COMP 307

Principles of Web Development

Lecture 8

Unit 3 – Frontend Design

Planning a website build

Contents

Design
SE Techniques
Team Management

Class Outline

- Frontend Design
  - Elements
  - Shapes
  - Viewports & Responsive
  - Single vs Multi-paged
  - Interactions
- Planning a website build
  - Software Engineering Techniques
  - Case study analysis
- Team Management

Contents

Design
SE Techniques
Team Management

Readings

- Full Stack Developer
  - Chapter 3, 4 & 9
- Internet Resources
  - https://www.indeed.com/career-advice/career-development/types-of-website-design
  - https://webflow.com/blog/types-of-websites
  - https://www.flux-academy.com/blog/12-popular-website-types-and-how-to-design-them

Contents

Design
SE Techniques
Team Management

Important META Ideas

Elements of a Good Design

- Small group of decision makers
- Website has a clear focus
- Dare to do less, but very well
- Keep site layout and architecture simple and extensible (less is more)
- Develop the content with the website's stakeholders (the ones who care, use, and fund the website)
- Test early, test incrementally, don't trust
- Defensive programming and design

Contents

Design
SE Techniques
Team Management

Budget Threat: Scope Creep

- When you run out of time or money
- Scope Creep: plans change multiple times
- Solution:
  - Define functional requirements (what they want to do)
  - Define technical requirements (speed & how to build it)
  - Do that early
  - Create two versions of the program
    - Version 1 you are building now
    - New ideas, changes, add to the documentation for version 2

Contents

Design
SE Techniques
Team Management

Spinning Straw into Gold

- Knowing which things to build first
  - What they need versus fancy additions
- Controlling scope creep well
  - Saying no! Planning well.
  - Leaving things for version 2
- Controlling heavy technology
  - Better to code some things by hand than to download a 1GB library because you need a single object.
  - Technology means more resources
    - Faster CPU = $$
    - More RAM = $$
    - Reduces the number of packets processed by the server per unit of time

Contents

Design
SE Techniques
Team Management

COMP 307
Principles of Web Development

Frontend Design

Planning a website build

Contents

Design
SE Techniques
Team Management

The Elements of a Plan

- Meta decisions
  - Type of website
  - Website flow
  - Website shape (or layout)
  - Standardizing elements
  - Colors and feel
- Some software engineering
  - Creating the design document
  - Organizing your team
  - Quality control and deployment

Before you build your website you need to make these meta decisions.

Planning on paper is better than talking!!

Contents

Design
SE Techniques
Team Management

Type of Website

- Single Page
  - The entire website exists within a single HTML document.
  - The menu moves the user up and down the page or causes web elements to hide and reveal or loads content from database.
- Multi-paged
  - This webpage has a landing page with a menu.
  - Each menu option links to another webpage dedicated to that option. Each webpage may contain links to further webpages.
- Static
  - Other than hyperlinks to other pages, this website is primarily informational with no (few) interactive elements.
- Dynamic
  - A website with a lot of interactive elements: mouse overs, upload/download, hiding/revealing, drawing, forms, queries, etc.

Contents

Design
SE Techniques
Team Management

Website Flow

- Fixed design
  - The website is built to function properly in one type of layout only.
  - For example, when you play a video game it forces you to go into full screen mode. The website table is set to 800 px wide.
- Responsive design
  - A website that has defined multiple **viewports**. These are **fixed** design layouts for a particular screen resolution.
  - For example, website that has a fixed PC design, but can switch to a different fixed tablet design, and can switch to a different fixed phone design.
- Liquid design
  - The website elements float on the page and will automatically reorder themselves as the screen resolution and device viewport changes.
  - This gives you less control over the look as compared to Responsive.

Contents

Design
SE Techniques
Team Management

Website Flow

- Modal
  - A single page that has a single function and restricts movement.
  - Eg: login page, registration page, submit form, survey
- Multi-modal
  - Like a dashboard
  - Many “boxed” regions on a single page. Each region displays information and a menu based on the “box’s” function.

Contents

Design
SE Techniques
Team Management

Shapes and Layouts

- The F and E Shape
  - Studies show user's eyes move in an F or E pattern
  - Websites that have this shape? Header on top, vertical menu on left, content on right, footer on bottom.
- The Z Shape
  - Users from western countries also use the Z eye movement.
  - Top menu scan horizontally, move down to read from left-to-right.
- The I Shape
  - Vertical scrolling websites with a menu at the top, possibly credits at the bottom. twitter, tiktok, etc
- Split Screen
  - Provides multiple shapes on the same page.

