1. What is Prototyping?

- **Definition:** Building a draft or an early version of a product or system to explore, demonstrate, and test design ideas for part or all of the system.
- **Place in Design Thinking:** Prototyping follows the Ideate phase and precedes the Test phase. It involves building real, tactile representations of ideas.

2. Prototyping Methods

- **Paper Prototyping:**
  - Mocking up designs using sketches and basic supplies (paper, pens, etc.).
  - Simulating system responses by manually swapping screens or elements.
  - Most useful in the earliest design stages.
- **Wireframes (with Annotations):**
  - Lo-fi prototypes showing the structure and components of a design, either hand-drawn or digital.
  - Most useful in early-to-mid design stages.
  - **Annotations:** Labels and notes explaining the goals, content, and function of wireframe elements to aid interpretation.
- **Interactive Prototyping:**
  - Creating realistic prototypes focusing on navigation or structure (or both).
  - Involves linking screens/pages to simulate transitions and navigation.
  - Can use lo-fi or hi-fi components.
- **Native Prototyping:**
  - Implementing and testing design ideas directly on the target technology platform (e.g., using React Studio, Bootstrap Studio).

3. Prototyping Theory

- **Three-dimensional Model of Prototyping:** Prototypes represent three dimensions of a design idea:
  - **Role:** The functions the system serves in the user's life (how it's useful).
  - **Look and Feel:** The sensory experience (what the user sees, hears, feels).
  - **Implementation:** The technical capabilities enabling the system's function (how it works).
  - **Integration:** Combining these dimensions creates a working prototype or pre-alpha product.
- **Prototyping Scope:** Refers to the breadth and depth of features/functioning represented.
  - **Horizontal Prototype:** Broad view of the entire system, focusing on user interaction over deep functionality. Good for testing discoverability and findability initially.
  - **Vertical Prototype:** Focuses on a single feature, providing its full functioning in depth. Good for testing usability of specific interactions.
- **Prototyping Fidelity:** The level of detail in the prototype.
  - **Low-fidelity (lo-fi):** Rough representations (e.g., sketches, paper prototypes).
    - *Benefits:* Lower cost, encourages broader feedback, allows rapid exploration of multiple ideas, helps identify high-level issues (structure, navigation).
    - *Limitations:* Needs facilitation, limited ability to find detailed breakdowns, lacks development specs, limited feasibility sense.
  - **High-fidelity (hi-fi):** Looks close to the final product (e.g., interactive mockups, native prototypes).
    - *Consideration:* The more "done" it looks, the narrower (more detailed, less conceptual) the feedback tends to be. Use no higher fidelity than necessary.

4. Choosing the Right Method

Consider these factors:

- **Design Team Goals:** What do you need to learn or test? (e.g., conceptual feedback, navigation testing, look & feel communication).
- **Capabilities and Resources:** What tools and skills does the team have?.
- **Available Time:** How much time can be dedicated to prototyping?.
- **Stage of Design:**
  - *Early (Conceptual):* Sketches, Paper Prototyping.
  - *Mid (Structure/Navigation):* Wireframes, Interactive Prototypes (lo-fi).
  - *Late (Look & Feel/Implementation):* Interactive Prototypes (hi-fi), Native Prototyping.

1. Mobile Input

- **Overview:** Mobile input primarily uses touch-sensitive screens offering direct manipulation and multi-touch gestures. Mobile devices have unique input and sensing capabilities.
- **Direct Manipulation:** Unlike the relative mapping of mouse/trackpad on desktops, mobile uses absolute mapping where the user directly interacts with screen elements.
- **Multi-touch Gestures:**
  - **Tap:** Brief touch, maps to a desktop click for selection or activation.
  - **Double-tap:** Two rapid taps, used for zooming, selecting items (accessibility), or selecting text (single word). Triple-tap selects a sentence, quadruple-tap selects a paragraph.
  - **Long-press:** Touching the surface for an extended time, used for contextual menus, content previews, or enabling editing modes.
  - **Drag/Swipe:** Moving a fingertip across the surface without losing contact. Used for scrolling, moving objects, or adjusting controls. Vertical swipes scroll content; horizontal swipes scroll carousels, open drawers, or navigate screens. Dragging can also move objects or operate controls like sliders.
  - **Pinch/Spread (Two-finger):** Bringing two fingers closer (pinch) or moving them apart (spread) to shrink or expand elements like maps or photos.
  - **Pinch/Spread (Three-finger):** Often mapped to OS-level actions like copy, cut, and paste.
  - **Flick:** Quickly brushing the surface.
  - **Press:** Extended touch.
  - **Rotate:** Touching with two fingers and moving in a circular direction.
- **Context of Use:** Consider how people hold devices (one-handed, two-handed, cradled) and the device size to determine reachable zones (Comfort, Reachable, Hard) when choosing gestures.
- **Motion Gestures:** Involve moving the device itself (e.g., shaking). Often application-specific or OS-customizable.
  - Examples include placing the phone to ear to answer/place a call.
  - Shaking to ignore a call.

2. Microinteractions

- **Definition:** Contained product moments revolving around a single use case (e.g., liking a post, pull-to-refresh). Think of them as single-purpose action-feedback pairs.
- **Structure:** Four parts:
  - **Trigger:** Initiates the microinteraction. Can be manual (user-initiated, e.g., pressing a button) or automatic (system-initiated based on conditions, e.g., new message notification).
  - **Rules:** Determine what happens (and doesn't) when triggered.
  - **Feedback:** How the system signals what's happening (visual, aural, haptic).
  - **Loops & Modes:** Meta-rules changing behavior based on context.
    - *Loops:* Determine duration, repetition (like a fridge door alarm), or changes over time (like a microwave reminder).
    - *Modes:* Switch system operation (e.g., "do not disturb," silent mode changing feedback from chime to vibration).
- **Pro Tip:** Keep microinteractions focused on a single task/action; don't let them become complex features.

3. Mobile Design Patterns

Mobile platforms are constrained, and patterns help overcome limitations.

- **Stacks:** Vertically organize elements (toolbar, content, navigation bar) to maximize vertical space.
- **Screen Carousels:** Full-screen horizontal arrays for displaying different instances of the same information type (e.g., weather for multiple cities).
- **Drawers:** Slide-out panels (usually from the side) providing navigation links or settings controls.
- **Lists:** Vertical stacks of items (text, controls, thumbnails) supporting vertical scrolling.
- **Grids:** Large continuous grids (or panes of grids) for vertical or horizontal scrolling, often used for visual content like photos or icons.
- **Carousels (Content):** A horizontal row of content items (images, cards) navigated by swiping left/right.
- **Swimlanes:** Stacked content carousels, allowing visual Browse through several rows/categories simultaneously.
- **Cards:** Rectangular containers with images/text serving as entry points to more detail. Can be organized in lists/grids, combining multimedia content.
- **Bars:** Horizontal or vertical strips with buttons/tabs.
  - *Tab Bars:* For navigation between components (often top or bottom).
  - *Toolbars:* Activate application or OS functions.
  - *Navigation Bars:* Link to previous content or next/previous items.
- **Search, Sorting, Filtering:** Provide ways to navigate large amounts of content via search boxes (text/voice input, history) and filtering/sorting options.
- **Landing Pages, Guided Tours:** Welcome/home screens acting as portals to common functions or guides. Can include guided tours or help overlays for new users.
- **Advanced Direct Manipulation:** Custom controls for content creation/editing in apps like image editors or media players (e.g., control handles, custom scrollbars).
- **Panes and Panels:** Multi-pane structures and pop-up panels, common on tablets, providing secondary windows similar to desktop apps.
- **AR/VR (Mixed Reality):** Overlaying digital objects/info onto the real-world view captured by the device camera.

