COMP6080

🍓 Other - Outline

Lecture 1.1

Author(s): Hayden Smith

(Download as PDF)

🤷‍♀️ Why COMP6080?

Web is exploding. Everyone wants web-based front-end developers.

- 28% of github code pushes are javascript
- In 2018, 1 in 4 job postings were for a javascript based framework (source: Hacker News Hiring Trends)

Within 10 weeks I want to make you all confident in building your own simple web apps, to build them quickly, and to build it to look how you want it to look and behave.

But first let's answer two questions:

1. What is a front-end developer?
2. Why is web exploding?

🎨 What Is A Front-End Developer?

Someone who writes code that is executed client-side, typically part of a end-user facing product.

In most cases, they write code for web browsers. This is a course about helping you make web browsers do things.

They largely use HTML to structure pages, CSS to style them, and Javascript to make them dynamic.

They build things for *people* to use.

🐝 Why COMP6080?

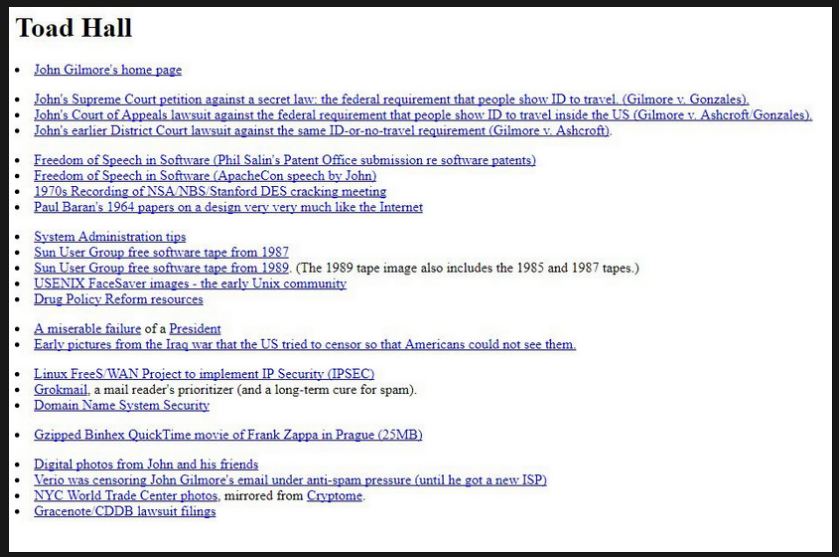
💥 Web is exploding.

🕵🏽 How did we get to this point?

🚶‍♀️ The Progress In Web

1️⃣ HTML Static Pages

Beginning of websites



Toad Hall

- John Gilmore's home page

- John's Supreme Court petition against a secret law: the federal requirement that people show ID to travel. (Gilmore v. Gonzales).
- John's Court of Appeals lawsuit against the federal requirement that people show ID to travel inside the US (Gilmore v. Ashcroft/Gonzales).
- John's earlier District Court lawsuit against the same ID-or-no-travel requirement (Gilmore v. Ashcroft).

- Freedom of Speech in Software (Phil Salin's Patent Office submission re software patents)
- Freedom of Speech in Software (ApacheCon speech by John)
- 1970s Recording of NSA/NBS/Stanford DES cracking meeting
- Paul Baran's 1964 papers on a design very very much like the Internet

- System Administration tips
- Sun User Group free software tape from 1987
- Sun User Group free software tape from 1989. (The 1989 tape image also includes the 1985 and 1987 tapes.)
- USENIX FaceSaver images - the early Unix community
- Drug Policy Reform resources

- A miserable failure of a President
- Early pictures from the Iraq war that the US tried to censor so that Americans could not see them.

- Linux FreeS/WAN Project to implement IP Security (IPSEC)
- Grokmail, a mail reader's prioritizer (and a long-term cure for spam).
- Domain Name System Security

- Gzipped Binhex QuickTime movie of Frank Zappa in Prague (25MB)

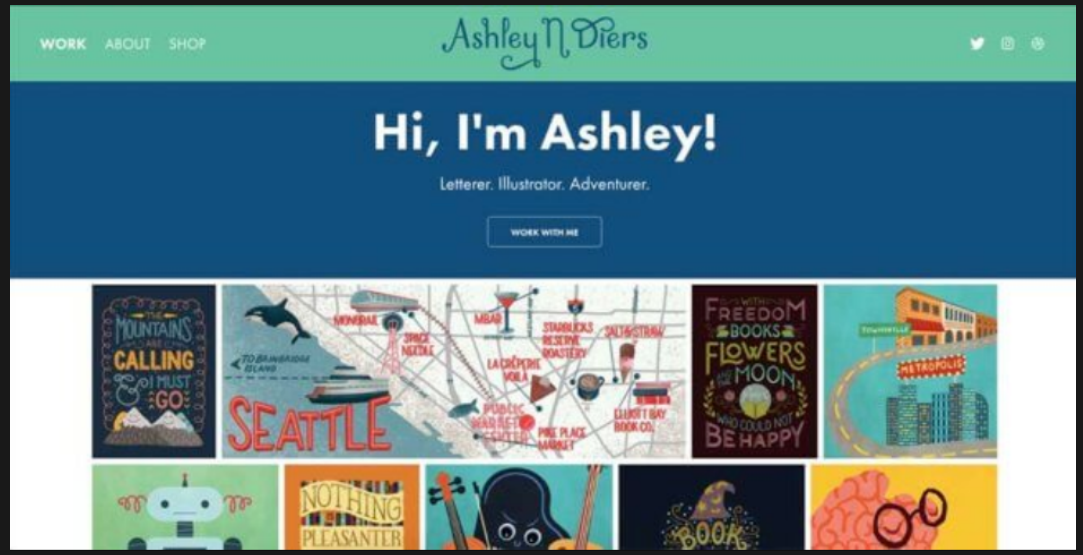
- Digital photos from John and his friends
- Verio was censoring John Gilmore's email under anti-spam pressure (until he got a new ISP)
- NYC World Trade Center photos, mirrored from Cryptome.
- Gracenote/CDDB lawsuit filings

