

REPORT INTERNSHIP

WEEK 1

Trainee: Phan Thiên Quân
Trainer/Mentor: Khoa Trần

A series of five parallel diagonal lines in a light blue color, extending from the bottom left towards the top right of the page.

APPS CYCLONE
FRONTEND - DEV

Contents

I. HTML	3
1.1 Basic	3
1.2 Form & Validation	11
Các cách Validate Form	13
1.3 Seo Basic	17
II. CSS	24
2.1. Basic	24
2.2 Making Layouts	26
2.3 Box Model	30
2.4 CSS Grid	30
2.5 Flex box	37
3. Responsive Web Design - Media Queries	42
III. BEM	45
3.1 Định nghĩa BEM	45
3.2 Quy tắc đặt tên	45
IV. SCSS	47
4.1 Scss Variables	48
4.2 SCSS Nested	48
4.3 Partial	49
4.4 Modules	49
4.5 Mixins	50
4.6 Extend/Inheritance	52
4.7 Operators	52
V. Tailwind CSS	55
5.1 Định nghĩa	55
5.2 Installation & Usage	55
VI. Javascript	57
6.1 Syntax & Basic	57
6.2 DOM Manipulation	60

6.3 Fetching API	64
AJAX	65
6.4 ES6.....	66
6.5 Library	66
VII. JQUERY	67
7.1 Syntax & Basic	67
7.2 DOM Manipulation	70

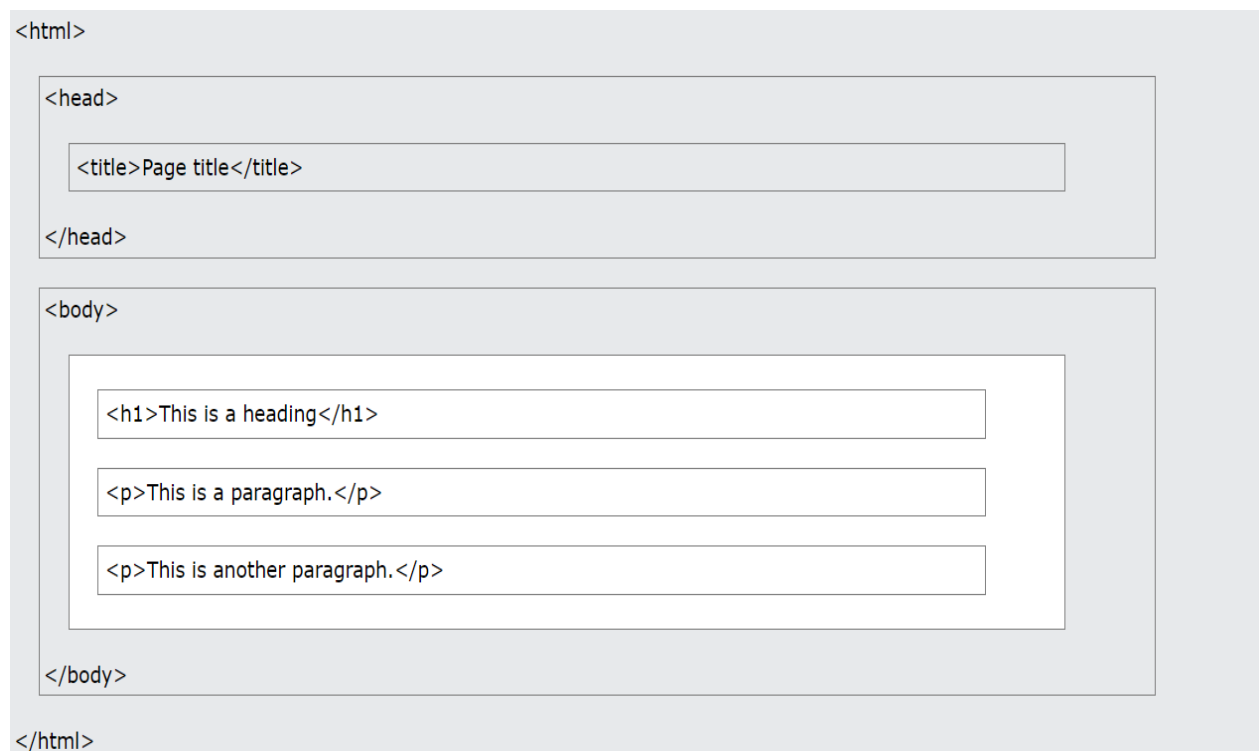
REPORT_WEEK 1

I. HTML

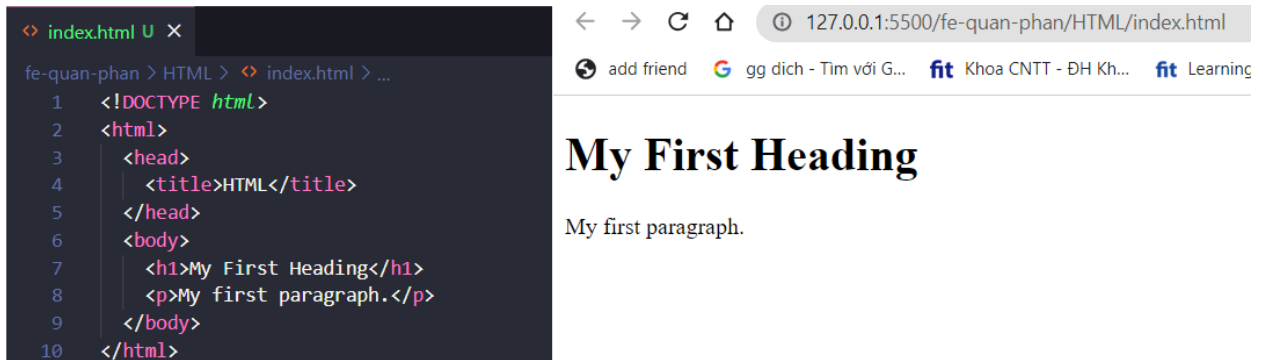
1.1 Basic

- **HTML** là viết tắt của Hyper Text Markup Language (ngôn ngữ đánh dấu siêu văn bản)
- **HTML** là ngôn ngữ đánh dấu tiêu chuẩn để tạo các trang Web
- **HTML** mô tả cấu trúc của một trang Web
- **HTML** bao gồm một loạt các elements
- Các phần tử HTML cho trình duyệt biết cách hiển thị nội dung
Các phần tử HTML gắn nhãn cho các phần nội dung như: 'đây là tiêu đề', 'đây là đoạn văn', 'đây là liên kết',

HTML Page Structure



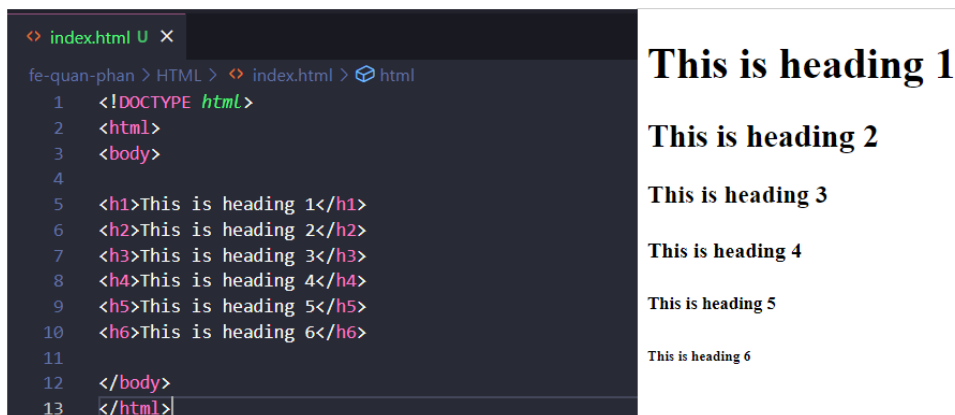
Example



- Tất cả HTML documents phải bắt đầu bằng khai báo bằng `<!DOCTYPE html>` (chỉ được xuất hiện 1 lần trong mỗi project và phải ở đầu trang)
 - Khai báo `<!DOCTYPE html>` xác định rằng tài liệu này là tài liệu HTML5
- HTML documents bắt đầu với thẻ `<html>` và kết thúc bằng thẻ đóng `</html>`
 - Phần tử `<html>` là phần tử gốc của trang HTML
- Phần `<head>` chứa thông tin về trang, bao gồm tiêu đề trang (title), meta và các tài nguyên bổ sung như CSS và JavaScript.
- Phần `<body>` là phần chứa nội dung hiển thị trên trình duyệt. Các phần tử chính như *header*, *nav*, *main*, và *footer* đại diện cho các phần của trang.

HTML Headings

Thẻ `<h1>` có kích thước lớn nhất, nhỏ dần cho tới `<h6>`



HTML Paragraph, Links, Images



Ta có thể thấy trong thẻ `img` có rất nhiều chữ màu xanh, đó là các **attributes**: **alt** – dùng để hiển thị chữ để mô tả một hình ảnh nếu như ảnh `src` bị lỗi không hiển thị ra, **width** và **height** là để điều chỉnh chiều dài và rộng của ảnh

HTML Style Guide

- Luôn có thẻ đóng và thẻ mở, ví dụ như là thẻ `<h1>` thì phải có `</h1>`
- Các thẻ luôn là các chữ viết thường, không được viết hoa, ví dụ như là `<body>`, **không được viết <BODY>**
- Tương tự đó các attribute cũng phải viết thường, ví dụ như trên ảnh ở trên ta có thẻ `<a>` thì attribute trong đó là `href`, **không được viết là HREF**
- Và khi khai báo attribute thì đằng sau phải luôn có dấu nháy kép, ví dụ `href="striped.com"`, **không được bỏ dấu nháy kép như này: href=striped.com**
- Không được bỏ qua thẻ `<title>`
- Khi viết code không có thẻ `<html>` hay là `<body>` thì vẫn hợp lệ

```
<!DOCTYPE html>
<head>
  <title>Page Title</title>
</head>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

This is a heading

This is a paragraph.

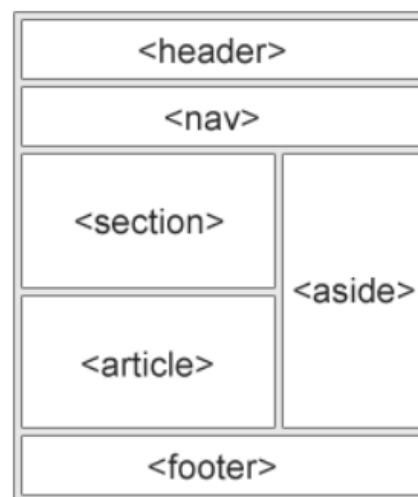
- Thẻ **<head>** cũng vẫn có thể bỏ qua được, nhưng tốt nhất 3 thẻ html, body và head không nên bỏ đi
- Luôn khai báo attribute 'lang' để khai báo ngôn ngữ trang web, hỗ trợ tìm kiếm trên trình duyệt `<html lang="en-us">`
- Để đảm bảo giải thích đúng và lập chỉ mục công cụ tìm kiếm chính xác, cả ngôn ngữ và ký tự mã hóa meta charset='charset' phải được xác định càng sớm càng tốt trong tài liệu HTML
- Set viewport : Khung nhìn là khu vực hiển thị của người dùng trên trang web. Nó thay đổi theo thiết bị:

`<meta name="viewport" content="width=device-width, initial-scale=1.0">`

- Comment code để ghi chú lại chỗ code đó dùng để làm gì, tô đen dòng đó rồi ấn phím tắt CTRL + /
- Dùng stylesheet để link file css tới
`<link rel="stylesheet" href="styles.css">`
- Loading javascript vào: `<script src="myscript.js">`

Semantic Elements in HTML

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



HTML <section> Element

Xác định 1 phần trong document

Ví dụ những nội dung mà thẻ này có thể sử dụng:

- Chapters
- Introduction
- News items
- Contact information

`<section>`

`<h1>WWF</h1>`

`<p>`The World Wide Fund for Nature (WWF) is an international organization working on issues regarding the conservation, research and restoration of the environment, formerly named the World Wildlife Fund. WWF was founded in 1961.`</p>`

`</section>`

HTML `<article>` Element

Được sử dụng để đánh dấu một phần tử HTML là một bài viết độc lập, một phần tử có thể tồn tại hoặc có thể được tái sử dụng một cách độc lập trên các trang web khác nhau.

`<article>`

`<h2>`Tiêu đề bài viết`</h2>`

`<p>`Mô tả ngắn về nội dung của bài viết.`</p>`

`<p>`Nội dung chi tiết của bài viết...`</p>`

`<p>`Thêm nội dung...`</p>`

`<p>`...`</p>`

`<footer>`

`<p>`Tác giả: John Doe`</p>`

`<p>`Ngày đăng: 5 tháng 7, 2023`</p>`

`</footer>`

`</article>`

HTML `<header>` Element

Đại diện cho một vùng chứa nội dung giới thiệu hoặc một tập hợp các liên kết điều hướng:

- Một hoặc nhiều thành phần tiêu đề (h1 - h6)
- Logo hoặc biểu tượng
- Thông tin tác giả

```
<article>
<header>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission:</p>
</header>
<p>WWF's mission is to stop the degradation of our planet's natural environment,
and build a future in which humans live in harmony with nature.</p>
</article>
```

HTML <footer> Element

Xác định một phần cuối trang của document như :

- Thông tin tác giả
- Thông tin bản quyền
- Thông tin liên lạc
- Sơ đồ trang web
- Liên kết trở lại đầu trang
- Tài liệu liên quan

```
<footer>
  <p>Author: Hege Refsnes</p>
  <p><a href="mailto:hege@example.com">hege@example.com</a></p>
</footer>
```

HTML <nav> Element

Xác định một tập hợp các link điều hướng trang web

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
```

```
<a href="/jquery/">jQuery</a>
</nav>
```

HTML <aside> Element

Xác định một số nội dung ngoài nội dung được đặt trong đó (như sidebar).

```
<p>My family and I visited The Epcot center this summer. The weather was nice, and
Epcot was amazing! I had a great summer together with my family!</p>
```

```
<aside>
```

```
<h4>Epcot Center</h4>
```

```
<p>Epcot is a theme park at Walt Disney World Resort featuring exciting attractions,
international pavilions, award-winning fireworks and seasonal special events.</p>
```

```
</aside>
```

Bảng tổng hợp

Tag	Description
<u><article></u>	Defines independent, self-contained content
<u><aside></u>	Defines content aside from the page content
<u><details></u>	Defines additional details that the user can view or hide
<u><figcaption></u>	Defines a caption for a <figure> element
<u><figure></u>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<u><footer></u>	Defines a footer for a document or section
<u><header></u>	Specifies a header for a document or section
<u><main></u>	Specifies the main content of a document
<u><mark></u>	Defines marked/highlighted text
<u><nav></u>	Defines navigation links
<u><section></u>	Defines a section in a document
<u><summary></u>	Defines a visible heading for a <details> element
<u><time></u>	Defines a date/time

Những thẻ thuộc nhóm inline

▪ <a>	▪ <abbr></abbr>	▪ <acronym></acronym>
▪ 	▪ <basefont></basefont>	▪ <bdo></bdo>
▪ <big></big>	▪ 	▪ <cite></cite>
▪ <code></code>	▪ <dfn></dfn>	▪
▪ 	▪ <i></i>	▪ <input></input>
▪ <kbd></kbd>	▪ <label></label>	▪ <q></q>
▪ <s></s>	▪ <samp></samp>	▪ <select></select>
▪ <small></small>	▪ 	▪ <strike></strike>
▪ 	▪	▪
▪ <textarea></textarea>	▪ <tt></tt>	▪ <u></u>
▪ <var></var>		

- Những thẻ thuộc nhóm inline là những thẻ cơ bản của HTML/XHTML, chỉ được dùng để chứa nội dung cho text hoặc các thẻ inline khác.
- Bản thân text cũng có thể coi thuộc nhóm inline.
- Những thẻ thuộc nhóm inline nên được bao ngoài bởi **nhóm các thẻ block**, vì các thẻ block sẽ lo nhiệm vụ dàn trang web, còn các thẻ thuộc nhóm inline chỉ để hiển thị nội dung cho văn bản.
- Những thẻ thuộc nhóm inline có thể được lồng vào nhau.
- Không được sử dụng các thẻ khác bên trong các thẻ inline

Thẻ thuộc nhóm Block

<address>	<article>	<aside>	<blockquote>	<canvas>
<dd>	<div>	<dl>	<dt>	<fieldset>
<figcaption>	<figure>	<footer>	<form>	<h1>-<h6>
<header>	<hr>		<main>	<nav>
<noscript>		<p>	<pre>	<section>
<table>	<tfoot>		<video>	

- Luôn bắt đầu trên một dòng mới và trình duyệt sẽ tự động thêm một số khoảng trắng (lẻ) trước và sau phần tử.
- Luôn chiếm toàn bộ chiều rộng có sẵn (kéo dài sang trái và phải hết mức có thể).
- Hai phần tử khối thường được sử dụng là: `<p>` và `<div>`

1.2 Form & Validation

- Thẻ `<form>` được dùng để tạo một form cho người dùng nhập liệu
- Để nhập thì ta có thể `<input>` và đi đôi là các attributes, một số attributes của thẻ `input`:

Type	Description
<code><input type="text"></code>	Displays a single-line text input field
<code><input type="radio"></code>	Displays a radio button (for selecting one of many choices)
<code><input type="checkbox"></code>	Displays a checkbox (for selecting zero or more of many choices)
<code><input type="submit"></code>	Displays a submit button (for submitting the form)
<code><input type="button"></code>	Displays a clickable button

Form Attributes

Action attribute: khi form được gửi đi thì sẽ thực hiện hành động gì đó, ví dụ hành động:

`<form action="/action_page.php">`

```

<!DOCTYPE html>
<html>
<body>

<h2>HTML Forms</h2>

<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>

<p>If you click the "Submit" button, the form-data will be sent to a page called
"/action_page.php".</p>

</body>
</html>

```

HTML Forms

First name:

Last name:

If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".