Contents

Design
SE Techniques
Team Management

Website Elements

- Header
  - Title of website (maybe company name)
  - Common info that appears on every page
  - May contain links and/or logo image
- Side bar
  - Informational, hyperlinks, or submenu
- Menu
  - A method to switch the view from one page to another page
- Body
  - The content of a particular page
  - May have pictures, diagrams, hyperlinks
- Footer
  - Copywrite information appears on every page
  - Less important links that appear on every page

Contents

Design
SE Techniques
Team Management

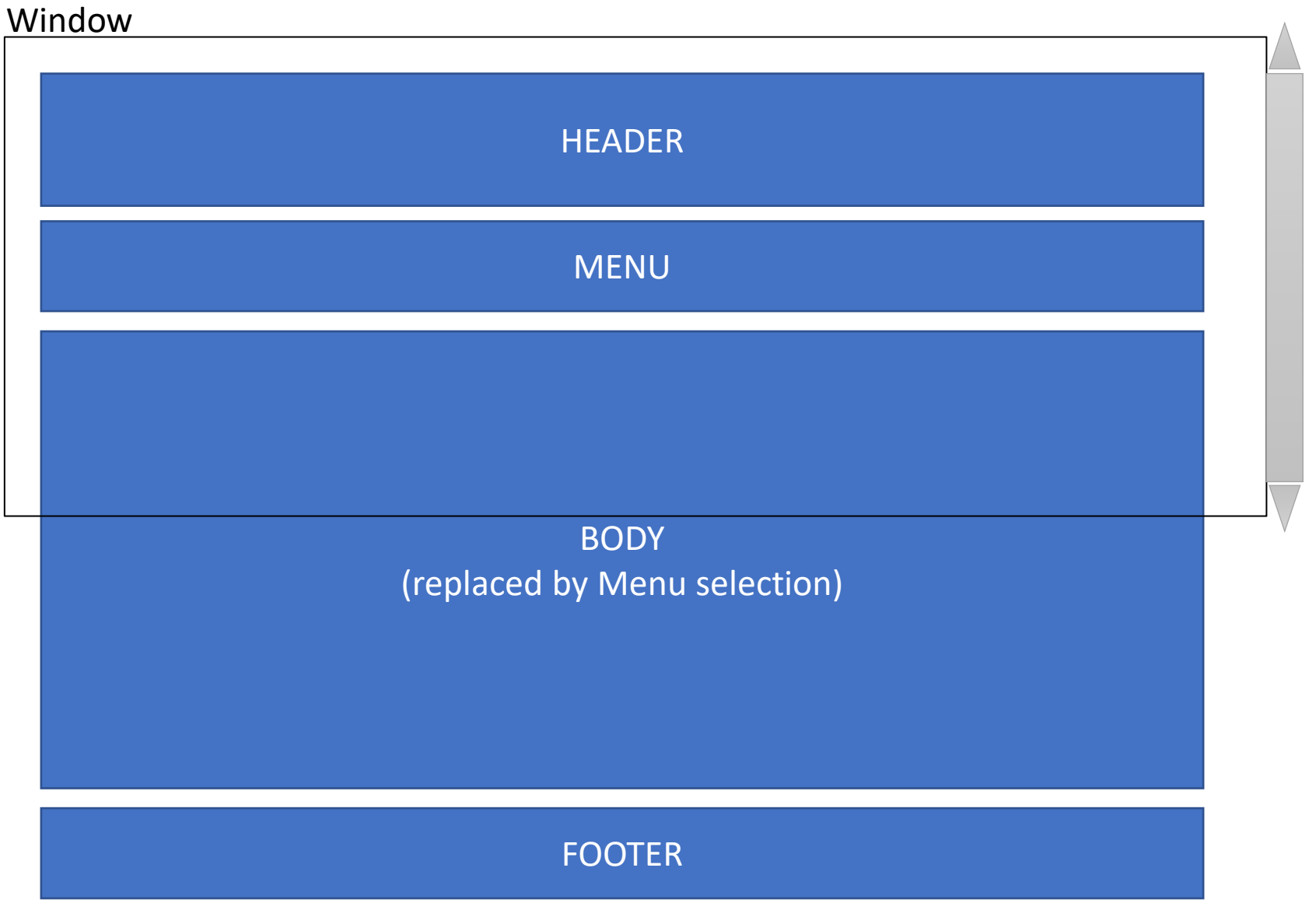
Website Elements

- Popup
  - A window that appears above the webpage
  - Modal popups expect a response before returning to page
  - Informational, announcement, warning, or input request
- Form
  - Consolidated region on page requesting information
  - Connected to backend program via submit button
- Information "Box" (area) (aside)
  - An area of the page interrupting the page's primary flow
  - A shaped region containing information

