Trường Đại học Công nghệ Thông tin – Đại học Quốc gia Hồ Chí Minh
Bộ môn An toàn thông tin

CSS Basic – Advanced

Ths. Trần Tuấn Dũng – dungtran@uit.edu.vn

Styling HTML with CSS

- CSS stands for **C**ascading **S**tyle **S**heets.
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
- CSS saves a lot of work. It can control the layout of multiple web pages all at once.
- The most common way to add CSS, is to keep the styles in separate CSS files

CSS Advantages

Makes website more flexible

- CSS is reusable
- Change stylesheet to change design of many pages
- Example: CSS Zen garden http://www.csszengarden.com/

Easier to maintain

- Cleaner HTML code
- Separates styles from HTML tags and page content
- Consistent look across entire website that is easily maintained by changing styles in one place.

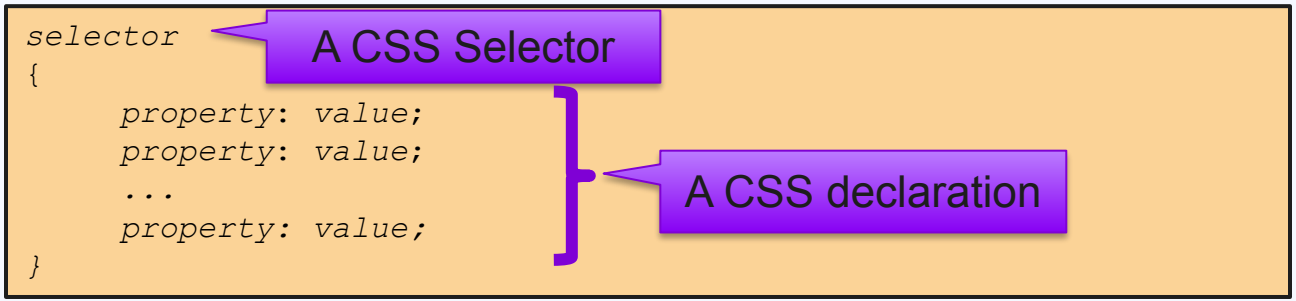
CSS Disadvantages

Not uniformly supported by all browsers.

- CSS works differently on different browsers. IE and Opera supports CSS as different logic.
- Chrome adheres to CSS standards more than IE

Adding style

- There are two aspects to adding style to a Web page via CSS
  - Specifying what the style looks like "Declaration"
  - Naming the HTML element "Selector"




```
p {
font-family: sans-serif;
color: red;
}
```

CSS comments

```css
/* This is a comment.
It can span many lines in the CSS file. */
p {
color: red; background-color: aqua;
}
```

CSS

- The // single-line comment style is NOT supported in CSS
- The <!-- ... --> HTML comment style is also NOT supported in CSS

CSS Selectors

Define which elements (or group of elements) will have the corresponding CSS applied

CSS selectors are used to "find" (or select) the HTML elements you want to style

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on tag, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

CSS Selectors - Examples

```
/*Select all p tag in the document*/
p {
  text-align: center;
  color: red;
}
/*Select all element has “class1” in classname*/
.class1 {
  text-align: center;
  color: red;
}
/*Select element has “id1” */
#id1 {
  text-align: center;
  color: red;
}
```

CSS Selectors - Examples

```
/*apply to all elements in the document*/
* {
  text-align: center;
  color: red;
}
/*Select all element has both “class1” and “class2” in classname*/
.class1.class2 {
  text-align: center;
  color: red;
}
/*Select all p elements which have direct parent is “id1”  */
#id1 > p {
  text-align: center;
  color: red;
}
```

Naming HTML elements

- There are two naming options for an HTML element: assigning "ID" names and "class names".
- An id declaration is the same as a class declaration, except that it should only be used specifically once per Web page

```
<h1 class="myboldandbluelook"> Introduction </h1>
```

```
.myboldandbluelook
{
        font-weight: bold;
        color: blue;
}
```

```
<h1 id="myboldandbluelook"> Introduction </h1>
```

```
#myboldandbluelook
{
        font-weight: bold;
        color: blue;
}
```

