



# Introduction to CSS

# Agenda

---

# Agenda

---

- 1 Background
- 2 Adding CSS to an HTML Page
- 3 CSS Selectors
- 4 Styling with CSS
- 5 Tools and Resources
- 6 Quiz

# Agenda

---

- 1 Background
- 2 Adding CSS to an HTML Page
- 3 CSS Selectors
- 4 Styling with CSS
- 5 Tools and Resources
- 6 Quiz

# Background

---

# CSS

Cascading Style Sheet

- CSS was released in 1996 by World Wide Web Consortium (W3C)
- CSS is a language that defines the style of an HTML document. It details how HTML elements should be displayed
- CSS provides styling features which are not available in HTML
- CSS is used to enhance the visual appearance on web pages by supporting styling using colors, animations, etc
- CSS also allows developers to create websites that render well on all kinds of devices
- The latest version of CSS is CSS 3.0

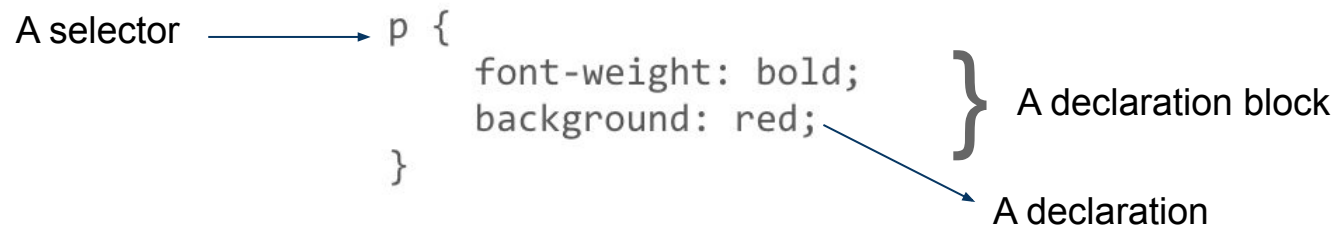
# CSS Syntax

- A basic CSS rule-set consists of two elements: a **selector** and a **declaration block**
- The selector points to the HTML element intended to be styled
- The declaration describes a rule for styling a particular element or a group of HTML elements. Declarations are terminated by semicolons (;) and one or more declarations make up the declaration block
- Each declaration includes a CSS property name and a value, separated by a colon (:)

A selector → `p {`  
`font-weight: bold;`  
`background: red;`  
`}`

→ A declaration block

→ A declaration

A diagram illustrating the components of a CSS rule. It shows a CSS rule: `p { font-weight: bold; background: red; }`. An arrow points from the text 'A selector' to the `p {` part. Another arrow points from the text 'A declaration block' to the closing curly brace `}`. A third arrow points from the text 'A declaration' to one of the declarations inside the block, `background: red;`.

- The CSS selectors, declaration block, semicolons, and curly braces must be written and properly used in order for the CSS to work properly.

# Agenda

---

- 1 Background
- 2 Adding CSS to an HTML Page
- 3 CSS Selectors
- 4 Styling with CSS
- 5 Tools and Resources
- 6 Quiz



# Adding CSS to an HTML Page

---

# Adding CSS to an HTML Page

- **Inline Styling:** achieved using the style attribute. Inline styling is used to apply a unique style for a single element. e.g

```
<h1 style="font-size: 36px; color:orange;">This is an inline styling</h1>
```

- **Internal Styling:** achieved using the <style> tags in the <head> section of the HTML document. Internal CSS styles can only be applied to a single page since the CSS is resident in the same file as the HTML. e.g

```
<!DOCTYPE html>
<html>
  <head>
    <title> Page Title </title>
    <style>
      a {
        color: green;
      }
    </style>
  </head>
  <body>
  </body>
</html>
```

# Adding CSS to an HTML Page

- **External Styling:** achieved using the `<link>` element goes inside the `<head>` section. It is possible to change the look and feel of an entire website by changing just the CSS file. The style sheet file must be saved with a `.css` extension e.g `style.css`, `homepage.css`. Each page must include a reference to the external style sheet file inside the `<link>` element as shown below:

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

- In the example above, the `href` attribute contains the path to `style.css`. The style will not be loaded if the path is not correctly specified

# Cascading Order

- If an element is assigned to multiple ( and possibly conflicting) styles, the style that is finally applied to that element is defined by the following cascading rule, in order of priority:
  1. Inline style
  2. Internal and External styles
  3. Browser default
- The inline CSS style overrides any similar style definition in the Internal or External style sheet for that element.