Target Attributes

Target attribute: chỉ định nơi hiển thị phản hồi nhận được sau khi bấm submit

Value	Description
_blank	The response is displayed in a new window or tab
_self	The response is displayed in the current window
_parent	The response is displayed in the parent frame
_top	The response is displayed in the full body of the window
framename	The response is displayed in a named iframe

Đó là 2 cái phổ biến, ngoài ra còn có **Method Attribute** (chỉ định phương thức http được sử dụng), **Autocomplete Attribute** (bật/tắt tự động điền), **Novalidate Attribute** (boolean attribute)

Bảng tổng hợp attributes của form

Attribute	Description
<u>accept-charset</u>	Specifies the character encodings used for form submission
<u>action</u>	Specifies where to send the form-data when a form is submitted
<u>autocomplete</u>	Specifies whether a form should have autocomplete on or off
<u>enctype</u>	Specifies how the form-data should be encoded when submitting it to the server (only for method="post")
<u>method</u>	Specifies the HTTP method to use when sending form-data
<u>name</u>	Specifies the name of the form
<u>novalidate</u>	Specifies that the form should not be validated when submitted
<u>rel</u>	Specifies the relationship between a linked resource and the current document
<u>target</u>	Specifies where to display the response that is received after submitting the form

Form Elements

<input>: Thẻ <input> được sử dụng để tạo một trường nhập liệu trong biểu mẫu. Nó có nhiều loại (type) khác nhau như text, password, checkbox, radio, date, email, v.v. Tùy thuộc vào loại, <input> có thể cho phép người dùng nhập văn bản, chọn một tùy chọn, hoặc thực hiện một hành động khác.

<label>: Thẻ <label> được sử dụng để tạo một nhãn cho một trường nhập liệu. Nó giúp kết nối trực tiếp với trường nhập liệu bằng việc sử dụng thuộc tính "for" hoặc bọc trường nhập liệu bên trong thẻ <label>. Việc sử dụng <label> cải thiện khả năng tương tác và truy cập cho người dùng.

<select>: Thẻ <select> tạo một danh sách thả xuống (dropdown) cho người dùng chọn một hoặc nhiều tùy chọn. Các tùy chọn được định nghĩa bên trong thẻ <select> bằng cách sử dụng thẻ <option>.

<textarea>: Thẻ <textarea> tạo một ô văn bản đa dòng để người dùng nhập nhiều dòng văn bản. Nội dung nhập vào trong <textarea> có thể được đọc và gửi dữ liệu tới máy chủ.

<button>: Thẻ <button> tạo một nút bấm trên biểu mẫu, cho phép người dùng thực hiện một hành động cụ thể khi nhấn vào. Hành động này có thể được xử lý bằng JavaScript hoặc gửi dữ liệu lên máy chủ.

<fieldset>: Thẻ <fieldset> tạo một nhóm các trường nhập liệu liên quan thành một nhóm logic. Nó thường được sử dụng với <legend> để cung cấp tiêu đề cho nhóm trường nhập liệu.

<legend>: Thẻ <legend> được sử dụng để định nghĩa tiêu đề cho một nhóm các trường nhập liệu trong <fieldset>. Tiêu đề giúp mô tả mục đích hoặc thông tin liên quan đến nhóm trường nhập liệu.

<datalist>: Thẻ <datalist> định nghĩa một danh sách các tùy chọn cho một trường nhập liệu <input>. Nó cung cấp một danh sách gợi ý cho người dùng khi họ bắt đầu nhập vào trường.

<output>: Thẻ <output> được sử dụng để hiển thị kết

Form Validation

Javascript cung cấp một cách để **validate form** trên máy khách trước khi gửi nó đến máy chủ. **Form Validation** thường thực hiện hai chức năng:

- **Basic Validation:** Trước hết, biểu mẫu phải được kiểm tra để đảm bảo rằng tất cả các trường bắt buộc đã được điền vào. Nó sẽ chỉ yêu cầu một vòng lặp qua từng trường trong biểu mẫu và kiểm tra dữ liệu.
- **Data Format Validation:** Thứ hai, dữ liệu được nhập vào phải được kiểm tra về hình thức và giá trị chính xác. Mã của bạn phải bao gồm logic thích hợp để kiểm tra tính đúng đắn của dữ liệu.

Các cách Validate Form

Về cơ bản, có 3 cách để **Validate Form**:

- Dữ liệu của **form** sẽ được gửi tới **server** (máy chủ), và việc xác thực (validate) sẽ được thực hiện tại phía máy chủ.

- Dữ liệu của **form** sẽ được xác thực tại phía **client** bằng cách sử dụng **Javascript**, điều này giúp **server** không phải làm việc quá nhiều, và tăng hiệu năng cho ứng dụng.
- Sử dụng cả 2 phương thức trên để xác thực **form**.

Ví dụ:

Basic Validation

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Validation</h2>

<p>Please input a number between 1 and 10:</p>

<input id="numb">

<button type="button" onclick="myFunction()">Submit</button>

<p id="demo"></p>

<script>
function myFunction() {
  // Get the value of the input field with id="numb"
  let x = document.getElementById("numb").value;
  // If x is Not a Number or less than one or greater than 10
  let text;
  if (isNaN(x) || x < 1 || x > 10) {
    text = "Input not valid";
  } else {
    text = "Input OK";
  }
  document.getElementById("demo").innerHTML = text;
}
</script>

</body>
</html>
```

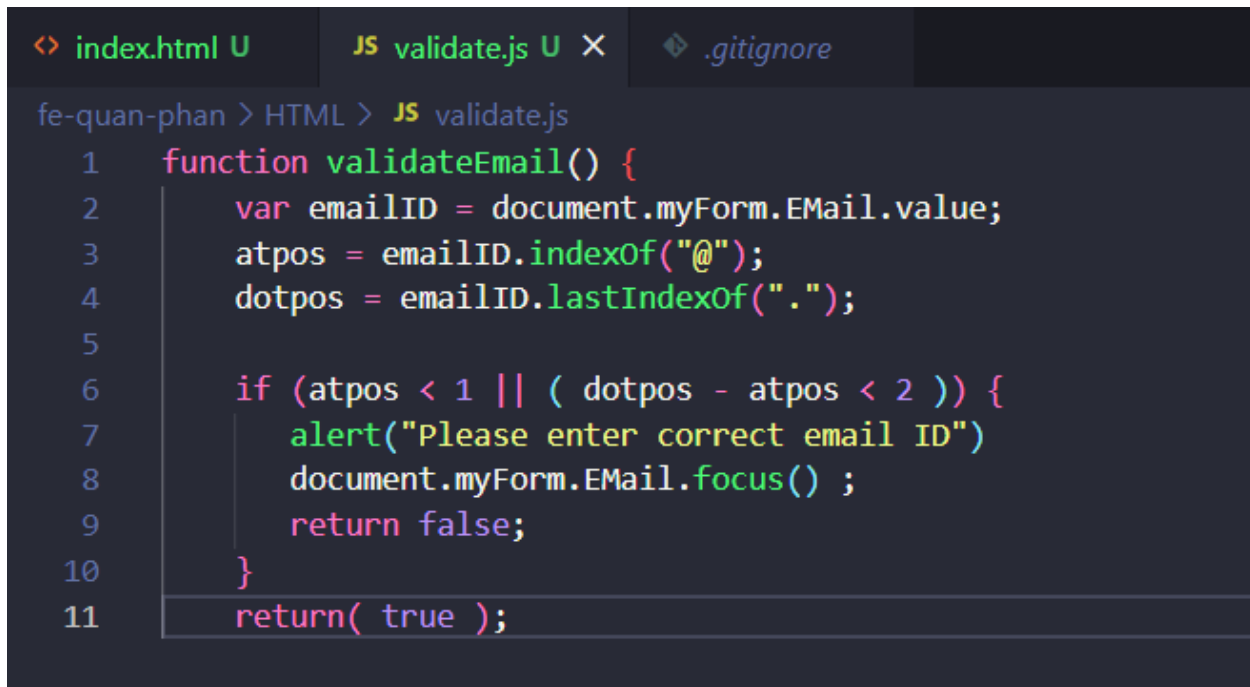
Automatic HTML Form Validation

→ Cái này là tự động ràng buộc bắt người dùng phải nhập vào, cái này ta sử dụng **required attribute**

```
<form action="/action_page.php" method="post">
  <input type="text" name="fname" required>
  <input type="submit" value="Submit">
</form>
```

Data Format Validation

Là quá trình đảm bảo đầu vào từ người dùng là chính xác và hợp lệ. Nó bao gồm kiểm tra các yêu cầu bắt buộc, định dạng ngày tháng và kiểu dữ liệu. Việc kiểm tra dữ liệu có thể thực hiện trên máy chủ sau khi gửi dữ liệu hoặc trên trình duyệt trước khi gửi đi. Mục đích chính là đảm bảo dữ liệu từ người dùng là đúng đắn và phù hợp.

A screenshot of a code editor with three tabs: 'index.html', 'JS validate.js', and '.gitignore'. The 'JS validate.js' tab is active, showing a JavaScript function named 'validateEmail'. The function takes the value from a form field with ID 'Email', finds the positions of '@' and '.' characters, and checks if the email format is valid. If invalid, it shows an alert and returns false; otherwise, it returns true.