Using CSS

- Inline

```
<h1 style="color:blue;text-align:center;">This is a heading</h1>
```

- Internal

```
<head>
<style>
h1 {color: maroon;margin-left: 40px;}
</style>
</head>
```

- External

```
<link rel="stylesheet" href="mystyle.css"> (CSS file locate in project)
<link href="http://www.google.com/uds/css/gsearch.css" rel="stylesheet"
type="text/css" />
```

- A page can link to multiple style sheet files
  - In case of a conflict (two sheets define a style for the same HTML element), the latter sheet's properties will be used

Colors

```
p {
color: red;
background-color: yellow;
}
```

This paragraph uses the style above *output*

property	description
color	color of the element's text
background-color	color that will appear behind the element

Specifying colors

```
p { color: red; }
h2 { color: rgb(128, 0, 196); }
h4 { color: #FF8800; }
```

This paragraph uses the first style above

This h2 uses the second style above.

This h4 uses the third style above.

- color names: aqua, black, blue, …
- RGB codes: red, green, and blue values from 0 (none) to 255 (full)
- hex codes: RGB values in base-16 from 00 (0, none) to FF (255, full)

Fonts

property	description
font-family	which font will be used
font-size	how large the letters will be drawn
font-style	used to enable/disable italic style
font-weight	used to enable/disable bold style

Complete list of font properties (https://www.w3schools.com/css/css_font.asp)

font-family

```
p {
font-family: Georgia;
}
h2 {
font-family: "Courier New";
}
```

CSS

This paragraph uses the first style above.

This h2 uses the second style above.

output

- Enclose multi-word font names in quotes

font-size

```
p {
        font-size: 24pt;
}
```

This paragraph uses the style above.

font-weight, font-style

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

This paragraph uses the style above.

- Either of the above can be set to normal to turn them off (e.g. headings)

Text

property	description
text-align	alignment of text within its element
text-decoration	decorations such as underlining

Complete list of text properties (https://www.w3schools.com/css/css_text.asp)

CSS Layout

Common ways to style the layout of website:

- Position
- Float
- Flexbox
- Grid

Position

The position property specifies the type of positioning method used for an element

There are five different position values:

- static: position by normal flow of web page
- relative: relatively to normal position
- fixed: fixed position in viewport
- absolute: relatively to position of nearest ancestor
- sticky: stay at relative and become fixed when specific scroll position is met

