Tài liệu: CSS Responsive

1. Định Nghĩa

CSS Responsive (Responsive Web Design - RWD) là phương pháp thiết kế web giúp giao diện tự động điều chỉnh kích thước, bố cục và nội dung để phù hợp với mọi thiết bị, từ điện thoại di động (iPhone, Samsung), máy tính bảng (iPad, Galaxy Tab) đến máy tính để bàn (laptop, PC). Mục tiêu là đảm bảo nội dung dễ đọc, dễ điều hướng và mang lại trải nghiệm người dùng mượt mà, bất kể kích thước màn hình (320px trên mobile hay 1920px trên desktop) hay độ phân giải (Retina, 4K).

Responsive CSS sử dụng các kỹ thuật như **đơn vị tương đối** (%, vw, vh, rem, em), **media queries**, **Flexbox**, **CSS Grid**, và **hình ảnh linh hoạt** để đạt được tính thích ứng.

2. Cách Hoạt Động

CSS Responsive hoạt động bằng cách kết hợp nhiều kỹ thuật CSS để xử lý các đặc điểm thiết bị (chiều rộng, chiều cao, độ phân giải, hướng màn hình). Dưới đây là giải thích chi tiết từng thành phần và cách chúng phối hợp, với các ví dụ cụ thể để minh họa.

2.1. Thẻ Meta Viewport

- **Vai trò**: Thẻ `<meta name="viewport" content="width=device-width, initial-scale=1.0">` trong HTML chỉ thị trình duyệt hiển thị trang web theo chiều rộng thực tế của thiết bị. Ví dụ, một iPhone 13 có chiều rộng vật lý 390px sẽ hiển thị đúng 390px CSS, thay vì giả lập màn hình desktop (thường 980px).
- **Cơ chế chi tiết**:
    1. Trình duyệt đọc thẻ meta viewport khi tải trang.
    2. `width=device-width` yêu cầu trình duyệt căn chỉnh chiều rộng nội dung với chiều rộng thiết bị.
    3. `initial-scale=1.0` đặt tỷ lệ phóng ban đầu là 1:1, ngăn trình duyệt tự động phóng to hoặc thu nhỏ.
- **Tác động thực tế**:
    - **Không có thẻ viewport**: Trang web trên iPhone có thể hiển thị như phiên bản desktop thu nhỏ, với chữ nhỏ xíu, buộc người dùng phải pinch-to-zoom để đọc.
    - **Có thẻ viewport**: Trang web hiển thị đúng tỷ lệ, chữ và hình ảnh rõ ràng, không cần zoom.
- **Ví dụ thực tế**: Trên trang web như Shopee, thẻ viewport đảm bảo danh sách sản phẩm hiển thị rõ ràng trên điện thoại mà không bị thu nhỏ như giao diện PC.

2.2. Đơn Vị Tương Đối

- **Vai trò**: Các đơn vị như **%, vw, vh, rem, em** cho phép kích thước phần tử tự điều chỉnh dựa trên kích thước màn hình hoặc font-size gốc, thay vì cố định bằng px.
- **Cơ chế chi tiết**:
    - **%**: Tính theo phần trăm chiều rộng/cao của phần tử cha.
      Ví dụ, `width: 80%` khiến phần tử chiếm 80% container cha.

- **vw/vh**: 1vw = 1% chiều rộng viewport, 1vh = 1% chiều cao viewport.
  Ví dụ, `font-size: 5vw` trên màn hình 360px sẽ cho chữ 18px (5% của 360px), trên màn hình 1920px sẽ là 96px.
- **rem/em**: rem dựa trên font-size gốc của `<html>`, em dựa trên font-size của phần tử cha.
  Ví dụ, nếu `html { font-size: 16px }`, thì `2rem` = 32px, nhưng nếu người dùng phóng to font trình duyệt, rem sẽ tự điều chỉnh.