HTML released 1993

🏃‍♀️ The Progress In Web

2️⃣ Basic CSS / Javascript

Now things don't like terrible

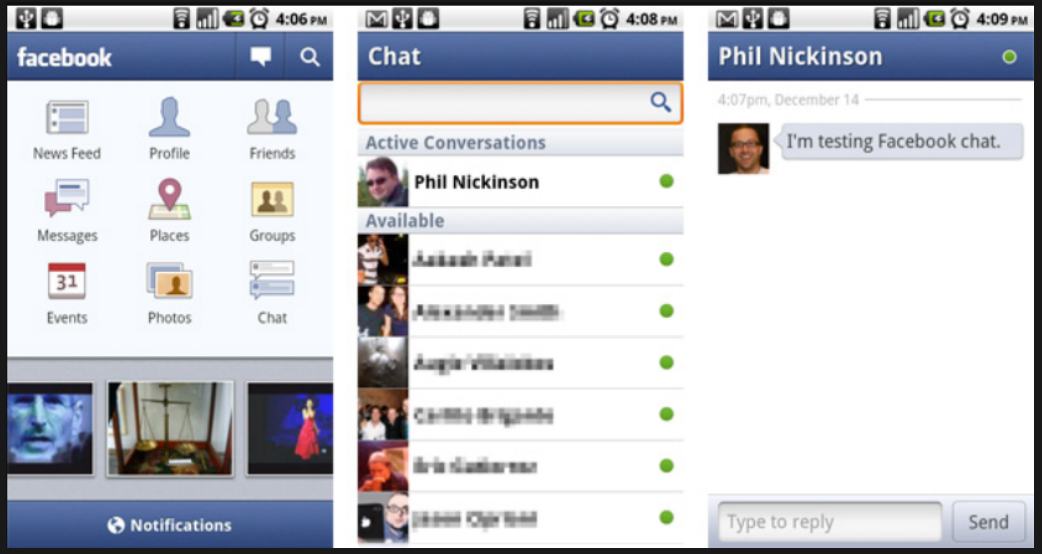


JS released 1995, CSS released 1996

 The Progress In Web

3️⃣ AJAX and background requests

A whole new class of websites, without the need to refresh



AJAX appeared 2000~

🏃‍♀️ The Progress In Web

4️⃣ NodeJS & Typescript

Type checking and common codebases - the beginning of heavy JS infrastructure

NodeJS released 2009, TS released 2012

🏃‍♀️ The Progress In Web

5️⃣ Hybrid Apps, Electron

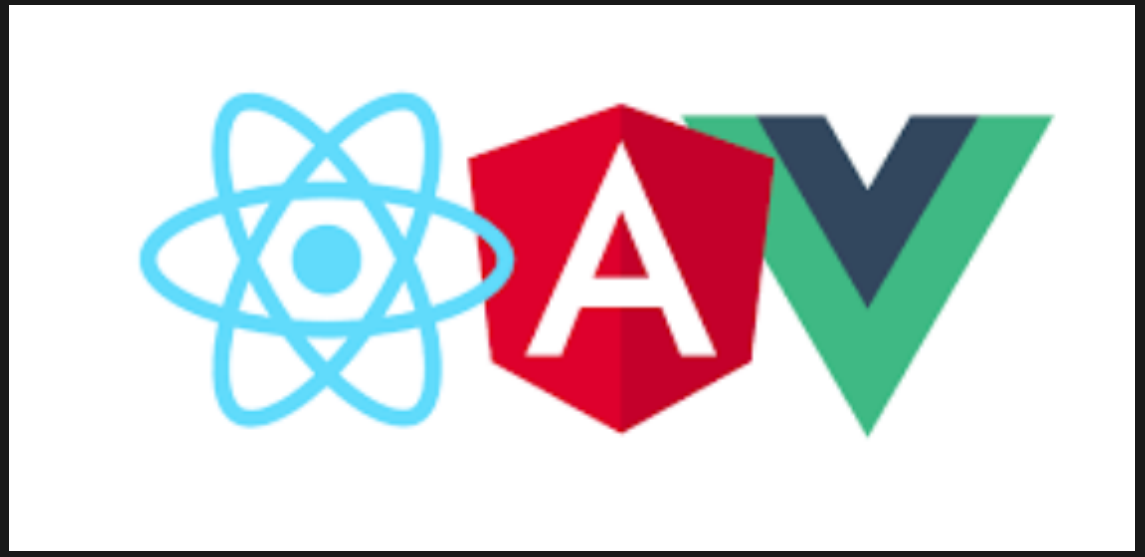
Type checking and common codebases - the beginning of heavy JS infrastructure

Electron, Apache Cordova

🏃‍♀️ The Progress In Web

6️⃣ Declarative Frameworks

Building of complex applications rapidly and in a scalable way. A web approach finally optimised for apps, rather than pages.



AngularJS released 2010, ReactJS released 2013, VueJS released 2014

The Progress In Web

In the vast majority of cases, your native desktop applications, native mobile applications, and web-apps or websites, can be built on a web-based, javascript based stack.

🧠 Assumed Knowledge

- Basics of GIT
- Basics of HTTP & Web Browsers
- High level understanding of scripting languages

This course has no other assumed knowledge. Experienced web developers will likely find this course slow.

If you lack this assumed knowledge, we have lectures to provide you all the help.

📗 Summarised Learning Outcomes

- Fundamentals of Javascript to design, construct, test, debug code
- Understanding HTML, CSS, DOM, to construct web pages
- Modern Javascript & CSS frameworks, to build componentised apps
- Understanding of asynchronous programming in the context of JS
- Basic knowledge of front-end security
- Awareness of UI/UX design, including accessibility

🌻 Approach To Teaching

Firstly, this is a challenging level 6 course. It will be hard to perform in the HD bracket without hard work or natural intellect.

