**Một vài nguyên tắc cơ bản khi test responsive web design**

* Text, control và hình ảnh được căn chỉnh đúng cách
* Higlight, đổi màu sắc khi hover và ở trạng thái lựa chọn
* Vùng có thể click thích hợp
* Màu sắc, độ phủ bóng và độ dốc phù hợp
* Kiểm tra padding chính xác
* Văn bản, hình ảnh, điều khiển và khung không chạy vào các cạnh của màn hình
* Kích cỡ phông chữ, kiểu dáng và màu sắc phù hợp với từng loại văn bản
* Scroll đoạn văn bản đã nhập (dữ liệu đã được nhập) được hiển thị chính xác
* Không bao giờ hiển thị thanh ngang của trang.
* Có thể đọc được trang web trên mọi độ phân giải.
* Nội dung được xác định là "quan trọng" cần phải được hiển thị trong tất cả các điểm phá vỡ. **Điểm phá vỡ (breakpoint)** Mỗi điểm phá vỡ đòi hỏi sự thích ứng của cách bố trí, với các mô-đun thay đổi vị trí và các quy tắc hiển thị. Một khả năng khác là có bố cục fluid, với text và ảnh có tỷ lệ thích hợp tương đối với chiều rộng của trang. **Mô-đun (modules)**

Kiểm tra vị trí của mô-đun khi bạn co và mở rộng cửa sổ trình duyệt hoặc khi bạn xoay thiết bị di động. Các mô-đun khác nhau có thể biến mất khi bạn chuyển từ máy tính để bàn sang điện thoại di động nhưng chắc chắn rằng bạn biết chính xác những mô-đun nào nên được hiển thị.

* Hiển thị chính xác tất cả hình ảnh website trên các thiết bị có kích thước màn hình khác nhau.
* Căn chỉnh và định vị hợp lý các khối văn bản và tiêu đề.
* Kích thước của hình ảnh, văn bản và các khối trang web khác phải được thay đổi tương ứng với kích thước màn hình.
* Màu sắc của tất cả các thành phần trang web phải được giữ nguyên, bất chấp phần mở rộng và đặc điểm của màn hình.
* Khi người dùng nhập văn bản, nó phải được hiển thị chính xác với kích thước ghim, phông chữ và tab.
* Duy trì tính chính xác của việc tái tạo màu sắc khi tương tác với giao diện - khi tab nằm trên các nút và liên kết, khi di chuột, v.v.
* Trang sẽ được cuộn mà không gặp bất kỳ sự cố và lỗi nào.
* Khả năng nhấp chuột của các vùng thích hợp trên các trang của trang web.

**1. Tùy devices chứ**

**2. Ngoài padding/margin element trên dưới trái phải thì còn cần check**

+ Font sizes, styles, colors

+ Line height, spacing

+ Width, height element ( button, text field, image)

+ Color scheme/ contrast

+ Alighment

+ transition/ animation

+ gesture functions

**Steps:**

Check phóng to, thu nhỏ màn hình

1. Phongs to màn hình trên 100%

2. Thu nhỏ màn hình dưới 100%

3. Check hiểr thị ở các độ phân giải màn hình 600x800

Result: 1,2,3 layout ko bị vỡ, các hình ảnh vẫn nét, Aagent làm gì trong heej toongsẹ

**3. Check với designer xem, tùy dự án/ yêu cầu**

**1. Vùng responsive thì không có cố định, mà phải tùy thuộc vào đối tượng thiết bị đáp ứng. Tuy nhiên khi code CSS thì thường có các vùng dưới đây (mình tham khảo trên w3school):**

- width <600px: phones

- 600 -> 768px: portrait tablets and large phones

- 768 -> 992p: landscape tablets

- 992 -> 1200px: laptops/desktops

- >1200px: 992 -> 1200px

https://www.w3schools.com/css/css\_rwd\_mediaqueries.asp

Tuy nhiên, chính xác nhận thì phải dựa vào phân giải của thiết bị, không phải cứ máy bé hơn thì độ phân giải bé hơn. Bạn có tham khảo độ phân giải của các màn hình ở website này: https://screensiz.es/

2. Perfect pixel là nói đến pixel nên bạn cần check tất cả nhưng thông tin liên quan đến kích thước pixel: font, height, width, padding, margin, line-height.. tất cả các attibute có đơn vị có thể quy đổi ra px, và khoảng cách giữa các element.

3. Vào page screen size bên trên, tìm một device bất kỳ có width/height 1200px rồi set theo là được. Khi code thì CSS chủ yếu theo width thôi, nên height đủ để theo design là được, nên để tỷ lệ theo frame trên design.

4. Theo đúng quy định thì phải check lại nếu phát hiện sai nhé, vì một số UI tùy thuộc vào data. Còn tùy vào bạn "fake data" như thế nào. Nếu data của bạn là từ code/database thì thường sẽ không có sai lệch, còn nếu bạn F12 để sửa text/data thì rất dễ lệch nhé, vì có nhiều CSS được sinh ra dựa vào data, và không update lại khi bạn sửa element (VD tooltip)

https://developer.chrome.com/docs/devtools/

**CÁC BẠN THƯỜNG CHECK NHỮNG GÌ CỦA WEB KHI SỬ DỤNG F12 (INSPECT) ĐỂ CHECK?**

e thường dùng:

- element: tìm element bằng xPath, hoặc để xem/edit attr

- console: xem dev có quên cmt dòng debug nào ko :))) jkd

- network: xem pageload với res

- application: xoá data của page khi bị cache

- lighthouse: xem điểm SEO :v

API, bắt Xpath, chỉnh responsive, CSS. lấy refresh token.