```
fe-quan-phan > HTML > JS validate.js
1  function validateEmail() {
2      var emailID = document.myForm.Email.value;
3      atpos = emailID.indexOf("@");
4      dotpos = emailID.lastIndexOf(".");
5
6      if (atpos < 1 || ( dotpos - atpos < 2 )) {
7          alert("Please enter correct email ID")
8          document.myForm.Email.focus() ;
9          return false;
10     }
11     return( true );
```

Ví dụ trên cho thấy cách xác thực địa chỉ email đã nhập. Địa chỉ email phải chứa ít nhất dấu '@' và dấu chấm (.). Ngoài ra, '@' không được là ký tự đầu tiên của địa chỉ email và dấu chấm cuối cùng ít nhất phải là một ký tự sau dấu '@'.

HTML Constraint Validation

- Ràng buộc xác thực **HTML Input Attributes**
- Ràng buộc xác thực **CSS Pseudo Selectors**
- Ràng buộc xác thực **DOM Properties and Methods**

1. Ràng buộc xác thực HTML Input Attributes

Một vài loại phần tử `<input>` mới được giới thiệu trong HTML5 có các thuộc tính (attribute) đặc biệt giúp trình duyệt biết cách để validate dữ liệu của nó một cách tự động. Dưới đây là danh sách một vài thuộc tính như vậy:

- `disabled`: Chỉ định rằng phần tử Input này sẽ bị vô hiệu hóa (disabled).
- `max`: Chỉ định giá trị lớn nhất của một phần tử Input
- `min`: Chỉ định giá trị nhỏ nhất của một phần tử Input
- `pattern`: Chỉ định mẫu giá trị của một phần tử Input
- `required`: Chỉ định rằng trường đầu vào là bắt buộc. Người dùng phải nhập dữ liệu.
- `type`: Chỉ định loại của một phần tử Input

2. Ràng buộc xác thực CSS Pseudo Selectors

- `:disabled`: Chọn các phần tử đầu vào có thuộc tính "disabled" được chỉ định
- `:invalid`: Chọn các phần tử đầu vào có giá trị không hợp lệ
- `:optional`: Chọn các phần tử đầu vào không có thuộc tính "required" được chỉ định
- `:required`: Chọn các phần tử đầu vào với thuộc tính "required" được chỉ định
- `:valid`: Chọn các phần tử đầu vào có giá trị hợp lệ

Constraint Validation HTML Input Attributes

Attribute	Description
disabled	Specifies that the input element should be disabled
max	Specifies the maximum value of an input element
min	Specifies the minimum value of an input element
pattern	Specifies the value pattern of an input element
required	Specifies that the input field requires an element
type	Specifies the type of an input element

Constraint Validation CSS Pseudo Selectors

Selector	Description
<code>:disabled</code>	Selects input elements with the "disabled" attribute specified
<code>:invalid</code>	Selects input elements with invalid values
<code>:optional</code>	Selects input elements with no "required" attribute specified
<code>:required</code>	Selects input elements with the "required" attribute specified
<code>:valid</code>	Selects input elements with valid values

1.3 Seo Basic

- SEO viết tắt của Search Engine Optimization, là những kỹ thuật/mẹo giúp trang web của bạn có thể được tìm thấy bởi các Search engine như là Google.

- Mục tiêu của SEO là tăng cường hiển thị của trang web trong kết quả tìm kiếm tự nhiên (organic search) và thu hút lượng truy cập từ khách hàng tiềm năng. Điều này đạt được bằng cách tối ưu hóa cấu trúc, nội dung và các yếu tố kỹ thuật khác của trang web để phù hợp với tiêu chí và thuật toán của các công cụ tìm kiếm.

- Có rất nhiều công cụ tìm kiếm nhưng google là phổ biến nhất

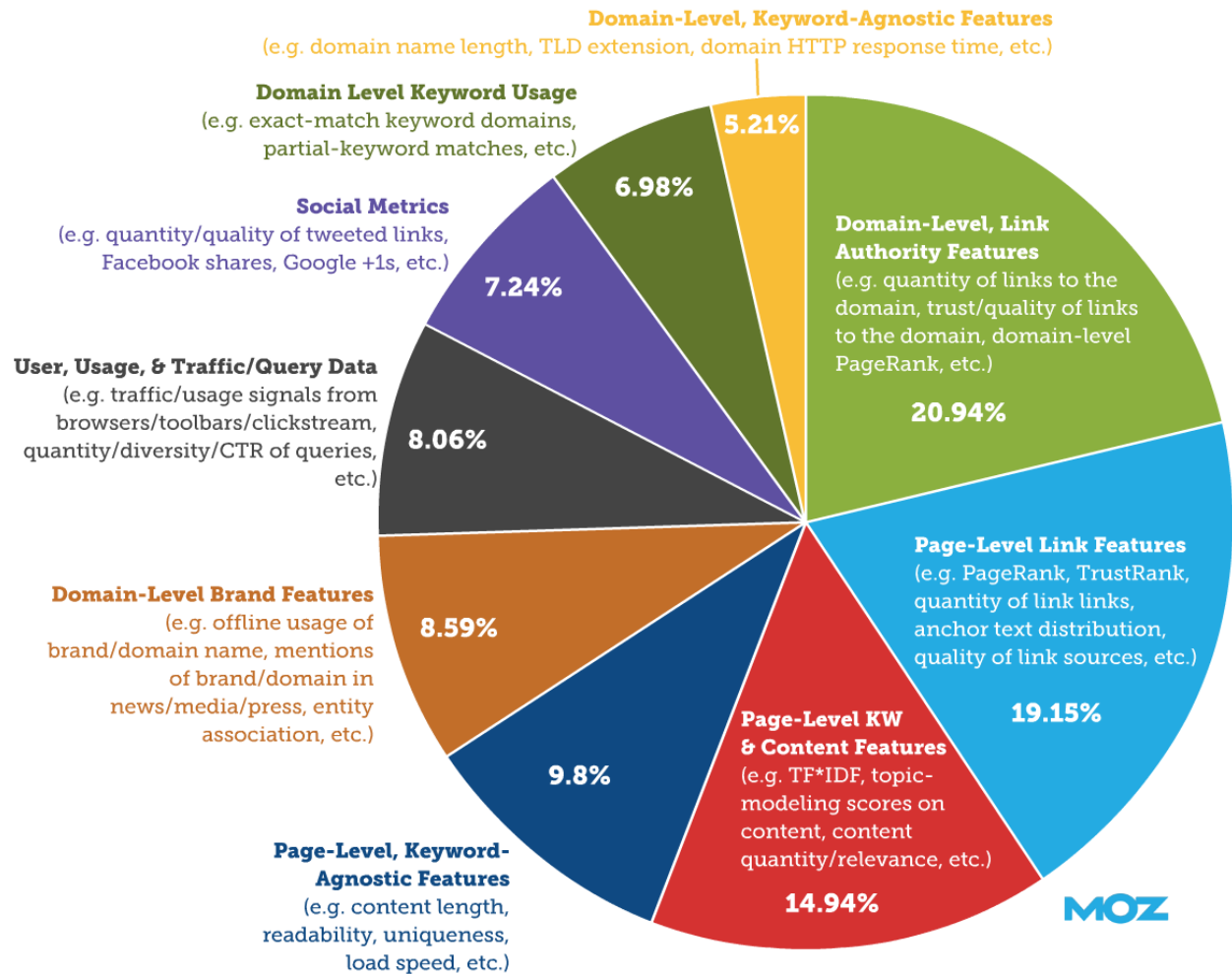
Core Search Entity	Share (%) Jan-15	Share (%) Feb-15	Point Change
Total Explicit Core Search	100.0%	100.0%	0
Google Sites	64.4%	64.5%	0.1
Microsoft Sites	19.7%	19.8%	0.1
Yahoo Sites	13.0%	12.8%	-0.2
Ask Network	1.8%	1.8%	0.0
AOL, Inc.	1.1%	1.1%	0.0

Google update rất nhiều vào thuật toán tìm kiếm của họ trong những năm gần đây. Thuật toán này cực kì phức tạp :

- Google sẽ tìm kiếm những pages chứa chất lượng cao và có lượng thông tin "liên quan" với câu hỏi của người search
- Google quyết định định nghĩa việc "liên quan" bằng kỹ thuật crawling nội dung và đánh giá(bằng thuật toán) liệu nội dung đó có liên quan với keyword user search hay ko
- Google quyết định chất lượng của 1 trang bằng một số phương tiện, một trong số đó chính là số lượng các site khác liên kết tới page/site của bạn.

Weighting of Thematic Clusters of Ranking Factors in Google

(based on survey responses by 128 SEO professionals in June 2013)



Tối ưu Keyword

Khối lượng tìm kiếm – Yếu tố đầu tiên cần xem xét là có bao nhiêu người đang thực sự tìm kiếm một từ khóa nhất định.

Số lượng tìm kiếm: Đánh giá mức độ người dùng tìm kiếm từ khóa đó.

Liên quan đến lĩnh vực: Xác định tính liên quan của từ khóa đến ngành kinh doanh của bạn.

Mức độ cạnh tranh: Xem xét chi phí và khả năng thành công khi sử dụng từ khóa đó.

Free Keyword Tool

Keyword or website URL

adopt a dog

Industry optional

Pets & Animals

Location optional

United States

NEW SEARCH

This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

Showing 25 of 500 keywords for **adopt a dog**

EMAIL ALL MY KEYWORDS

Keywords	Search volume	CPC	Competition	Search volume	CPC	Competition
rescue dogs	90,500	\$1.77	High	6,240	\$0.05	High
dog rescue near me	165,000	\$2.19	High	1,500	\$0.05	High
dog shelters near me	201,000	\$2.46	High	3,620	\$0.05	High
dogs for adoption near me	301,000	\$1.90	High	9,480	\$0.05	High
dog pound	90,500	\$1.85	High	3,020	\$0.05	High
puppies for adoption near me	110,000	\$1.44	High	50	\$0.05	High
puppies for adoption	110,000	\$1.23	High	210	\$0.14	High
boxer puppies for sale	40,500	\$0.67	High	80	\$1.36	High

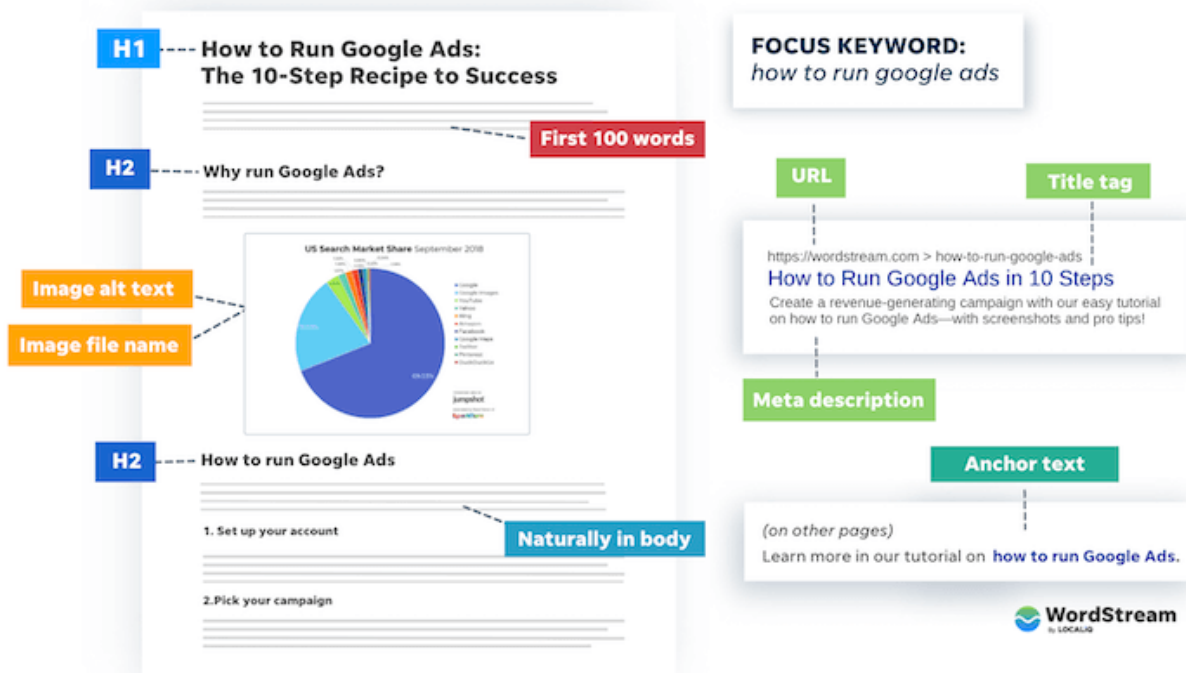
→ Nói chung phần này chủ yếu nói về cách tìm keyword, rồi đánh giá độ cạnh tranh của keyword đó, trên mạng có nhiều web để có thể thống kê keyword, đánh giá % rồi chọn lựa keyword phù hợp thôi

Tối ưu On-Page

Sau khi bạn có danh sách từ khóa, bước tiếp theo là thực sự triển khai các từ khóa được nhắm mục tiêu vào nội dung trang web của bạn.

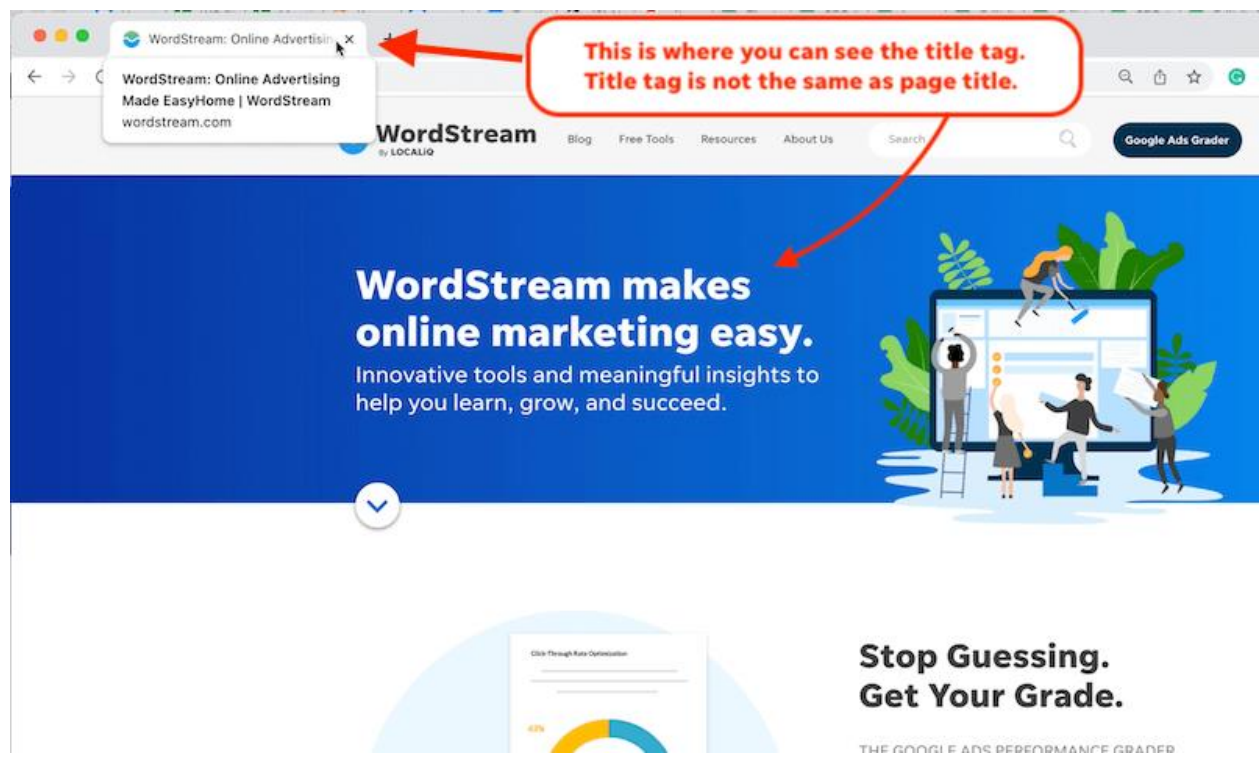
Mỗi trang trên trang web của bạn nên nhắm mục tiêu một thuật ngữ cốt lõi và một “rủ” các thuật ngữ có liên quan. Đây là giao diện của một trang được tối ưu hóa cho SEO trên trang:

ON-PAGE SEO: KEYWORD PLACEMENT



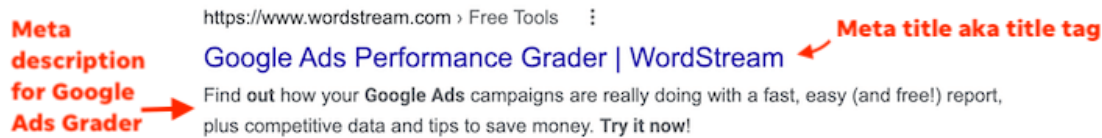
Title Tag

Đặt keyword lên title tag, độ dài khoảng 55-60 từ thì sẽ phù hợp so với rule



Meta Descriptions

→ tăng traffic của web của lên.



Body Content

1. Nội dung đủ và độc đáo: Tránh việc có nhiều trang với nội dung ngắn và lặp lại chỉ thay đổi title tag. Hãy tập trung vào việc tạo nội dung đa dạng và giảm thiểu nội dung trùng lặp, để tăng tính độc nhất và chất lượng của trang web.
2. Hấp dẫn người dùng: Google đánh giá cao trải nghiệm người dùng và thời gian người dùng dừng lại trên trang web của bạn. Tạo nội dung có khả năng thu hút người dùng, ví dụ như câu hỏi/trả lời, và tối ưu tốc độ tải trang để người dùng không chán nản và tìm kiếm trang khác.
3. Khả năng chia sẻ: Xem xét việc tạo nội dung mà người dùng sẵn sàng chia sẻ và liên kết đến trang web của bạn. Điều này có thể đạt được bằng cách cung cấp thông tin hữu ích, giải đáp câu hỏi của người dùng và đáp ứng nhu cầu của họ.

Alt Attribute

Cái này là thuộc tính của image, mỗi khi ảnh lỗi không mở lên thì sẽ có alt hiện lên để dự phòng

→ công cụ tìm kiếm không đọc image, mà sẽ đọc thông qua alt nên là phải ghi alt một cách để công cụ hiểu được, ảnh hưởng tới traffic web rất nhiều

URL Structure

→ Tạo web theo url ngắn gọn súc tích, thân thiện giúp dễ dàng share, copy → thân thiện hơn

Schema & Markup

Đánh dấu lược đồ không làm cho trang của bạn hiển thị cao hơn trong kết quả tìm kiếm (hiện tại nó không phải là yếu tố xếp hạng). Nó cung cấp cho danh sách của bạn một số

"bất động sản" bổ sung trong kết quả tìm kiếm, giống như cách tiện ích mở rộng quảng cáo làm cho quảng cáo Google Ads

Hotels Near Portland International Airport ... - Marriott
www.marriott.com/hotels/.../pwmlo-courtyard-portl... ▾ Marriott International ▾
★★★★★ Rating: 4.6 - 78 reviews
Our **hotel** welcomes you with stylish rooms and close proximity to Portland International Jetport. ... 100 Southborough Drive South **Portland Maine** 04106 USA.

Portland, ME, Waterfront Hotel | Visit Residence Inn ...
www.residenceinndowntownportland.com/ ▾
The Residence Inn by **Marriott**® Portland Downtown/Waterfront offers traditional, ...
Discover Comfort and Convenience at Our **Portland, Maine, Waterfront Hotel**.

Portland Marriott at Sable Oaks (South Portland, Maine ...
www.tripadvisor.com > ... > **South Portland Hotels** ▾ TripAdvisor LLC ▾
★★★★★ Rating: 4 - 388 reviews - Price range: \$\$
#4 of 12 **Hotels** in South Portland. Certificate of Excellence. **Hotel** website. **Hotel** deals.
Hotel packages. 200 Sable Oaks Drive, South **Portland, ME** 04106.

Các vấn đề kỹ thuật chung trong SEO

Tốc độ website

sử dụng tool <https://developers.google.com/speed/pagespeed/insights/> của Google để kiểm tra tốc độ website

Thân thiện với mobile?

Sử dụng tool <https://www.google.com/webmasters/tools/mobile-friendly/>

Header response

cần đảm bảo response của bạn trả về phải đúng chuẩn. Như vậy thì công cụ tìm kiếm mới có thể biết được web của bạn đang tồn tại hay không. Hãy sử dụng các công cụ check header để kiểm tra vấn đề này.

Redirects

Việc thao tác redirect cho website một cách không hợp lý sẽ ảnh hưởng nghiêm trọng đến kết quả search của bạn. Ví dụ nội dung trên example.com/page của bạn đang được traffic khá ổn trên công cụ tìm kiếm, và bạn không muốn chuyển tất cả mọi nội dung sang bên example.com/different-url/newpage.html vì việc này có thể khiến bạn bị mất traffic trong một khoảng time ngắn hoặc dài hạn. Nếu bạn cần phải di chuyển nội dung, bạn sẽ cần phải chắc chắn rằng bạn sẽ redirect nội dung mãi mãi hay chỉ là ngắn hạn tạm thời.

Duplicate Content

Hãy dùng tool Webmaster Tools để kiểm tra đánh giá lượng lặp về nội dung trong web của bạn.

XML Sitemap

XML sitemap sẽ hỗ trợ Google, Bing hiểu về site của bạn và có thể tra cứu tất cả nội dung web của bạn. Sử dụng tool này để tạo XML sitemap <https://www.xml-sitemaps.com/>

Robots.txt, Meta NoIndex, & Meta NoFollow

Robots.txt, Meta NoIndex và Meta NoFollow là các phương pháp được sử dụng trong SEO để chỉ định cho công cụ tìm kiếm cách xử lý nội dung cụ thể trên trang web. Dưới đây là mô tả ngắn về mỗi phương pháp:

+ Robots.txt: Đây là một tệp tin có tên là "robots.txt" được đặt tại thư mục gốc của trang web (ví dụ: yoursite.com/robots.txt). Tệp tin này cho phép bạn chỉ định cho các công cụ tìm kiếm cách xử lý nội dung trên trang web của bạn. Bằng cách sử dụng robots.txt, bạn có thể cho biết cho công cụ tìm kiếm không nên duyệt qua một phần cụ thể của trang web, chẳng hạn như không duyệt các trang trong một phần mục đích nội bộ hoặc không duyệt các trang bản nháp. Điều này giúp bạn kiểm soát việc xếp hạng và hiển thị của trang web trên các công cụ tìm kiếm.

+ Meta NoIndex: Meta NoIndex là một thẻ HTML được thêm vào phần đầu của một trang web để chỉ định cho công cụ tìm kiếm không nên lập chỉ mục (index) nội dung của trang đó. Khi công cụ tìm kiếm gặp thẻ Meta NoIndex, nó sẽ bỏ qua việc lập chỉ mục trang và không hiển thị trong kết quả tìm kiếm. Thẻ này thường được sử dụng cho các trang bản nháp, trang đăng nhập, hoặc các trang không mong muốn xuất hiện trong kết quả tìm kiếm.

+ Meta NoFollow: Meta NoFollow cũng là một thẻ HTML được thêm vào phần đầu của một trang web, nhưng chức năng của nó khác so với Meta NoIndex. Thẻ Meta NoFollow chỉ định cho công cụ tìm kiếm không theo dõi (follow) các liên kết trên trang đó. Điều này có nghĩa là công cụ tìm kiếm sẽ không theo dõi các liên kết và không xem chúng là tài nguyên hữu ích để xếp hạng trang. Thẻ này thường được sử dụng để ngăn chặn công cụ tìm kiếm theo dõi các liên kết không mong muốn hoặc không đáng tin cậy trên trang web.

Theo dõi, đánh giá kết quả SEO

Keyword rankings: lên các web rồi check ranking của keyword đó xem nó ở rank nào

Organic traffic: cho biết số lượng users đến trang web của mình, vào những trang nhỏ nào trong web mình

→ Sử dụng Google Analytic để theo dõi Organic traffic

II. CSS

2.1. Basic

- Là viết tắt của Cascading Style Sheets

- được sử dụng để định dạng bố cục của một trang web.

- Với CSS, ta có thể kiểm soát màu sắc, phông chữ, kích thước của văn bản, khoảng cách giữa các phần tử, cách các phần tử được định vị và trình bày, hình nền hoặc màu nền nào sẽ được sử dụng, cách hiển thị khác nhau cho các thiết bị và kích thước màn hình khác nhau

Using CSS

Ta có thể thêm css vào html bằng 3 cách sau:

1. **Inline:** sử dụng style attribute trong html elements
2. **Internal:** sử dụng thẻ <style> trong phần <head> trong html
3. **External:** sử dụng thẻ <link> là link tới một file css nằm ngoài html

Ví dụ cho Inline Css