Secondly, learning front-end development will be a bit different from other languages you've learned:

- Front-end is a 'breadth' topic, which is unlike many other languages
- A lot less time is spent solving algorithmic or computational problems, and a lot more time is spent just trying to fine tune your visuals or find the right method/library/approach to get the expected behaviour
- You will still feel like you have a lot to learn by the end of the course

💻 Course Site

Our course site is a custom content management system called Eckles. Eckles was written by Hayden in January 2022.

It is a ReactJS application with a lightweight NodeJS server. The source code is public for all all students, and we encourage you to contribute.

🔨 Why Was This Built?

- COMP6080 is trying to pioneer a new style of learning in CSE, and all previous software was not considered appropriate for this style
- We want to give you a real example to reference throughout the course and something we can all have fun with together.

🥰 The Team

We have a team of staff from UNSW who support you through your teaching. We also have a helping hand with some pre-recorded lectures from Canva.

👀 Assessment Structure

There is no direct assessment associated with Lectures or Tutorials

Item	Due	Weighting
Assignments	Due weeks 3, 4, 7, 10	80%
Exam	Exam Period	20%

📝 Important Notes About The Course

- This is a "hard" course for some - it's a breadth course (unique for many of you)
- Assignment due dates are as late as possible (includes no late penalty)
- We consider Vanilla JS a critical part of teaching - a course just in ReactJS wouldn't do you justice.
- Pre-recorded industry lectures are always being reviewed
- We can't teach everything! So we have to cut some things, and sometimes have bonus lectures.
- We don't have labs, because this is a level 6 course.

👨🏽‍🏫 Teaching Methods

🟢 Lectures

- Avg 2 hours of pre-recorded per week (tightly broken up)
- Avg 2 hours live lectures (Mon 6pm-8pm) (ideally no new content, all demos)
- Lectures are categorised into levels of importance for you.

🟡 Tutorials

- There are 6 tutorial times a week, covering 3 streams of content (available on the schedule page)
- All tutorials are online
- You can attend none, any, or all tutorials
- We will record tutorials and snip them up later
- Tutorials are also your self-learning exercises
- Please respect your tutor probably needs to leave at the end

👨🏽‍🏫 Teaching Methods

🔵 Help Sessions

- Chance to talk to tutors and get help on matters to do with tutorial exercises and assignments
- Help sessions will contain 1-5 tutors who will be split between assisting students with questions, and marking labs off
- Please pay attention to how many tutors are in a given help session - we do our best to predict demand and adjust but please attend help sessions being prepared to wait

🟣 Assignments - In-Depth Skills

- Ass1: Static HTML/CSS
- Ass2: Intro to DOM
- Ass3: Building an app with HTML/CSS/VanillaJS
- Ass4: Building and testing an app with ReactJS (pair)

🔥 Assignments are now due 4 days later this term 🔥

🙏 Getting Help

If you need help with something, go here:

1. EdStem (sidebar on Webcms3)
2. Help Sessions
3. cs6080@cse.unsw.edu.au

☃️ Getting Setup

⌨️ Running Your Code

- In this course you'll need to use: NodeJS, NPM, Web Browser, HTML, CSS, ReactJS, and more.
- All of these tools can be easily run on your local machine (recommended) for OSX, Linux, or Windows.
- If you aren't comfortable with local installs, you can complete the course using gitlab (we have installed all relevant programs there).

💾 Gitlab

- gitlab is the online tool we're using to manage our git repositories.
- We will provide a demo of it's usage.
- There are git resources on Eckles.

📚 Resources

- Web is a very well-resourced set of tools on the internet. Self-guided research will generally be adequate.
- The biggest issue will be finding the appropriate resources.
- Lectures may include resources on slides, and we also have a course-wide resources page here.

🎭 Style Guide

Information about style guides for particular languages can be found on various parts of our resources pages.

 Feedback

Throughout term, you can leave anonymous feedback by clicking on the link in the Eckles sidebar "Feedback"

💝 Getting Along

- Understand the expectations around student conduct.
- Create an inclusive learning environment.
- Let's all treat each other with respect and understanding.

⚒️ A Taste: "Making Web Browsers Do Things"

- Most things we do in this course are various combinations of:
  - **HTML**: Page structure
  - **CSS**: Page aesthetics
  - **Javascript**: Page dynamics
- Other topics we discuss are not about technology, but rather how to use it effectively:
  - Accessibility, product design, UI/UX, testing
- (Optional) Let's do a short demo to explore what is coming up in the first weeks

🦻 Feedback

Or go to the form here.

COMP6080
Web Front-End Programming

HTML Intro

HTML

This is the basic "webpage" scaffold on virtually every website on the internet. It's made up of "HTML".

```
<!doctype html>
<html>
  <head>
    <title>Title</title>
    <!-- More meta -->
  </head>
  <body>
    Hi
    <!-- More meta -->
  </body>
</html>
```

What is HTML?

HTML stands for "Hypertext Markup Lanaguge". It is a markup language that provides the structure for webpages. It does not provide the aesthetics/style (that's what CSS is for), and it does not deal with dynamic state (that's what Javascript is for).

A standard HTML tag consists of:

1. Tag name
2. (optional) Series of attribute/value pairs
3. (optional) Inner HTML

```
<tag1 attr1="value1" attr2="value2">
  <tag2>Text</tag2>
  <tag3>Other</tag3>
</tag1>
```

What is a web browser?

A web browser is essentially a tool that takes in HTML, CSS, Javascript, and more, and renders dynamic web pages. Think of a web browser a little bit like a compiler/interpret that takes in our HTML as "source code" and produces something that is compiled for the user/client.

This "source code" is available in most web browsers, and is easy to find and analyse.

<!DOCTYPE html> <html></html>

<!DOCTYPE html> is required by the specification on modern websites. However, if you omit it, many browsers will render the page anyway.

```
<!doctype html>
<html>
  <head>
    <title>Title</title>
    <!-- More meta -->
  </head>
  <body>
    Hi
    <!-- More meta -->
  </body>
</html>
```

<head></head>

Contains meta information of the page. Nothing in this section is rendered in the browser.

