CSS

▼ inline ≠ block ≠ inline-block trong display

- **inline**: item nằm trên 1 dòng, vượt quá độ dài sẽ tự động xuống dòng. Chỉ điều chỉnh được margin và padding của left, right (span)
- **block**: item luôn được xuống dòng và chiếm toàn bộ width nếu không được set. Ta điều chỉnh được margin, padding của cả 4 hướng (div)
- **inline-block**: các item được sắp xếp trên cùng 1 hàng như inline nhưng sẽ có thuộc tính của block như có thể điều chỉnh 4 hướng của margin và padđing

▼ BEM là gì ?

- BEM viết tắt của Blocks, Elements, Modifiers, là một phương pháp đặt tên class cho HTML và CSS.
- Được phát triển tại Yandex giúp lập trình viên hiểu rõ hơn mối quan hệ giữa HTML và CSS trong dự án front end

```
/* Một Block (khối) độc lập */
.btn {}

/* Element (phần tử) con, phụ thuộc vào Block ở trên */
.btn__price {}

/* Modifier (bộ điều chỉnh) thay đổi trạng thái của Block hoặc Element */
.btn--orange {}
.btn--big {}
.btn__price--bold {}
```

▼ Box-model là gì ?

- là một hộp chữ nhật bao quanh mọi element trong HTML
- được dùng để xác định chiều cao và rộng cảu
- CSS Box Model giống như là một cái hộp bao quanh element của chúng ta và trong đó có rất nhiều lớp dày mỏng khác nhau, các lớp dày mỏng đó bao gồm: margins, border, padding và cuối cùng là phần nội dung của chúng ta (text và ảnh).

- Nó bao gồm:
  - **Content:** như đã nói ở trên đây là phần mà text và hình ảnh của chúng ta xuất hiện
  - **Padding:** là một khoảng trống kế tiếp bọc xung quanh **content**
  - **Border**: phần khung bao bọc xung quanh **padding** và **content**
  - **Margin**: cuối cùng, **margin** là phần ngoài cùng của Box Model, chỉ là một khoảng trống không màu

▼ **Sự khác biệt giữa các thuộc tính Box Sizing?**

- Thuộc tính CSS box-sizing quy định cách tính tổng chiều rộng và chiều cao của một phần tử.
  - **Context-box:** Giá trị chiều rộng và chiều cao mặc định chỉ áp dụng cho nội dung của phần tử. Padding và border nằm ở bên ngoài hộp.
  - **Padding-box:** Giá trị chiều rộng và chiều cao mặc định chỉ áp dụng cho nội dung của phần tử và padding của nó. Border nằm ở bên ngoài hộp. Hiện tại chỉ có Firefox hỗ trợ padding-box.
  - **Border-box:** Giá trị chiều rộng và chiều cao áp dụng cho nội dung, padding và border.

▼ `* { box-sizing: border-box }` **là gì?**

- Nó điều chỉnh tất cả phần tử có bao gồm padding, border trong không gian phần tử cho tính toàn chiều dài và chiều rộng.
- Trong `box-sizing: border-box`, chiều cao phần tử được tính toán với: height + padding dọc + độ dài border dọc. Còn chiều dài là width + padding ngang + độ dài border ngang.

▼ **CSS selector là gì ?**

- CSS Selector giống như là đường dẫn, chỉ định để cho CSS biết bạn đang muốn điều chỉnh, tạo kiểu cho phần tử HTML nào vậy.