```
<h1 style="color:blue;">A Blue Heading</h1>
```

```
<p style="color:red;">A red paragraph.</p>
```

Ví dụ cho Internal Css

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1 {color: blue;}
p {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Khai báo css trong thẻ
<style> nằm trong <head>

Ví dụ cho External Css

Gọi tới file css có tên là
styles.css trong thẻ <link>
nằm trong <head>

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```
body {
background-color: powderblue;
}
h1 {
color: blue;
}
p {
color: red;
}
```

file styles.css

2.2 Making Layouts

CSS Display

- Thuộc tính có tên là **display**, dùng để kiểm soát layout
- Dùng để đổi định dạng của thẻ, như ở trên đã nói 2 cái là các thẻ **inline** và các thẻ **block**, thì thuộc tính display này chuyển đổi các định dạng kiểu đó

Ví dụ: để tạm ẩn 1 cái gì đó đi thì sử dụng display:none

→ ví dụ thẻ <div> gì đó là 1 thẻ block, bây giờ ta muốn đổi sang kiểu inline thì ta sử dụng display:inline

Ngoài display ra thì còn có visibility

| Property | Description |
|-------------------|---|
| <u>display</u> | Specifies how an element should be displayed |
| <u>visibility</u> | Specifies whether or not an element should be visible |

CSS Position

- Thuộc tính có tên là position, giúp định dạng cái type của position mà mình muốn, gồm có 5 giá trị là:

1. Static
2. Relative
3. Fixed
4. Absolute
5. Sticky

Position: static

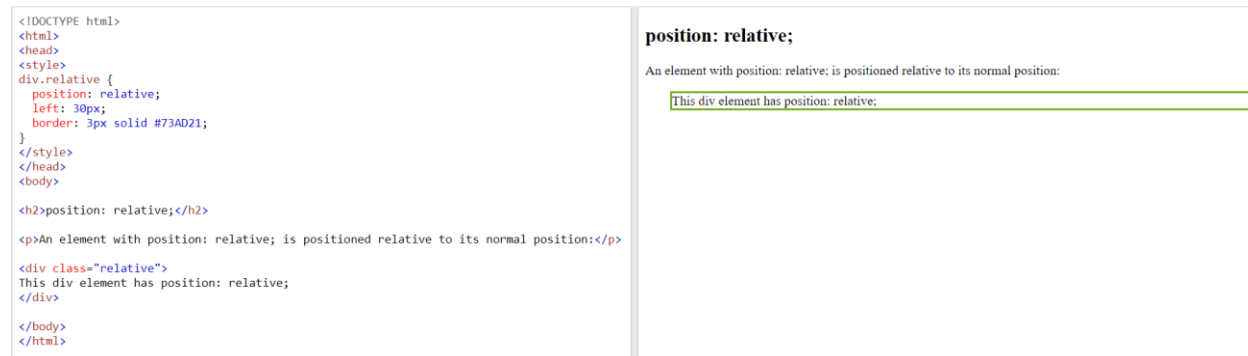
- Thuộc tính này là default khi khai báo rồi, nó đã mặc định từ đầu
- Định vị theo quy trình bình thường của trang, không bị ảnh hưởng bởi thuộc tính trên dưới trái phải và z-index
- Các thành phần sẽ nằm đúng theo thứ tự của văn bản.

```
div.static {
  position: static;
  border: 3px solid #73AD21;
}
```

→ bỏ **position:static** trong ví dụ trên đi thì nó vẫn chả thay đổi gì vì mặc định đã là static rồi

Position: relative

- Được hiểu là vị trí tương đối của các thành phần, không ảnh hưởng tới vị trí ban đầu hay các thành phần khác
- Đặt các thuộc tính top, right, bottom và left của một phần tử được định vị tương đối sẽ khiến nó bị điều chỉnh khỏi vị trí bình thường. Nội dung khác sẽ không được điều chỉnh để phù hợp với bất kỳ khoảng trống nào do phần tử để lại.



Vậy khác nhau giữa **relative** và **static** là gì ? với sự khác biệt có thể thấy qua ví dụ trên, đối với **static** thì chắc chắn sẽ **không thể** sử dụng thuộc tính left, right, top, bottom được bởi vì các thuộc tính đó không ảnh hưởng khi sử dụng static, còn với **relative** thì có thể sử dụng 4 thuộc tính đó

Position: absolute

- Có vị trí tuyệt đối, được định vị so với tổ tiên được định vị gần nhất (kiểu như dựa vào thằng cha gần nhất của nó)
- Nếu không có tổ tiên gần nhất, nó sẽ định vị theo body và có nút cuộn trang kèm theo
- Khi sử dụng thì nó sẽ bị mất cái luồng định vị và sẽ chồng lấp lên phần tử khác

Ví dụ

This <div> element has position: relative;

This <div> element has position: absolute;

Thẻ <div> trên là tổ tiên gần nhất, có thuộc tính là relative

Thẻ <div> dưới là thuộc tính absolute, nên khi sử dụng nó sẽ bị mất đi vị trí mặc định và nhảy lên chồng lên tổ tiên

Ví dụ tổng cho 3 thuộc tính mới đề cập ở trên

position: static



position: relative



Không ảnh hưởng tới bố cục xung quanh khi di chuyển ô màu xanh lá

position: absolute



với relative ở trên, vị trí của ô xanh dựa trên vị trí gốc là cái viền chấm chấm, do ở đây để top 20px và left 20px nên nó rời ra

Position: fixed

- Là một vị trí cố định, nó giúp được phần tử cố định ở một chỗ ví dụ như là khi ta cuộn trang hay làm bất cứ điều gì tác động, sử dụng được 4 thuộc tính right left top bottom

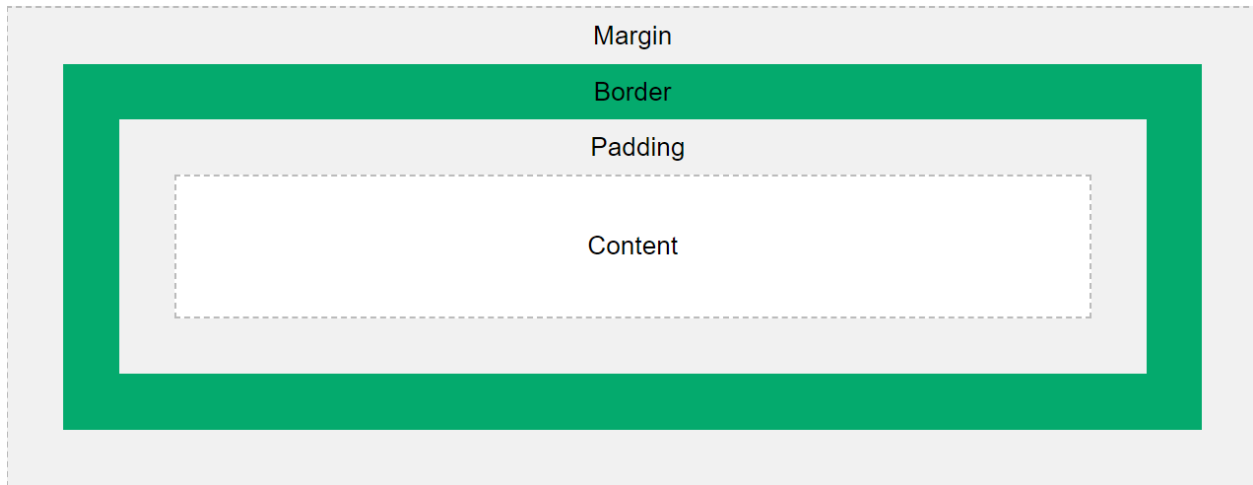


Position: Sticky

- Có vị trí dính dính lại, được định vị dựa trên vị trí cuộn trang của user, nói chung khá giống position:fixed

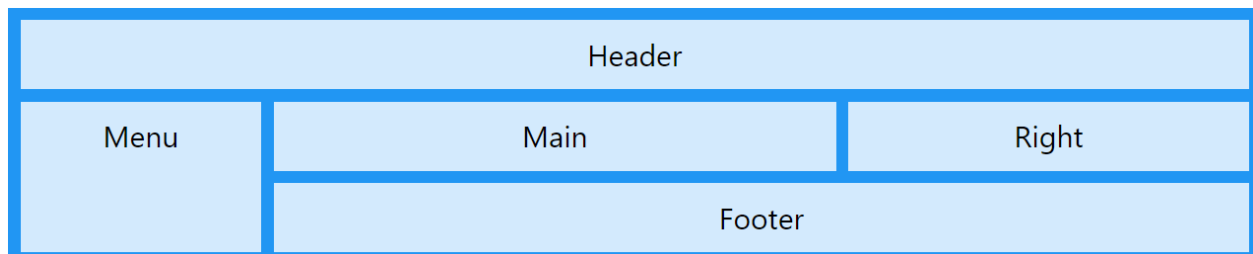


2.3 Box Model



- + **Content** - Nội dung của hộp, nơi văn bản và hình ảnh xuất hiện
- + **Padding** - Xóa một khu vực xung quanh nội dung. Phần đệm trong suốt
- + **Border** - Đường viền bao quanh Padding và Content
- + **Margin** - Xóa vùng bên ngoài đường viền. Biên độ trong suốt

2.4 CSS Grid



- Css grid giống như kiểu lúa, làm layout theo nhiều hướng nhiều chiều, linh hoạt hơn , là một hệ thống bố cục dựa trên lưới hai chiều

Grid cho Parent

- [display](#)
- [grid-template-columns](#)
- [grid-template-rows](#)
- [grid-template-areas](#)
- [grid-template](#)
- [grid-column-gap](#)
- [grid-row-gap](#)
- [grid-gap](#)
- [justify-items](#)
- [align-items](#)
- [place-items](#)
- [justify-content](#)
- [align-content](#)
- [place-content](#)
- [grid-auto-columns](#)
- [grid-auto-rows](#)
- [grid-auto-flow](#)
- [grid](#)

Grid cho Children

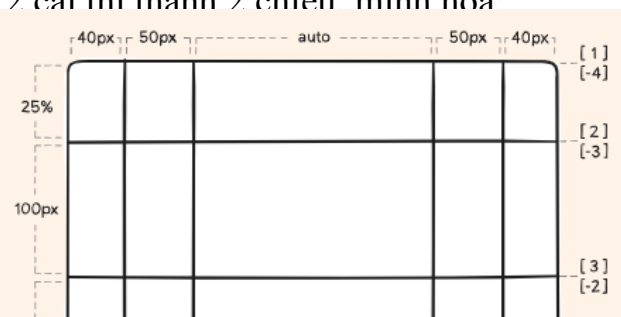
- [grid-column-start](#)
- [grid-column-end](#)
- [grid-row-start](#)
- [grid-row-end](#)
- [grid-column](#)
- [grid-row](#)
- [grid-area](#)
- [justify-self](#)
- [align-self](#)
- [place-self](#)

* Nói về parent (một số thuộc tính phổ biến)

- display thì sử dụng display grid hoặc inline-grid, cũng như cái tên, một cái block một cái inline

- grid-template-columns và grid-template-rows thì như kiểu ta chia kích thước của dòng và cột, linh động hơn và sử dụng cả 2 cái thì thành 2 chiều minh họa:

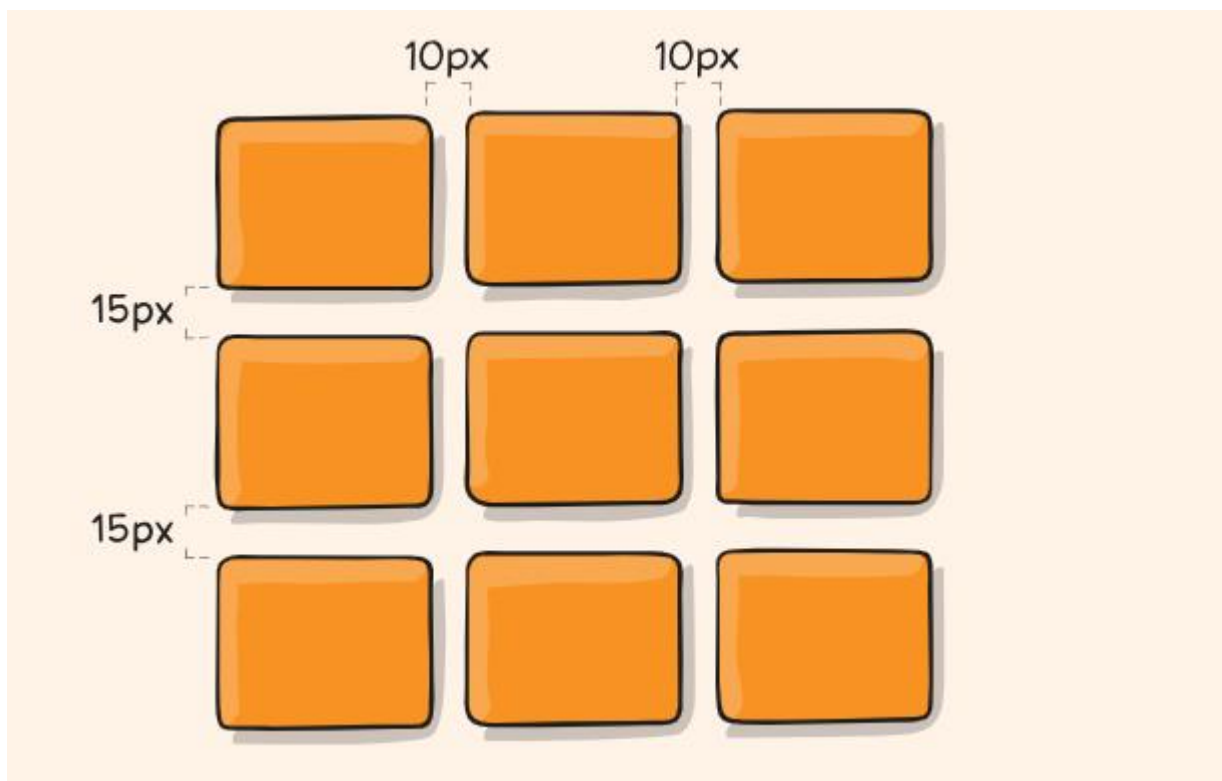
Dựa vào đây thấy rõ chia theo cột thì



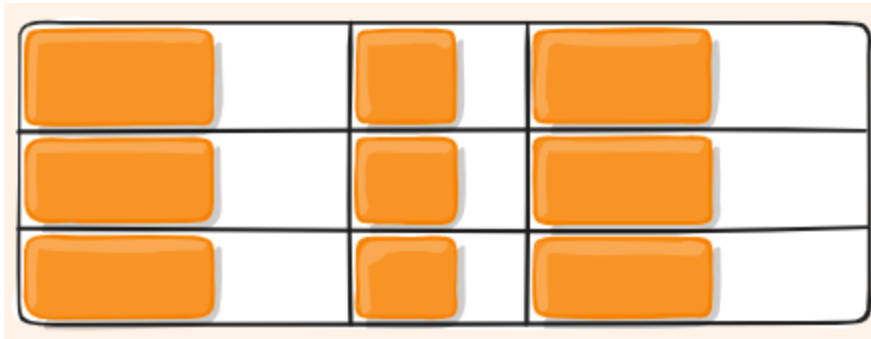
40px 50px auto 50px 40px và 5 cột,

Còn dòng thì 3 dòng với 25% 100px auto

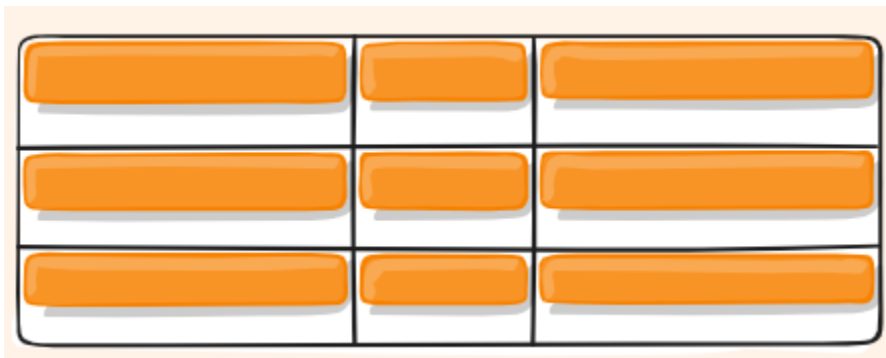
- Tiếp theo là column-gap, row-gap, grid-column-gap, grid-row-gap, kiểu như chỉ ra cái khoảng cách giữa dòng và cột, hình minh họa:



- justify-item: giống align (trục ngang) , gồm start, end, center, stretch, start thì là khởi đầu bên tay trái, center là ở giữa còn stretch là trải dài lấp hết ô



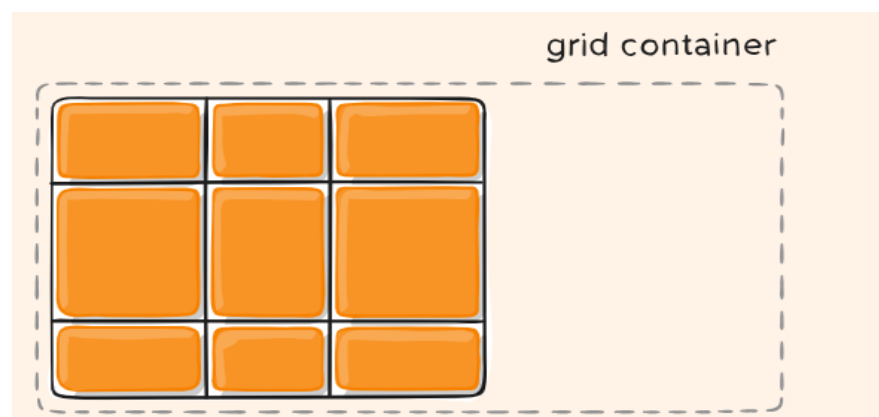
- align-item: giống align của justify-item nhưng align này theo chiều từ trên xuống, tức theo trục y (trục dọc)



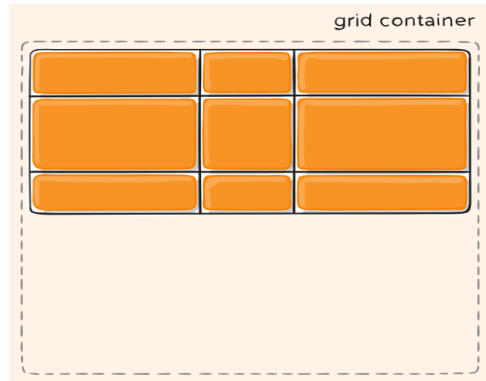
- gap thì gồm 2 giá trị, 1 là row, 1 là column truyền vô, ví dụ gap:10px 15px thì 10px là khoảng cách của các row, 15px là khoảng cách của các column

- justify-content: vùng lưới grid nhỏ hơn vùng chứa cái lưới thì ta sử dụng cái này, nó đối lập với align-items, tức là cái này sẽ theo trục ngang

Tất cả đều định dạng theo “start”



- align-content thì đối lập với justify-content thôi, thay vì trục ngang thì này là trục dọc



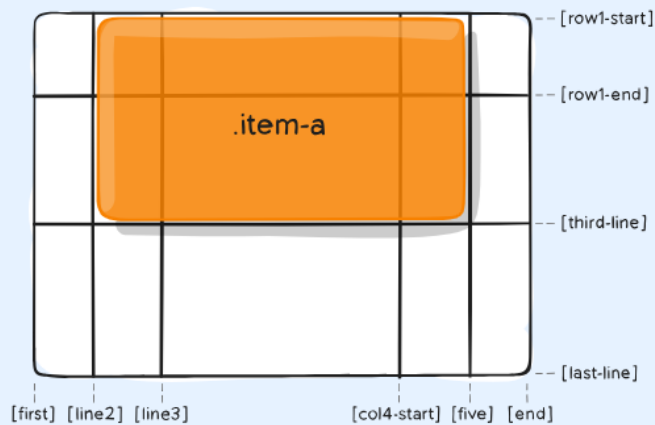
Ngoài ra còn rất nhiều thuộc tính nữa

* Nói về children

grid-column-start, grid-column-end, grid-row-start, grid-row-end: xác định vị trí của một mục lưới trong lưới bằng cách tham chiếu đến các đường lưới cụ thể

Ví dụ

```
.item-a {  
  grid-column-start: 2;  
  grid-column-end: five;  
  grid-row-start: row1-start;  
  grid-row-end: 3;  
}
```

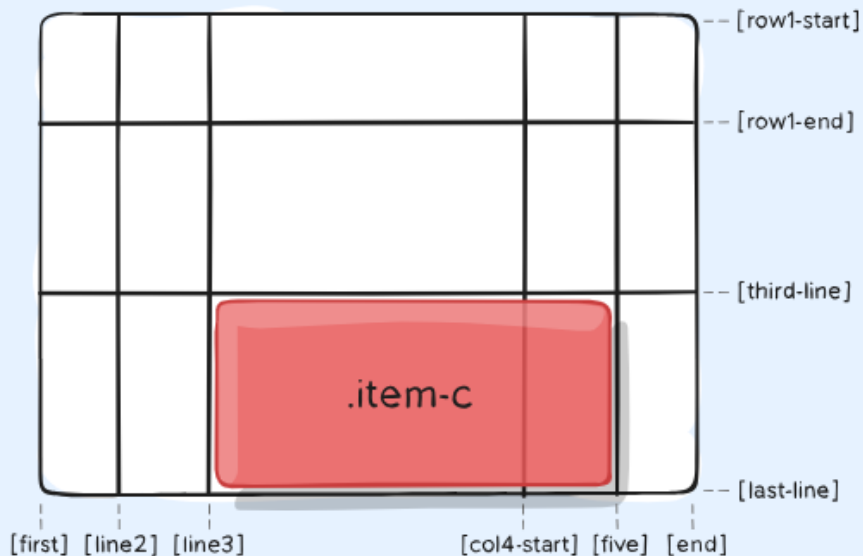


- grid-column, grid-row: viết tắt cho *grid-column-start*, *grid-column-end*, *grid-row-start*, *grid-row-end*

```
.item {  
  grid-column: <start-line> / <end-line> | <start-line> / s  
  grid-row: <start-line> / <end-line> | <start-line> / span  
}
```

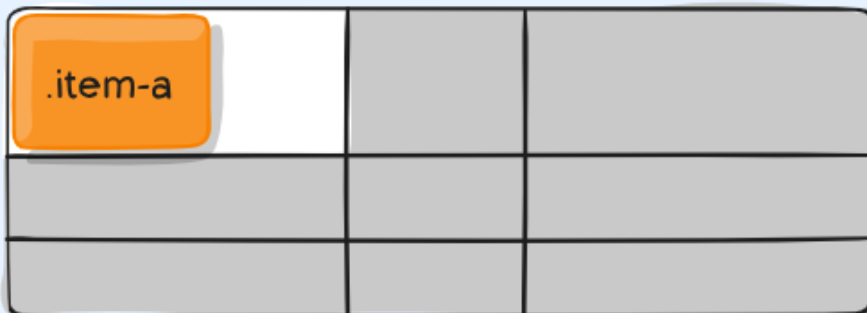
Example:

```
.item-c {  
  grid-column: 3 / span 2;  
  grid-row: third-line / 4;  
}
```



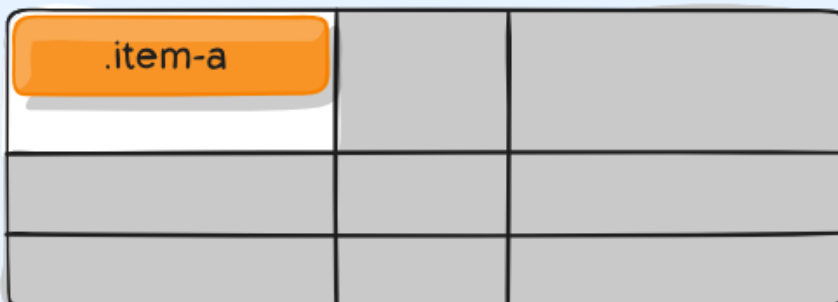
- justify-self: chỉnh cho một ô lưới con, một ô item, còn bên parent là chỉnh cho cả khối grid, cũng đều có các giá trị như start,end,center và stretch giống nhau và cái này theo trục ngang

```
.item-a {  
  justify-self: start;  
}
```



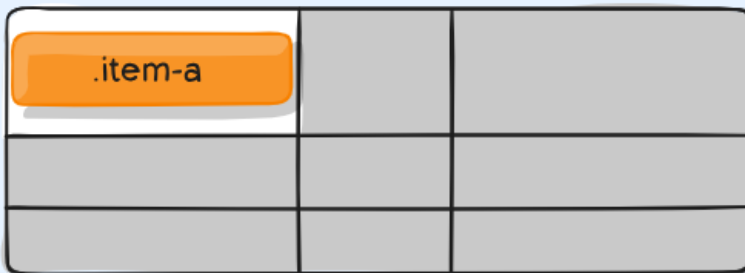
- align-self thì đối lập với justify-self và theo trục dọc

```
.item-a {  
  align-self: start;  
}
```



- place-self: có tất cả thuộc tính của justify-self và align-self

```
.item-a {  
  place-self: center stretch;  
}
```

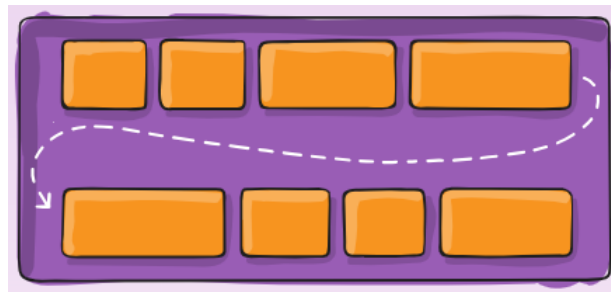


2.5 Flex box

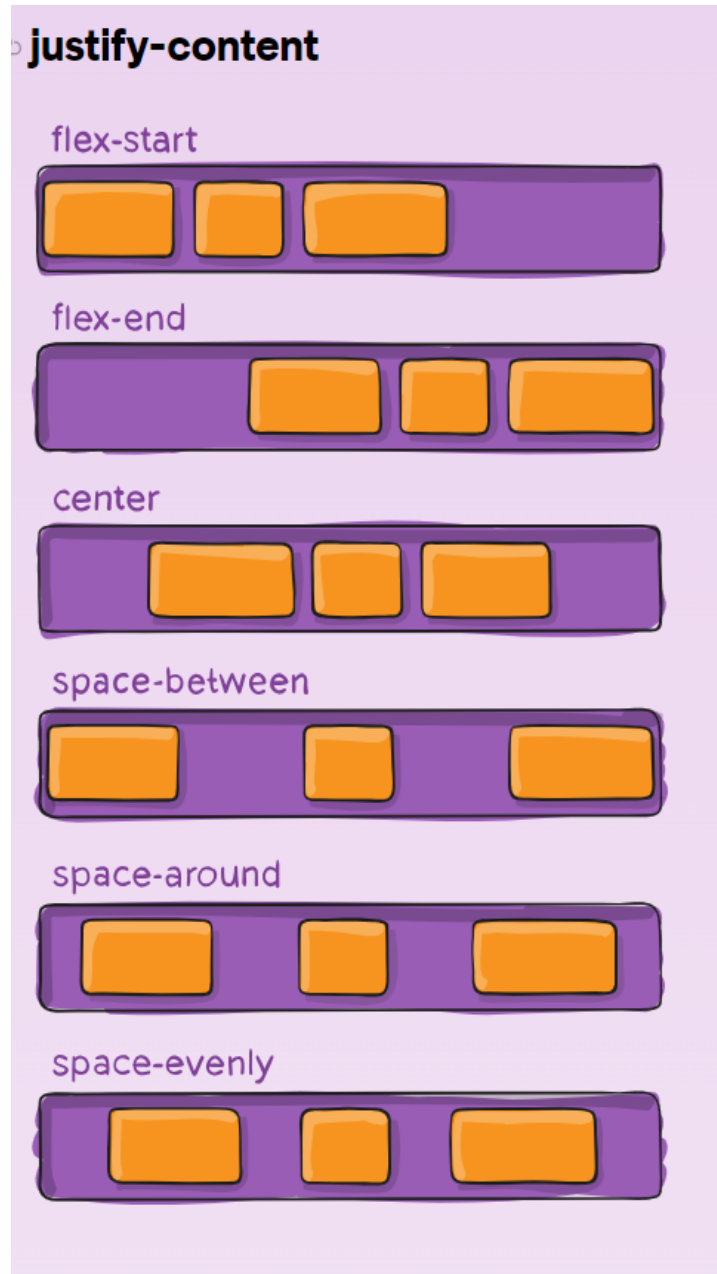
- Mục đích sử dụng là để các phần tử nằm trên một hàng hoặc một cột, trải dàn ra bố cục cho phù hợp và đỡ bị rối
- Nó cũng chia ra các properties cho parent và cho children giống css grid nhưng khác grid, không linh hoạt bằng grid nhưng lại rất ổn định cho các phần tử nằm theo hàng theo cột

* Nói về Parent

- Sử dụng display:flex để khởi tạo flex cho parent container
- flex-direction: nói về hướng của cái parent này là dọc hay ngang, ngoài ra còn có thêm column-reverse và row-reverse là để đảo ngược
- flex-wrap: mục đích như là nếu cái hàng ngang của parent của mình nó dài quá thì có cho nó xuống dòng hay không, gồm các giá trị như wrap, nowrap và wrap reverse