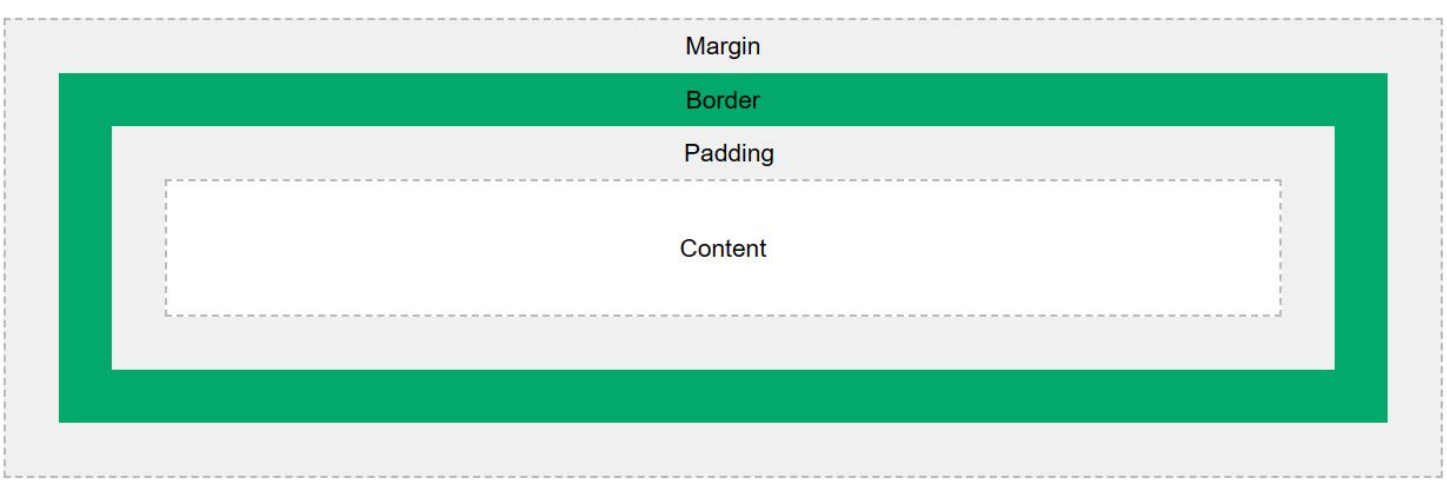
CSS Box Model

An element is wrap inside a “box” shown as follow. Consist of: margin, border, padding, content. Like a pool, the wall is border, water is padding, and people is content

To calculate size of an element:

Width = Content width + padding left + padding right + border left + border right

Height = Content height + padding top + padding bottom + border top + border bottom



Padding

Padding is used to create space around an element's content, inside of any defined borders

Like we “pad” the content with an extra layer


```
padding-top: 50px;
padding-right: 50px;
padding-bottom: 50px;
padding-left: 50px;
```

```
padding-top: 50px;
padding-right: 50px;
padding-bottom: 50px;
padding-left: 50px;
```

Padding

Normal

```
padding-top: 50px;
padding-right: 50px;
padding-bottom: 50px;
padding-left: 50px;
```

or

Shorthand

```
padding: 50px
```

```
padding-top: 50px;
padding-right: 80px;
padding-bottom: 40px;
padding-left: 20px;
```

or

```
padding: 50px 80px 40px 20px
```

Border

Border is like a box, hold everything inside

Border

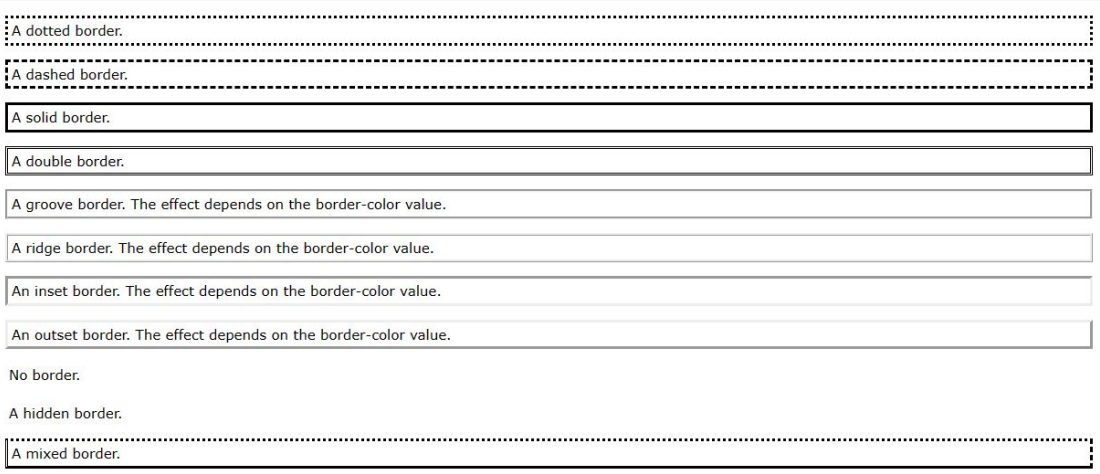
Same value syntax with padding

Normal		Shorthand
border-top: 50px; border-right: 50px; border-bottom: 50px; border-left: 50px;	or	border: 50px
border-top: 50px; border-right: 80px; border-bottom: 40px; border-left: 20px;	or	border: 50px 80px 40px 20px

Border

Border with style

```
border-style: dotted
border-style: dashed
border-style: solid
border-style: double
border-style: groove
border-style: ridge
border-style: inset
border-style: outset
border-style: none
border-style: hidden
border-style: dotted dashed solid double
```



Border

To create modern rounded border


```
border-radius: 12px;
```

Margin

Margins are spaces between elements

Margin values have same syntax with padding and border

Float

- The float property is used for positioning and layout on web pages.
- The float property can have one of the following values:
  - left - The element floats to the left of its container
  - right- The element floats to the right of its container
  - none - The element does not float (will be displayed just where it occurs in the text). This is default
  - inherit - The element inherits the float value of its parent
- In its simplest use, the float property can be used to wrap text around images.