We usually use this to:

- Give meta-information about the page
- Load in CSS stylesheets
- Load in Javascript code

```
<head>
  <title>My page</title>
  <meta charset="utf-8" />
  <link rel="icon" href="favicon.ico" />
  <link rel="stylesheet" type="text/css" href="styles.css" />
  <script type="text/javascript" src="script.js"></script>
</head>
```

<body></body>

Everything inside <body> renders on the webpage. <body> only begins rendering after all the meta information in <head></head> has been processed by the web browser.

Most tags associated with the body focus on rendering something...

Layout Tags

Layout tags help us separate our page into separate structures. Many of these tags have limited inherent properties, and are just semantically meaningful ways of distinguishing or identifying particular key parts of the webpage.

```
<!-- This is an HTML comment -->

<div></div> <!-- A generic "box grouping" element -->
<span></span> <!-- A generic "grouping" element -->
<p></p> <!-- A paragraph -->
<h1></h1> <!-- 1st biggest header -->
<h6></h6> <!-- 6th biggest header (also h2, h3, h4, h5) -->
<ul><li></li><li></li></ul> <!-- A unordered list of items -->
<ol><li></li><li></li></ol> <!-- A ordered list of items -->
<table></table> <!-- A table of information -->
<b></b> <!-- Bold text -->
<i></i> <!-- Italic -->
<u></u> <!-- Underlined text -->
<!-- So much more.... -->
```

Formatting Tags

Formatting tags usually have some basic visual properties that are assigned to all elements inside those tags.

```
<!-- This is an HTML comment -->

<b></b> <!-- Bold text -->
<i></i> <!-- Italic -->
<u></u> <!-- Underlined text -->
<!-- So much more.... -->
```

Links: <a>

The "anchor" tag is how we link to another resource from our current page. It takes in a URL that may be relative or absolute.

There are other interesting properties of the anchor too, including:

- **title** attribute which is the hover-over tooltip text
- **target="_blank"** to open the link in a new tab

```
<a href="/settings" title="Open settings page">Settings</a>

<a href="https://en.wikipedia.org" target="_blank">Wikipedia</a>

<!-- A rare but interesting use case -->
<a href="/image.png" download><img src="download-icon.png" /></a>
```

Images: <img />

- An image tag is one of many HTML properties known as a "void" tag. Since it doesn't need to contain any inner HTML, we don't need a close tag.
- We can specify image height, width, and source of the image
- We can also specify the "alternate" text in case the image doesn't load

```
1 <img src="cat.jpg" alt="Image of the cat" width="300" height="600" />
```

<form></form>

Forms will be covered in much more depth in another lecture. In particular, the dynamic interaction with a form and it's results will be covered later. We will not cover **action** or **method** of a form just yet.

Forms provide the structure to collect information from a user and then submit it. Key parts of forms include:

- inputs & labels
- textarea
- select
- button
- submit

<input />

Form inputs consist of the a number of properties:

- **type:** the type of the field (text, radio)
- **name**: name of the attribute during submission
- **value**: the default value to the field that will be sent when submitting
- **placeholder:** background text to hint at what value to input
- **disabled:** boolean as to whether the field is disabled

```
<!-- There are more input types -->

<input type="text" placeholder="Hayden Smith" />

<input type="radio" id="yes" name="answer" value="yes" checked />
<input type="radio" id="no" name="answer" value="no" />

<input type="checkbox" id="no" name="answer" value="no" checked />
<input type="checkbox" id="yes" name="answer" value="yes" />

<input type="hidden" name="messageID" value="message123" />
```

<textarea></textarea>

Textareas are essentially "extended response text inputs"

```
<textarea rows="5" cols="40" placeholder="Write here">
  Default value
</textarea>
```

<select></select>

<select> is a dropdown box that contains options inside of it.

```
1 <select name="animal">
2   <option value="" selected>Select your favourite animal</option>
3   <option value="dog">Dog</optio>
4   <option value="cat">Cat</option>
5 </select>
```

<label></label>

Labels group text to an input so that when text is clicked, the field is selected or focused

```
1 <label for="dog">I am a dog</label>
2 <input type="checkbox" value="dog" id="dog" />
```

<button></button>

You can make a form button with <button></button>. These can have a lot of functionality added to them later with Javascript.

An input of type "submit" is also a button that will automatically submit the form (covered later).

```
<button>
  This is a general button
</button>

<input type="submit" value="This is a submit button" />
```

Interesting HTML

```
<!-- iframes allow us to include a "view" to another webpage in our own -->
<iframe src="https://google.com" width="400" height="400"></iframe>

<!--
HTML5 brought with it a number of other interesting features.
These include the ability to render the playing of audio, and video
-->

<audio src="music.mp3" controls>
  Browser does not support audio
</audio>

<video src="movie.mp4" type="video/mp4" controls>
  Browser does not support video
</video>
```

Extra Information

- The <html> tag often has an attribute <html lang="en-AU"> that search engines use to understand the language your website is in.

COMP6080
Web Front-End Programming

HTML
Image Types

Images

Images are files that are used to render a collection of pixels on a screen that provide a visual.

There are countless image formats, but we will explore some of the key ones used in web browser. This is a great article.

JPG **PNG** **GIF** **BMP** **SVG**

Two Key Categories

Raster Images

Raster image files display an image where every pixel has a defined colour (e.g. RGBA) and position.

When enlarged, the original image is just stretched, leading to a lower quality image.