Tab Console và Network nha 😘

Còn tab Element: What Font

CSS Viewer

Grid Ruler

JSONVue

Check console: Check log messages , Check status code, nhiều khi bị lỗi API, vào Console để xác định nó bị lỗi 50, 409, 400...

Ko get được property...

Extension for check color code, do px, fontsize, height, width: vô extension của google r tìm:

Measure Dimensions

ColorZilla

CSSViewer

Window Resizer

Responsive Viewer

**HOW TO WRITE TESTCASE CORRECT:**

Bạn nên liệt kê tất cả các màn hình cần kiểm tra. Mỗi màn hình là 1 dòng (trong Ms Excel).

Mỗi cột là Hình ảnh sản phẩm, Tên sản phẩm,… các thông tin cần kiểm tra khác.

Mỗi ô giao nhau của dòng và cột bạn ghi giá trị cần kiểm tra. Ô nào không có (ko tồn tại, ko hợp lệ,… bạn điền “-“)

Mỗi ô có giá trị là cần đi kiểm tra (là 1 test case)

Trên jira thì mỗi màn hình cần ktra là 1 task.

Trong phần mô tả task đó e gạch đầu dòng các thứ cần ktra.

hoặc tạo 1 task: kiểm tra width

Và mỗi subtask là 1 màn hình cần kiểm tra.

Trong phần mô tả của mỗi subtask bạn ghi thông số cần kiểm tra.

Test xong màn hình nào cập nhật trạng thái pass/fail luôn.

về cách làm việc em nên hỏi lại chị lead sai chỗ nào, cải thiện như nào, rõ ràng rành mạch. mà trước khi hỏi thì tự coi lại những vấn đề em đang nghi vấn, ghi rõ ràng ra giấy trước rồi mới hỏi. ko sợ hỏi ngu chỉ sợ ko hỏi. giải quyết vấn đề trên rồi mới nói tới test case thiếu gì.

**PROCEDURE TEST**

Truy cập đc vào web,

Test UI cho web

Test hết function của web ở mỗi role,

Test flow navigation của web

Test performance của web

Load test cho web

Test api cho web

Test web trên multi browsers

Test responsive mode của web

Test hết đống trên trên mobiles

**Test scenario** hay còn gọi là kịch bản test. Nó tương đi theo 1 luồng hoạt động giống như người dùng lúc thao tác ấy. Template nó như test case thôi nhưng thứ tự các case nó thành 1 luồng từ đầu đến cuối của 1 thao tác.

Ví dụ:

1. Login account ko tồn tại

2. Login account có tồn tại

3. Kiểm tra dữ liệu hiển thị màn dashboard(tổng quan)

4. Click cho tiết thông tin gì đó trên dashboard

5. Click menu A

6. Tạo mới dữ liệu user invalid data

7. Tạo mới dữ liệu user valid data

8. Kiểm tra dữ kiệu sau tạo mới thành công

9. Thực hiện edit thông tin user......

**Cơ bản là cần đủ thông tin để đọc vào cái title là biết ngay vấn đề, mình thường hay dùng cú pháp kiểu: Bị gì + khi nào + ở đâu.**

Ví dụ: hiển thị text A thay vì text B khi click nút C trên màn hình D

Về actual thì có thể mô tả tương tự title của bug

Về expected thì mình hay dùng cú pháp kiểu: cái này nên như thế này

Ví dụ: text B nên (should be) hiển thị trên màn hình D sau khi click nút C

Tuy nhiên còn tuỳ vào là bug gì và phản ứng của bug ra sao thì sẽ có title, actual và expected linh động để phù hợp, chứ ko nhất thiết phải rạp khuôn

Em nghĩ c nên xào nấu lại CV xíu, test trên những layer nào (UI, API, functional, ..), impact ra sao, feature phức tạp ko. Càng ko có kinh nghiệm thì mình thêm mắm muối xíu cho nó ngon ngon xem sao chị.

Kinh nghiệm là thử và sai, check kết quả và lặp lại. Mỗi lần test 1 kiểu CV nên rải tầm tầm vài cty thôi. Chịu khó sửa CV bám theo JD.

Em mới ra trường và vào làm ở 1 cty được 2 hôm. Hôm nay thì chị leader bảo 1 bé (nhỏ tuổi hơn em nhưng làm cũng lâu rồi) train cho em làm file A. Hiện file A em vẫn chưa làm xong, mai em sẽ hoàn thành nốt. Vậy khi em làm xong thì em báo cáo đã xong và nộp file đó cho chị leader đúng không mọi người 😊

Trường hợp này sau khi làm xong e gửi cho bạn hướng dẫn em, nhờ bạn kiểm tra trước. Sau đó gửi mail cho bạn leader và cc bạn hướng dẫn vào