- **Quy trình**:
  1. Trình duyệt tính toán kích thước viewport (chiều rộng, chiều cao).
  2. Các đơn vị tương đối được quy đổi thành pixel dựa trên viewport hoặc font-size hiện tại.
  3. Khi viewport thay đổi (xoay màn hình, thay đổi thiết bị), trình duyệt tính lại giá trị, tự động điều chỉnh bố cục.
- **Ví dụ thực tế**:
  - Một div có `width: 90%` sẽ là 324px trên mobile 360px và 1728px trên desktop 1920px.
  - Một tiêu đề có `font-size: 2rem` sẽ tự động lớn hơn trên thiết bị có font-size gốc lớn hơn (do cài đặt trình duyệt hoặc thiết bị).

Ví dụ minh họa:

```
.container {
  width: 90%; /* 90% chiều rộng container cha */
  padding: 2vw; /* 2% chiều rộng viewport */
  font-size: 1.5rem; /* Tương đối với font-size gốc */
}
.text {
  margin: 2vh; /* 2% chiều cao viewport */
}
```

2.3. Media Queries

- **Vai trò**: Media queries cho phép áp dụng CSS khác nhau dựa trên đặc điểm thiết bị, như chiều rộng (`max-width`, `min-width`), chiều cao (`max-height`), hướng (`orientation: portrait/landscape`), hoặc độ phân giải (`min-resolution`).
- **Cơ chế chi tiết**:
  1. Trình duyệt kiểm tra điều kiện trong media query, ví dụ: `@media screen and (max-width: 600px)`.
  2. Nếu điều kiện đúng (màn hình ≤ 600px), các quy tắc CSS bên trong được áp dụng.
  3. Nếu điều kiện sai, trình duyệt bỏ qua và áp dụng quy tắc mặc định hoặc media query khác.
  4. Media queries được xử lý theo thứ tự, quy tắc sau có thể ghi đè quy tắc trước nếu cụ thể hơn.
- **Quy trình**:
  - Trình duyệt liên tục kiểm tra viewport khi tải trang hoặc khi người dùng thay đổi kích thước màn hình/xoay thiết bị.
  - Các quy tắc CSS tương ứng được kích hoạt ngay lập tức, thay đổi bố cục, kích thước chữ, hoặc ẩn/hiện phần tử.
- **Ví dụ thực tế**:
  - Trên mobile (dưới 600px), ẩn thanh menu desktop và hiển thị nút hamburger.
  - Trên tablet (601px-1024px), hiển thị menu dạng cột thay vì hàng.

- Trên desktop (>1024px), hiển thị menu dạng hàng với các mục đầy đủ.

- **Ví dụ minh họa:**

```
/* Mặc định cho desktop */
.menu {
  display: flex;
  flex-direction: row;
}
.hamburger {
  display: none;
}
/* Mobile: dưới 600px */
@media screen and (max-width: 600px) {
  .menu {
    display: none; /* Ẩn menu */
  }
  .hamburger {
    display: block; /* Hiện nút hamburger */
  }
}
/* Tablet: 601px-1024px */
@media screen and (min-width: 601px) and (max-width: 1024px) {
  .menu {
    flex-direction: column; /* Menu dạng cột */
  }
}
```

2.4. Bố Cục Linh Hoạt (Flexbox và CSS Grid)

- **Flexbox**:
  - **Vai trò**: Tạo bố cục 1 chiều (hàng hoặc cột) linh hoạt, tự động điều chỉnh kích thước và vị trí các phần tử con.
  - **Cơ chế chi tiết**:
    1. Container được đặt `display: flex`.
    2. Các thuộc tính như `flex-wrap: wrap` cho phép phần tử xuống dòng khi không đủ chỗ.
    3. `flex: 1 1 200px` khiến phần tử con co giãn linh hoạt, với chiều rộng tối thiểu 200px.
    4. Khi viewport thu nhỏ, các phần tử tự động tái sắp xếp (xuống dòng hoặc thu nhỏ).
  - **Ví dụ thực tế**: Một danh sách sản phẩm 4 cột trên desktop tự động chuyển thành 2 cột trên tablet và 1 cột trên mobile.
  - **Ví dụ minh họa**:

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

```
  gap: 20px;
}
.item {
  flex: 1 1 200px; /* Tối thiểu 200px, co giãn đều */
  background: #f4f4f4;
  padding: 10px;
}
```