GIF BMP JPG PNG

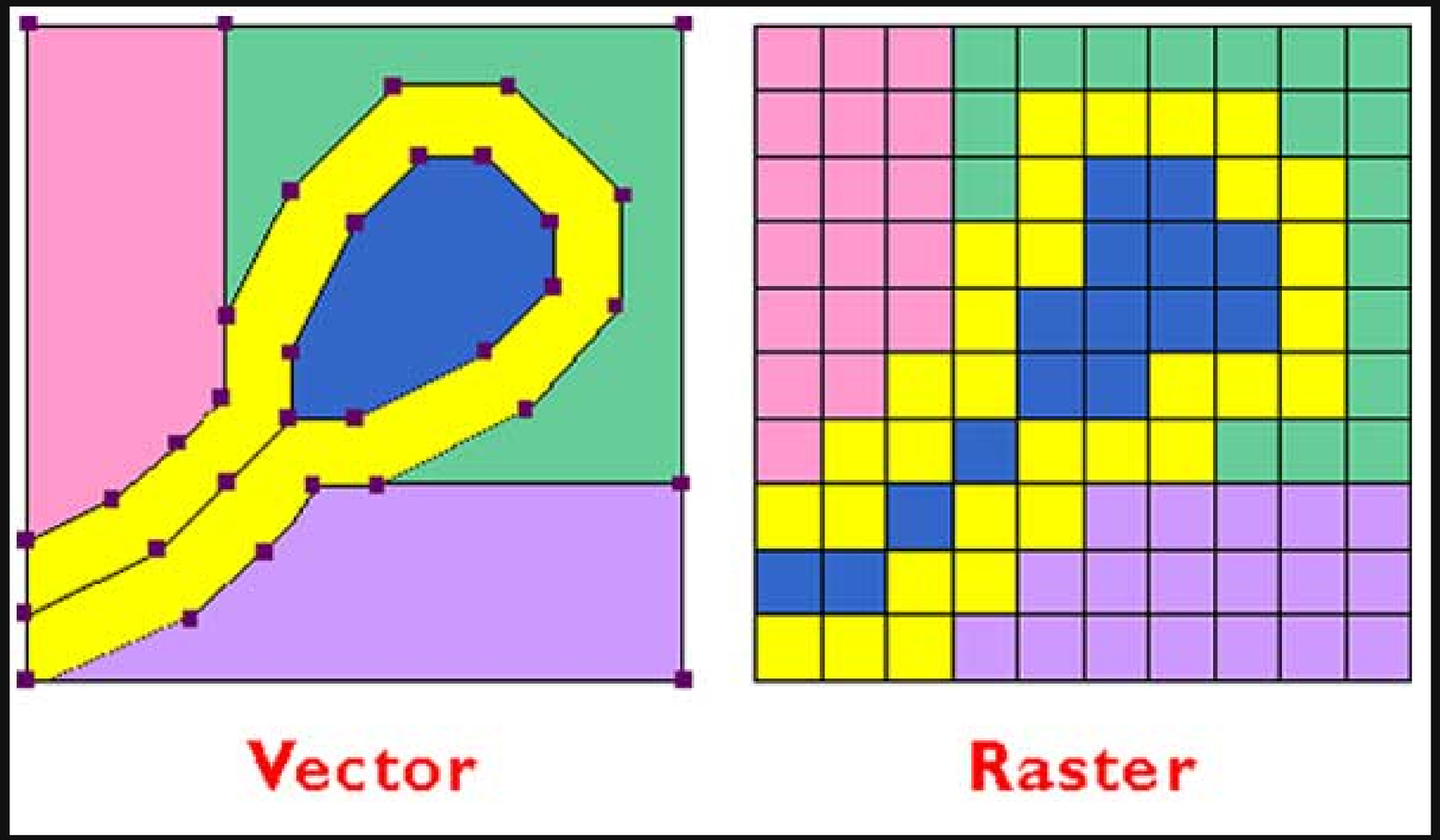
Vector Images

Vector image files store a series of geometric instructions (lines, shapes, colours) that are rendered on-the-fly by the browser. Vector image files are typically used for icons & animations.

Vector images do not distort when enlarged

SVG

Two Key Categories



Source: Psu.edu

Vector Images

Vector images, specifically SVGs, will be discussed in a separate SVG lecture.

Raster Images

	BMP	**GIF**	**JPG**	**PNG**
General Format	Variable	8 bit	24 bit	16-24 bit
Compression	No	Yes, Lossless	Yes, Lossy	Yes, Lossless
Uses	No	Image animations*	High resolution photos	Most non-photo use cases

* Nowadays a .webm is an increasingly common format for animations than GIF

Base64 Encoding

Instead of loading an image as a resource via HTTP, you can instead encode it directly into your page of API response.

This reduces the number of HTTP connections required (there is a finite amount that can run in parallel), but does increase the overall amount of data needing to be processed, and will naturally slow down the request being made that contains that information.

Commonly used for SMALL images.

CSS

(Cascading Style Sheets)

Used to describe the presentation of HTML document

Import CSS

To add styles to the document could be used of the following methods:

Import external file with styles

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

Add inline styles to specific element

```
<div style="color: red">text</div>
```

Add styles directly to the document

```
<style>
  .class {
    color: red;
  }
</style>
```

Which one to use depends on the situation. Usually most of the styles stores in the external file, but some the most critical styles could be added to the document or as inline styles

CSS Structure

CSS file built from the list of CSS Rules

CSS Rule has 3 parts:

Selector - define the elements to which the rule applies

Property - one of the style property

Value - one of the possible value for the given property

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

Universal selector

`*` - Selects all the elements on the page

Better to avoid using universal selector unless it's necessary

```
<h1>Hello world</h1>
<div>This is my <span>Article</span></div>
```

```
* {
  color: blue;
}
```

Type selector

tagName - Selects all the elements with the given node type

```
<span>one</span>
<span>two</span>
<span>three</span>
```

```
span {
  color: blue;
}
```

Class selector

.className - Selects all the elements with the given class

Page could contains any number of elements with the same class name

```
<p class="text">The first block of text</p>
<p class="text">The second block of text</p>
```

```
.text {
  margin-top: 10px;
}
```

ID selector

#id - Selects the element with the given id

Page should contains only one element for each ID

```
<div id="error">The password is incorrect</div>
```

```
#error {
  color: red;
}
```

Attribute selector

selector[attr=value] - Selects all the elements with the given attribute pair - name/value

```
<input type="radio" name="vote" value="one" />
<input type="radio" name="vote" value="two" />
```

```
input[type="radio"] {
  margin: 0 10px;
}
```