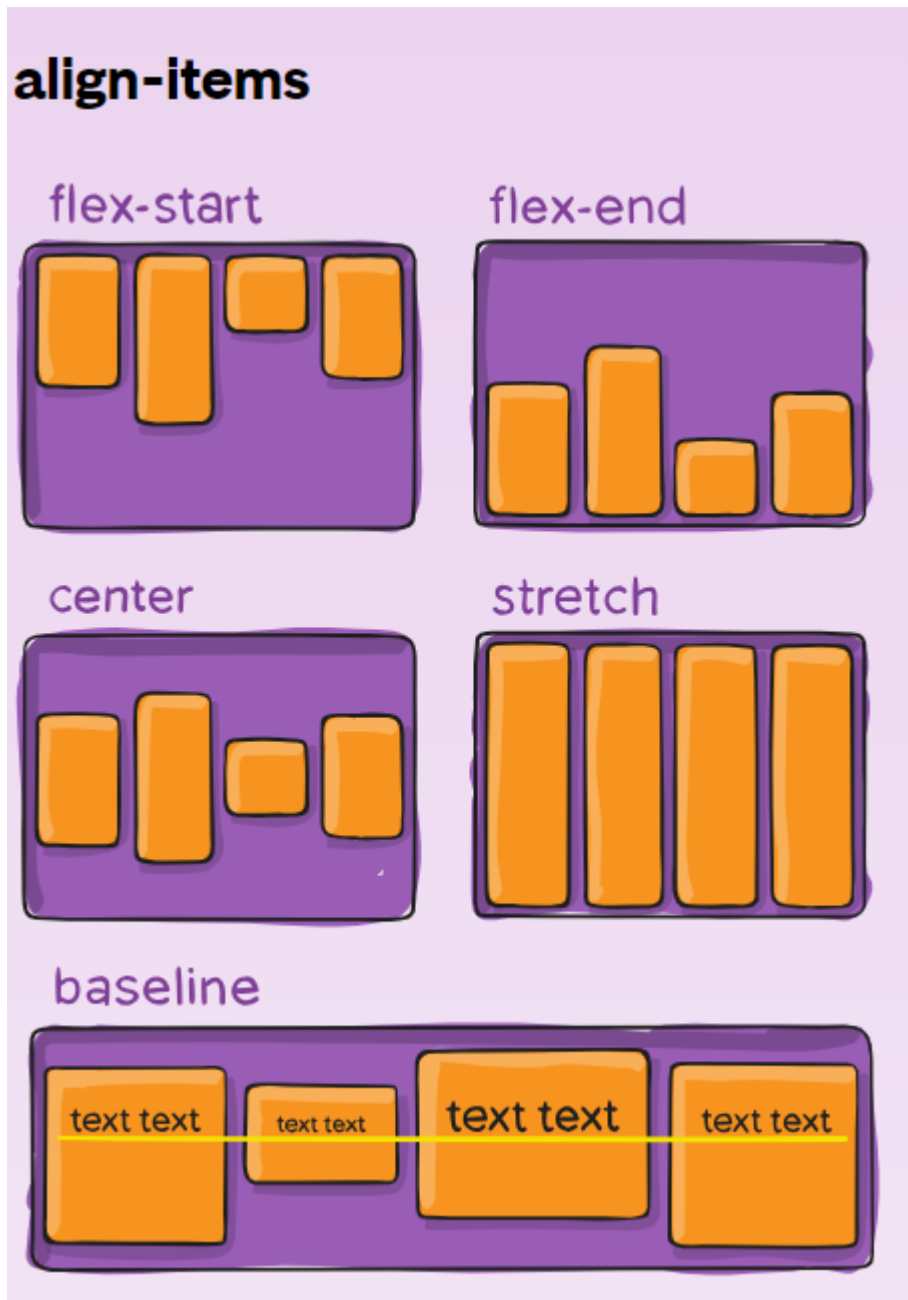


- flex-flow: viết tắt của flex-wrap và flex-direction, giá trị mặc định là nowrap, ta có thể sử dụng flex-flow:column wrap
- justify-content: căn các phần tử trong parent theo các định dạng như flex start, end, center, space-between (trải đều và 2 phần tử đầu và cuối dính sát vào lề trái lề phải), space-around (giống space-between nhưng 2 phần tử đầu không dính sát vào lề trái phải), space-evenly: các mục được phân phối sao cho khoảng cách giữa hai mục bất kỳ (và khoảng cách đến các cạnh) bằng nhau.

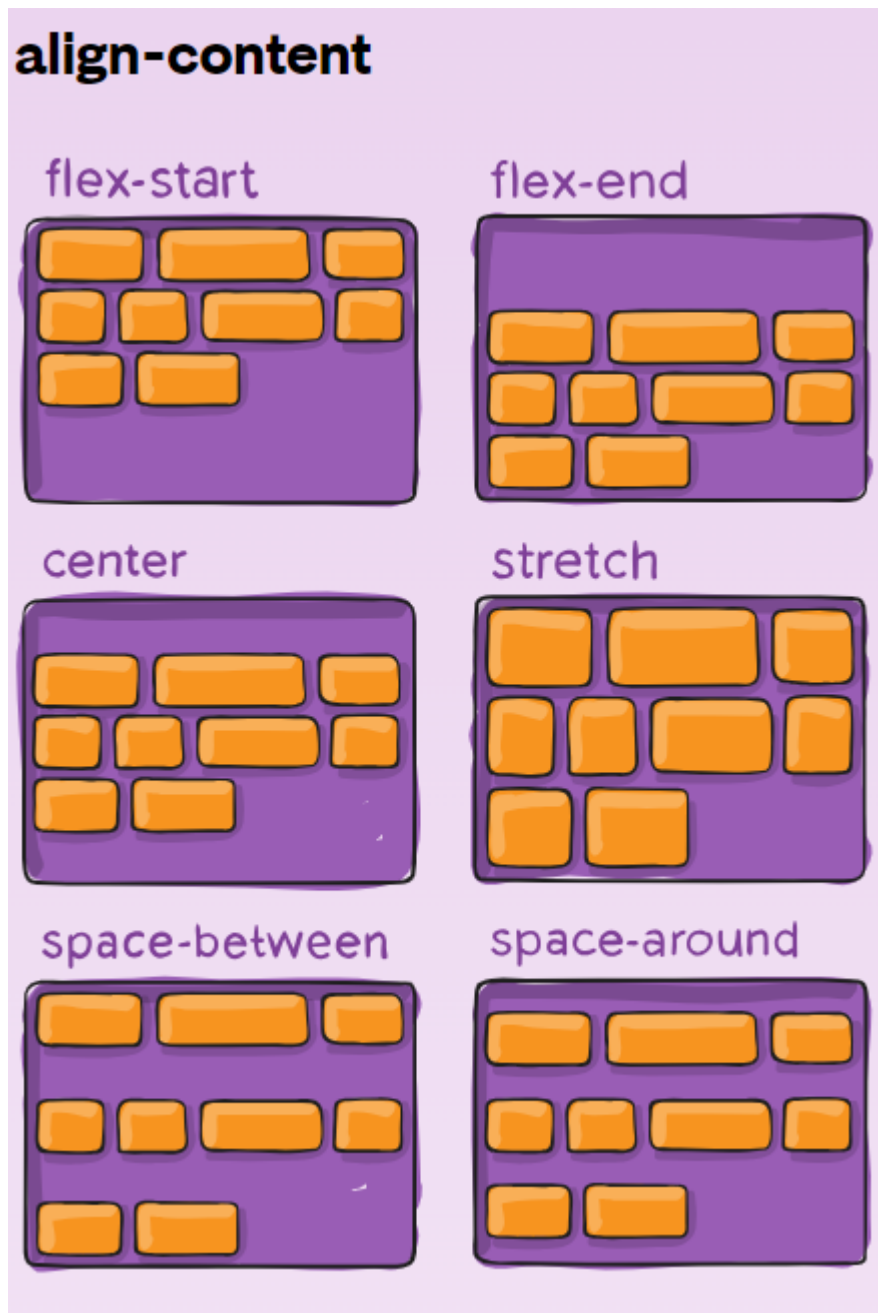


- align-items: giống trên nhưng theo trục dọc

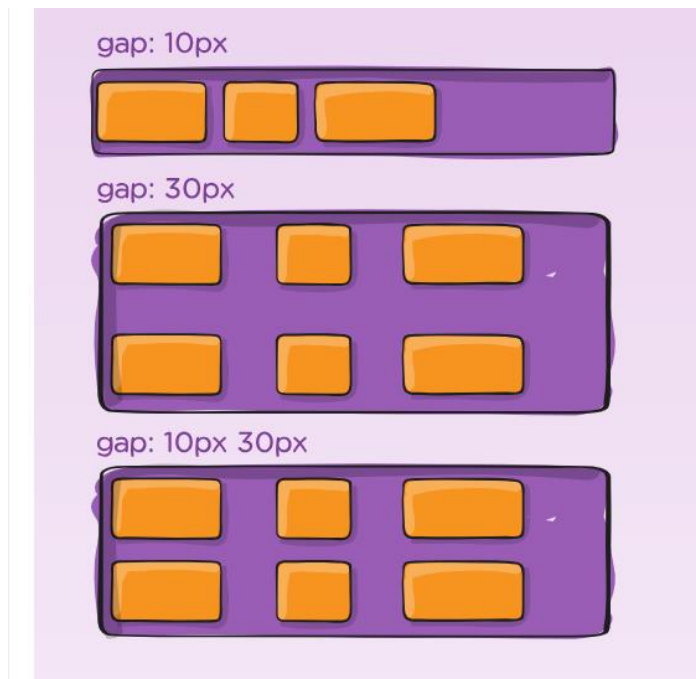
Hình minh họa nhìn dễ hiểu hơn



- align-content: được sử dụng để căn chỉnh và điều chỉnh khoảng cách giữa các hàng (theo chiều dọc) bên trong một flex container. Nó chỉ có tác dụng khi có nhiều hàng trong flex container.

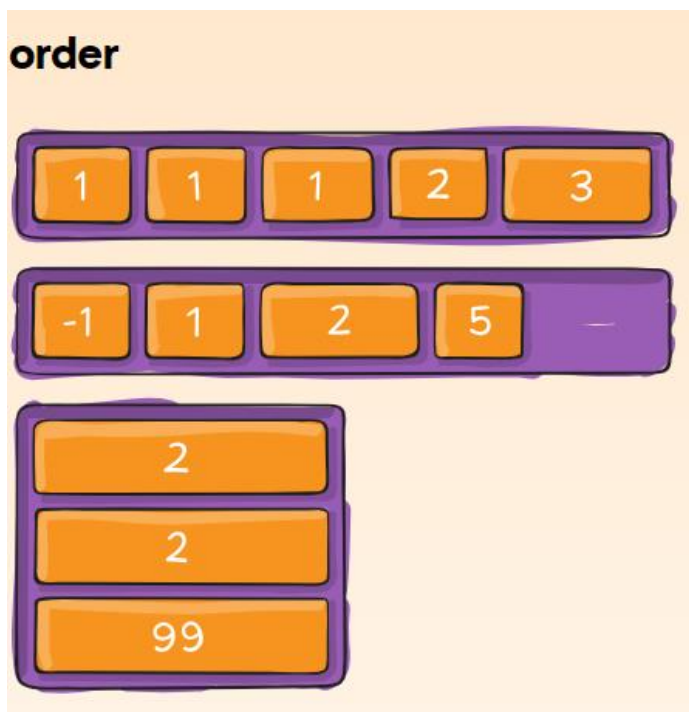


- gap, row-gap, column-gap: cách khoảng cách giữa các dòng, các cột

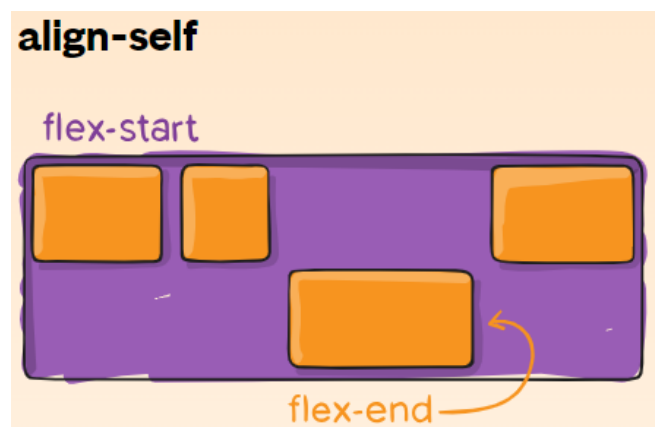


* Nói về Children

- order: thứ tự của item children trên 1 hàng / cột, mặc định là 0, muốn nó ở vị trí nào thì cứ order:1 hoặc 2 3 tùy theo mong muốn



- **flex-grow**: để xác định mức độ mở rộng của các item linh hoạt trong flex container khi không đủ không gian, giống như khi có 3 items, set cho nó **flex-grow:1** thì mỗi item có thể mở rộng và sẽ chia tỷ lệ bằng nhau các phần còn lại trong flex container khi không đủ không gian.
- **flex-shrink**: xác định khả năng co lại, ví dụ **flex-shrink** được đặt là 1 cho mỗi item. Điều này có nghĩa là mỗi item có khả năng co lại và sẽ chia tỷ lệ bằng nhau khi không đủ không gian.
- **flex-basis**: được sử dụng để xác định kích thước ban đầu của một item trong flex container trước khi sử dụng thuộc tính **flex-grow**, **flex-shrink**, và **flex** tổng hợp.
- **flex**: tổng hợp của **shrink** và **basis**
- **align-self**: thay đổi vị trí bắt đầu của 1 item nhất định



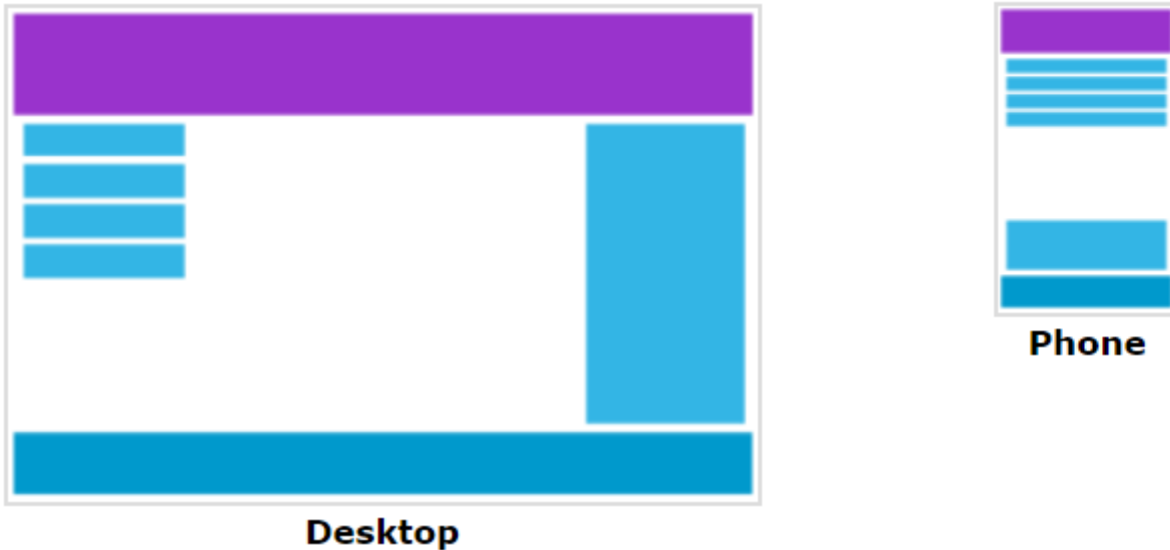
3. Responsive Web Design - Media Queries

- Sử dụng **@media** để chỉ bao gồm một khối thuộc tính CSS nếu một điều kiện nhất định là đúng.