- **CSS Grid**:
  - **Vai trò**: Tạo bố cục 2 chiều (hàng và cột) với khả năng điều chỉnh tự động dựa trên viewport.
  - **Cơ chế chi tiết**:
    1. Container được đặt `display: grid`.
    2. `grid-template-columns: repeat(auto-fit, minmax(200px, 1fr))` tạo số cột linh hoạt, mỗi cột tối thiểu 200px và co giãn đều.
    3. Khi viewport thay đổi, Grid tự động tái cấu trúc số cột/hàng.
  - **Ví dụ thực tế**: Một gallery ảnh hiển thị 4 cột trên desktop, 2 cột trên tablet, 1 cột trên mobile, chỉ với một dòng CSS.
  - **Ví dụ minh họa**:

```
.gallery {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 15px;
}
.gallery img {
  width: 100%;
  height: auto;
}
```

2.5. Hình Ảnh Linh Hoạt

- **Vai trò**: Đảm bảo hình ảnh không vượt quá container và giữ tỷ lệ trên mọi màn hình.
- **Cơ chế chi tiết**:
  1. `max-width: 100%` giới hạn chiều rộng hình ảnh không vượt quá container.
  2. `height: auto` giữ tỷ lệ hình ảnh, tránh méo mó.
  3. Kỹ thuật nâng cao: Sử dụng `srcset` trong HTML hoặc `image-set()` trong CSS để tải hình ảnh phù hợp với độ phân giải thiết bị (ví dụ, ảnh nhỏ cho mobile, ảnh lớn cho desktop).
- **Quy trình**:
  1. Trình duyệt tính chiều rộng container.
  2. Hình ảnh co giãn để vừa container, giữ tỷ lệ.
  3. Nếu dùng `srcset`, trình duyệt chọn nguồn ảnh phù hợp dựa trên DPI hoặc kích thước màn hình.
- **Ví dụ thực tế**: Một ảnh sản phẩm 1200px trên Shopee hiển thị full-width (360px) trên mobile mà không bị cắt, nhưng vẫn sắc nét trên desktop 1920px.
- **Ví dụ minh họa**:

```
  img {
    max-width: 100%;
    height: auto;
  }
  /* Sử dụng image-set cho độ phân giải */
  .responsive-img {
    background-image: image-set(
      url('low-res.jpg') 1x,
      url('high-res.jpg') 2x
    );
  }
```

2.6. Quy Trình Tổng Thể

1. **Khởi tạo**: Trình duyệt đọc thẻ meta viewport để xác định tỷ lệ hiển thị (ví dụ, 360px trên iPhone).
2. **Tính toán viewport**: Trình duyệt lấy chiều rộng, chiều cao, và đặc điểm thiết bị (portrait/landscape, DPI).
3. **Áp dụng đơn vị tương đối**: Các giá trị như %, vw, rem được quy đổi thành pixel dựa trên viewport hoặc font-size.
4. **Kiểm tra media queries**: Trình duyệt áp dụng quy tắc CSS phù hợp với điều kiện (ví dụ, `max-width: 600px` cho mobile).
5. **Sắp xếp bố cục**: Flexbox hoặc Grid tái cấu trúc các phần tử dựa trên không gian viewport.
6. **Điều chỉnh hình ảnh**: Hình ảnh co giãn để vừa container, hoặc tải phiên bản phù hợp qua `srcset`.
7. **Cập nhật động**: Khi người dùng xoay thiết bị, thay đổi kích thước cửa sổ, hoặc chuyển thiết bị, trình duyệt lặp lại quy trình để cập nhật giao diện.

3. Cú Pháp

3.1 Đơn Vị Tương Đối

```
.container {
  width: 90%; /* 90% chiều rộng container cha */
  padding: 2vw; /* 2% chiều rộng viewport */
  font-size: 1.5rem; /* Tương đối với font-size gốc */
}
.text {
  margin: 2vh; /* 2% chiều cao viewport */
}
```

3.2 Media Queries

```
/* Mặc định */
.navbar {
  display: flex;
  justify-content: space-between;
}
/* Mobile */
```

```
@media screen and (max-width: 600px) {
  .navbar {
    flex-direction: column;
    align-items: center;
  }
}
/* Tablet */
@media screen and (min-width: 601px) and (max-width: 1024px) {
  .navbar {
    flex-wrap: wrap;
  }
}
```