Contents

Design
SE Techniques
Team Management

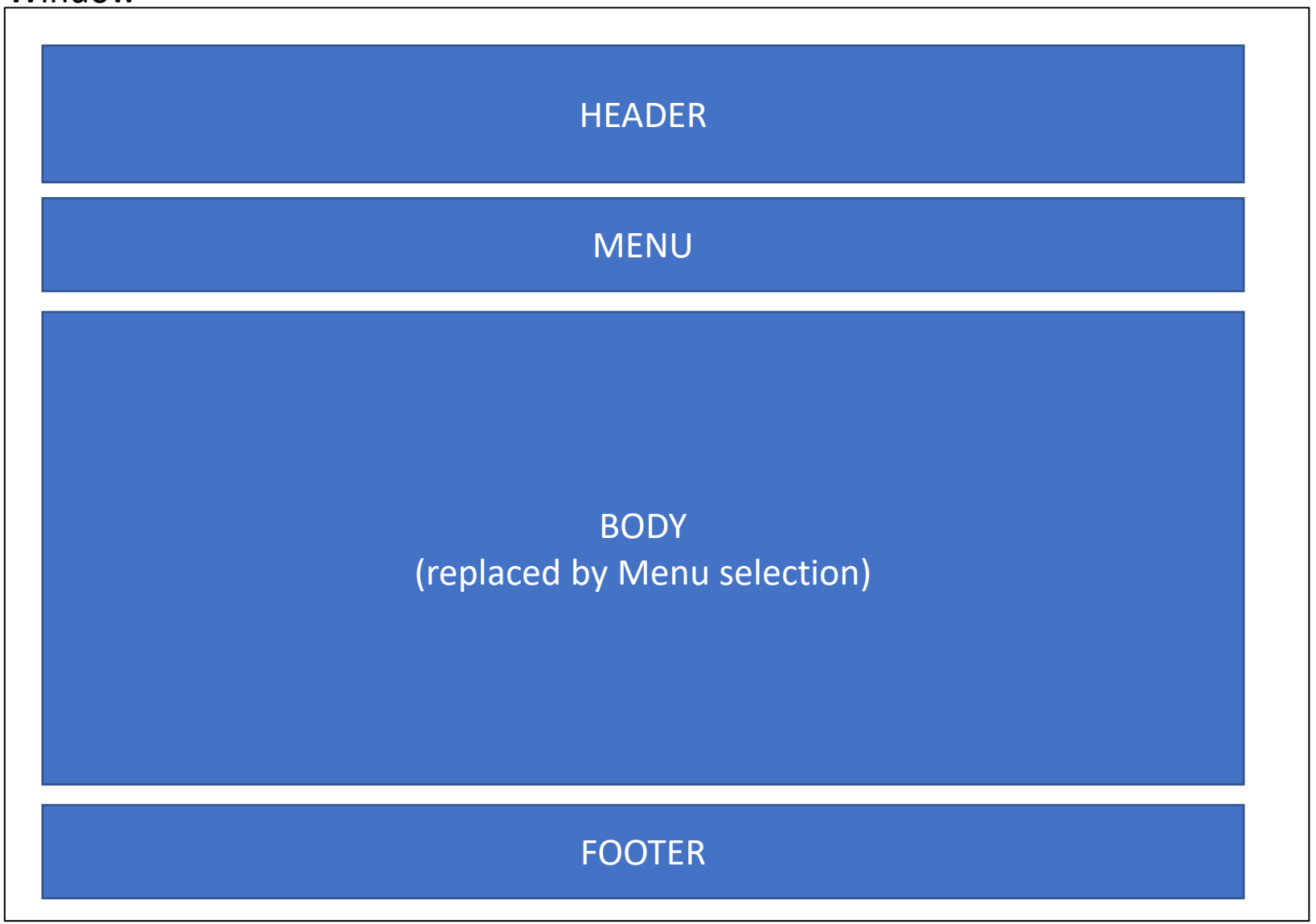
Shapes: Scroll style



Contents

Multi-paged-static
Building websites
Case studies

Shapes: Fixed style

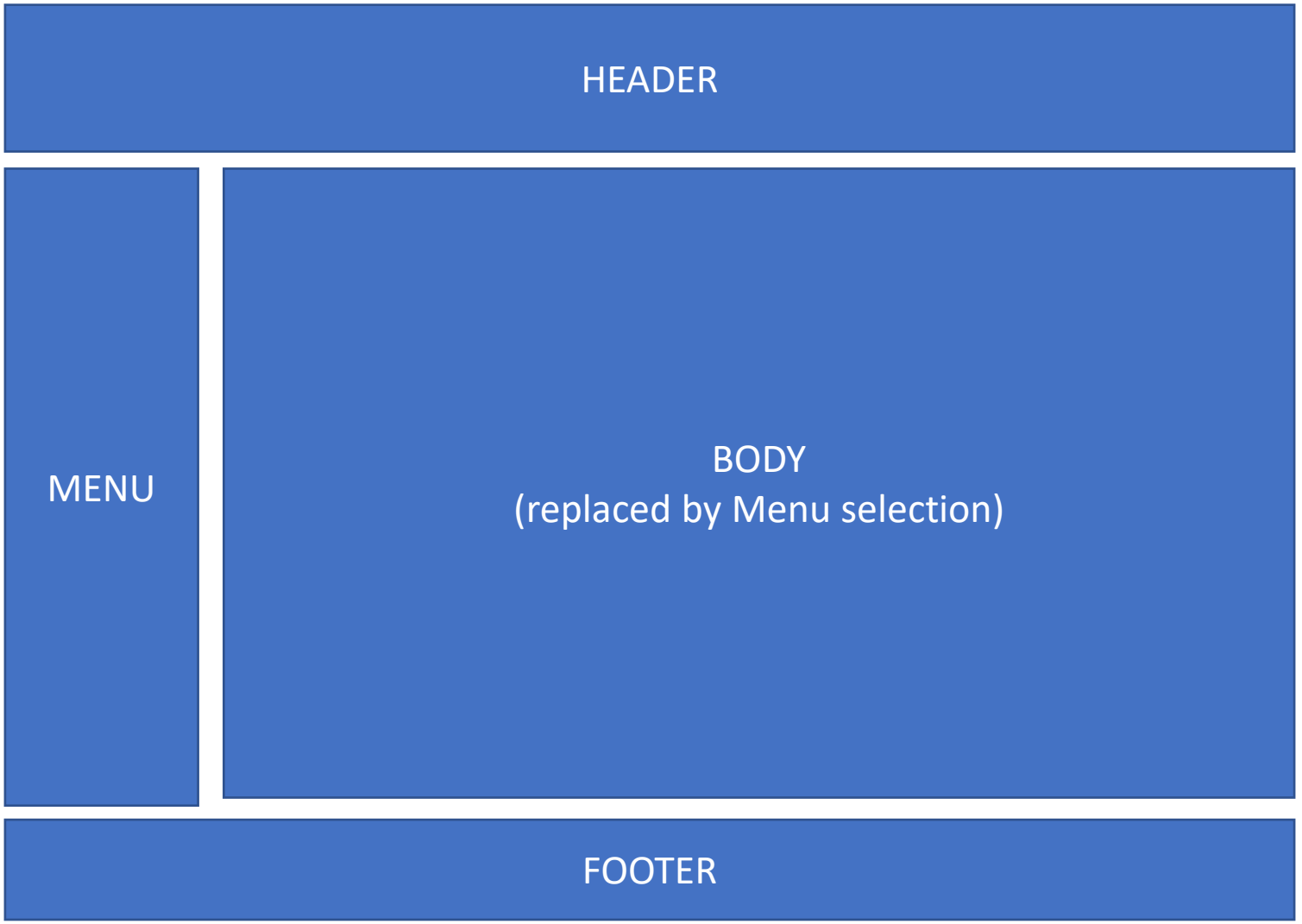


Contents

Multi-paged-static
Building websites
Case studies

Shapes: Table style

(can scroll of be fixed)



Contents

Multi-paged-static
Building websites
Case studies

Shapes: Popup style

(often fixed style or overlay style)

HEADER & MENU

Popup
(resulted from Menu selection)

FOOTER

Contents

Multi-paged-static
Building websites
Case studies

Shapes: Indexed style

(often scroll style)

HEADER

INDEX
(clicking on this menu auto scrolls down the page to beginning of section)

BODY
(contains all the pages of the website in one page in sections)

FOOTER

Contents

Multi-paged-static
Building websites
Case studies

Shapes: Multi-style

Public

HEADER

MENU

BODY

FOOTER

login

Private

HEADER

DASHBOARD

BODY
(each page a different style with different tech)

(input form, report, graph, spreadsheet, database access)

FOOTER

Contents

Multi-paged-static
Building websites
Case studies

Shapes: Immersive style

Public

HEADER

MENU

BODY

FOOTER

login

Private

CONTROL BOARD

GUI / GAME
(Canvas)

STATUS INFO

Contents

Multi-paged-static
Building websites
Case studies

Standardizing Elements

- Professional design
  - Pick a common color pallet for all pages
  - Pick a common feel: flow, mouse usage, interaction
  - Divide pages into shape groups & standardize each group
    - Public pages vs private pages
    - Landing page vs product pages
- Question: How might we standardize…
  - Personal Website
  - Blog Website
  - What would a “fun” feel look like?
  - What about “professional cooking” website look like?

Contents

Design
SE Techniques
Team Management

Software Engineering Techniques

Planning a website build

Contents

Design
SE Techniques
Team Management

Planning a website build

- Figure out how it will look and behave first
- Then figure out what the backend needs to look like to support the look & behavior
  - Optimal tools
  - Think about “load” and “response time”
- Websites crash recovery plan
  - Notifications, quick relaunch vs quick reinstall
  - Operational backups
- Maintain a repo with a good wiki

Contents

Multi-paged-static
Building websites
Case studies

Steps in planning a website

1. Decide on the purpose, scope, look, and feel
2. Create a "storyboard"
3. Create a "wireframe/mockup" of important pages (or all pages if time permits)
4. Create the development environment
    1. Repo
    2. Directory structure
    3. Development rules with the team / division of labour
5. Build templates (to control the look)
6. Using the templates to build the website
7. Don't forget about security, load times, recovery