Ví dụ như muốn thay đổi màu web khi màn hình đạt 600px co lại:

```
@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

Add a breakpoint: thêm một điểm dừng để hoạt động theo từng điểm dừng màn hình khác nhau



- Media types: all, print và screen

- Viewport: chế độ xem

- Thẻ `<meta>` viewport thiết lập cho trang web hiển thị tương ứng với kích thước của từng thiết bị khác nhau.
- `width=device-width`: đặt chiều rộng của trang web theo chiều rộng màn hình của thiết bị (sẽ thay đổi tùy theo thiết bị).
- `initial-scale=1.0`: thiết lập mức độ phóng ban đầu khi trang được trình duyệt tải lần đầu tiên, người dùng sẽ không thể zoom khi thuộc tính này có giá trị bằng 1.

Các breakpoint thông dụng hiện nay

`/* Extra small devices (phones, 600px and down) */`

`@media only screen and (max-width: 600px) {...}`

`/* Small devices (portrait tablets and large phones, 600px and up) */`

`@media only screen and (min-width: 600px) {...}`

`/* Medium devices (landscape tablets, 768px and up) */`

`@media only screen and (min-width: 768px) {...}`

```
/* Large devices (laptops/desktops, 992px and up) */
```

```
@media only screen and (min-width: 992px) {...}
```

```
/* Extra large devices (large laptops and desktops, 1200px and up) */
```

```
@media only screen and (min-width: 1200px) {...}
```

Thêm 1 số ví dụ

```
/* If the screen size is 600px wide or less, hide the element */
```

```
@media only screen and (max-width: 600px) {
```

```
  div.example {
```

```
    display: none;
```

```
  }
```

```
}
```

```
/* If the screen size is 601px or more, set the font-size of <div> to 80px */
```

```
@media only screen and (min-width: 601px) {
```

```
  div.example {
```

```
    font-size: 80px;
```

```
  }
```

```
}
```

```
/* If the screen size is 600px or less, set the font-size of <div> to 30px */
```

```
@media only screen and (max-width: 600px) {
```

```
  div.example {
```

```
    font-size: 30px;
```

```
  }
```

III. BEM

- BEM là quy tắc viết tên class trong html css
- Là viết tắt của Block, Element, Modifier
- viết css cho có tổ chức, giống như best practice

3.1 Định nghĩa BEM

| Thành phần | Block (Khối) | Element (Phần tử) | Modifier (Sửa đổi) |
|------------|--|---|---|
| Định nghĩa | Là thực thể độc lập có ý nghĩa riêng. Tuy các block có thể được lồng vào nhau và tương tác với nhau, nhưng về mặt ngữ nghĩa các block vẫn bình đẳng, không có sự ưu tiên hoặc thứ bậc. | Là một phần của block và không có ý nghĩa độc lập, hay có thể coi element là phần tử con của block, element được gắn về mặt ngữ nghĩa với block của nó. | Là sửa đổi trên một block hoặc element, được dùng để thay đổi về ngoài, hành vi hoặc trạng thái của block hay element đó. |
| Ví dụ | <code>header, container, menu, checkbox, input</code> | <code>menu item, list item, checkbox caption, header title</code> | <code>disabled, highlighted, checked, fixed, size big, color yellow</code> |

3.2 Quy tắc đặt tên

→ BLOCK

- Tên block có thể bao gồm các chữ cái Latinh, chữ số và dấu gạch ngang.
- Tạo CSS class: thêm một tiền tố vào phía trước. VD: `.block`

Cách dùng trong HTML

- Bất kỳ node (nút) DOM nào cũng có thể là một block nếu nó có một class name.
- VD:

```
<div class="block">...</div>
```

Cách dùng trong CSS

- Chỉ sử dụng bộ chọn class
- Không dùng tên tag (thẻ) hoặc id
- Không phụ thuộc vào các block/element khác trên một trang
- VD:

```
.block { color: #042; }
```

→ ELEMENT

- Tên element có thể bao gồm các chữ cái Latinh, chữ số, dấu gạch ngang và dấu gạch dưới.
- Tạo class CSS: tên block cộng với hai dấu gạch dưới, cộng với tên element.
- VD: `.block__element`

HTML

- Bất kỳ node DOM nào trong một block đều có thể là một element.
- Trong một block nhất định, tất cả các element đều bằng nhau về mặt ngữ nghĩa.
- VD:

```
<div class="block">
    ...
    <span class="block__elem"> </span>
</div>
```

Cách dùng trong CSS

- Chỉ sử dụng bộ chọn class
- Không dùng tên tag (thẻ) hoặc id
- Không phụ thuộc vào các block/element khác trên một trang
- VD:

- Nên:

```
.block__elem { color: #042; }
```

- Không nên:

```
.block .block__elem { color: #042; }
```

```
div.block__elem { color: #042; }
```

→ MODIFIER

Quy tắc đặt tên

- Tên của modifier có thể bao gồm các chữ cái Latinh, chữ số, dấu gạch ngang và dấu gạch dưới.
- Tạo class CSS: tên block hoặc element cộng với hai dấu gạch ngang, cộng với tên modifier.

- Dấu cách trong các modifier dài (chứa 2 tiếng trở lên) được thay thế bằng dấu gạch ngang.
- VD:

```
.block--mod { }
.block__elem--mod { }
.block--color-black { }
.block--color-red { }
```

HTML

- Modifier là tên class mà bạn thêm vào node DOM block/element.
- Tăng thêm các modifier class vào các block/element mà chúng ta cần sửa đổi và giữ lại class ban đầu của block/element đó.
- VD:

- Nên (thêm các class mới có chứa modifier, và giữ nguyên class cũ **.block**):

```
<div class="block block--mod">...</div>
<div class="block block--size-big block--shadow-yes">...</div>
```

- Không nên (thay thế class cũ):

```
<div class="block--mod">...</div>
```

Cách dùng trong CSS

- Sử dụng modifier class làm bộ chọn CSS. VD:

```
.block--hidden { }
```

- Thay đổi các element dựa trên block có chứa modifier. VD:

```
.block--mod .block__elem { }
```

- Element có modifier. VD:

```
.block__elem--mod { }
```

IV. SCSS

- Giúp giảm thiểu code css và nhìn code có tổ chức hơn, đỡ rối mắt
- Tái sử dụng code, sửa 1 lần, áp dụng toàn project
- Là phần mở rộng hơn của css thôi, ngoài ra syntax không khác gì mấy

4.1 Scss Variables

- Sử dụng biến để lưu trữ thông tin

Sass Variable Syntax:

➤ `$variablename: value;`

Ví dụ:

```
$myFont: Helvetica, sans-serif;
$myColor: red;
$myFontSize: 18px;
$myWidth: 680px;

body {
  font-family: $myFont;
  font-size: $myFontSize;
  color: $myColor;
}

#container {
  width: $myWidth;
}
```

4.2 SCSS Nested

- Cho phép lồng các css lại với nhau thì vì css riêng lẻ như thông thường, tại sao phải lồng? Ví dụ có 1 block chứa rất nhiều block nhỏ, khi css thì rất dài và nhiều code, thì với nested này ta có thể lồng cha với con, sửa rất tiện mỗi khi sai

| | |
|--|--|
| <pre>nav { ul { margin: 0; padding: 0; list-style: none; } li { display: inline-block; } a { display: block; padding: 6px 12px; text-decoration: none; } }</pre> | <pre>nav ul { margin: 0; padding: 0; list-style: none; } nav li { display: inline-block; } nav a { display: block; padding: 6px 12px; text-decoration: none; }</pre> |
| SCSS | CSS |

4.3 Partial

Định nghĩa từ docs của SCSS: Trong SCSS (Sass), partials là các file CSS được tách ra thành các phần nhỏ để sử dụng và quản lý dễ dàng hơn trong quá trình phát triển. Các partials trong SCSS thường có đuôi .scss và bắt đầu bằng dấu gạch dưới _.

→ Có nghĩa là có thể tách các file riêng biệt ra , ví dụ như ta tách ra ‘_header.scss’, ‘_navigation.scss’,... , sau đó nhập các partials này vào một tệp SCSS chính ví dụ như ‘main.scss’ bằng cách sử dụng @import

VÍ DỤ:

```
// _variables.scss

$primary-color: blue;
$secondary-color: green;

// main.scss

@import 'variables';

.header { background-color: $primary-color; }

.footer { background-color: $secondary-color; }
```

4.4 Modules

Khi sử dụng SCSS modules, có thể tạo ra các file SCSS riêng cho mỗi module hoặc thành phần trong ứng dụng của bạn. Mỗi file SCSS module này chứa tất cả các luật CSS cần thiết để tạo ra giao diện cho module đó. Bằng cách này, bạn có thể tách biệt các module và đảm bảo rằng các luật CSS không xung đột với nhau.

VÍ DỤ:

```
// _button.scss

.button {

background-color: blue;

color: white;

padding: 10px 20px;

border-radius: 4px;

}
```

```
// main.scss

@import 'button';

.header {
  .button {
    margin-right: 10px;
  }
}
```

4.5 Mixins

Mixin tương tự như các function. có thể sử dụng ở khắp mọi nơi (phải include) và chúng ta có thể truyền tham số để tùy biến.

Lệnh `@mixin` cho phép bạn tạo mã CSS để sử dụng lại trên toàn bộ trang web.

Chỉ thị `@include` được tạo để cho phép bạn sử dụng (bao gồm) mixin.

Syntax

Mixin có 2 loại là:

→ Không tham số

```
@mixin mixin_name {
// SASS selectors...
}
```

Ví dụ:

```
@mixin important-text {
  color: red;
  font-size: 25px;
  font-weight: bold;
  border: 1px solid blue;
}

@include important-text
```

Sử dụng mixin mới khai báo

Syntax:

```
selector {  
  @include mixin-name;  
}
```

Ví dụ:

```
.danger {  
  @include important-text; // sử dụng ví dụ ở trên  
  background-color: green;  
}
```

→ Có tham số

```
@mixin mixin_name(param1, param2...)  
{  
  // SASS selectors...  
}
```

Ví dụ

```
/* Define mixin with two arguments */  
@mixin bordered($color, $width) {  
  border: $width solid $color;  
}  
  
.myArticle {  
  @include bordered(blue, 1px); // Call mixin with two values  
}
```

Ngoài ra còn có thể truyền tham số mặc định được, có thể dùng mixin cho Vendor Prefix được

Ví dụ vendor prefix:

```
@mixin transform($property) {  
  -webkit-transform: $property;  
  -ms-transform: $property;  
  transform: $property;  
}
```

```

}

.myBox {
  @include transform(rotate(20deg));
}

```

4.6 Extend/Inheritance

@extend cho phép bạn chia sẻ một tập hợp các thuộc tính CSS từ bộ chọn này sang bộ chọn khác.

Ví dụ

```

.button-basic {
  border: none;
  padding: 15px 30px;
  text-align: center;
  font-size: 16px;
  cursor: pointer;
}

.button-report {
  @extend .button-basic;
  background-color: red;
}

.button-submit {
  @extend .button-basic;
  background-color: green;
  color: white;
}

```

→ Có thể thấy được rằng button report và button submit đều kế thừa thuộc tính từ button basic

4.7 Operators

Sass hỗ trợ một số toán tử hữu ích để làm việc với các giá trị khác nhau. Chúng bao gồm các toán tử toán học tiêu chuẩn như + và *, cũng như các toán tử cho nhiều loại khác:

Arithmetic Operators

→ Cộng trừ nhân chia

operator	description
+	this operator is used for addition.
-	this operator is used for subtraction.
*	this operator is used for multiplication.
/	this operator is used for division.
%	this operator is used for remainder.

Ví dụ

```
h2 {  
  font-size: 15px + 2em; // Show error due to incompatible units  
  font-size: 15px + 2; // 17px  
}
```

```
h2 {  
  width: 3px * 5 + 5px; // 20px  
  width: 3 * (5px + 5px); // 30px  
  width: 3px + (6px / 2) * 3; // 12px  
}
```

Equality Operators & Comparison Operators

→ được sử dụng trong các câu điều kiện. Các toán tử này được sử dụng để kiểm tra xem hai giá trị có giống nhau hay không và trả về các giá trị Boolean.

Operator	Description
= =	It is equal to operator.
!=	It is not equal to operator.

Operator	Description
>	It specifies a greater than comparison operator.
>=	It specifies a greater than or equal to comparison operator.
<	It specifies a less than comparison operator.
<=	It specifies less than or equal to comparison operator.

Ví dụ:

Bằng (=): 5px = 5px sẽ cho kết quả true.

Không bằng (!=): 10px != 5px sẽ cho kết quả true.

Lớn hơn (>), nhỏ hơn (<), lớn hơn hoặc bằng (>=), nhỏ hơn hoặc bằng (<=): 10px > 5px sẽ cho kết quả true.

Logic Opearators

→ Các toán tử logic được sử dụng để kiểm tra nhiều điều kiện trong một biểu thức điều kiện.

Operator	Conditions	Description
And	x and y	It specifies true if x and y both are true.
Or	x or y	It specifies true if either x or y is true.
Not	x	It is true if x is not true.

→ Và (and): true and false sẽ cho kết quả false.

→ Hoặc (or): true or false sẽ cho kết quả true.

→ Phủ định (not): not true sẽ cho kết quả false.

V. Tailwind CSS

5.1 Định nghĩa

- Là 1 framework CSS, tương tự như Bootstrap
- Giúp xây dựng website một cách nhanh chóng nhất với các thuộc tính CSS đã được gán thành những class riêng, khi dùng chúng ta chỉ có việc gọi ra để dùng

5.2 Installation & Usage

Cài đặt thông qua link CDN

```
<script src="https://cdn.tailwindcss.com"></script>
```

Hoặc

```
<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
```

Lưu ý: Có một số hạn chế khi sử dụng CDN.

- Không thể sử dụng tùy chỉnh chủ đề mặc định của Tailwind
- Không thể sử dụng các lệnh như `@apply`, `@variants`,...
- Không thể cài đặt plugin của bên thứ ba Tốc độ tải trang của trang web phụ thuộc vào tốc độ kết nối internet của người dùng và khoảng cách đến máy chủ CDN.
- Nếu kết nối internet chậm hoặc khoảng cách đến máy chủ CDN quá xa, thời gian tải trang sẽ mất nhiều thời gian.
- Bạn cần phải trả phí để sử dụng các tính năng, dịch vụ CDN cao cấp. Bạn không thể kiểm soát việc lưu trữ tệp CSS của bạn trên máy chủ CDN. Điều này có thể ảnh hưởng đến việc kiểm soát và bảo mật dữ liệu của bạn.
- Vì vậy, trước khi sử dụng CDN, bạn nên cân nhắc các hạn chế này để đảm bảo rằng việc sử dụng CDN là phù hợp với nhu cầu

Cài đặt thông qua NPM

Bước 1: Tailwind có sẵn trên npm và bạn có thể cài đặt nó bằng lệnh sau:

```
npm install tailwindcss postcss autoprefixer
```

Bước 2: Tạo tệp cấu hình Tailwind CSS bằng lệnh sau:

npx tailwindcss init

Bước 3: Tạo một postcss.config.js file trong thư mục gốc của thư mục dự án và thêm nội dung sau:

```
module.exports = {
  plugins: [
    require('tailwindcss'),
    require('autoprefixer'),
  ],
}
```

Bước 4: Sau đó, tạo một styles.css file trong thư mục dự án của bạn src (hoặc bất kỳ thư mục nào khác mà bạn thích) và thêm nội dung sau:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Bước 5: Ở thao tác này, bạn sẽ nhập các kiểu cơ sở, thành phần (components) và utilities styles từ Tailwind CSS.

Cuối cùng, thêm tập lệnh vào package.json file của bạn để tạo CSS:

```
"scripts": {
  "build": "postcss src/styles.css -o dist/styles.css"
}
```

Tập lệnh này sẽ biên dịch src/styles.css file của bạn và xuất nó thành một dist/styles.css file.

Bước 6: Chạy lệnh sau để xây dựng CSS :

npm run build

Cài đặt thông qua Yarn

→ Giống với các câu lệnh npm, chỉ thay đổi ở chỗ là thay chữ ‘npm’ thành ‘yarn’

VI. Javascript

- JavaScript là ngôn ngữ lập trình phổ biến dùng để tạo ra các trang web tương tác. Được tích hợp và nhúng vào HTML giúp website trở nên sống động hơn.

- JavaScript đóng vai trò như một phần của trang web, thực thi cho phép **Client-Side Script** từ phía người dùng cũng như phía máy chủ (Nodejs) tạo ra các trang web động.

6.1 Syntax & Basic

Mở đóng thẻ

Tất cả các đoạn mã JavaScript đều được đặt trong cặp thẻ đóng mở `<script></script>`. Ví dụ cụ thể như sau:

```
<script language="javascript">  
    alert("Hello World!");  
</script>
```

Cách đặt thẻ Script

1. Internal: viết trong file html hiện tại.
2. External: viết ra một file js khác rồi import vào.
3. Inline: viết trực tiếp trong thẻ HTML.

Datatypes

Có 8 loại:

1. String
2. Number
3. BigInt
4. Boolean
5. Undefined
6. Null
7. Symbol
8. Object

Object Datatypes

Kiểu dữ liệu mà đối tượng object có thể chứa

1. An object
2. An array
3. A date

Variable

Có 3 loại: var, let, const

Sự khác nhau:

- + var: có phạm vi toàn cục hoặc phạm vi hàm, nghĩa là nó có thể được truy cập từ mọi nơi trong cùng một tệp tin hoặc cùng một hàm; có thể tái khai báo; có hoisting nghĩa là biến có thể được truy cập trước cả khi nó được khai báo
- + let: phạm vi block , tức là scope nó được khai báo với 1 scope nhất định , riêng lẻ chứ không có phạm vi toàn cục như var; không thể tái khai báo nhưng gán giá trị mới thì được; không có hoisting
- + const: phạm vi scope giống let, nhưng giá trị của biến không thay đổi được khi đã gán giá trị , không thể tái khai báo, ghi đè, không hoisting

JavaScript Assignment Operators

→ Giống các ngôn ngữ khác

JavaScript Comparison Operators

→ Giống các ngôn ngữ khác, nhưng có thêm toán tử “ === “ tức là để so sánh giá trị và kiểu dữ liệu

JavaScript Logical Operators

→ Giống các ngôn ngữ khác

JavaScript Bitwise Operators

→ Giống các ngôn ngữ khác

JavaScript String Methods

String length	String trim()
String slice()	String trimStart()
String substring()	String trimEnd()
String substr()	String padStart()
String replace()	String padEnd()
String replaceAll()	String charAt()
String toUpperCase()	String charCodeAt()
String toLowerCase()	String split()
String concat()	

JavaScript Array Methods

Array length	Array join()
Array toString()	Array delete()
Array pop()	Array concat()
Array push()	Array flat()
Array shift()	Array splice()
Array unshift()	Array slice()

The methods are listed in the order they appear in this tutorial page

JavaScript Array Iteration

Gồm có ForEach(), map(), flatMap(), filter(), some(), indexOf(), find(), includes(),, ngoài ra còn nhiều hàm khác

JavaScript JSON

- Viết tắt của Javascript Object Notation

Thông qua ví dụ cho dễ hiểu :

