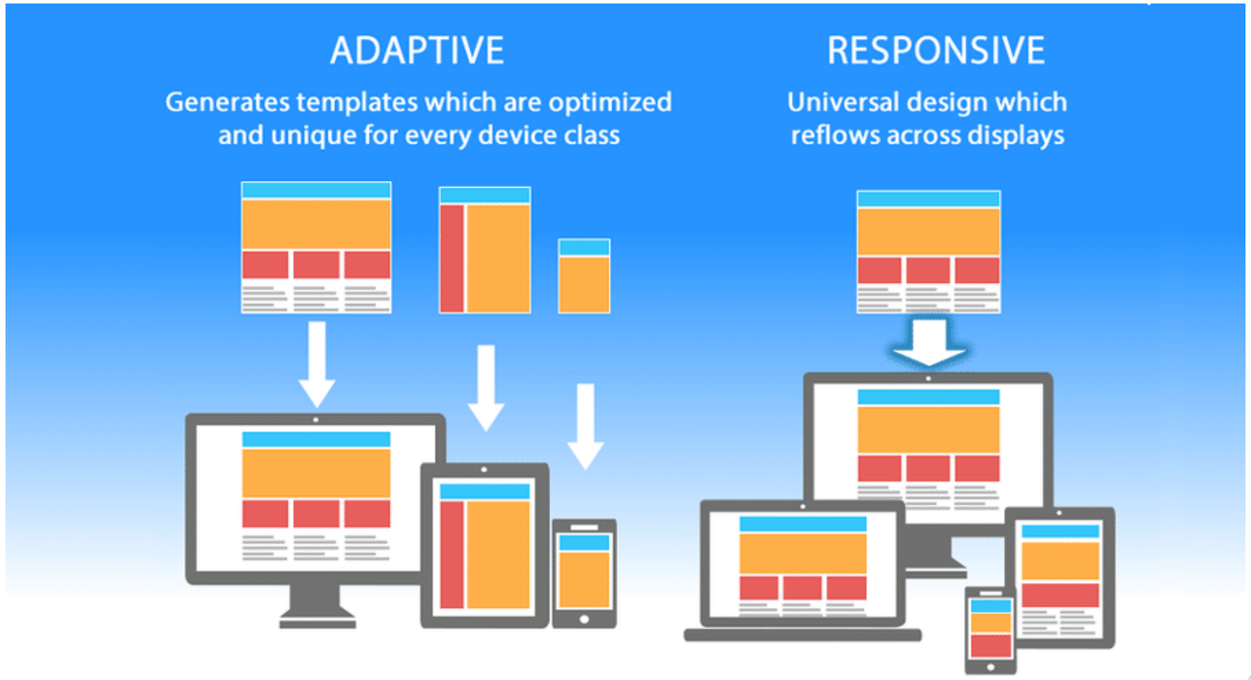


…

RESPONSIVE WEB DESIGN

See: https://kinsta.com/blog/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


ADAPTIVE
Generates templates which are optimized and unique for every device class
RESPONSIVE
Universal design which reflows across displays

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">
    ```

  - Avoid large fixed-width elements, or make assumption on viewport size
  - ***Mobile-first*** design, and scale/rearrange elements using CSS `@media` queries

RESPONSIVE HANDLING OF IMAGES

- Images can be scaled with parent size

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

- The picture element 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: e.g., 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 compiled 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*](https://codepen.io/miocene/pen/NWRWQpX)
- You don't need to learn everything.
  Know the syntax and learn reading the ***documentations***!

READ FURTHER…

w3schools.com CSS Tutorial

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

MDN Introduction to CSS

- https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS

State of CSS 2021

- https://2021.stateofcss.com