▼ **Một số loại selector:**

- **Universal Selector:** hoạt động như một ký tự đại diện cho tất cả phần tử trong trang.

```
* {
    color: "green";
}
```

- **Element Type Selector:** selector loại này ứng với một hoặc nhiều phần tử HTML cùng tên.

```
ul {
    line-style: none;
}
```

- **ID Selector:** selector này ứng với bất kỳ phần tử HTML nào có thuộc tính ID có cùng giá trị với giá trị của selector.

```
#container {
    width: 960px;
}
```

- **Class Selector:** tương tự như ID Selector nhưng thay vì ứng với ID thì nó ứng với thuộc tính class.

```
.box {
    padding: 10px;
}
```

- **Descendant Combinator:** giúp bạn kết hợp hai hoặc nhiều selector để có thể chỉ định phần tử cụ thể.

```
#container .box {
  float: left;
}
```

- **Child Combinator:** selector sử dụng bộ child combinator tương tự như descendant combinator, ngoại trừ việc nó chỉ nhắm đến các phần tử con.

```
#container> .box {
  float: left;
  padding-bottom: 15px;
}
```

- **General Sibling Combinator:** selector này so với các phần tử có quan hệ anh chị em với phần tử tương ứng.

```
h2 ~ p {
  margin-bottom: 20px;
}
```

- **Adjacent Sibling Combinator:** selector sử dụng ký tự `+` và gần giống với General Sibling Combinator. Sự khác biệt là phần tử được nhắm phải là anh chị ruột thịt chứ không phải anh chị em chung chung.

```
p + p {
  margin-bottom: 0;
}
```

- **Attribute Selector:** nhắm đến các phần tử dựa trên sự xuất hiện và giá trị của thuộc tính HTML. Được khai báo bằng dấu ngoặc vuông.

```
input [type=”text”] {
  width: 200px;
}
```

▼ **Có thể hiện thị 1 trang web trong 1 trang web khác không ?**

- Có
- Sử dụng iframe để nhúng 1 web khác vào
- Có thể nhúng 1 website, video hoặc image, …

▼ **z-index dùng để làm gì ?**

- z-index được sử dụng để chỉ định cách xếp chồng theo chiều sâu của các phần tử chồng lên nhau xảy ra tại thời điểm định vị nó.

- Nó chỉ định thứ tự ngăn xếp theo chiều sâu của các phần tử được định vị giúp xác định cách hiển thị các phần tử diễn ra như thế nào trong trường hợp chồng chéo.

▼ **Sự khác biệt giữa reset và normalize CSS?**

- **Reset CSS:** nhằm mục đích xoá tất cả thiết lập style mặc định từ trình duyệt. Ví dụ như margin, padding, font-size của tất cả phần tử đó được reset lại giống nhau.
- **Normalize CSS:** nhằm mục đích làm cho các style mặc định nhất quán trên trình duyệt. Nó cũng sửa các lỗi phổ biến trên trình duyệt.

▼ **Float là gì ?**

- dùng để định vị phần tử theo chiều ngang về bên trái hoặc phải

▼ **Phần tử Pseudo và các lớp Pseudo là gì?**

- **Phần tử pseudo** cho phép ta tạo các mục thường không tồn tại trong DOM.
  - ::before
  - ::after
  - ::first-letter
  - ::first-line
  - ::selection
- **Lớp pseudo** chọn các phần tử thông thường nhưng trong các điều kiện nhất định như khi người dùng di chuột qua liên kết.
  - :link
  - :visited
  - :hover
  - :active
  - :focus

▼ **CSS sprite là gì ?**

- CSS Sprite dùng cho kết hợp nhiều hình ảnh thành một hình ảnh lớn. Nó thường dùng cho biểu diễn icons. Các ưu điểm của nó là:

    - Giảm số lượng yêu cầu HTTP để lấy nhiều ảnh vì nó cho phép chỉ gửi một yêu cầu.
    - Nó giúp tải trước các nội dung giúp hiển thị các icon hoặc hình ảnh khi di chuột và các pseudo-state khác.
    - Khi có nhiều hình ảnh, trình duyệt sẽ thực hiện các lệnh gọi riêng biệt để lấy hình ảnh cho từng hình ảnh đó.

▼ Tích hợp css vào HTML có bao nhiêu cách ?

- inline: css trực tiếp trong element bằng thuộc tính style
- external: tạo 1 file css riêng và nhúng vào
- internal: style thông qua thẻ <style> đặc ở <head>

▼ Lợi thế của dùng translate() thay vì position absolute?