Float

Example - float: right;

The following example specifies that an image should float to the right in a text:

```
img {
    float: right;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac...

Float: Clear

The clear property specifies what elements can float beside the cleared element and on which side.

The clear property can have one of the following values:

- none - Allows floating elements on both sides. This is default
- left - No floating elements allowed on the left side
- right- No floating elements allowed on the right side
- both - No floating elements allowed on either the left or the right side
- inherit - The element inherits the clear value of its parent

Without Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

With Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

Clearfix

If a floated element is taller than the containing element, it will "overflow" outside of its container. We can then add a clearfix hack to solve this problem

Without Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

With Clearfix

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

```
.clearfix {
  overflow: auto;
}
```

or

```
.clearfix {
  clear: right; /* left/both/none/inherit */
  display: inline-block;
}
```

Flexbox

Flexbox makes it easy to create responsive layout

To use flexbox, we need a flex container and flex items

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

Then apply the flexbox, flex items will be placed horizontally

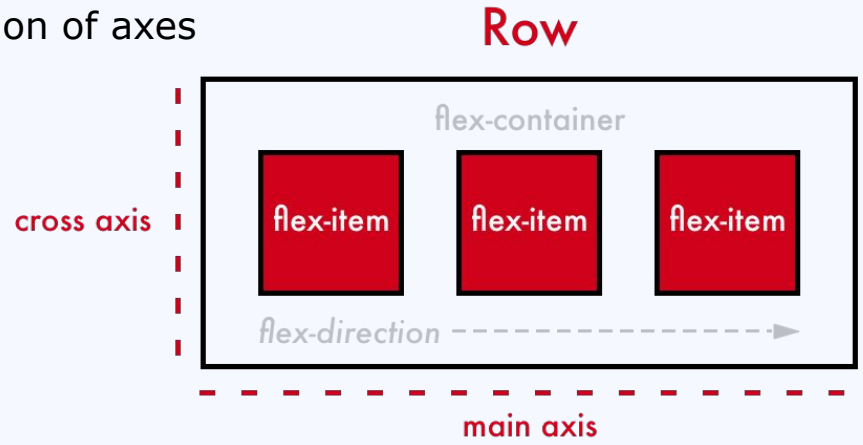
```
.flex-container {
  display: flex;
}
```

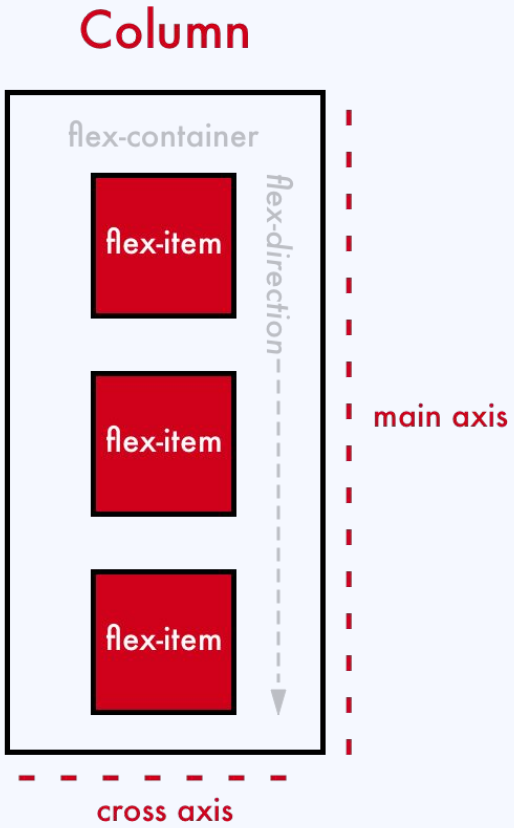
Flexbox

A flex container will have 2 axes:

- Cross axis
- Main axis

Changing `flex-direction` will change the direction of axes





Flexbox

Align items of flex box:

- `justify-content:` align along main-axis
- `align-items:` align along cross-axis

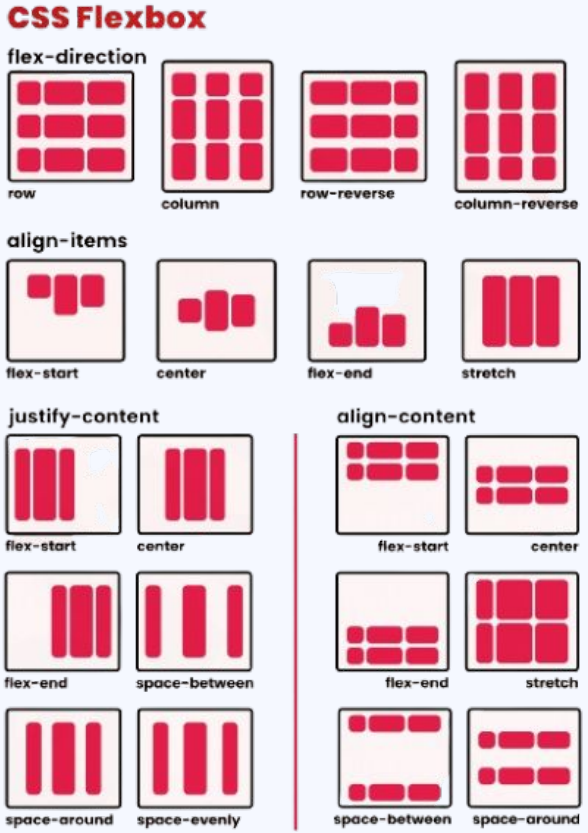
When using `flex-direction: row`

- `justify-content:` align horizontally
- `align-items:` align vertically

When using `flex-direction: column`

- `justify-content:` align vertically
- `align-items:` align horizontally

Because the direction of axes has changed



Grid

The **CSS Grid Layout Module** offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use **floats** and **positioning**.

Using grid make easier to create complex layout comparing to HTML Table

Grid

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