1. Introduction to Conversational Interfaces (CIs)

- **Definition:** Technology supporting conversational interaction with virtual personal assistants (VPAs) via speech and other modalities.
- **Components:** Integrate technologies like:
  - Speech Recognition (audio to words)
  - Spoken Language Understanding (words to meaning)
  - Dialog Management (decides next system action)
  - Response Generation (meaning to words)
  - Text-to-Speech Synthesis (words to audio)
- **Value Proposition:** While often less effective, efficient, and satisfactory than GUIs, CIs deliver value:
  - In contexts with constraints (e.g., driving).
  - For accessibility (addressing vision, motor, cognitive deficiencies).
  - By streamlining tasks like installation, login, payment, notifications within a conversational paradigm.

2. Design Principles for Conversational Interfaces

- **Gricean Maxims:** Based on the cooperative principle for effective conversation:
  - **Quality:** Be truthful and accurate.
  - **Quantity:** Provide just the right amount of information.
  - **Relevance:** Provide appropriate and relevant information.
  - **Manner:** Be clear and cooperative.
- **Multimodality:** Most CIs are multimodal, using visual info, speech, touch, etc..
  - **Principle:** Leverage other modalities (visuals, vibrations) where appropriate. Use them for breaks, decision-making, interruptions.
  - **Caveat:** Ensure the interface still works speech-only; integrate components cohesively.
- **Interaction Paradigms:**
  - **Command-and-Control:** Speech input maps directly to system functions (e.g., voice assistants using wake words or button presses). Involves wake word/button → indicate listening → execute function.
  - **Conversational:** Interaction mimics human conversation characteristics (turn-taking, depth, markers) (e.g., chatbots, task assistants).
- **Turn-taking:** Core structure involving one speaker at a time and token exchange.
  - **Principles:** One speaker at a time; explicit signaling for turn exchange; handle interruptions (difficult).
- **Conversational Markers:** Speech cues indicating conversation state or direction.
  - **Types:** Timelines ("First," "Finally"), Acknowledgements ("Got it," "Sorry about that"), Positive Feedback ("Good job").
- **Confirmations:** Improve usability and transparency. Can be speech or non-speech based.
  - **Explicit:** Require user confirmation ("Is that right?"). Best for irreversible/critical actions.
  - **Implicit:** Let user know what was understood via results or next prompt ("Ok, setting reminder...").
- **Error Handling:** Address deviations from expected flow.
  - **Types of Errors:** No speech detected; speech detected but not recognized; recognized correctly but wrong system action; recognized incorrectly.
- **Flowcharting:** Common method to model/prototype CI interactions, showing flows based on system state, user behavior, etc..
- **Conversational Interface Heuristics:** Guidelines for CI design (e.g., Wei & Landay's 17 heuristics).
  - *Example (S5):* Pay attention to context. Remember prior user input in the conversation; use known context to pre-fill information (but confirm).
  - *Example (S12):* Confirm input intelligently. Use implicit confirmation often, explicit for critical actions.

3. Experience Prototyping

- **The Problem:** Conventional prototyping methods have limited support for CIs because human interaction relies on tacit knowledge (knowing more than we can tell).
- **Definition:** Prototyping the *holistic experience* of interacting with a product/system. Any representation designed to understand, explore, or communicate what it might be *like* to engage with the design.
- **Why it Works:** By acting out interactions, we apply our tacit knowledge of conversational norms ("What I do, I understand!").
- **When to Use:**
  - Understanding existing user experiences/context.
  - Exploring and evaluating design ideas.
  - Communicating ideas.
- **What to Prototype:**
  - System behavior.
  - User behavior.
  - Interactions with context.
- **How to Do It (Steps):**
  1. **Define Context:** Where does the interaction occur? (e.g., on a bus, packing luggage).
  2. **Develop Scenarios:** What concrete interactions need support? (e.g., buying a ticket, cooking a meal).
  3. **Identify Design Goals:** What role does the design play? What capabilities does it offer? (e.g., find/filter flights, provide daily reminders).
  4. **Set up the Environment:** Represent the context physically (e.g., use props).
  5. **Act out Interaction:** How does the interaction unfold? How do user and system behave? (Often involves **Bodystorming**: physically experiencing a situation for insights).
  6. **Develop Insight:** What was learned about system/user behavior and context interactions?.
- **Pro Tip:** Experience prototyping can feel awkward; push through it to focus on design.

1. Computers as Social Actors (CASA)

- **Definition:** A paradigm stating that humans *mindlessly* apply social heuristics from human-human interaction to computers, treating them as social agents. Mindlessness relies on past distinctions without active thought.
- **Implications:** People attribute social characteristics to computers, such as gender, ethnicity, group membership, personality, and expect social norms like politeness and reciprocity.

2. Social-Psychological Theories in Agent Design

- **Similarity-Attraction Theory:** "Likeness begets liking." People are attracted to agents similar to themselves.
  - *Explicit Characteristics:* Age, gender, language, accent.
  - *Implicit Characteristics:* Personality (e.g., extroversion), interaction style (formal/informal).
  - *User Preferences.*
- **Example:** Robots matching user introversion/extroversion through eye contact increased user motivation (though intrinsic motivation is stronger).
- **Consistency-Attraction Theory:** People prefer agents that behave consistently, as it reduces cognitive load and increases predictability.
  - *Internal Consistency:* Behaviors, appearance, function are mutually consistent.
  - *External Consistency:* Design aligns with user expectations and preferences.
- **Application:** Creating believable characters requires consistent backstory, verbal/nonverbal behaviors, and cultural expression.