- Translate() không làm cho trình duyệt kích hoạt repaint layout, mà chỉ thực hiện soạn thảo.
- Còn position: absolute làm trình duyệt phải vẽ lại các luồng DOM.

→ Thế nên `translate()` đem về hiệu suất tốt hơn

▼ Liệu margin-top hoặc margin-bottom có ảnh hưởng đến các phần tử inline không?

- Không, nó không ảnh hưởng đến các phần tử inline. Các phần tử inline ở cùng dòng với nội dung của trang.

▼ Khi nào xảy ra DOM-reflow ?

- Dom-reflow là quá trình browser tính toán lại vị trí và hình dạng của element trong document, nhằm mục đích hiển thị lại 1 phần hoặc toàn bộ DOM
- Reflow xảy ra khi:
    - Chèn, xóa, update element trong DOM
    - Sửa đổi nội dung trang
    - Thay đổi style css

▼ Làm thế nào để căn giữa 1 thẻ p trong thẻ div ?

```
<div>
  <p>Hello</p>
</div>
```

```
// can giua theo chieu ngang
div {
  text-align: center;
}

// can giua theo chieu doc
div {
  position: relative;

  p {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    margin: auto;
  }
}
```

▼ Làm sao để căn giữa một div trong một div khác?

▼ transform

```
.cn {
  position: relative;
  width: 500px;
  height: 500px;
}

.inner {
  position: absolute;
  top: 50%; left: 50%;
  transform: translate(-50%,-50%);
  width: 200px;
  height: 200px;
}
```

▼ flex-box

```
.cn {
  display: flex;
  justify-content: center;
  align-items: center;
}
```

▼ grid

```
<div class="wrap_grid">
  <div id="container">vertical aligned text<br />some more text here
  </div>
</div>
```

```
.wrap-grid {
  display: grid;
  place-content: center;
}
```

▼ **Các các ẩn đi 1 phần tử trong css**

- display: none → phần tử sẽ không xuất hiện trong DOM
- visibility: hidden → phần tử có xuất hiện trong DOM nhưng không hiển thị lên màn hình
- position: absolute → set top, right, bottom hoặc left về số âm để di chuyển element ra ngoài màn hình
- transform: translateX(-999px) | translateY(-999px) | scale(0)
- opacity: 0 → ẩn đi, nó chỉ vô hình nhưng ta vẫn có thể add event lên nó

▼ **grid và flex khác nhau gì ?**

- grid là bố cục 2 chiều cả ngang và dọc → tiếp cận theo hướng layout
- flex 1 chiều, theo chiều ngang hoặc dọc → tiếp cận theo hướng nội dung
- Nếu biết rõ nội dung ta cần trình bày thì nên dùng flex ngược lại dùng grid
- Flex phù hợp web có bố cục đơn giản, grid thì phù hợp UI phức tạp

▼ **Grid là gì?**

- hệ thống layout 2 chiều theo trục x và y
- tổ hợp của đường ngang và dọc cắt nhau bao gồm các hàng và cột
- Các phần tử sẽ được đặt trên các hàng và cột này

▼ **Flexbox là gì?**

- là hệ thống bố cục một chiều (ngang hoặc dọc)
- giúp căn chỉnh bố trí những item trong container một cách linh hoạt ngay cả khi kích thước chưa xác định hoặc kích thước động
- Flex bao gồm:
  - flex-container (parent)
    - flex-direction
    - flex-wrap
    - flex-content
    - align-items
    - align-content
  - flex-item (child)
    - order
    - align-self
    - flex-grow
    - flex-shink
    - flex-basis

▼ **Giải thích Position trong css?**

> 💡 Normal flow là cách trình duyệt hiển thị những block element từ trên xuống dưới và mỗi block sẽ chiếm toàn bộ chiều ngang của container (div, p), ngược lại thì là inline(a, span, img)