# Agenda

---

- 1 Background
- 2 Adding CSS to an HTML Page
- 3 **CSS Selectors**
- 4 Styling with CSS
- 5 Tools and Resources
- 6 Quiz

# CSS Selectors

---

# CSS Selectors

- CSS selectors are used to target HTML elements using their element name, id, class, attribute, and grouping
- The selector is a major element in a CSS rule-set definition

# The Element Selector

- The element selector targets HTML elements based on the element name. They are used to customize properties which are unique to a particular HTML tag on the web page.

```
p {  
    font-weight: bold;  
    background: red;  
}
```

- The code block above selects all `<p>` elements on a page and sets their background red with a bold font
- This rule will apply to every `<p>` element on any page this style is added



# The Id Selector

- The id selector uses the id attribute of an HTML element to select a specific element. Ideally, the id of an element should be unique on a web page, so the id selector targets **one** unique element. It uses the id attribute of an HTML element to select the target element
- The id selector takes priority over other selectors, If two kinds of CSS selectors target the same element
- The id selector is declared by a hash (#) character, followed by the id of the element. For example, the style below targets an HTML element with attribute `id="currency"`

```
#currency {  
    font-weight: bold;  
    color: dark-gray;  
}
```

# The Class Selector

- The class selector selects elements with the specified class attribute. Multiple HTML elements can have the same class attribute value.
- A period (.) character, followed by the name of the class is used to target elements with the specified class as their class attribute. For example, the style below targets all elements with attribute `class='title'`

```
.title {  
    color: purple;  
    font-weight: 38px;  
}
```

# Grouping Selectors

- For cleaner and maintainable code, it is common practice to group the selectors. When grouping selectors, the selectors are separated with a comma before the declaration block.
- Selectors are grouped when multiple HTML elements have the same style definition. For example, the style below targets all header tags

```
h1,  
h2,  
h3,  
h4,  
h5,  
h6{  
    color: dark-gray;  
    text-transform: uppercase;  
    font-family: 'Helvetica';  
}
```

# Agenda

---

- 1 Background
- 2 Adding CSS to an HTML Page
- 3 CSS Selectors
- 4 Styling with CSS
- 5 Tools and Resources
- 6 Quiz

# Now, let's dive in

---

# Styling with CSS

- We have looked at basic CSS syntax, how to add styles to HTML documents and the different CSS selectors. In this section, we will discuss how to style web pages with CSS

# CSS Colors

---

# CSS Colors

- Colors are used to enhance the look and feel of a website. Colors in CSS can be specified by using any of these 3 ways:
  1. **Color name** e.g yellow, blue, purple, red
  2. **RGB (Red Green Blue) value** e.g `rgb(0, 0, 250)`
  3. **Hexadecimal value** e.g `#FFFFFF`

```
p{  
  color: black;  
}
```

```
p{  
  color: #000000;  
}
```

```
p{  
  color: rgb(0, 0, 0);  
}
```

- The styles above all apply black font color to the `<p>` element

Check [here](#) for modern browser supported colors with their RGB values and Hex color codes



# The RGB Model

- The RGB model is used to assign color values by using their red, green, and blue composition using the formula **rgb(red, green, blue)**.
- Each parameter (red, green, and blue) defines the intensity of the color. Each parameter value ranges from 0 to 255. You can assign different values to each parameter to generate a color or choice
- An enhancement to the RGB model is the RGBA. The additional parameter 'A' stands for Alpha. Alpha defines the opacity of the color specified using the RGB values
- The parameter alpha can range from 0 to 1. Full transparency is achieved by setting alpha to 0 while full opacity is achieved by setting alpha to 1

**Task:** Experiment with the RGB and RGBA models

# The Hexadecimal Specification

- Just like the RGB model, the Hex format the red, green, and blue intensity to define colors in **Hexadecimal format** `#rrggbb`. In practice, rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff.
- For example, `#ff0000` is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).
- Hexadecimal values are not case-sensitive. `#fa0001` is the same as `#FA0001`.