Combinators

Combinators establish a relationship between selectors

```html
<div class="A">
  <span class="B">one</span>
  <span class="C">two</span>
  <span class="B">three</span>
  <span class="B">four</span>
  <div>
    <span class="B">five</span>
  </div>
</div>
<span class="B">six</span>
```

```css
/* Descendant: all .B elements inside .A */
/* one, three, four, five */
.A .B { color: red; }

/* Child: all .B which are direct childs of .A */
/* one, three, four */
.A > .B { color: red; }

/* Adjacent sibling: .B that follows immediately after .C */
/* three */
.C + .B { color: red; }

/* General sibling: all sibling .B that follows after .C */
/* three, four */
.C ~ .B { color: red; }
```

Pseudo-classes

selector:pseudo-class - Selectes the elements with a special state

```css
/* The button which is the first/last element in its container */
button:first-child { background-color: blue; }
button:last-child { background-color: blue; }

/* The button over which the user's pointer is hovering  */
button:hover { background-color: blue; }

/* The input which received focus */
input:focus { color: red; }

/* Disabled input */
input:disabled { background-color: #ccc; }

/* Links that a user has already visited */
a:visited { color: red; }
```

Pseudo-elements

selector::pseudo-element - Used to create cosmetic content for the element or allows to style a specific part of the element

```
/**
 * Only the first letter in the .text elements
 * will have a red color
 */
.text::first-letter {
  color: red;
}

/**
 * After the .text elements will shown "*"
 * with the given styles
 */
.text::after {
  content: '*';
  color: red;
}
```

Cascade

In CSS rules applied to the element in the order which they are written in the document

That means the value of the property from the last rule will override values of the same property for all other rules which are target this element

```css
.title {
  /* Will be overridden by the next rule */
  font-size: 36px;
}

.title {
  /* Will be used for .title */
  font-size: 40px;
}
```

Inheritance

Many properties are inheriting their values from the parents of the element

So you don't need to write a rule for every element

And in opposite need to be careful and **reset** the styles from the parents if they are not expected for this particular element

```
<article>
  My first <span>article</span>
</article>
```

```
article {
  color: #444;
}

span {
  /**
   * Will also have "color: #444" from its parent
   * even it's not defined here
   */
}
```

Specificity

If an element has multiple rules with the same property, to decide what value should be used, beside cascade browser will look on the specificity of each selector and choose the highest one

Specificity calculated by the number of each selector type:

X-0-0-0 : inline styles have more specificity than any selectors

0-N-0-0 : number of id selectors

0-0-N-0 : number of class selectors

0-0-0-N : number of type selectors

https://specifishity.com

```
<div id="id" class="class1 class2">text</div>
```

```
/* Specificity: 0121 */
div#id.class1.class2 {
  /* Will be used as has more specificity */
  color: red;
}
/* Specificity: 0110 */
#id.class1 {
  color: blue;
}
```

!important

Overrides any other declarations of the property applied to the element (even inline styles)

Better to avoid using **!important** rule as it hard to debug and maintain the code when it contains it

```
<div
  id="id"
  class="class1 class2"
  style="color: red;"
>text</div>
```

```
.class1 {
  /* Will overrides any other declarations */
  color: green!important;
}

div#id.class1.class2 {
  color: blue;
}
```

CSS

(Cascading Style Sheets)

Formatting

Text

CSS has various ways to style the text

The most common properties:

color - change the color of the text

font-size - change the size of the text

font-family - specify the list of fonts that will be used to render the text

font-weight - sets the boldness of the text

font-style - sets whether the text renders italic/oblique or normal

text-decoration - adds decoratives lines on the text

```html
<div>
  <span class="size">size</span>
  <span class="font">font</span>
  <span class="color">color</span>
  <span class="bold">bold</span>
  <span class="italic">italic</span>
  <span class="underline">underline</span>
</div>
```

```css
.size { font-size: 2em; }
.font { font-family: "Comic Sans MS"; }
.color { color: red; }
.bold { font-weight: 600; }
.italic { font-style: italic; }
.underline { text-decoration: underline; }
```

size font color **bold** *italic* underline

Background image

Sets the image as a background of the element

CSS has multiple properties which allow to correctly render the background image

background-image - sets the image as a background

backgorund-repeat - sets how image should repeat if its smaller than the element

background-position - sets the initial position of the image

background-size - sets the size of the image

```css
.image {
  width: 100px;
  height: 100px;
  border: 5px solid blue;
  background-image: url("cat.jpg");
  background-size: contain;
  background-position: center;
  background-repeat: no-repeat;
}
```

It couldn't be interacted by the user, so you shouldn't render any meaningful images this way

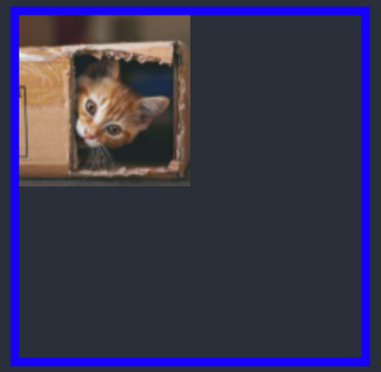
Background size

Change the size of the background image to fits it to the element

contain - scale the image to fits the element

cover - scale the image as large as possible and crop the overflow part

```
.image { background-size: contain; }
```



```
.image { background-size: cover; }
```

Background position

Sets position for X and Y coordinates of the element

```
image { background-position: center; }
```

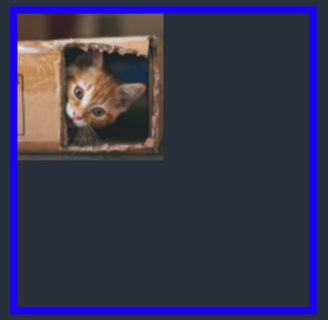
```
.image { background-position: top right; }
```

Background repeat

By default if image is smaller than the element it will repeats until it fills the whole element. It could be changed to repeat only in a specific direction or not repeat at all

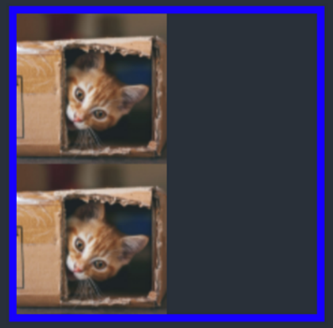
```
.image { background-repeat: repeat; }
```

```
.image { background-repeat: no-repeat; }
```



```
.image { background-repeat: repeat-x; }
```

```
.image { background-repeat: repeat-y; }
```