```
{  
  "employees": [  
    {"firstName": "John", "lastName": "Doe"},  
    {"firstName": "Anna", "lastName": "Smith"},  
    {"firstName": "Peter", "lastName": "Jones"}  
  ]  
}
```

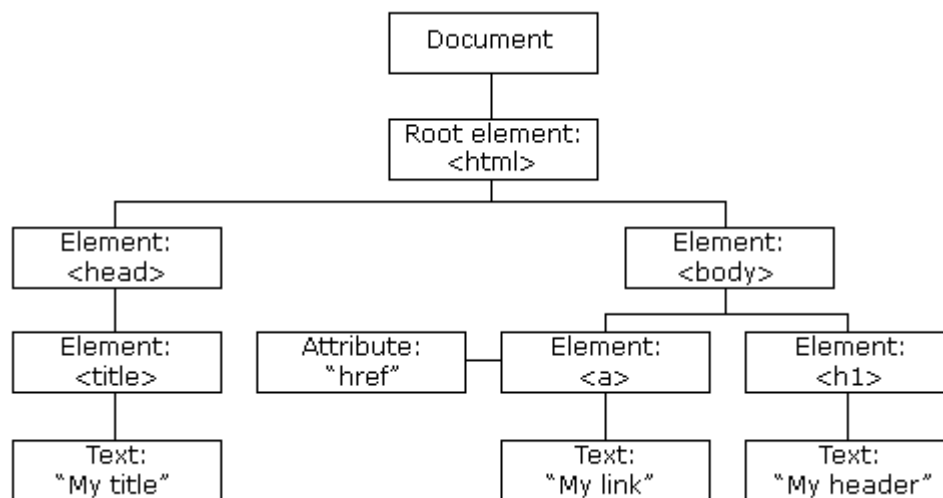
]
}

→ Giống như cặp key-value, không cố định 1 kiểu dữ liệu, có thể xài nhiều kiểu dữ liệu,
Dữ liệu được phân cách bởi dấu phẩy, nếu khai báo mảng thì phải xài dấu ngoặc vuông
như trên hình, bắt đầu 1 json thì phải là bắt đầu bằng ngoặc nhọn

6.2 DOM Manipulation

- DOM là viết tắt của Document Object Model

- Khi page khởi tạo , load sẽ tạo 1 DOM, HTML DOM có cấu trúc dưới đây



Với mô hình đối tượng, JavaScript có được tất cả sức mạnh cần thiết để tạo HTML động:

- JavaScript có thể thay đổi tất cả các phần tử (elements) HTML trong trang
- JavaScript có thể thay đổi tất cả các thuộc tính (attributes) HTML trong trang
- JavaScript có thể thay đổi tất cả các kiểu CSS trong trang
- JavaScript có thể xóa các phần tử và thuộc tính HTML hiện có
- JavaScript có thể thêm các phần tử và thuộc tính HTML mới
- JavaScript có thể phản ứng với tất cả các sự kiện HTML hiện có trong trang
- JavaScript có thể tạo các sự kiện HTML mới trong trang

Vậy kết luận về HTML DOM :

Các **elements** HTML như các đối tượng

Các **attributes** của tất cả các phần tử HTML

Các **methods** để truy cập tất cả các phần tử HTML

Các **events** cho tất cả các phần tử HTML

Nói cách khác: HTML DOM là một tiêu chuẩn về cách lấy, thay đổi, thêm hoặc xóa các phần tử HTML.

DOM methods

- HTML Dom methods là các hành động mà mình có thể biểu diễn trên HTML elements
 - HTML Dom properties là các giá trị của HTML elements mà ta có thể set hoặc thay đổi
- methods như kiểu là hành động thêm hoặc xóa 1 HTML elements

Ví dụ có sau:

```
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

getElementById → methods, inner HTML → property

DOM Document

- Nếu muốn truy cập tới bất kì element nào trong HTML, ta phải bắt đầu bằng việc truy cập document object

Ví dụ:

Nếu muốn tìm HTML Elements

Method	Description
<code>document.getElementById(<i>id</i>)</code>	Find an element by element id
<code>document.getElementsByTagName(<i>name</i>)</code>	Find elements by tag name
<code>document.getElementsByClassName(<i>name</i>)</code>	Find elements by class name

Nếu muốn thay đổi HTML Elements

Property	Description
<code>element.innerHTML = <i>new html content</i></code>	Change the inner HTML of an element
<code>element.attribute = <i>new value</i></code>	Change the attribute value of an HTML element
<code>element.style.property = <i>new style</i></code>	Change the style of an HTML element
Method	Description
<code>element.setAttribute(<i>attribute</i>, <i>value</i>)</code>	Change the attribute value of an HTML element

Nếu muốn thêm hay xóa Elements

Method	Description
<code>document.createElement(<i>element</i>)</code>	Create an HTML element
<code>document.removeChild(<i>element</i>)</code>	Remove an HTML element
<code>document.appendChild(<i>element</i>)</code>	Add an HTML element
<code>document.replaceChild(<i>new</i>, <i>old</i>)</code>	Replace an HTML element
<code>document.write(<i>text</i>)</code>	Write into the HTML output stream

DOM Elements

Tìm phần tử HTML theo id

Tìm phần tử HTML theo tên thẻ

Tìm phần tử HTML theo tên lớp

Tìm các phần tử HTML bằng bộ chọn CSS

Tìm phần tử HTML bằng HTML object collections

Ví dụ

Tìm phần tử thông qua id:

```
const element = document.getElementById("intro");
```

Tìm phần tử thông qua Tag Name:

```
const element = document.getElementsByTagName("p");
```

Tìm phần tử thông qua Class Name:

```
const x = document.getElementsByClassName("intro");
```

Tìm phần tử thông qua CSS Selectors

```
const x = document.querySelectorAll("p.intro");
```

DOM Events

- Là các sự kiện để cho trang web phong phú hơn, tạo cảm giác với user để dễ nhận biết và thao tác giữa các elements trong html ví dụ như các trường hợp người dùng bấm chuột vào các button, di chuột trên element, hiệu ứng load trang, load các dữ liệu, nhập liệu form,...

VD1:

```
<h1 onclick="this.innerHTML = 'Oops!'">Click on this text!</h1>
```

VD2:

```
<h1 onclick="changeText(this)">Click on this text!</h1>
```

```
<script>
function changeText(id) {
  id.innerHTML = "Oops!";
}
</script>
```

VD3:

```
<script>
document.getElementById("myBtn").onclick = displayDate;
</script>
```


6.3 Fetching API

- thực hiện các yêu cầu HTTP tới máy chủ (server) từ trình duyệt web.

Sending a request

- **fetch()** method chỉ require về 1 url, truyền url này vào trong và gọi tới api

```
let response = fetch(url);
```

- **fetch()** method này return về 1 Promise, ta sẽ sử dụng **then()** và **catch()** để handle **fetch(url)**

```
.then(response => {  
    // handle the response  
})  
.catch(error => {  
    // handle the error  
});
```

→ Khi mà một request thành công, resource có tồn tại, tại thời điểm này promise sẽ resolve into 1 Response object (là 1 API wrapper cho fetch resource).

Reading the Response

Responses có phương thức truy cập vào nội dung trả về. Ví dụ, **Response.json()** trả về một promise resolves dạng JSON.

Tổng kết: Đầu tiên bước 1 ta sẽ fetch một url nào đó. Fetch trả về một promise, promise sẽ được resolve, giá trị trả về ta sẽ validate nó bởi 1 hàm validate do ta tự định nghĩa. Bước 2, khi validate sẽ nhận đầu vào là response và kiểm tra xem có hợp lệ không (status là 2xx), nếu không, sẽ có lỗi sẽ chuyển đến hàm **.catch** thực thi. Còn nếu không có lỗi thì tức là response valid và sẽ được chuyển tới hàm để đọc response as Json. Bước 3, tại hàm đọc response này, sẽ đọc nội dung của response bằng phương thức **Response.Json()**,

phương thức này trả về một promise resolves dạng json, khi promise dc resolves, dữ liệu Json sẽ dc chuyển tới hàm để xuất ra kết quả

AJAX

AJAX là từ viết tắt của từ **A**synchronous **J**avaScript **A**nd **X**ML - **Bất đồng bộ** trong Javascript và XML.

- **Asynchrononous** (*Bất đồng bộ*): Bất đồng bộ có nghĩa là một chương trình có thể xử lý không theo tuần tự các hàm, không có quy trình, có thể nhảy đi bỏ qua bước nào đó. Ích lợi dễ thấy nhất của bất đồng bộ là chương trình có thể xử lý nhiều công việc một lúc.
- **Javascript**: Đây là một ngôn ngữ lập trình rất phổ biến hiện nay. Trong số rất nhiều chức năng của nó là khả năng quản lý nội dung động của website và hỗ trợ tương tác với người dùng.
- **XML**: Đây là một dạng ngôn ngữ gần giống với **HTML**, tên đầy đủ là **eXtensible Markup Language**. Nếu HTML được dùng để hiển thị dữ liệu, XML được thiết kế để chứa dữ liệu.

- là một kỹ thuật cho phép gửi và nhận dữ liệu từ server mà không cần tải lại trang web. Nó cho phép giao tiếp mượt mà và cập nhật nội dung của trang web một cách động.

Ví dụ

```
// Tạo đối tượng XMLHttpRequest
```

```
var xhttp = new XMLHttpRequest();
```

```
// Thiết lập callback function để xử lý dữ liệu nhận được
```

```
xhttp.onreadystatechange = function() {
```

```
    if (this.readyState == 4 && this.status == 200) {
```

```
        // Xử lý dữ liệu nhận được
```

```
        var response = JSON.parse(this.responseText);
```

```
        console.log(response);
```

```

    }

};

// Mở kết nối và gửi yêu cầu đến server

xhttp.open("GET", "https://api.example.com/data", true);

xhttp.send();

```

6.4 ES6

- phiên bản nâng cấp , bổ sung nhiều tính năng mới như biến let , const (thay đổi scope), arrow function, toán tử spread (...), Map, Set, Class, Promise, Rest parameter, String.includes(),...

6.5 Library

Lodash

- Lodash là một thư viện tiện ích JavaScript nhất quán, các modules có hiệu suất cao, và quan trọng rất mạnh mẽ bao gồm hầu hết các điểm chức năng logic có thể gặp trong quá trình phát triển front-end. Sử dụng Lodash có thể cải thiện đáng kể hiệu quả phát triển của mỗi dự án.

- Dùng để xử lý Array, Object, Function, Collection ...

Sau khi tìm hiểu thì thấy đa số không khuyên dùng Lodash, mà thay vào đó nên sử dụng ES6

Swiper

- Được sử dụng để tạo các carousel (đối tượng trượt) và các giao diện trượt khác trên các trang web và ứng dụng di động. Nó cung cấp một cách dễ dàng để tạo ra các trình trượt hình ảnh, trình trượt quảng cáo, trình trượt sản phẩm, và nhiều kiểu trình trượt khác một cách linh hoạt.

- Swiper hỗ trợ nhiều tính năng như trượt ngang (horizontal) hoặc trượt dọc (vertical), trượt tự động, trượt vô hạn (loop), trượt đa phương tiện (multimedia slide), chỉ mục trang

(pagination), điều hướng trượt (navigation), và nhiều hiệu ứng trượt khác nhau. Nó cũng cho phép tùy chỉnh và điều chỉnh các thuộc tính và sự kiện của trình trượt.

VII. JQUERY

7.1 Syntax & Basic

- jQuery là một thư viện JavaScript mã nguồn mở phổ biến, được sử dụng để giảm bớt sự phức tạp và tiết kiệm thời gian khi viết mã JavaScript trên các trang web. Nó cung cấp các phương thức và chức năng để thao tác và tương tác với các phần tử HTML, xử lý sự kiện, thay đổi nội dung trang web và thực hiện các yêu cầu Ajax một cách dễ dàng.

jQuery giúp các nhà phát triển web thao tác với DOM (Document Object Model) một cách thuận tiện hơn và đơn giản hóa việc tương tác với HTML, CSS và JavaScript. Với jQuery, bạn có thể chọn các phần tử HTML dễ dàng bằng cú pháp CSS-style selector và thực hiện các hành động trên chúng như thêm, xóa, sửa đổi thuộc tính, thay đổi nội dung và xử lý sự kiện.

Sử dụng JQuery: thông qua cách tải về hoặc gắn link CDN

Jquery Syntax

`$(selector).action()`

- Dấu \$ để xác định/truy cập jQuery
- selector là để query tới HTML elements
- action() được thực hiện trên các elements

ví dụ

`$(this).hide()` - ẩn đi element hiện tại

`$("p").hide()` - ẩn đi tất cả element <p>

`$(".test").hide()` - ẩn tất cả element với class="test".

`$("#test").hide()` - ẩn element với id="test".

Jquery Selector

Ví dụ :

```
$(document).ready(function(){  
    $("button").click(function(){  
        $("p").hide();  
    });  
});
```

Jquery Events

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

Jquery & Ajax

jQuery cung cấp một số phương thức cho chức năng AJAX.

Với các phương thức jQuery AJAX, bạn có thể yêu cầu văn bản, HTML, XML hoặc JSON từ một máy chủ từ xa bằng cách sử dụng cả HTTP Get và HTTP Post - Và bạn có thể tải trực tiếp dữ liệu bên ngoài vào các phần tử HTML đã chọn trên trang web của mình!

Không có jQuery, viết mã AJAX có thể hơi phức tạp một chút!

→ Kết hợp jQuery và Ajax cho phép bạn tương tác với server một cách bất đồng bộ và cập nhật nội dung của trang web mà không cần tải lại toàn bộ trang. jQuery cung cấp các phương thức và chức năng đơn giản để thực hiện các yêu cầu Ajax một cách dễ dàng.

Ví dụ

```

40
41 <!DOCTYPE html>
42 <html>
43 <head>
44   <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
45 </head>
46 <body>
47   <button id="loadData">Load Data</button>
48   <div id="dataContainer"></div>
49
50   <script>
51     $(document).ready(function() {
52       $("#loadData").click(function() {
53         $.ajax({
54           url: "https://api.example.com/data",
55           type: "GET",
56           success: function(data) {
57
58             $("#dataContainer").text(data);
59           },
60           error: function() {
61             alert("Đã xảy ra lỗi khi tải dữ liệu!");    "liệu": Unknown word.
62           }
63         });
64       });
65     });
66   </script>
67 </body>
68 </html>

```

Trong ví dụ trên, chúng ta sử dụng jQuery để chọn phần tử có id="loadData" là nút "Load Data".

Khi người dùng nhấp vào nút đó, chúng ta sử dụng phương thức \$.ajax() của jQuery để gửi một yêu cầu GET đến URL <https://api.example.com/data>.

Trong phần cấu hình của \$.ajax(), chúng ta thiết lập URL, loại yêu cầu (type: "GET"), và cung cấp một hàm success để xử lý dữ liệu nhận được từ server.

Trong ví dụ này, chúng ta chỉ đơn giản hiển thị dữ liệu đó bằng cách sử dụng phương thức text() của jQuery để đặt nội dung vào phần tử có id="dataContainer".

Nếu yêu cầu gặp lỗi, chúng ta sử dụng hàm error để hiển thị thông báo lỗi đơn giản.

Khi người dùng nhấp vào nút "Load Data", yêu cầu Ajax sẽ được gửi đi và kết quả trả về từ server sẽ được hiển thị trong phần tử có id="dataContainer".

7.2 DOM Manipulation

- Là quá trình sử dụng các phương thức và chức năng của thư viện JQuery để thay đổi, điều chỉnh và tương tác với các phần tử DOM trong trang web.

→ Điều này bao gồm việc chọn các phần tử, thay đổi nội dung, thêm hoặc xóa các phần tử, điều chỉnh thuộc tính và kiểu dáng, và thực hiện các hiệu ứng hoặc sự kiện trên các phần tử DOM.

Ví dụ

Xóa HTML Elements

```
$("#id02").remove();
```

Code javascript tương ứng:

```
document.getElementById("id02").remove();
```

Get Parent Element

```
myParent = $("#02").parent().prop("nodeName");
```

Code javascript tương ứng:

```
myParent = document.getElementById("02").parentNode.nodeName;
```