Contents

Multi-paged-static
Building websites
Case studies

The Storyboard

Contents

Multi-paged-static
Building websites
Case studies

The Storyboard

- A technique used to plan a website
  - Used to design any website (not only static)
- Goals:
  - What pages do I need?
  - How will the website flow?
  - What technology do I need?
  - How much time will it take to build?

general things

not specifics

Contents

Multi-paged-static
Building websites
Case studies

Use-Case

- A static website for a professor
  - Landing page
  - About page
  - Research page
  - Login page for research group
  - Private projects pages

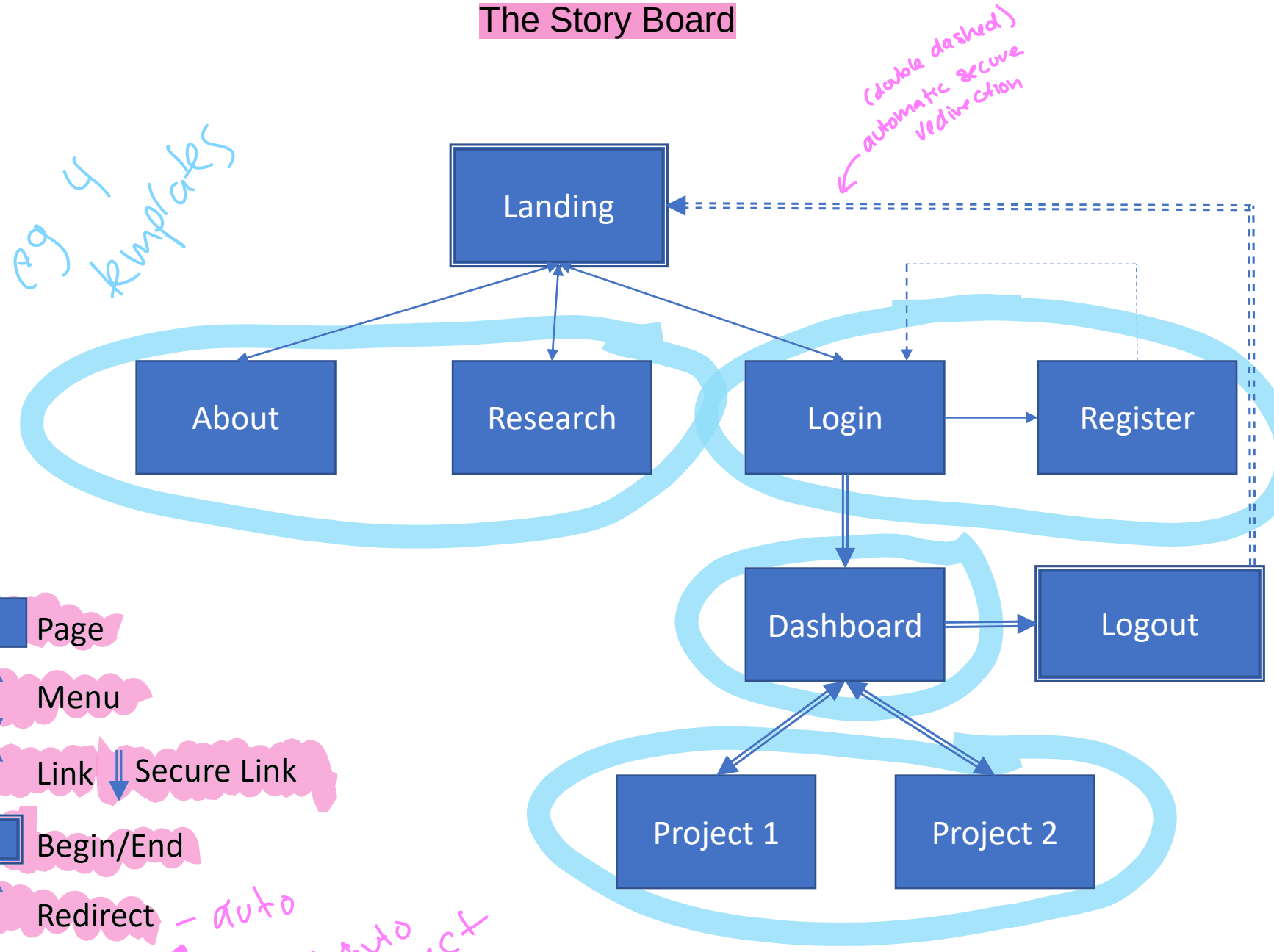
Contents

Multi-paged-static
Building websites
Case studies

COMP 307
Principles of Web Development

Use-Case

The Story Board



Contents

Multi-paged-static
Building websites
Case studies

Use-Case

Technology Estimation

Page	Technology	Externals	Time
Landing	Static, HTML & CSS, responsive	N/A	
About	Static, HTML & CSS, responsive	N/A	
Research	Static, HTML & CSS, responsive	N/A	
Login	Static, HTML & CSS, responsive	PHP+SQL & Ticket	
Registration	Static, HTML & CSS, responsive	PHP+SQL & redirect	
Logout	Static, HTML & CSS, responsive	PHP+SQL & redirect	
Dashboard	Static, HTML & CSS, responsive	Ticket	
Project 1	Static, HTML & CSS, responsive	Ticket	
Project 2	Static, HTML & CSS, responsive	Ticket	

Security

Contents

Multi-paged-static
Building websites
Case studies

Wireframe / Mockups

Contents

Multi-paged-static
Building websites
Case studies

Purpose

To come to a group understanding about the look and behavior of the website pages.