3. Designing Character Speech

- **Politeness Theory:** Individuals use communication strategies to express concern for others and minimize threats to "face" (self-esteem/public image).
  - **Face:**
    - *Positive Face:* Need for self-image to be accepted, appreciated, approved.
    - *Negative Face:* Need for independence, freedom from imposition.
  - **Face Saving vs. Threatening:** Saving face involves showing deference (negative face) or solidarity (positive face). Threatening face opposes the other's wants/desires.
  - **Politeness Types:**
    - *Positive Politeness:* Avoiding offense through friendliness, emphasizing common ground.
    - *Negative Politeness:* Avoiding offense through deference, minimizing imposition.
  - **Effective Politeness Strategies (Do Use):**
    - Gratitude ("I really appreciate...")
    - Deference ("Nice work so far...")
    - Indirectness ("By the way, where did you find...")
    - "Please" (mid-sentence) ("Could you please say more...")
    - Apologizing ("Sorry to bother you...")
    - Counterfactual Modals ("Could/Would you...")
    - Greeting ("Hey, I just tried to...")
  - **Ineffective/Rude Strategies (Don't Use):**
    - Direct Start ("So can you retrieve it or not?")
    - Factuality ("In fact you did link...")
    - 2nd Person Start ("You've reverted yourself...")
    - "Please" Start ("Please do not remove warnings...")
    - Direct Question ("What is your native language?")
    - Negative Lexicon ("If you're going to accuse me...")
    - **Hedging:** Using words/phrases to mitigate the force or certainty of a statement (e.g., "sort of," "I think," "possibly"). More hedging generally implies more negative politeness.
- **Expressing Expertise:** Convincing users the agent is knowledgeable.
  - **Dimensions:** Rhetorical ability (speaking skill) & Practical knowledge (experience).
  - **Cues:** Goodwill (showing concern for listener), Prior expertise references, Organized information flow, Metaphors (accessibility), Fluency (reduced pauses, confidence).

4. Personality in Artificial Agents

- **Definition:** Individual differences in characteristic patterns of thinking, feeling, and behaving.
- **Models:**
  - **Personality Traits (e.g., Big Five):** Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism (OCEAN).
  - **Personality Types (e.g., Myers-Briggs – MBTI):** Posits distinct categories (16 types) based on combinations of preferences (Extraversion/Introversion, Sensing/Intuition, Thinking/Feeling, Judging/Perceiving).
- **Design Strategies:**
  - **Personality Matching:**
    - *Same Personality:* Agent matches user (leverages similarity-attraction).
    - *Complementary Personality:* Agent has a personality that complements the user's (e.g., task-focused agent for a visionary user).
  - **Personality Expression:** Conveying personality through:
    - Language use (politeness, expertise cues)
    - Visible cues (gaze, gesture, facial expression, appearance)
    - Marketing & Branding
    - Specialization & Domain Knowledge
  - **Persona Development:** Creating detailed fictional characters based on user research to represent user types. Agents can be designed to align with or interact effectively with specific user personas.

1. Why Evaluate?

- User experience (UX) design aims to create meaningful and relevant experiences.
- Evaluation, specifically the **Test** phase, is a core part of the iterative design process (Empathize, Define, Ideate, Prototype, Test, Implement).
- Testing-based methods involve empirical testing with users representing the target population.

2. Redefining Usability (The Five-E Model)

1. **Effective:** How completely and accurately users achieve goals.
   - *Evaluation:* Use tasks of varying difficulty; measure accuracy, undetected errors.
2. **Efficient:** How quickly users can complete work.
   - *Evaluation:* Time typical tasks (needs working software/hi-fi prototype); assess subjective speed via interviews.
3. **Engaging:** How well the interface draws users in and is pleasant/satisfying.
   - *Evaluation:* Use satisfaction surveys/interviews; comparative preference tests; allow users to abandon the product.
4. **Error Tolerant:** How well the product prevents errors and helps users recover.
   - *Evaluation:* Design scenarios likely to cause errors; observe recovery ease/accuracy.
5. **Easy to Learn:** How well the product supports initial orientation and continued learning.
   - *Evaluation:* Control instruction level; recruit users with varied experience; mix frequent and infrequent/unusual tasks.

Note: The importance (weight) of each "E" varies depending on the product/context (e.g., Engaging is more critical for a museum site, Error Tolerant for a registration form).

3. Usability Testing Basics

- **Definition:** Observing users performing tasks with a design solution and asking them about their experience, noting performance, behavior, and verbal feedback.
- **When to Use:**
  - **Formative Testing:** Done *throughout* the design process. Aims to diagnose and fix problems. Typically uses a small number of users (e.g., 5 users can find 85% of issues) repeatedly to inform design iterations. Considered a "discount" method.
  - **Summative Testing:** Done at the *end* of the design process. Aims to establish baseline usability. Involves larger numbers of users, often comparative, uses metrics and statistics.
- **Testing Contexts:**
  - **Laboratory Testing:** Controlled setting with specialized equipment (recording, eye-tracking, observation rooms).
  - **Field Testing:** Testing in the user's natural environment (target setting) with target users.
  - **Guerilla Testing:** Low-cost field testing in public spaces, recruiting passersby.
  - **Remote Testing:** Testing over the internet. Can be *Moderated* (facilitator guides user) or *Unmoderated* (user completes tasks independently, behavior logged, e.g., A/B testing).

4. Designing a Usability Test (The 4 Dimensions)

- A structured approach involves defining:
  1. **Why (Goals):**
     - Define the purpose of the test. Formulate goals as questions the test should answer.
     - Goals should specify:
       - *Desired Outcomes:* What the design aims to achieve (e.g., improve accessibility, reduce errors).
       - *Basis for Comparison:* Is the outcome measured against a baseline (e.g., previous version, competitor, standard benchmark)?
  2. **What (Scope, Tasks, Scenarios):**
     - Determine *what* aspects of the system/prototype to test (scope relates to horizontal vs. vertical prototypes).
     - Define specific *Tasks* (sequences of actions users perform).
     - Create *Scenarios* (brief stories providing context and goals for tasks).
     - *Scenario Ordering:* Typically order from general to specific, simple to complex, short to long.
  3. **How (Approach, Metrics):**
     - Method depends on goals (formative vs. summative, single vs. comparative).
     - Decide on data collection:
       - *Qualitative:* Observations, user comments, answers to questions.
       - *Quantitative:* Measurements of performance, errors, ratings.
  4. **Who (Users, Team):**
     - Define *Participants:* Recruit users representative of target *subgroups* (based on experience, demographics, domain knowledge, etc.).
     - Assign *Team Roles:* Moderator (guides participant), Note-taker (records data), Observer(s), Technician (operates system/equipment).
- **Test Plan (Protocol):** A document detailing the Why, What, How, and Who. Includes scripts, checklists, consent forms, etc.

5. Measurement

- **Performance Metrics:**
  - *Task Success:* Completion rate (binary or levels).
  - *Time on Task:* Time taken to complete.
  - *Errors:* Frequency and type of mistakes.
  - *Efficiency:* Effort required (e.g., clicks, steps).
  - *Learnability:* Change in performance over time/repeated tasks.
- **Self-Report Metrics:** User perceptions and experiences.
  - *Likert Scales:* Numerical rating scales with descriptive anchors (e.g., 1=Strongly Disagree to 5=Strongly Agree).
  - **Standardized Questionnaires:**
    - **SUS (System Usability Scale):** 10 items, widely used general usability measure.
    - **USE (Usefulness, Ease of Use, Ease of Learning, Satisfaction):** 4 sub-scales.
  - *Qualitative Feedback:* Open-ended questions, narratives.
- **Issue-Based Metrics:** Identifying usability problems.
  - *Identification:* Through task actions, behavior observation, verbal/non-verbal cues (frustration, confusion, surprise).
  - *Severity Ratings:* Prioritize issues based on impact (user experience, business), frequency, and cost. Levels:
    - *Low:* Annoyance, doesn't cause task failure.
    - *Medium:* Causes significant difficulty, but not task failure.
    - *High:* Directly leads to task failure.