**Task:** Experiment with the Hexadecimal color specification

# The HSL Specification

- HSL stands Hue, Saturation and Lightness. Color can be specified using HSL in the format **hsl(hue, saturation, lightness)**
- Hue is a degree on the [color wheel](#) from 0 to 360. Pure Red is represented by 0 or 360, 120 represents green, 240 represents blue.
- Saturation is a percentage value that defines the intensity of the color, 100% is the full color
- Lightness is also described with a percentage parameter. 0% is black, 50% is in between and 100% is white e.g



```
.title {  
  background-color: hsl(120, 100%, 50%);  
}
```

**Task:** Experiment with the HSL color specification

# Styling Backgrounds in CSS

---

# Styling Backgrounds

Background styling can be applied to almost all HTML elements. Below are CSS background properties are used for background styling of elements:

- **background-color**: specifies the background color of an element
- **background-image**: specifies an image to be used as the background of an element
- **background-repeat**: the image is repeated by default. This property controls the repeat behavior of the background image
- **background-position**: specifies the position of the background image
- **background-attachment**: specifies the behaviour of the background image when the page is scrolled.
- **background**: specifies all the background properties in one single property. This is called the background shorthand property.

# Styling Backgrounds in Action

```
body {  
  background-color: #ff0000;  
  background-image: url("backdrop.png");  
  background-repeat: repeat-x;  
  background-position: left top;  
  background-attachment: fixed;  
}
```

```
body {  
  background: #ff0000 url("backdrop.png") repeat-x right top;  
}
```

# CSS Borders

---

# CSS Borders

CSS border can be applied to almost any HTML element. It defines a border around the specified HTML element. Below are CSS border styling properties:

- **border-style:** specifies what kind of border to display. It can accept any of the values dotted, dashed, solid, double, groove, ridge, inset, outset, none, hidden.
- **border-width:** specifies the width of the four borders. The width can be specified using one of the three predefined values (thin, medium, or thick) or using units (px, pt, cm, em, etc) and accepts one to four values in the order top, right, bottom, left
- **border-color:** used to set the color of the four borders. This property accepts any of the color specification modes discussed in the previous section and accepts one to four values in the order top, right, bottom, left
- **border-radius:** used to add rounded borders to an element
- **border:** used as a shorthand property for properties: border-width, border-style, border-color



# CSS Borders in Action

```
.item {  
    border-color: green;  
    border-right-color: white;  
    border-radius: 1px 2px 3px 4px;  
    border-style: dotted  
    border-width: 2px  
}
```

```
.item {  
    border: 3px solid #ff0000;  
}
```

# CSS Margins

---

# CSS Margins

CSS margins are used to create space around elements, outside of any defined borders. The properties for setting the CSS margins are:

- **margin-top, margin-right, margin-bottom, margin-left:** specifies the margin for each side of an element. The accepted values are auto, length (in units px, pt, cm, etc), % and inherit

```
input {  
    margin-top: 10px;  
    margin-bottom: 10px;  
    margin-right: 10px;  
    margin-left: 10px;  
}
```

- **margin:** used as a shorthand property for the individual margin properties above. It accepts one to four values in the order top, right, bottom, left

```
input {  
    margin: 10px 20px 10px 20px;  
}
```

# CSS Paddings

---

# CSS Paddings

CSS paddings are used to generate space around an element's content, inside of any defined borders.