```
.grid-container {
  display: grid;
}
```

1	2	3
4	5	6
7	8	9

Grid Layout Control

`grid-template-columns` defines the number of columns in your grid layout

When set to auto, all column will have same width

`grid-template-rows` defines height of each row

```
/* Create 4 column */
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto auto;
}
```

```
.grid-container {
  display: grid;
  grid-template-columns: 80px 200px auto 40px;
}
```

Grid Layout Control

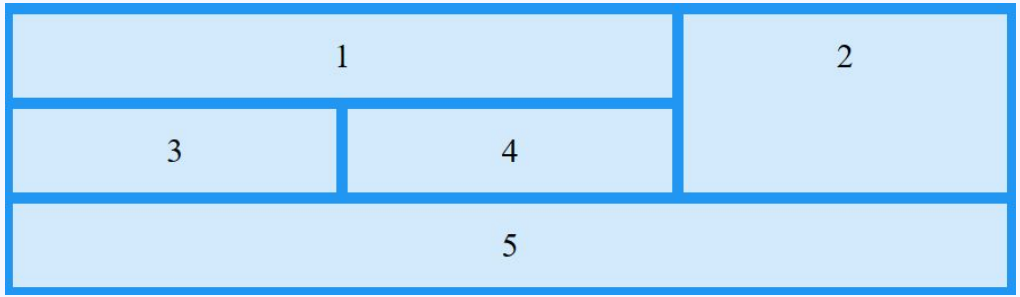
grid-row and grid-column control the placement of grid items

Allow we to control the "span" of each items, and order of each item

```
.item1 {
 grid-column: 1 / span 2;
 grid-row: 1;
}

.item2 {
 grid-column: 3;
 grid-row: 1 / span 2;
}

.item5 {
 grid-column: 1 / span 3;
 grid-row: 3;
}
```



Grid Layout Control

We can pre-define the layout by naming the area

```
.grid-container {
  display: grid;
  grid-template-areas:
    'header header header header header header'
    'menu main main main right right'
    'menu footer footer footer footer footer';
}
```

```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }
```

Reset CSS

All element have their default CSS

Causing inconvenience when editing the style of the document

Reset CSS is to remove or adjust some default CSS of the browser

Value of new default CSS can be will depend of each project

Most common Reset CSS is