1. What is Design Thinking?

- **An Approach:** A hands-on, user-centric approach to innovative problem-solving.
- **A Process:** An iterative process typically involving phases focused on understanding the user, exploring potential solutions, and materializing those solutions. (The lecture uses the NN/g 6-phase model).

2. Phases of Design Thinking (NN/g Model)

Phase Group 1: UNDERSTAND

- **1. Empathize:**
  - **Goal:** Conduct research to develop an understanding of your users (their needs, preferences, expectations, what they do, say, think, feel).
  - **Key Method: Contextual Inquiry**
    - **Definition:** A method for gathering data about the user and their work/life in their actual environment.
    - **Process:**
      1. Go to where the user works or lives.
      2. Observe the user as they perform relevant tasks.
      3. Talk to the user about their work/actions *in context*.
    - **Tips:** It's an *inquiry* (learning from the user as the master), not just an interview. Ask questions but don't distract. Focus on the context.
- **2. Define:**
  - **Goal:** Combine all research findings (data) and analyze them to observe where users' problems exist and derive insights.
  - **Key Method: Affinity Diagramming**
    - **Definition:** A method for organizing qualitative/messy data into clusters based on similarity or "affinity." Helps make sense of observations.
    - **Bottom-Up Process:**
      1. Start with disorganized notes (e.g., observations from contextual inquiry on sticky notes).
      2. Group related notes together based on likeness.
      3. Consult with the team on groupings.
      4. Re-organize if needed.
      5. Identify and label the common theme for each group.
    - **Outcome:** The themes identified represent findings and insights that guide design.

Phase Group 2: EXPLORE

- **3. Ideate:**
  - **Goal:** Generate a wide range of crazy, creative ideas to address the problems/insights identified in the Define phase.
  - **Nature:** Active, creative, exploratory, iterative, fast-moving, collaborative process.
  - **Key Considerations:** Workspace setup, team dynamics, process structure, rules of engagement, idea capture method.
  - **IDEO's Rules (Example):** Defer judgment, Encourage wild ideas, Build on others' ideas, Stay focused, One conversation at a time, Be visual, Go for quantity.
  - **Representing Ideas:**
    - **Conceptual Design:** An abstract characterization of the envisioned solution's context, use, or experience.
    - **Sketching:** Quick, inexpensive, disposable drawings to explore many ideas. Effective for communication, doesn't require artistic skill, can be annotated.
    - **Storyboarding:** A sequence of visual frames (like a comic strip) illustrating how a user interacts with the envisioned system within a specific context (showing setting, actions, outcomes).

Phase Group 3: MATERIALIZE

- **4. Prototype:**
  - **Goal:** Build real, tactile representations (from low- to high-fidelity) for a range of ideas. (Covered in Lec 16).
- **5. Test:**
  - **Goal:** Return to users for feedback on the prototypes. (Covered in Lec 25).
- **6. Implement:**
  - **Goal:** Put the refined vision into effect (build the actual product/service).

Note: Design thinking is iterative, not strictly linear. You might loop back to earlier phases based on discoveries in later phases (e.g., testing reveals a need to redefine the problem).

1. Elements of Design (The Building Blocks)

- **Space:** The canvas area.
  - *Positive Space:* Area occupied by the main subject/elements.
  - *Negative Space:* Area surrounding the subject (can be used creatively).
- **Line:** The most basic element; can divide space, draw attention, organize, create textures, and form other elements.
- **Shape:** An area defined by a contour or outline.
  - *Geometric:* Mathematically regular (squares, circles, triangles).
  - *Organic:* Irregular, natural (leaves, ink splatters).
  - *Abstract:* Stylized representations (icons, symbols).
- **Size (Scale):** The relative extent or dimensions of elements.
- **Pattern:** Systematic repetition or duplication of elements (lines, shapes).
- **Texture:** The perceived tactile or visual surface quality (smooth, rough, bumpy).
- **Value:** The lightness or darkness (intensity) of an element or color.

2. Principles of Design (How to Use the Elements)

- **Focal Point (Emphasis):** The area of primary visual interest where the viewer's attention is directed first.
- **Contrast:** Juxtaposing strikingly different elements (e.g., light/dark, rough/smooth, large/small) to highlight differences, guide attention, or create interest.
- **Balance:** Arranging elements to create a sense of equilibrium.
  - *Symmetrical:* Elements mirrored across an axis.
  - *Asymmetrical:* Elements balanced by visual weight without mirroring.
- **Movement:** Organizing elements to suggest flow and direct the viewer's eye through the design in a specific path.
- **Rhythm:** Using elements in a patterned or recurring way to communicate movement, order, or energy.
- **Perspective:** Creating an illusion of depth or distance on a flat surface.
- **Unity (Harmony):** Ensuring all parts of the design work together cohesively to form a coherent whole; consistency in the use of elements.

3. Key Components for UX Visual Design

- **Type (Typography):**
  - **Definitions:**
    - *Type:* Printed letters/characters.
    - *Font:* A specific style of type (e.g., Arial Regular 12pt).
    - *Typeface:* A family of related fonts (e.g., Arial, including Regular, Bold, Italic).
    - *Glyph:* An individual character (letter, number, symbol).
  - **Categories:**
    - *Old Style:* Serifs (small strokes at ends of letters) taper. (e.g., Garamond).
    - *Modern:* Very thin, straight serifs, high contrast strokes.
    - *Slab Serif:* Thick, block-like serifs. (e.g., Rockwell).
    - *Sans Serif:* No serifs, often uniform stroke weight. (e.g., Gill Sans, Helvetica).
    - *Script:* Mimics cursive handwriting.
    - *Decorative:* Stylized fonts for specific contexts/moods (e.g., Gothic).
  - **Parameters:** Weight (light, bold), Style (italic, oblique), Caps (all caps, small caps).
  - **Usage:** Type creates hierarchy and guides the eye. Pay attention to *leading* (line spacing), *tracking* (letter spacing), *kerning* (space between specific letter pairs), and avoiding *widows/orphans* (isolated lines/words).
- **Color:**
  - **Purpose:** Creates emphasis, organizes content, evokes emotion.
  - **Color Wheel:**
    - *Primary Colors:* Red, Yellow, Blue.
    - *Secondary Colors:* Green, Orange, Purple (mixing primaries).
    - *Tertiary Colors:* Mixing primary and secondary.
    - *Tints:* Add white. *Shades:* Add black. *Tones:* Add grey.
    - *Warm vs. Cool Colors:* Affects mood.
  - **Color Harmonies (Palettes):**
    - *Analogous:* Adjacent colors on the wheel.
    - *Complementary:* Opposite colors on the wheel (high contrast).
    - *Split Complementary:* A base color + two colors adjacent to its complement.
    - *Triadic:* Three equidistant colors.
    - *Monochromatic:* Variations (tints/shades/tones) of one hue.
    - *Achromatic:* Neutrals (black, white, grey).
  - **Accessibility:** Account for color vision deficiencies (use contrast in value/saturation, size, patterns, not just hue; use tools to check).
- **Images:**
  - **Types:** Photos, illustrations, icons, infographics, etc.
  - **Formats:** *Raster* (pixel-based, photos) vs. *Vector* (path-based, logos/icons, scalable).
  - **Usage:** Should complement and support content/goals, not distract. Choose appropriate realism level (realistic to abstract). Use motivational images subtly.