Helpful links

- **https://www.w3.org/TR/CSS2/** - specification
- **https://caniuse.com** - checks the browsers support of the css properties
- **https://developer.mozilla.org/en-US/docs/Web/CSS/Reference** - documentation from Mozilla corporation
- **https://csstriggers.com** - checks what property triggers while rendering
- **https://css-tricks.com** - the portal with bunch of helpful tricks and articles

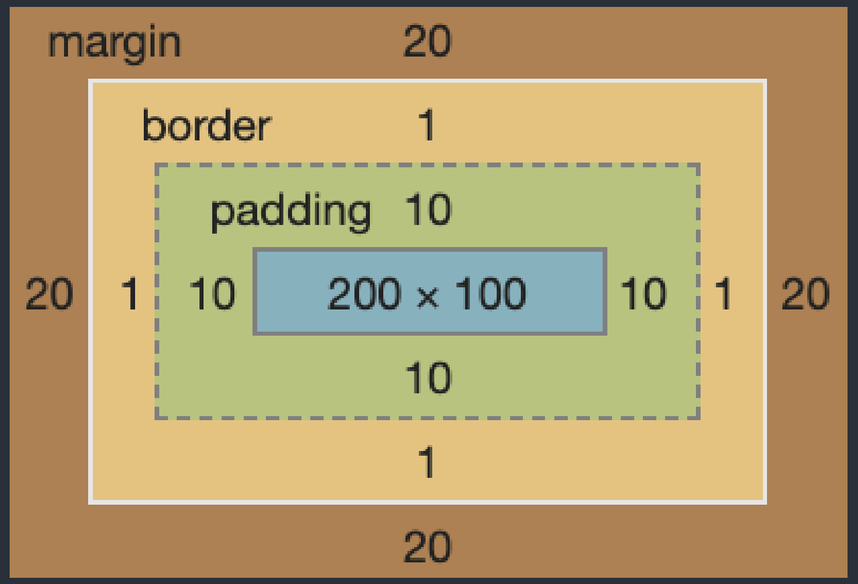
CSS

(Cascading Style Sheets)

Layouts

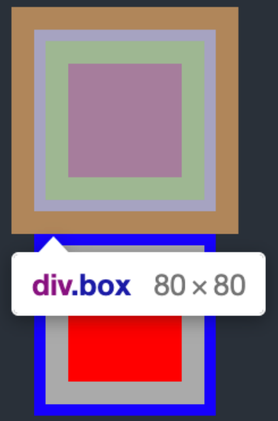
Box model

Each element represented as a rectangular box for the rendering




```
<div class="box"><div class="content"></div></div>
<div class="box"><div class="content"></div></div>
```

```
.box {
  width: 50px;
  height: 50px;
  padding: 10px;
  margin: 10px;
  border: 5px solid blue;
  background-color: #aaa;
}
.content {
  height: 100%;
  background-color: red;
}
```



Block elements

Start with a new line and stretch to the full width of the container

Could contain block and inline elements

Some properties could be applied only to block elements (such as width, height, margin, padding...)

```
<div>
  text before
  <div class="block">
    block <span>element</span>
    <div>inner block</div>
  </div>
  text after
</div>
```

```
.block {
  display: block;
  border: 5px solid blue;
}
```

text before

block element
inner block

text after

Inline elements

Have the size of the content

Could contain only text and other inline elements

Don’t break the flow

***Inline-block** - creates a block element (could use all the properties from the block element) which behaves in the flow as inline

```
<div>
  text before
  <span class="inline">
    inline
    <span class="inlineBlock">element</span>
  </span>
  text after
</div>
```

```
.inline {
  display: inline;
  color: red;
}

.inlineBlock {
  display: inline-block;
  border: 5px solid blue;
}
```

Display: none

Completely removes the element from the flow

```
<div>
  text before
  <div class="none">
    removed block
  </div>
  text after
</div>
```

```
.none {
  display: none;
}
```

text before text after

Visibility

Change visibility of the element, but keep it in the flow

```
<div>
  hide
  <span class="visibility">the element</span>
  and preserve the space
</div>
```

```
.visibility {
  visibility: hidden;
}
```

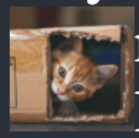
hide and preserve the space

Float

Removes an element from the normal flow and puts it to the left/right of the container and allows to other inline elements wrap around it

```
<div>
  very very long long text
  <img src="cat.jpg" class="float"/>
  more of very long long text
  more of very long long text
</div>
```

```
.float {
  float: left;
  width: 50px;
}
```

very very long long text  more of very long long text more of very long long text

Static position

Positions an element based on the normal flow

Offset properties have no effect on it

The default value of the position property

```html
<div class="container">
  Shows
  <span class="position">the position</span>
  of the element
</div>
```

```css
.container {
  position: relative;
  border: 5px solid blue;
  padding: 20px;
}

.position {
  color: red;
  position: static;
  top: 20px;
  left: 20px;
}
```

Relative position

Positioned an element based on the normal flow

Offset properties will apply on it based on its default position

```
<div class="container">
  Shows
  <span class="position">the position</span>
  of the element
</div>
```

```
.container {
  position: relative;
  border: 5px solid blue;
  padding: 20px;
}

.position {
  color: red;
  position: relative;
  top: 20px;
  left: 20px;
}
```

Absolute position

Removes an element from the normal flow

It will be positioned by the offset properties based on the closest parent element with not static position

```
<div class="container">
  Shows
  <span class="position">the position</span>
  of the element
</div>
```

```
.container {
  position: relative;
  border: 5px solid blue;
  padding: 20px;
}

.position {
  color: red;
  position: absolute;
  top: 0;
  left: 0;
}
```