```
* {
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}
```

Or using common CSS Normalizer from CDNjs

normalize - Libraries - cdnjs - The #1 free and open source CDN built to make life easier for developers

CSS Specificity

If there are two or more CSS rules that point to the same element, the selector with the **highest specificity** value will be chosen

Specificity can be understand as "priority", CSS rule with higher priority will be chosen over the lower priority

Rules of Specificity:

- *: 0
- Tag name (or pseudo-element): 1
- Class name (or pseudo-class or attribute selector): 10
- ID: 100
- Inline: 1000

CSS Specificity – Calculation

Selector	Specificity Value	Calculation
p	1	1
p.test	11	1 + 10
p#demo	101	1 + 100
p.test1.test2	21	1 + 10 + 10

```
.intro {background-color: yellow;}
h1 {background-color: red;}

<h1 class="intro">This is a heading</h1>
```

CSS !important

The !important rule in CSS is used to add more importance to a property/value than normal.

In fact, if you use the !important rule, it will override **ALL** previous styling rules for that specific property on that element!

```
p {
  background-color: red !important;
}
```

Even we have higher specificity selectors apply to p tags, above CSS will always be used

If we have multiples !important, CSS with normal higher specificity will be applied

CSS pseudo-classes & pseudo-elements

A **pseudo-class** is used to define a special state of an element

A **pseudo-element** is used to style specified parts of an element

Pseudo-class use one ":"	Pseudo-element use two "::"
.element:pseudo-class	.element::pseudo-element

Examples

Style the hover state of div

```
.mouseover:hover {
  background-color: #008CBA;
}
```

Style the first line of p

```
p::first-line {
 color: #ff0000;
 font-variant: small-caps;
}
```

CSS Units

CSS has several different units for expressing a length

From w3school, we will have a lot of available units

Using correct unit will make responsive easier

CSS Units – Common units

CSS units divide into 2 categories:

- Absolute units
- Relative units

While we have lots of units, some common are:

Absolute units: display the corresponding value correctly on every screen

- px: pixels
- pt: points

Relative units: Scale relatively with other property

- em: Relative with element's font size
- rem (root em): Relative with root element font size
- vw (view-width): 1% of view screen's width (the browser window size)
- vw (view-height): 1% of view screen's height (the browser window size)
- %: Relative to parent element

CSS Responsive

The **CSS Grid Layout Module** offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use **floats** and **positioning**.

Using grid make easier to create complex layout comparing

Exercies

1. Calculate the specificity of each selector:

- `div.test1.test2 p#id1 {…}`
- `h1#sitetitle > .logo {…}`

2. What css property can be used to create a circle:

2. Use **CSS Flexbox** to create this navbar (style don't have to be exactly accurate):

Homework

Create this CV using both HTML and CSS

LỚP LẬP TRÌNH ỨNG DỤNG WEB
Khoa Mạng máy tính & Truyền thông

CSS v1 CSS v2 CSS v3 CSS v0 (Not style)

Nguyễn Văn A
Lập trình viên Frontend

Ngày sinh 23/10/1991
Giới tính Nam
Địa chỉ TP Thuận An, Bình Dương

LIÊN HỆ VỚI TÔI

- **ĐT:** 0984752222
- **Email:** avn@uit.edu.vn
- **Fb:** An Nguyễn

Lorem ipsum dolor sit amet, consectetuer adipiscing elit

MỤC TIÊU NGHỀ NGHIỆP

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

HỌC VẤN

Đại học Công nghệ thông tin - ĐHQG TP HCM

Truyền thông và Mạng máy tính

GPA: 9.0

Tốt nghiệp: Giỏi

THPT Chuyên Hùng Vương - Bình Dương

Ban Tự Nhiên

Tốt nghiệp: Giỏi

CÔNG VIỆC

Vị trí	Đơn vị	Thời gian
Lập trình viên	FPT Quận 9	Tháng 3/2019 - 5/2021
Leader	Công ty ABC	Tháng 6/2021 - 10/2025
Giám đốc	Công ty BX	Tháng 9/2025 - nay
Ernst Handel	Roland Mendel	Austria