4. Applying Visual Design

- Analyze existing designs to identify elements and principles.
- Practice applying principles intentionally.
- Use checklists/cheat sheets.
- Ensure every element serves a purpose within the overall design principles.

1. Designing for the Desktop (Context)

- **WIMP Paradigm:** The traditional desktop interface model: **W**indows, **I**cons, **M**enus, **P**ointer. Dates back to Xerox Alto (1973).
- **Windows:**
  - *Definition:* Resizable containers for applications.
  - *Types:*
    - **Primary:** Contain main application functionality (e.g., document canvas).
    - **Secondary:** Support primary windows (e.g., dialog boxes, tool palettes, modal panes). Can be docked, stacked, or floating.
  - *Organization:* Can overlap or be tiled across the screen.
  - *Structures:* Combine dedicated panes (e.g., email: folders pane, message list pane, message preview pane).
- **Menus:** List application functions; serve educational/reference purposes.
- **Toolbars/Sidebars/Tooltips:** Provide quick visual and manipulation access to *frequently used* functions.
- **Tool Palettes:** Provide *advanced controls* for specific functions (unlike toolbars for frequent functions).
- **Pointing:** Enables direct manipulation of objects on the canvas.

2. Designing for the Web

- **Desktop vs. Web:**
  - *Desktop Apps:* Typically dynamic, persistent screens enabling complex tasks.
  - *Websites:* Interconnected *pages* focused on navigating and accessing content.
- **Web Applications (SPAs):** Single-Page Applications that provide desktop-like functionality within a webpage, often using similar UI conventions.
- **The Page:** The fundamental building block of web content.
- **Web Navigation:**
  - **Primary Navigation:** Menus/menu bars reflecting major site sections (top-level structure).
  - **Secondary Navigation:** Comprehensive links to specific content (e.g., site maps, side navigation, footer links).
  - **Breadcrumbs/Hierarchical Lists:** Help users understand their location and navigate back up the site structure ("get home").
- **Organizing Page Content:**
  - **The Fold:** The imaginary line separating content visible on initial load ("above the fold") from content requiring scrolling ("below the fold"). Content above the fold receives significantly more attention.
  - **Handling Large Content:**
    - *Pagination:* Breaking content into discrete pages.
    - *Infinite Scroll:* Loading content incrementally as the user scrolls down.
- **Search:**
  - An alternative to browse/navigation.
  - **Faceted Search:** Allows users to refine search results using filters and sorting options based on content attributes (e.g., price, brand, size).

3. Layout Design (Arranging Elements on the Page/Canvas)

Builds on visual design elements/principles:

- **Create a Focal Point:** Direct the viewer's eye to the most important element first.
- **Use Compositional Guides:**
  - **Golden Ratio (1:1.618):** Mathematical proportion creating aesthetically pleasing, often asymmetrical balance.
  - **Rule of Thirds:** Divide the canvas into a 3x3 grid; place key elements along lines or at intersections for visual interest.
  - **Grids:** Provide an underlying structure for organizing elements in an orderly, balanced way. Essential for responsive design.
- **Integrate Type:** Use headlines, text size, and weight to create hierarchy and guide reading flow.
- **Place Imagery:** Use images to create focal points or imply movement. Position strategically (e.g., top, facing content flow). Avoid interrupting headlines or awkwardly wrapping text.
- **Use Negative Space (Whitespace):** Empty areas around elements; essential for visual breathing room, balance, and directing focus.
- **Group Elements (Gestalt Principles):** Users perceive relationships based on arrangement:
  - *Proximity:* Items close together are seen as a group.
  - *Similarity:* Items that look similar are seen as a group.
  - *Continuity:* The eye follows smooth lines or curves.
  - *Closure:* The mind fills in gaps to perceive complete shapes.
- **Create Visual Hierarchy:** Use size, color, contrast, po-

sition, etc., to signal the relative importance of elements.
- **Exploit Visual Scan Patterns:** Arrange content considering how users typically scan pages (e.g., F-pattern: headlines, first sentences; Z-pattern: simple layouts).
- **Create Contrast and Emphasis:** Manipulate element features (spatial position, size, color, texture, typography) to make key elements stand out and establish hierarchy.

4. Design Patterns in UX

- **Definition:** Reusable, standard solutions to common UI design problems related to structure, organization, components, and controls.
- **Examples:** Master/Detail, Palette/Canvas, Column Browse, Spreadsheet, Wizard, Form, Search/Results, Dashboard, Portal, Tabs.
- **Pros:** Speed up design, improve quality, leverage user familiarity, provide starting points.
- **Cons:** May not fit every problem, may stifle innovation if applied rigidly, may not exist for novel interaction types.

1. What is Interaction Design?

- **Definition:** Defining behaviors for a system that engages the full spectrum of its user's perception, cognition, and movements.
- **vs. Visual Design:** Interaction design is more closely related to user behavior and context, dealing with aspects like navigation models which visual design typically doesn't cover.
- **Five Dimensions (ID):**
  1. **1D (Words):** Text, labels, instructions.
  2. **2D (Visual Representations):** Graphics, images, icons, typography.
  3. **3D (Physical Objects/Space):** The hardware, device, or physical context (e.g., mouse, keyboard, mobile phone).
  4. **4D (Time):** How the interaction unfolds over time (e.g., animations, sound, response time).
  5. **5D (Behavior):** How users act and how the system responds (e.g., clicking, swiping, system feedback).

2. Interaction Design Paradigms

- **Implementation-centric Design:**
  - *Definition:* Interaction directly maps to how system functions are implemented internally. User interacts with low-level system components.
  - *Pros:* Easy to build and debug.
  - *Cons:* High learning curve (user must understand system internals), requires user skill, difficult for system to perform high-level actions based on user goals.
- **Metaphoric Design:**
  - *Definition:* Design follows a real-world metaphor familiar to users (e.g., desktop, folders, trash can, bookshelf).
  - *Goal:* Leverage users' existing knowledge to reduce learning time.
  - *Leeda:* Can be applied broadly (Global Metaphor like Magic Cap's 'Downtown') or locally. Mixed metaphors combine different domains.
  - *Global Metaphor Cons:* Can be hard to scale, may not exist for new concepts, cultural differences, difficult to adapt as capabilities evolve.
- **Idiomatic Design:**
  - *Definition:* Design uses unique, learned conventions or "idioms" specific to the interface (e.g., mouse-cursor mapping, swiping gestures, pull-to-refresh).
  - *Focus:* Learnability is key. Relies on creating intuitive, discoverable, and memorable interactions, even if not based on real-world metaphors.