- **padding-top, padding-right, padding-bottom, padding-left:** specifies the padding for each side of an element. The accepted values are length (in units px, pt, cm, etc), % and inherit

```
input {  
    padding-top: 10px;  
    padding-bottom: 10px;  
    padding-right: 10px;  
    padding-left: 10px;  
}
```

- **padding:** used as a shorthand property for the individual padding properties above. It accepts one to four values in the order top, right, bottom, left

```
input {  
    padding: 10px 20px 10px 20px;  
}
```

# CSS Box Model

---

# CSS Box Model

- All HTML elements can be considered as boxes. The CSS term "box model" is used when talking about design and layout and is very important in correctly sizing page elements
- The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.



Source: <https://www.bitdegree.org/learn/css-box-model/>

# CSS Box Model

- The box model allows us to add a border around elements, and to define space between elements. Total dimension of an element is calculated by accounting for width/height, paddings, border, and margins

- Total element width should be calculated as:

```
width = content width + left padding + right padding + left border + right border + left margin + right margin
```

- Total element height should be calculated as:

```
height = content height + top padding + bottom padding + top border + bottom border + top margin + bottom margin
```

- That is why some elements with the same height and width values but with different margin, padding and border specification appear in different sizes on the screen.
- The CSS `box-sizing` property allows developers specify the dimension of HTML elements without having to worry about the issues raised above.



# CSS Box-sizing

- CSS box-sizing property was introduced in CSS3. It includes the padding and the border as part of the element size. This makes it easier to define dimensions without having to manually calculate everything
- To use the box-sizing property, assign the border-box value to the property

```
box-sizing: border-box;
```

- An easy way to maintain this style across all elements is as shown below:

```
* {  
  box-sizing: border-box;  
}
```

# CSS Pseudo Classes

---

# CSS Pseudo Classes

- CSS pseudo classes are used to target elements a certain state. For example, what happens to an element when clicked or when the mouse pointer is moved over it. A common example is how the color of textual link change color after they have been clicked. The link is in **a:link** state before the click and in **a:visited** state after being clicked.
- Pseudo classes have to be attached to a CSS selector to work. To define a pseudo class style, a selector is defined first, followed by a semicolon (:) and a pseudo class before the curly braces

```
a:hover {  
    color: pink;  
}
```

- Note that they are not separated by spaces.
- Check [here](#) for all pseudo classes and their usage

# CSS Pseudo Elements

---

# CSS Pseudo Elements

- CSS pseudo elements are used to style specified parts of an element. For example, they can be used to style the first letter or first line of an element
- They are not to be confused with pseudo classes discussed in the previous slide as pseudo classes are used to target elements in a certain state.
- A Pseudo element must also be attached to a CSS selector to work. But it uses a double colon (::) instead of the single colon used for separating a pseudo class and its selector

```
div::first-letter {  
    color: violet;  
    font-weight: bold;  
    font-size: 40px;  
}
```

- Available pseudo elements are - **::after**, **::before**, **::first-letter**, **::first-line**, **::selection**
- Check [here](#) for all pseudo elements and their usage

# CSS Positioning

---

# CSS Positioning

- CSS **position** property specifies the type of positioning used for an element. The different position values are: **static**, **relative**, **fixed**, **absolute** and **sticky**

```
.picture {  
    position: absolute;  
    top: 0;  
    bottom: 0;  
    right: 0;  
    left: 0;  
}
```

- CSS properties **top**, **bottom**, **left** and **right** are used with the **position** property. These properties work differently depending on the position value
- **static**: elements are positioned static by default. It does not position elements in any special way. Elements are positioned according to the normal flow of the page
- **relative**: elements are positioned according to the normal flow of the page relative to its normal position. Setting the top, right, bottom, and left properties with relative positioning will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element

# CSS Positioning

- **fixed**: elements are positioned in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element. A fixed element does not leave a gap in the page where it would normally have been located
- **absolute**: elements are positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like `fixed`). A positioned ancestor is one whose position is anything except `static`. If an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling
- **sticky**: elements are positioned based on the user's scroll position. A sticky positioned element toggles between `relative` and `fixed`, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport, then it sticks in place like `fixed`

**Task:** Experiment with the different position values to be more comfortable with them



# CSS Units

---

# CSS Units

- You have probably noticed the use of different units like px, pt, cm, %, em, etc in the code examples provided so far
- CSS accepts different units for declaring length taking values such as width, margin, padding, font-size, border-width, etc
- Length is a number followed by a length unit such as 3px, 20em, etc. There should be no spaces between the number and the unit. A unit is optional for value 0. Negative lengths are allowed for some CSS properties
- There are two types of length units: absolute and relative.