3.3 Flexbox

```
.container {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
}
.item {
  flex: 1 1 200px; /* Co giãn, tối thiểu 200px */
}
```

3.4 CSS Grid

```
.gallery {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 15px;
}
```

3.5 Hình Ảnh Linh Hoạt

```
img {
  max-width: 100%;
  height: auto;
}
```

4. Ví Dụ Minh Hoạ

Dưới đây là ví dụ hoàn chỉnh để minh họa CSS Responsive:

Ví dụ: Trang Web Sản Phẩm

```
<!DOCTYPE html>
<html lang="vi">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Responsive Product Page</title>
  <style>
    * {
      box-sizing: border-box;
      margin: 0;
      padding: 0;
    }
    body {
      font-family: Arial, sans-serif;
      line-height: 1.6;
    }
    .container {
      width: 90%;
      margin: 0 auto;
      display: grid;
      grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
      gap: 20px;
      padding: 20px;
    }
    .product {
      background: #f4f4f4;
      padding: 20px;
      border-radius: 8px;
      text-align: center;
    }
    img {
      max-width: 100%;
      height: auto;
      border-radius: 8px;
    }
    h2 {
      font-size: 1.5rem;
    }
    p {
      font-size: 1rem;
    }
    @media screen and (max-width: 600px) {
      .container {
        grid-template-columns: 1fr;
      }
      h2 {
        font-size: 1.2rem;
      }
```

```
      p {
        font-size: 0.9rem;
      }
    }
    @media screen and (min-width: 601px) and (max-width: 1024px) {
      .container {
        grid-template-columns: repeat(2, 1fr);
      }
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="product">
      <img src="https://via.placeholder.com/300" alt="Product 1">
      <h2>Sản Phẩm 1</h2>
      <p>Mô tả sản phẩm 1, hiển thị tốt trên mọi thiết bị.</p>
    </div>
    <div class="product">
      <img src="https://via.placeholder.com/300" alt="Product 2">
      <h2>Sản Phẩm 2</h2>
      <p>Mô tả sản phẩm 2, tự động điều chỉnh kích thước.</p>
    </div>
    <div class="product">
      <img src="https://via.placeholder.com/300" alt="Product 3">
      <h2>Sản Phẩm 3</h2>
      <p>Mô tả sản phẩm 3, tối ưu cho mobile và desktop.</p>
    </div>
  </div>
</body>
</html>
```

5. Ứng Dụng Thực Tế

CSS Responsive được sử dụng trong các trường hợp sau:

- **Thương mại điện tử**: Shopee hiển thị danh sách sản phẩm 4 cột trên desktop, 2 cột trên tablet, 1 cột trên mobile, với hình ảnh và nút mua hàng co giãn phù hợp.
- **Blog tin tức**: VNExpress điều chỉnh kích thước chữ, ẩn cột phụ trên mobile, và hiển thị quảng cáo ở vị trí tối ưu theo thiết bị.
- **Ứng dụng web**: Gmail dùng Flexbox để xếp email dạng danh sách trên mobile và dạng lưới trên desktop, với media queries để điều chỉnh padding/font-size.
- **Portfolio**: Nhà thiết kế sử dụng Grid để hiển thị dự án dạng lưới trên desktop, cuộn dọc trên mobile, với hình ảnh tự co giãn.
- **Mạng xã hội**: X sử dụng responsive để hiển thị bài viết, hình ảnh, và video ở dạng cột đơn trên mobile, hoặc đa cột trên desktop.

Lợi ích:

- **Trải nghiệm người dùng**: Nội dung dễ đọc, nút dễ bấm trên mọi thiết bị.
- **SEO**: Google ưu tiên trang responsive, tăng thứ hạng tìm kiếm.
- **Tiết kiệm chi phí**: Một giao diện duy nhất thay vì thiết kế riêng cho mobile/desktop.

Lưu ý khi triển khai:

- Kiểm tra trên thiết bị thực tế (iPhone, Android, iPad) và trình duyệt (Chrome, Safari, Firefox).
- Dùng Chrome DevTools để mô phỏng các kích thước màn hình (320px, 768px, 1200px).
- Tối ưu hiệu suất: Nén hình ảnh, dùng srcset, và loại bỏ CSS không cần thiết.