TASK DESCRIPTION

Contents

Synopsys

This **FINAL INDIVIDUAL** task combines all your knowledge and skills.
Every student will defend it to interviewers personally.

Your goal is to create minimalistic e-commerce website.
Implementation is expected as set of static HTML files with responsive layout with (RWD techniques, Fluid layout).
You should add necessary basic JS.

Usage of CSS/JS libraries/frameworks is disallowed (if there are no special approval) and penalty might be applied if you use one.

There is list of extra miles that are not required to implement but will be a bonus when implemented.

You can use Babel CLI with babel-preset-env to compile ES2015 syntax to ES5.1 (target should include IE11!). You shouldn't use other bundling tools and penalty might be applied if you use one.

If you want to use polyfill for some browser features – you should ask if it is approved for this task.

Pages to make

- Homepage (`1_home`)
- Category Listing (`2_category-all`)
- Product Details Page (`3_product-details`)
- Shopping bag (`4_shopping-bag`)
- "Thank you" page (when user completes order) (`5_thank-you`)

Acceptance criteria

Your solution would be evaluated be the following criteria:

Markup

- For screen width larger than 414px (including) pages should use mobile layout (414px files).
- For screen width larger than 768px (including) page should use tablet layout (768px files). Page layout can be switched from mobile to tablet and from tablet portrait layout to landscape layout (1024px files) earlier than 768px and 1024px respectively if you see that this wouldn't break layout.
- For screen width larger than 1240px page should use desktop layout (1440px files).
- Main content area shouldn't stretch more than 1240px.
- Used HTML tags should be semantic.
- Semantic names of classes and ids.
- Correct using of CSS3 rules.
- Markup must be compatible with list of supported browsers.
- All links and buttons have noticeable hover and active state (see style guide for reference).
- Penalty might be applied to noticeably deviations from design. Do your best to make your layout as close as possible to design.

NOTE: Primary source of layout is `PNG` folder. Each page has multiple screenshots for various window width. You should reflect those changes of layout in your work. PSD files are secondary, as they were adapted to simplify some aspects - use them just for reference (font properties, pick colors).

NOTE: There should be no horizontal scroll bar unless device with is less than 414px. In this case, layout should not be broken and horizontal scroll appears.

Functionality

List or requirements for basic functionality (navigation, user interaction)

All pages (header)

- Click on search button expands search field (see screenshots with `search-expanded` in file name) and makes it focused.
- If search field is empty – click on search button should collapse search field.
- If search field contains any value – click on search button should navigate user to "Category Listing" page. (Only navigation, no filtering should be applied).
- Click on logo should navigate user to "Homepage"
- Click on menu items should navigate user to "Category Listing" page.

Homepage

- Click on squares with text should navigate user to "Product Details" page.
- Click on squares without text should navigate user to "Category Listing" page.
- Slider at the bottom of "Homepage" should scroll across static images when user clicks on arrows (use images of same width for simplicity).
- Click on left arrow should do nothing if left edge of the first item is visible to user.
- Click on right arrow should do nothing if right edge of the last item is visible to user.
- Click on images in carousel should navigate user to "Category Listing" or "Product Details" pages in alternating order (click on first item should navigate user to Category Listing, click on second - Product Details, click on third - Category Listing and so on.)

Category Listing page

- Use stock images for products in categories
- Click on block with product item (image, title, price) should navigate user to "Product Details" page.
- Slider at the bottom of the page should follow all requirements as for "Homepages"

Product details page

- Implement photos switcher
    - Click on thumbnail should replace main image with full size image
    - Thumbnail with currently newly displayed image should be hidden and thumbnail with previously displayed image shown.
- Click on buttons in Size section should highlight it. Only one size can be selected at time.
- Click on "Add to cart" button should update items count in header.
  Items count should be increased by one each time user click button.

NOTE: Full cart implementation is listed in **Extra miles** section.

Shopping bag (Cart)

- When user navigates to shopping bag page, it displays 3 different items with Qty field equal to 1.
- Items count in header should be equal sum of Qty fields for all items.
- Change in Qty field should update items count in header.
- Click on "X" button for product should remove it from page and decrease count in the header by Qty field value for this product.
- Click on "Order now" should navigate user to "Thank you page".
- Click on "Continue shopping" should navigate user to "Homepage".

- List of items in cart and their counts should reset after page reload until full cart functionality (with persistent storage) will be implemented.

NOTE: Full cart implementation is listed in **Extra miles** section

Extra miles

This is list of bonus tasks that are not required to implement, but will give you a bonus if implemented.

Shopping bag

(full shopping bag functionality with JS)

- Create one more "Product details" page with different name and price (so you will be able to add different items to shopping bag)
- Create persistent storage for Shopping bag data so it is saved when user navigated across the site (use LocalStorage API, use `http-server` Node.js package)
- (All pages) Items count in header should always display actual number of products in shopping bag.
- (Product details) Click on "Add to cart" button should add product to persistent shopping bag data storage
    - If two items in bag has same name and size - they should be combined on Shopping bag page. Qty field in shopping bag item should reflect number of similar items.
    - If two items in bag has same name, but different size - they shouldn't be combined.
- (Product details) Every time when user click on "Add to cart" button - items count header should be recalculated.
- (Shopping bag) Total price should be recalculated on every product addition, removal or change of Qty field value with respect to items Qty and prices.
- (Shopping bag) Every time when user click on "X" for product - item should be removed from the Shopping bag (page and persistent storage). Items count in header and total price below items list should be recalculated.
- (Shopping bag) "Order now" should remove all items from persistent storage and navigate user to "Thank you page".

You can create and use your own stub/mock data.

Search field keyboard functionality

- Enter key press while search filed is active should navigate user to "Category Listing" page.
- Escape kay press while search field is active should collapse it.

Simple email form validation

- When user clicks on "Add" button in "Your e-mail" field it should be validated.
  If user entered "valid" e-mail – "We added you to our list" should be displayed above field.
  If user entered invalid e-mail – "Please enter valid e-mail" should be displayed above field.
- E-mail is considered as "valid" if it:
    - Has one "@" character

- Has at least one "." character after "@"
- Has at least one character before “@"
- Has at least one character between "@" and "."
- Has at least one character after "." (which is after "@")

"Valid" examples: test@example.com, a@a.a, 1@1.1

"Invalid" examples: @example.com, email@test

Browser support

Your implementation should be compatible with latest versions of following browsers:

- Chrome
- Firefox
- Edge
- IE11

Tools

- Code editor of your choice
- Chrome DevTools
- Adobe Photoshop
- `http-server` package (https://www.npmjs.com/package/http-server), `browser-sync` (https://www.npmjs.com/package/browser-sync) or any other server that can server static html files to avoid problems with LocalStorage.
- etc.

Style guide

Use `style guide.pdf` as reference for colors, sizes, fonts, hover state etc.

Recommended file structure

Files and folders structure may vary depending on your implementation.