# Absolute length

- Absolute length units are fixed and a length expressed in any of these will appear as exactly that size
- Absolute length units are sometimes not proportionately displayed on varying screen sizes. Something considered too big on a small screen can be just fine on a large screen
- The absolute length units are:
  - **cm** - centimeters
  - **mm** - millimeters
  - **in** - inches (1in = 96px = 2.54cm)
  - **px** - pixels (1px = 1/96th of 1in)
  - **pt** - points (1pt = 1/72 of 1in)
  - **pc** - picas (1pc = 12 pt)
- Pixels (px) are relative to the viewing device. For low resolution devices, 1px is one device pixel (dot) of the display. For high resolution screens ,1px implies multiple device pixels

# Relative length

- Relative length units specify a length relative to another length property. Relative length units scale better between different rendering devices
- The absolute length units are:
  - **em** - relative to the font-size of the element
  - **ex** - relative to the x-height of the current font (rarely used)
  - **ch** - relative to width of the "0" (zero)
  - **rem** - relative to font-size of the root element
  - **vw** - relative to 1% of the width of the viewport\*
  - **vh** - relative to 1% of the height of the viewport\*
  - **vmin** - relative to 1% of viewport smaller dimension
  - **vmax** - relative to 1% of viewport larger dimension
  - **%** - relative to the parent element

**Task:** Experiment with the Absolute and Relative length units to better understand how they work

# Media Queries

---

# Media Queries

- The @media rule was introduced in CSS2 to define different style rules for different media types such as computer screens, printers, etc.
- CSS3 media queries makes it possible to define how a website appears depending on the desktop viewport height and width dimensions, the size dimensions of the device itself, resolution, and screen orientation (portrait/landscape). Media queries are used to make web applications responsive and display nicely on a wider range of devices.



Source: <https://clix2brix.com/web-design-scottsdale-why-local-businesses-need-responsive-website/>

# Media Queries

- A media query consists of a media type and can contain one or more expressions, which either evaluates to true or false

```
@media screen and (min-width: 480px) {  
  body {  
    background-color: white;  
  }  
}
```

- The styles defined in the media query are applied if the specified media type matches the type of device the document is being displayed on and all expressions in the media query are true. Supported CSS3 media types are:
  - all - used for all media type devices
  - print - used for printers
  - screen - used for computer screens, tablets, smart-phones etc
  - speech - used for screen readers that "reads" the page out loud

```
<link rel="stylesheet" media="mediatype and|not|only (expressions)" ref="print.css">
```