- **static**: vị trí mặc định, theo dòng chảy thông thường của trang, tuy nhiên ta không thể set **L, T, R, B, z-index** cho nó
- **relative**: tuân theo quy luật của dòng chảy thông thường của trang nhưng có thể set các giá trị **L, T, R, B, z-index**
- **absolute**: element sẽ bị loại bỏ khỏi normal flow và nó sẽ nằm tương đối so với thuộc tính cha gần nhất của nó mà element cha đó phải có thuộc tính position là **relative**, **absolute**, **fixed** hoặc **sticky**.
    - Nếu không có thằng element cha nào mà có các thuộc tính trên thì nó sẽ nằm tương đối với root-element là thẻ html
- **fixed**: tương tự absolute nhưng khác là nó chỉ hiện thị tương đối so với thẻ html (root-element)
- **sticky**: là sự kết hợp giữa **relative** và **fixed**. Tức là nó vẫn nằm trong normal flow của trang như **relative** nhưng nó sẽ trở thành **fixed** nếu ta cuộn xuống đúng vị trí của nó. Và nó chỉ hoạt động trên container chứa nó.

▼ **overflow là gì ?**

- dùng để xử lý khi kích thước nội dung vượt quá kích thược container

▼ **Độ đặc hiệu, độ cụ thể trong css là gì ?**

- Tính đặc hiệu hay độ ưu tiên (specificity) là cách mà trình duyệt quyết định sẽ áp dụng thuộc tính css nào với một phần tử khi có nhiều quy tắc css cùng trỏ đến phần tử đó.
- Inline style sẽ được ưu tiên so với ID rồi đến giá trị lớp (pseudo-class hoặc attribute selector), universal selector **(*)** sẽ không có độ ưu tiên. ID Selector có độ ưu tiên cao hơn attribute selector.

▼ **Đơn vị trong CSS?**

- Absolute units: px, pt, cm, mm, …
- Relative units: rem, em, %, vw, vh, vmin, vmax, …
    - em: giá trị phụ thuộc vào phần tử cha gần nhất hoặc chính nó, được xác định thông qua thuộc tính font-size
    - rem: tương tự như em nhưng phụ thuộc vào root

- vw: tính theo tỉ lệ **chiều rộng khung nhìn** thiết bị. 1 vw = 1/100 width view-port.
  - Ví dụ: màn hình của bạn có chiều rộng 1100px thì 1vw = 11px
- vh: tương tự vw nhưng theo height view-port

▼ **Cách css hoạt động?**

- Ngôn ngữ CSS được thiết kế để sử dụng cùng với ngôn ngữ "đánh dấu" như HTML.
- CSS xác định cách các phần tử HTML được định dạng - kiểm soát bố cục, màu sắc, phông chữ của chúng, ...
- Khi trình duyệt hiển thị một document, nó phải kết hợp nội dung của document với thông tin style của nó.
- Nó xử lý document theo một số giai đoạn, mà chúng ta đã liệt kê bên dưới.
  1. Trình duyệt tải HTML (ví dụ: nhận nó từ mạng).
  2. Nó chuyển đổi HTML thành DOM.
  3. Sau đó, trình duyệt sẽ tìm nạp hầu hết các tài nguyên được liên kết với tài liệu HTML, chẳng hạn như hình ảnh và video được nhúng và CSS được liên kết.
  4. Trình duyệt phân tích cú pháp CSS đã nạp và sắp xếp các quy tắc khác nhau theo kiểu selector của chúng thành các "nhóm" khác nhau, ví dụ: phần tử, lớp, ID, ... Dựa trên các selector mà nó tìm thấy, nó sẽ tìm ra các quy tắc nên được áp dụng cho các nút nào trong DOM và đính kèm kiểu cho chúng theo yêu cầu (bước trung gian này được gọi là cây render).
  5. Cây render được bố trí trong cấu trúc mà nó sẽ xuất hiện sau khi các quy tắc đã được áp dụng cho nó.
  6. Hiển thị trực quan của trang được hiển thị trên màn hình.