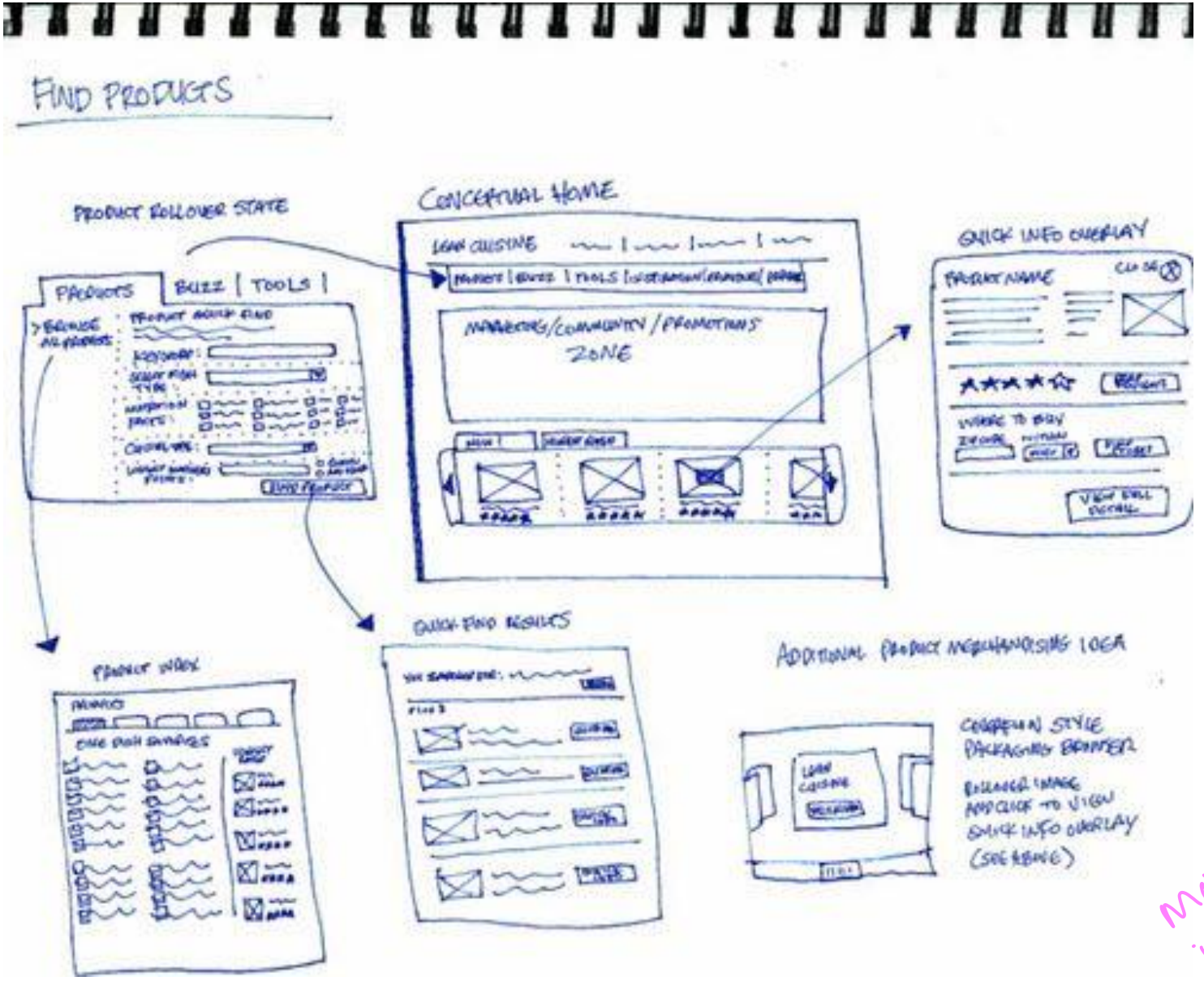
Often the members of the group debate the pros and cons of each page design / layout and technology / libraries needed.

Contents

Multi-paged-static
Building websites
Case studies

COMP 307
Principles of Web Development

Title wireframes based on the Storyboard box name



Arrows describe interactions (press a button, press a tab, etc.)

mockup is one level up

Contents

Multi-paged-static
Building websites
Case studies

Prof website example

What should the wireframe / mockups look like for the Home Page and the About Page from the Prof webpage question?

Keep in mind:

1. What will the page look like in picture-form
2. Interacting with an element results in a labeled arrow to a new wireframe containing the result, unless it is simple then loop back with only a label.
3. Provide language below the wireframe to orient the user
4. Group related wireframes on a common page.