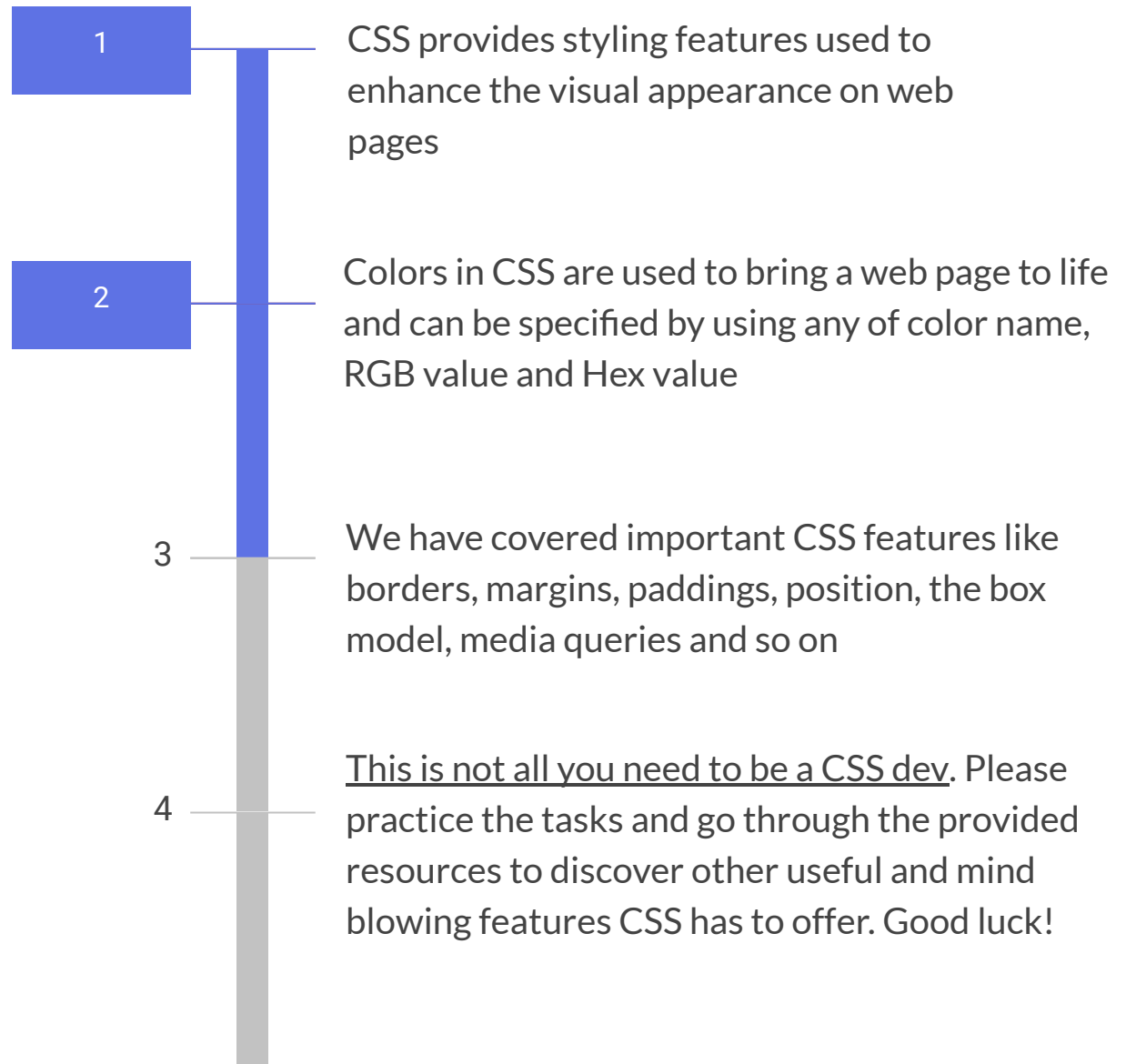
**Task:** Experiment with media queries for different screen sizes to understand how they work

# Summary

---



# Summary



# Tools & Resources

- [https://www.w3schools.com/css/css\\_intro.asp](https://www.w3schools.com/css/css_intro.asp)
- <https://www.bitdegree.org/learn/css-tutorial/>
- <https://drive.google.com/file/d/1RiTs-ifE8daoKoyq3tAt5li46a3YGjV5/view>
- <https://www.bitdegree.org/learn/css-basics/>
- <https://htmlcolorcodes.com/color-names/>

## References

- <https://www.w3schools.com/css/default.asp>
- <https://www.bitdegree.org/learn/css-tutorial/>
- <https://www.bitdegree.org/learn/css-basics/>

# Task 1

(Due on Friday, November 1 2019)

- Style the registration page designed for your ecommerce website
- Push your attempt to your Github repository
- Submit the link to your mentors

# Task 2

(Due on Sunday, November 10 2019)

- Create the first version of your ecommerce homepage.
- Push your attempt to your Github and Dufuna CodeCamp repository
- Submit the link to your mentors

# CSS Mentors on Slack

- Folashade: @folashade
- Toluwanimi: @Toluwanimi
- Femi: @Femi Oye
- Bolade: @Bolade Oye
- Jide: @Jide Oye
- Taiwo: @Taiwo

Congratulations on completing this phase! Let's style it.....

---