Fixed position

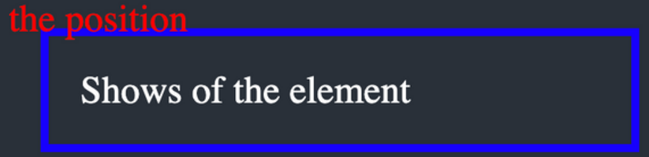
Removes an element from the normal flow

Offset properties will apply on it based on the viewport

```html
<div class="container">
  Shows
  <span class="position">the position</span>
  of the element
</div>
```

```css
.container {
  position: relative;
  border: 5px solid blue;
  padding: 20px;
}

.position {
  color: red;
  position: fixed;
  top: 0;
  left: 0;
}
```



Overflow

Sets how an element will show its content when it overflows the edges

visible - content will be rendered outside the edges (default value)

hidden - hide the content outside the edges. Doesn't allow to scroll

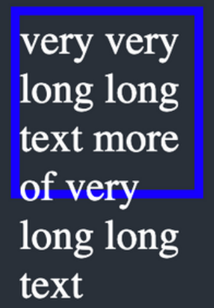
auto - hide the content outside the edges, but allows scrolling to the hidden content

scroll - hide the content outside the edges. Always show scrollbars, even when content fits the element

```
<div class="overflow">
  very very long long text
  more of very long long text
</div>

.overflow {
  overflow: visible;
  width: 100px;
  height: 100px;
  border: 5px solid blue;
}
```

overflow: visible



overflow: hidden

overflow: auto

Helpful links

- **https://www.w3.org/TR/CSS2/** - specification
- **https://caniuse.com** - checks the browsers support of the css properties
- **https://developer.mozilla.org/en-US/docs/Web/CSS/Reference** - documentation from Mozilla corporation
- **https://csstriggers.com** - checks what property triggers while rendering
- **https://css-tricks.com** - the portal with bunch of helpful tricks and articles

CSS

(Cascading Style Sheets)

Layouts

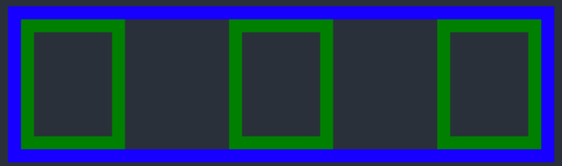
Flexbox

Flexbox defines layout where children could be positioned in any direction and change their size based on the available space

```
<div class="flex">
  <div class="child"></div>
  <div class="child"></div>
  <div class="child"></div>
</div>
```

```
.flex {
  display: flex;
  justify-content: space-between;
  width: 200px;
  height: 50px;
  border: 5px solid blue;
}

.child {
  width: 30px;
  border: 5px solid green;
}
```



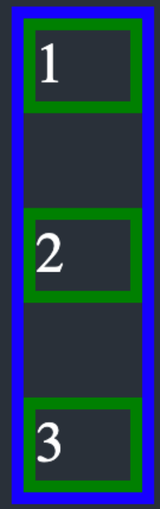
Flex: flex-direction

Defines the direction in which flex items will be placed

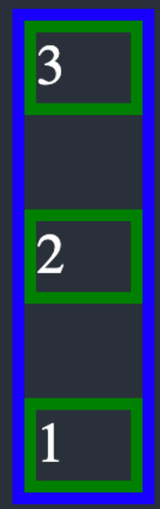
row

row-reverse

column



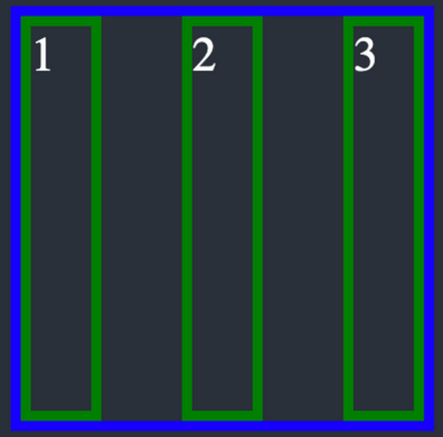
column-reverse



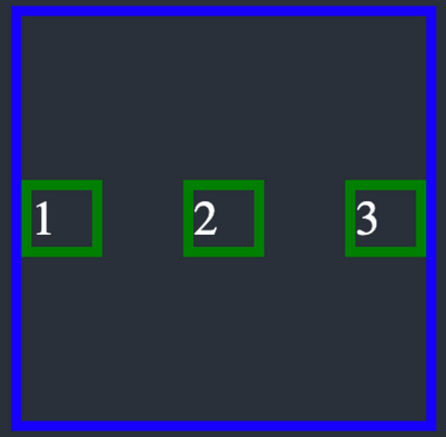
Flex: align-items

Defines how items will be positioned inside the flex element

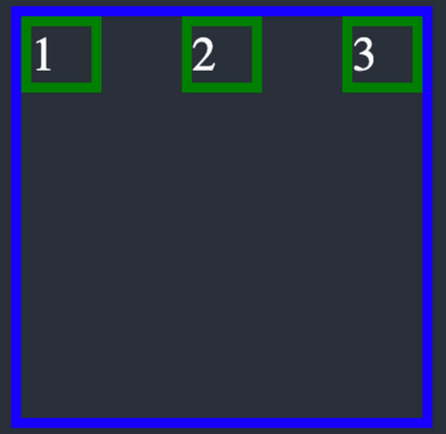
stretch



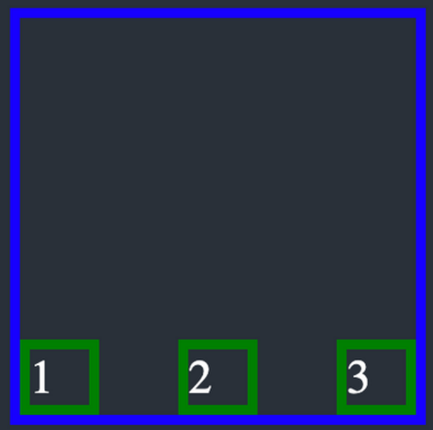
center



flex-start



flex-end



Flex: justify-content

Defines how space between items will be processed

space-around

flex-start

center

space-between

flex-end