3. Affordances

- **Definition:** The perceived properties of an object or design element that suggest how it can be interacted with (e.g., a button looks 'pressable', a handle looks 'pullable'). Concept from J. Gibson, adapted by D. Norman for design.
- **Types in Design (Gaver):**
  - **Perceptible Affordance:** The action is possible, and the design clearly communicates it (Ideal).
  - **Hidden Affordance:** An action is possible, but the design doesn't clearly communicate it (Requires learning/discovery, e.g., swipe gestures, some touch controls).
  - **False Affordance:** The design suggests an action is possible, but it isn't, or it works differently than expected (Leads to errors/frustration, e.g., underlined text that isn't a link, a door handle that says "Push").
  - **Correct Rejection:** An action is not possible, and the design doesn't suggest it.

4. Principles of Navigation

- **Wayfinding:** Helping users understand where they are, where they can go, and how to get there.
  - *Elements:* Signage (clear labels, icons), Environmental Clues (consistent layout, visual style), Maps (site maps, overviews).
- **Cost:** The time and effort required for navigation.
  - *Minimize:* Context switches, errors, delays, number of steps.
- **Aids:** Specific UI elements that support navigation.
  - *Global:* Main menus, tabs, persistent sidebars (site-wide).
  - *Utility:* Links for sign-in, help, search, print (common tasks).
  - *Associative/In-line:* Contextual links within content ("related articles").
- **Models:** Common structures for organizing interface components/pages.
  - *Hub and Spoke:* Central screen linking to distinct sections (common in apps).
  - *Fully Connected:* All main sections link to each other (simple websites).
  - *Multi-level:* Hierarchical structure (e.g., site sections with subsections).
  - *Stepwise:* Linear or branching sequence for tasks (wizards, checkouts).
  - *Pyramid:* Stepwise, but allows returning to a central hub from any step.
  - *Pan-and-Zoom:* Navigating a large, continuous information space (maps, large images, long documents).
  - *Flat Navigation:* Single main workspace, minimal navigation between screens (simple tools).
  - *Modal Panel:* Flat navigation + temporary overlays for specific tasks/inputs.
  - *Clear Entry Points:* Direct access to key functions, bypassing main navigation.
  - *Bookmarks:* User-created shortcuts to specific locations/states.
  - *Escape Hatch:* A way to quickly return to a main/home screen from deep within the structure.

1. Introduction to Usability Evaluation

- **Definition:** Assessing the effectiveness, efficiency, and user satisfaction with design solutions.
- **Types:**
  - **Testing-based:** Empirical testing with representative users.
  - **Expert-review-based (Usability Inspection):** Evaluation by experts following established protocols. (Focus of this lecture).

2. Heuristic Evaluation (Jakob Nielsen)

- **Definition:** A small set of evaluators examines an interface to judge its compliance with recognized usability principles (heuristics).
- **Nielsen's 10 Usability Heuristics:**
  1. **Visibility of system status:** Keep users informed about ongoing processes with appropriate feedback in reasonable time (e.g., progress bars, loading indicators).
  2. **Match between system and the real world:** Use language and concepts familiar to the user, not system-oriented jargon. Follow real-world conventions and logical order.
  3. **User control and freedom:** Provide clearly marked "emergency exits" (e.g., cancel buttons) and support undo/redo functionality.
  4. **Consistency and standards:** Use consistent terminology, actions, and design elements throughout the interface (internal consistency). Follow established platform and industry conventions (external consistency).
  5. **Error prevention:** Design to prevent errors from occurring in the first place. Use confirmation dialogs for critical actions and constraints (e.g., disabling invalid options).
  6. **Recognition rather than recall:** Minimize user memory load by making objects, actions, and options visible. Avoid requiring users to remember information between different parts of the interface.
  7. **Flexibility and efficiency of use:** Provide accelerators (e.g., keyboard shortcuts, gestures) for expert users while still being easy for novices. Allow customization for frequent actions.
  8. **Aesthetic and minimalist design:** Interfaces should not contain irrelevant or rarely needed information. Keep content and visual design focused on essentials.
  9. **Help users recognize, diagnose, and recover from errors:** Error messages should be in plain language, indicate the problem precisely, and constructively suggest a solution.
  10. **Help and documentation:** While aiming for a system usable without documentation, provide help that is easy to search, task-focused, lists concrete steps, and is concise.
- **Process:**
  1. Select 3–5 experts and the heuristics.
  2. Each expert individually inspects the interface against the heuristics.
  3. Experts consolidate findings, rate severity, prioritize issues, brainstorm solutions, and report.
- **Number of Evaluators:** Rule of thumb is 3–5; finds most major issues with diminishing returns beyond that.
- **Reporting:** Typically highlights top 3–5 problems. Reports include location, heuristic violated, description, severity rating, justification, and suggestions.
- **Severity Ratings (Example Scale):**
  - *0:* Not a usability problem.
  - *1:* Cosmetic problem (fix if time).
  - *2:* Minor usability problem (low priority).
  - *3:* Major usability problem (high priority).
  - *4:* Usability catastrophe (imperative to fix).
- **Pros:** Inexpensive, fast, intuitive, can be used early and often, good training tool.
- **Cons:** Doesn't involve real users, may miss issues or generate false positives, doesn't give a comprehensive interaction understanding.

3. Cognitive Walkthrough

- **Definition:** An expert review method focused on evaluating the *learnability* and *discoverability* of an interface, especially for first-time users.
- **Requirements:** A prototype (or system), a defined user profile, a set of specific user tasks, and the sequence of actions needed to complete those tasks.
- **Process:** For *each step* in the action sequence for a given task, the evaluator asks and answers four questions from the perspective of the target user:
  1. **Will the user try and achieve the right outcome?** (Does the interface support the user's goal and mental model at this step?)
  2. **Will the user notice that the correct action is available?** (Is the necessary control/element visible and recognizable?)
  3. **Will the user associate the correct action with the outcome they expect?** (Is it clear that this specific action will move them towards their goal?)
  4. **If the correct action is performed, will the user see that progress is being made towards their intended outcome?** (Is there clear feedback indicating the action was successful and they are on the right path?)
- **Pros:** Excellent for evaluating walk-up-and-use interfaces, new concepts, and designs for diverse users. Can be done frequently at any design stage. Good at finding learnability issues.
- **Cons:** Primarily focuses on learnability/discoverability, not overall efficiency or satisfaction. Best used in conjunction with user testing. Can be time-consuming for complex interfaces.

1. What is Accessibility?

- **Usability:** The effectiveness, efficiency, and satisfaction with which *specified users* can achieve *specified tasks* in a *particular environment* (ISO 9241-11).
- **Accessibility:** The usability of a product, service, environment, or facility by people with the *widest range of capabilities* (ISO 9241-20). It's about making systems usable by as many people as possible, with a primary focus on individuals with disabilities.