Contents

Multi-paged-static
Building websites
Case studies

Example

What would be the wireframe groupings for the professor website question?

Free Wireframe Tools

Not in any order:

- https://miro.com/aq/paid-search/free-wireframe-tools-for-web-design
- https://www.justinmind.com/free-wireframe-tool
- https://www.lucidchart.com/pages/landing/wireframe-software

important to be standardized across a team

Templates

Contents

Multi-paged-static
Building websites
Case studies

Purpose

This simplifies development.

Each webpage will follow a standard design. Building a generic webpage that uses that standard design that can be copy-and-pasted for each web page as a starting point is very useful in development.

Spend time to build this perfectly.

Contents

Multi-paged-static
Building websites
Case studies

Examples

Landing Page may look different from Public informational pages.

Login Page.

Internal “private” Pages will look different from public pages.

Dashboard.

Contents

Multi-paged-static
Building websites
Case studies

Note

These template pages are fully functional.

For example, if the website has 5 public informational pages, then build a single fully functional template page using Lorem Ipsum language. This can then be used by the other developers as a starting point for the actual public informational pages.

https://loremipsum.io/

Contents

Multi-paged-static
Building websites
Case studies

Example

What pages would you pick to first built templates in the professor website questions?

- Why
- What in the template would be fully implemented and what would be left partially implemented?
- What would the groupings look like?

Directory Structure

Contents

Multi-paged-static
Building websites
Case studies

Directory Structure

- Separate your concerns
  - Root directory
  - Scripts
  - Compiled programs
  - Databases
  - Images
  - Functional backup website
- Code repository

Contents

Multi-paged-static
Building websites
Case studies

Directory Structure

- public_html
  - homepage.html
  - cgi_bin (scripts & compiled)
    - common
    - page1
    - page2
  - database
  - assets – pictures, css
  - backup_website (or prev_website)

Backup has the exact same directory but with the previous working code.

Contents

Multi-paged-static
Building websites
Case studies

Team Management

Planning a website build

Contents

Design
SE Techniques
Team Management

Team Management

- Organizing your team
- Quality Control
- Deployment

Contents

Design
SE Techniques
Team Management

Team Management

- Organizing your team
  - Assign non-overlapping jobs to each member
    - Non-overlapping to track responsibility
  - Divide the project into “assignments” before starting
    - Create a calendar of due dates for each assignment
    - Take note of dates where code merges from two members
      - Take special care on those days to make sure it all works out
    - Do this for the entire project before writing code
  - Create an online Deployment Environment (public website) & identical Beta Environment (testing website)
  - Create a repo
- Quality Control
- Deployment

Contents

Design
SE Techniques
Team Management

Team Management

- Organizing your team
- Quality Control
  - Define a workflow, for example:
    - Get assignment & note due date
    - Develop locally and backup to repo
    - Deploy on Beta before due date
    - Give your assignment to another to test on Beta, fix errors
    - Once good, go to next assignment
  - Track things publicly
    - Check-off calendar
    - Bug list. Assigned a bug to a team member
  - Go to the Prof early if there are serious problems
- Deployment

Contents

Design
SE Techniques
Team Management

Team Management

- Organizing your team
- Quality Control
- Deployment
  - Divide the project into 3 stages by time (or purposes)
    - E.g. 4-month project: stage A) 2 months, stage B) 1 month, Stage C) last month
  - At each end of stage
    - Copy Beta environment into Deployment environment
    - Fix runtime issues in deployment environment & reconfigure Beta environment so that it does not happen again.
  - Best not to do a single big deployment at the end
    - Often the deployment environment does not behave like the Beta environment