2. Understanding Disability

- **Definition (CDC):** A condition of the body or mind (impairment) making it harder to do certain activities (activity limitation) and interact with the world (participation restrictions).
- **Variability:** Can change over time (age, recovery); severity can fluctuate. Most disabilities are acquired, not congenital.
- **Dimensions of Impairment:**
  - *Anatomical:* Differences in body function (sensory, physical, cognitive).
    - *Sensory:* Visual (blindness, low vision, color blindness), Auditory (deafness, hearing loss), Tactile, Olfactory/Gustatory, Balance/Spatial.
    - *Physical:* Motor/Mobility issues affecting limbs or movement.
    - *Cognitive:* Affecting memory, attention, problem-solving, communication, etc.
    - *Seizures:* Sensitivity to light/motion (e.g., photosensitive epilepsy).
  - *Temporal (Microsoft Persona Spectrum):*
    - *Permanent:* Long-term/congenital (e.g., blindness, limb difference).
    - *Temporary:* Short-term, improves over time (e.g., broken arm, ear infection).
    - *Situational:* Context-induced limitation (e.g., driving, noisy bar, new parent holding baby).
- **Models of Disability:**
  - *Medical Model:* Views disability as a personal health condition residing within the individual.
  - *Social Model:* Views disability as arising from the *mismatch* between an individual's abilities and the barriers present in the environment or society. *Disability = Mismatch(Ability + Context).*

3. Accessible Design Approaches

- **Goal:** Create products/environments usable by all people to the greatest extent possible.
- **Universal Design (Ron Mace):** Designing for everyone from the start, without needing adaptation or specialized design later.
  - *Premise:* Designs benefiting individuals with disabilities often benefit everyone (e.g., curb cuts, closed captions).
  - **7 Principles:**
    1. **Equitable Use:** Useful and marketable to people with diverse abilities. Provide same/equivalent means of use; avoid segregation; ensure equal privacy/safety; be appealing.
    2. **Flexibility in Use:** Accommodates a wide range of preferences and abilities (choice of method, L/R handed use, accuracy, adaptable pace).
    3. **Simple and Intuitive Use:** Easy to understand regardless of user's experience, knowledge, language, or concentration level (eliminate complexity, be consistent, support literacy, arrange by importance, provide feedback).
    4. **Perceptible Information:** Communicates necessary information effectively regardless of ambient conditions or user's sensory abilities (use redundant modes—visual, verbal, tactile; ensure contrast; maximize legibility; make elements describable; support assistive tech).
    5. **Tolerance for Error:** Minimizes hazards and adverse consequences of accidental actions (arrange to minimize errors, provide warnings, use fail-safes, discourage unconscious action in critical tasks).
    6. **Low Physical Effort:** Can be used efficiently and comfortably with minimum fatigue (allow neutral body position, reasonable force, minimize repetition/sustained effort).
    7. **Size and Space for Approach and Use:** Appropriate size/space provided for approach, reach, manipulation, and use regardless of body size, posture, or mobility (clear line of sight, comfortable reach, accommodate grip size, space for assistive devices).
- **Web Content Accessibility Guidelines (WCAG):** International standard providing technical requirements for web content accessibility.
  - **Conformance Levels:** A (minimum), AA, AAA (highest).
  - **Key Principles (POUR):** Content must be Perceivable, Operable, Understandable, Robust.
  - **Examples (Level A):** Labels for form controls, meaningful heading sequence, sufficient color contrast, usable keyboard navigation, meaningful error messages.
  - **Testing:** Manual inspection and automated tools (e.g., WAVE, axe).
  - **Note:** WCAG sets the minimum bar (the "floor").

4. Assistive Technologies (AT)

- **Definition:** Specialized tools used by individuals with disabilities to bridge accessibility gaps when universal design is insufficient.
- **Examples:**
  - **Vision:** Screen Readers (read screen content aloud), Screen Magnifiers, Text Readers, Refreshable Braille Displays.
  - **Motor:** Alternative Input Devices (Head/Mouth Wands, Eye Tracking, Motion Tracking, Single-Switch Access like Sip-and-Puff), Speech Recognition/Voice Input.
  - **Speech:** Augmentative and Alternative Communication (AAC) devices.
- **Inline Styles:** `style={{ fontSize: 16, color: 'blue' }}`
- **Combining Styles:**
  `style={styles.base, styles.override, { marginTop: 10 }}`. Styles later in the array override earlier ones.
- **Image:** `source={{ uri: 'network_url' }}`
- **<Button title="..." onPress=.../>**, cannot be styled
- **TouchableNativeFeedback** only works on Android, **TouchableHighlight** for iOS.
- **<ScrollView>** renders all children at once.
- **Animated:**
  - `useNativeDriver: true`: Essential for smooth performance. Cannot be used for layout properties (width, height, top, left, etc.).
  - `.start(callback)`: Takes an optional callback function that runs on completion.
- **Navigator** must be nested inside of a `NavigationContainer`.
- **Stack Navigation:** Can push a screen onto the history stack via `navigation.push(screenName, params)`
  - `screenName` is the name of the screen to navigate to.
  - `params` (optional) is received as `props.route.params`.
- **accessibilityLabel:** Primary information (what the element is).
- **accessibilityHint:** Auxiliary information (what happens when interacted with).
- **Passing Data to Dynamically Generated Screens:**
  - When creating screens dynamically (e.g., from an array using `.map()`), use a render callback function as the `component` or `children` prop of `<Screen>` to pass item-specific data as props to the screen component.

```
{
    items.map((item) => (
        <Drawer.Screen key={item.id} name={item.name}>
            {(props) => <MyScreenComponent {...props} {...item} />}
        </Drawer.Screen>
    ));
}
```

- **Auth:** `"Authorization": "Bearer ${jwtToken}"`
- **Deployment:**
  - Install CLI: `npm install -g eas-cli`
  - Build: `eas build -p android` (generates .apk/.aab), `eas build -p ios` (generates .ipa)
- `await` is only allowed within `async` functions and at the top-level of our code.
- **Wit.ai Response Structure:** The JSON response from Wit.ai typically includes:
  - **text:** The original user utterance.
  - **intents:** A list of potential matching intents with confidence scores.
  - **entities:** An object containing detected entities and their values.
    - **Keywords:** Entity value must match a predefined list (can include synonyms, e.g., "spooky" and "halloween" map to the value spooky).
    - **Free Text:** Entity value is not restricted to a predefined list.
    - **Free Text & Keywords:** Uses a predefined list but is also open to new, unseen values.
  - **traits:** Other detected characteristics (not detailed in this lecture).
- **A message has a** `role` and `content`
  - **platform: By OpenAI; content restrictions and moderation. Cannot be changed.**
  - **developer: By you; directive and purpose of agent.**
  - **assistant: By agent; response to user queries.**
  - **user: By client; queries and questions.**
- **Not doing** `resp.json()`, but `const reader = resp.body.getReader(); const decoder = new TextDecoder("utf-8");`

```
app.get("/messages", (req, res) => {
    const chatroom = req.query.chatroom;
    const comment = req.body.comment;
    res.status(200).send({ msg: "..." });
});
```