Contents

Design
SE Techniques
Team Management

Case study analysis

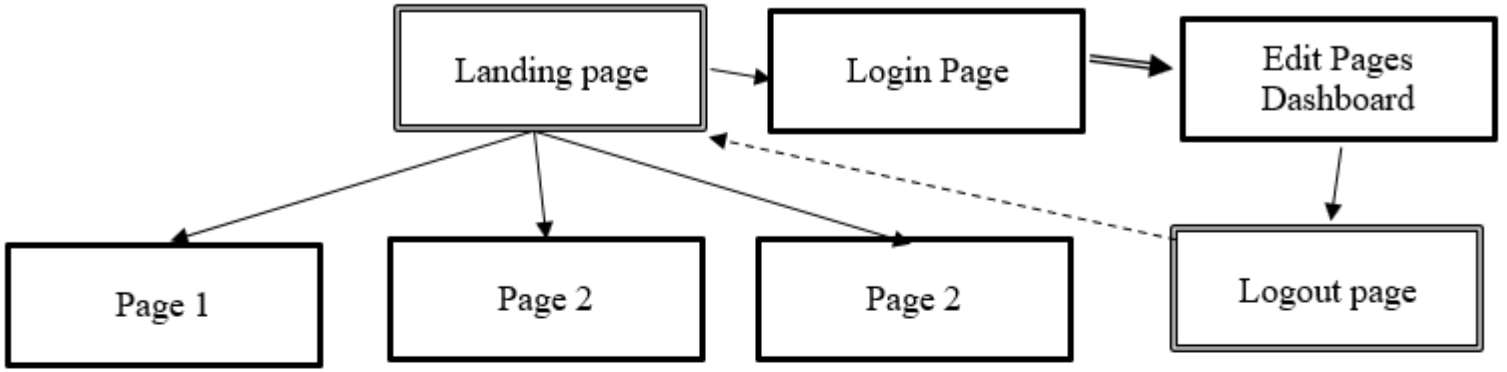
Static multi-paged websites

Contents

Multi-paged-static
Building websites
Case studies

Case Study 1: Personal Website with Login to Edit pages

The below storyboard depicts the website:



This website has two sections: a public section that contains the landing page, three information pages (Page 1, Page 2, and Page 3), and a login page. The private section requires a successful login from the login page, which gives access to the Dashboard and the Logout page. Other supporting technologies can be added to this website, like a databases/CSV files, etc., if you feel it is required.

Specific requirements for thus use-case: the number of users who are permitted to login is very small. It includes the owner of the website and at most two other content providers. The landing page welcomes the users to the site and provides a menu to three public information pages and the login page. The informational pages are each single pages. The site is responsive. The Dashboard permits the user to change the contents of the informational pages and the home page without requiring the user to access the command-line. The Dashboard presents the user with a simple textarea to change the informational pages and the ability to identify which page/page-part to change. The website has a standard look with every public page having the same header, footer, and menu.

To answer this question, do the following:

- Must be an optimal solution

A. Draw 2 Technology Estimation tables. One table for the Frontend of the website and a second table for the backend of the website.
B. Indicate the number of backend servers you are using, and why.
C. Using the provided storyboard do the following:
    a. Indicate where the database(s) or CSV or Data files interact with the storyboard.
    b. What was your reason for selecting the type of database / CSV / data file(s)?
    c. Indicate which part of the website is single-paged, multi-paged, backend-generated, or populated by dynamic content (your solution may not use all these techniques).

Contents

Multi-paged-static
Building websites
Case studies

COMP 307
Principles of Web Development

Contents

Multi-paged-static
Building websites
Case studies

Case Study 2: Truck Load Web Service

A "load board" is a service to independent delivery truck drivers. When an independent trucker is sitting in their truck with nothing to do, they can connect to a "load board" service to find an available delivery. The "load bord" is based on a spreadsheet. It lists the order, how much it is worth, the order's destination, and the phone number to call. The truck driver picks one of the orders and calls the phone number to confirm that the order is theirs. Then the trucker makes the pickup and delivery.

This type of website has three users: shipper, trucker, and sysop. The shipper is the person who places available shipments on the "load board" for a trucker. The trucker is the person who is looking for orders to deliver. The sysop is the person who manages the spreadsheet of orders. A shipper wants a way to add an available order to the spreadsheet. A shipper wants a way to remove an order from the spreadsheet when it has been picked up by a trucker. A trucker wants a way to search the spreadsheet for an available order. A trucker wants a way to pick an order and contact the shipper. A sysop wants a way to delete orders that have been on the spreadsheet for too long.

The complexity of the use-case is that shippers and truckers already have their own specific shipping and truck-driving applications. The sysop does not want to force the shippers and truckers to use a new website. Instead, the sysop wants to provide a web API that can be integrated into the shipping and truck-driving applications. Truckers most often access the Internet using their cell phone. Shippers most often access the Internet from their desktops.

Do not worry about the shipping and truck-driving application integration. Just focus on the API for the sysop's web service. Please note that a web service is not exactly a website. A website is about web pages. A web service is about an API that can be invoked by other applications. A web service does have a landing page, but the main part of the service is the API.

More specifically, this use-case asks for a single landing page that describes the service and a single informational webpage that describes the REST API signatures. The service is free. The server provides the API implementation and technological infrastructure to perform the REST API requests. Base your implantation on what we have seen during class time.

For this use case, do the following:

- We want an optimal solution

A. Draw a well formatted storyboard: pages, functions, databases, etc.
B. Provide all the REST API signatures from the use-case, assume the server root is www.orders.com.
C. How is the API implemented (for example, language, asynchronous, synchronous)? Which API (if using more than one technique)?
D. Are you using: single-paged, multi-paged, server-generated? Where (if using more than one)?
E. Are you using: responsive, dynamic, static? Where (if using more than one)?
F. One or more servers, and why.
G. If needed, you can provide a very short single additional paragraph to help explain your solution.

Prepare for Next Class

- Assignments
  - Mini 3 due
  - Mini 4 out
- No labs this week
- Do on your own
  - Try to solve the lecture case studies on your own (or with your fiends). Find an optimal solution.

Contents

Multi-paged-static
Building websites
Case studies