- **Middleware**
  - **Functions that execute before the route handler. They have access to** `req`, `res`, **and a** `next` **function.**
  - **Purpose: Used for tasks like body parsing (**`app.use(express.json())`**), logging, authentication, setting CORS headers, etc.**
  - **Execution: Middleware is executed in the order it's added using** `app.use()`**. Call** `next()` **to pass control to the next middleware or route handler.**
- `INSERT INTO TableName (Column1, Column2) VALUES (Value1, Value2);`
- `DELETE FROM TableName WHERE Condition;`
- `db.exec('SQL')`: **Execute arbitrary SQL, ignores results (e.g., CREATE TABLE).**
- `db.run('SQL', [param1, param2])`: **Run INSERT, UPDATE, DELETE, ignores results.**
- `db.get('SQL', [param1, param2])`: **Get the first matching row as an object.**
- `db.all('SQL', [param1, param2])`: **Get all matching rows as an array of objects.**
- **HTML Tags**
  - `<strong>` and `<em>`
  - `<table>`, `<thead>`, `<tbody>`, `<tr>`, `<th>`, `<td>`
  - `<label>`: label an input, should always include a for attribute
- **CSS**
  - . applies to class, # applies to id
  - p.intro applies to `<p class="intro">`
  - `<link rel="stylesheet" href="styles.css">`
- **JS**
  - Any value that is not `false`, `undefined`, `null`, `0`, `NaN`, or `""` returns true
  - Arrays don't have to be the same type
  - loops:
    - `for (let attr in course) {}` // loop through object
    - `for (let item of items) {}` // loop through array
  - `JSON.stringify(obj)` converts object to string
  - `JSON.parse(str)` converts string to object
  - Data fetched from an API does an implicit `JSON.parse`
  - Requests can be synchronous or asynchronous. Asynchronous requests are recommended as they are non-blocking.
  - then and catch take a callback function as an argument.
  - Declarative functions: `filter`, `map`, `some`, `every`, `reduce`
    - `[2, 4, -1.1, 7.2].reduce((prev, curr) => prev + curr, 1.2) // 13.3`
- **Syntactic Sugar**
  - Spread Operator: `...`
  - Nullish Coalescing Operator: `??`
    - checks for `null` or `undefined`
    - `x = a ?? b` // if a is null or undefined, x = b
- **Copying**

```
let myComplexObj = { name: "Cole", pets: ["pepper", "spud"] };
let refCopy = myComplexObj; // reference copy
// nested attributes are still references
let shallowCopy = { ...myComplexObj };
let deepCopy = JSON.parse(JSON.stringify(myComplexObj)); // deep copy
```

- **React**
  - `<button onClick={() => alert("Hello")}>Click Me</Button>`
- **Empty State**

```
const [name, setName] = useState("James");
console.log(name); // James
setName("Jim");
console.log(name); // still James
```

- **Responsive Design**

```
<div class="container">
    <div class="row">
        <div class="col-12 col-md-6 col-lg-3"></div>
    </div>
</div>

<Container>
    <Row>
        <Col xs={12} md={6} lg={3}></Col>
    </Row>
```

- **Pagination**

```
<Pagination>
    <Pagination.Item active={false}>1</Pagination.Item>
</Pagination>
```

- **Handling Text input**
  - in a controlled way using its value and tracking `onChange` events
  - in an uncontrolled way using `useRef`
- **Pass in props**

```
const [pizza, setPizza] = useState();
<Recipe name="name" ingredients="ingredients" />
<Recipe {...pizza} />
```

- **useContext**

```
// TODO
```

- **cookies vs session vs local**
- Cookies hold a small amount of data. When set as HTTPOnly, they hold that data securely. For us, that data is a JSON Web Token (JWT) exchanged for the user's credentials.
- For a successful request with credentials, the server will send back a Set-Cookie response header to your browser. This includes your newly-issued JWT!
- For requests that either set or use cookies, we must include credentials. This needs to be done for any requests related to logging in, logging out, or creating a post!
- JWT is a kind of token, which can be stored in cookies.

```
fetch("http://localhost:3001/api", {
    method: "POST",
    credentials: "include",
    headers: {
        "Content-Type": "application/json",
    },
    body: JSON.stringify({ name: "Cole" }),
});
```

- **Type of Routers**
  - BrowserRouter: What you typically think of!
  - MemoryRouter: Same as BrowserRouter, but the path is hidden from the browser in memory!
  - HashRouter: Support for deploying SPAs.
  - StaticRouter: Used for server-side rendering.
- **Routing Example**

```
<BrowserRouter>
    <Routes>
        <Route path="/" element={<Layout />}>
            <Route index element={<Home /> } />
            <Route path="about-us" element={<AboutUs /> } />
            <Route path="*" element={<Home /> } />
        </Route>
    </Routes>
</BrowserRouter>
```

- **Outlet**

```
<>
    <Navbar>{/* some nav links */}</Navbar>
    <Outlet />
</>
```

- **CRUD Operations via HTTP**
  - Create: POST
  - Read: GET
  - Update: PUT
  - Delete: DELETE
- **HTTP request**
  - Request Message Header
    - Request Line
    - Request Header Fields
  - A blank line
  - Request Message Body
    - For a JSON body, we need an additional HTTP header `Content-Type: application/json`
- **HTTP response**
  - Response Message Header
    - Status Line
    - Response Header Fields
  - A blank line
  - Response Message Body
- **HTTP Code**
  - 100s – Informational
  - 200 – OK
  - 300s – Redirection
  - 304 – Not Modified
  - 400s – Client Error
  - 400 – Bad Request
  - 401 – Unauthorized
  - 404 – Not Found
  - 409 – Conflict
  - 413 – Request Entity Too Large
  - 429 – Too Many Requests
  - 500s – Server Error
- The browser will pre-flight (i.e., ask permission first) any complex request like POST; that's why you may see two requests in your network log! This is because of Cross-Origin Resource Sharing (CORS), which can get quite complex!
- **useRef**

```
const inputVal = useRef();
return (
    <div>
        <label>Input</label>
        <input ref={inputVal}></input>
        // The value of the input is inputVal.current.value
    </div>
);
```

- **memo**

```
// Will not re-render if return is true
export default memo(GroceryList, (prevProps, nextProps) => {
    return (
        prevProps.apples === nextProps.apples &&
        prevProps.bananas === nextProps.bananas &&
        prevProps.coconuts === nextProps.coconuts
    );
});
```

- **useEffect**
  - When components mount (appear for the first time):
    - The parent component's render phase completes first
    - Then the child component's render phase completes
    - After rendering, the child component's `useEffect` runs first
    - Finally, the parent component's `useEffect` runs
  - When components update:
    - The parent component renders
    - The child component renders
    - The child component's `useEffect` cleanup function (if provided) runs
    - The child component's `useEffect` runs
    - The parent component's `useEffect` cleanup function (if provided) runs
    - The parent component's `useEffect` runs