Week 10: Flexbox, Frameworks, and Bootstrap

From manual CSS layout decisions to reusable responsive interface systems

Core Ideas

- Flexbox explains how page sections line up before any framework is used
- Bootstrap is easier when students already understand parent/child layout thinking
- Today moves from custom CSS layout to reusable framework classes
- The goal is not memorizing classes; the goal is choosing the right layout tool

Today's Path

1. Build layout logic with Flexbox
2. Compare custom CSS with framework utilities
3. Rebuild patterns using Bootstrap grid and components

Class Goal

Students should leave with one practical page section they can build both manually and with Bootstrap.

Why Flexbox Before Bootstrap?

Framework classes are easier when the underlying layout idea is already clear

Core Ideas

- Bootstrap grid and navbars are built on layout principles students already know
- Flexbox teaches parent-first thinking: the container controls the group
- gap, alignment, wrapping, and direction are the language behind many modern layouts
- If Flexbox feels clear, Bootstrap feels like a shortcut instead of magic

Teacher Move

Show the same navbar idea twice: first with display:flex, then with Bootstrap classes.

Avoid This

Do not let students jump directly to copied Bootstrap snippets without explaining why the layout works.

From Isolated Boxes to Page Sections

Good spacing is not enough; page elements must relate to each other intentionally

Core Ideas

- A page is made of groups: navbars, hero rows, card sections, forms, and footers
- Layout decides how those groups breathe, align, and wrap
- Flexbox is strongest for one-dimensional arrangements: row or column
- Students should identify the parent container before writing layout rules

Quick Diagnosis

Ask: Which elements belong together? Which element should become the flex container?

Mini Example

```
.navbar { display:flex; justify-content:space-between; align-items:center; gap:1rem; }
```

Flex Container and Flex Items

The parent becomes the layout manager; the children become coordinated items

Core Ideas

- display:flex is placed on the parent, not randomly on every child
- Direct children become flex items
- Many layout decisions happen at the parent level
- This creates consistent spacing and alignment without many manual margins

Code Pattern

```
.card-row {
  display:flex;
  gap:1rem;
  flex-wrap:wrap;
}
```

Common Check

If justify-content or align-items is not working, first verify that the correct parent has display:flex.

Direction, Wrap, and Flow

Flexbox becomes predictable when students control flow deliberately

Core Ideas

- flex-direction:row creates horizontal flow
- flex-direction:column creates vertical flow
- flex-wrap:wrap allows items to move to a new line
- gap creates clean spacing without margin fights

When to Use

Use row for navigation, hero content, and card rows. Use column for stacked cards, forms, and mobile sections.

Bridge to Bootstrap

Bootstrap grid classes behave like a structured way to control wrapping and column width across breakpoints.

Main Axis and Cross Axis

Axis thinking is the difference between guessing and controlling alignment

Core Ideas

- justify-content works along the main axis
- align-items works along the cross axis
- The main axis changes when flex-direction changes
- Students should say the axis out loud before choosing a property

Class Demo

Toggle flex-direction from row to column and ask why justify-content suddenly appears to behave differently.

Rule of Thumb

First decide direction, then decide distribution, then decide alignment.

Pattern 1: Menus and Simple Bars

Flexbox is ideal for navigation bars, toolbar rows, and grouped links

Core Ideas

- Logo and links can be separated with justify-content:space-between
- gap keeps link spacing clean and readable
- align-items:center prevents vertical misalignment
- Hover states make navigation feel intentional

Code Pattern

```
.nav { display:flex; justify-content:space-between; align-items:center; }
.nav-links { display:flex; gap:1rem; }
```

Bootstrap Preview

Later, navbar and spacing utilities will package this same idea into reusable classes.

Pattern 2: Hero and Two-Column Layouts

A hero section usually balances message, action, and visual support

Core Ideas

- Use a flex row when text and image should sit side by side
- Use max-width to keep text readable
- Use gap instead of many one-off margins
- On smaller screens, the layout may need to stack

Design Decision

A good hero answers three questions: What is this page? Why should I care? What should I do next?

Bootstrap Preview

Bootstrap grid will let students create similar two-column layouts with col-md-* classes.

Pattern 3: Card Rows and Gap

Repeated content needs a layout system, not hand-tuned spacing

Core Ideas

- Cards should align as a group, not as isolated boxes
- flex-wrap prevents overflow on narrow screens
- gap creates consistent spacing between repeated items
- Cards are a natural bridge from custom CSS to Bootstrap components

Code Pattern

```
.cards { display:flex; flex-wrap:wrap;
gap:1rem; }
.card { flex:1 1 220px; }
```

Class Check

Ask students to resize the browser and observe whether cards wrap cleanly.

Common Flexbox Mistakes

Most Flexbox problems come from choosing the wrong parent or wrong axis

Core Ideas

- Putting display:flex on the child instead of the group parent
- Using justify-content when align-items is needed
- Forgetting that direction changes the axes
- Adding many fixes without testing after each change

Debug Order

1. Find the parent
2. Confirm direct children
3. Check direction
4. Add gap/wrap
5. Align only after structure is correct

Teacher Note

This slide is worth keeping short but practical: students remember debugging steps better than property lists.

Layout Sprint: Build Before Bootstrap

Students first build the section manually, then rebuild it faster with Bootstrap

Task Requirements

- Create a navbar with logo and links
- Add a hero row with text, button, and image placeholder
- Add a three-card section that wraps cleanly
- Use display:flex, gap, justify-content, align-items, and flex-wrap

Success Criteria

The page should remain readable, aligned, and spacious when the browser width changes.

Bridge Question

After the manual version works, ask: Which parts felt repetitive enough that a framework could help?

What a CSS Framework Does

A framework gives prebuilt styling rules and layout utilities

Core Ideas

- Frameworks speed up repetitive design work
- They provide ready-made classes for spacing, layout, color, and components
- They help students build clean results quickly
- They also teach naming systems and reusable UI thinking

Simple Definition

A CSS framework is a prepared styling toolkit that reduces the amount of custom CSS needed for common interface patterns.

Balanced View

Frameworks are useful, but students should still understand the CSS ideas underneath them.

Why Frameworks Became Popular

Speed, consistency, and repeatable patterns matter in real projects

Core Ideas

- Teams need consistent buttons, forms, spacing, and layout behavior
- Frameworks reduce repeated work across many pages
- They make prototyping much faster
- They provide a shared vocabulary for common interface parts

Main Advantage

Students can focus more on structure and decisions when basic component styling is already available.

Main Caution

A framework should not replace understanding. It should amplify it.

Popular CSS Frameworks

Framework awareness helps students understand the ecosystem

Core Ideas

- Bootstrap is widely known for layout utilities and components
- Tailwind CSS is utility-first and highly configurable
- Bulma uses readable class-based layout rules
- Foundation is another established responsive framework

Classroom Choice

Bootstrap is often the easiest first framework because its class names are readable and its component set is broad.

Teaching Angle

This lesson uses Bootstrap as a practical entry point, not as the only possible workflow.

Introducing Bootstrap

Bootstrap gives responsive layout tools and ready-made UI components

Core Ideas

- Bootstrap combines layout utilities and component styles
- It supports responsive containers, grid columns, tables, buttons, alerts, forms, and more
- Students can assemble polished screens quickly with class-based HTML
- It is especially useful for education because results appear fast

Big Idea

Bootstrap does not remove the need for HTML and CSS knowledge. It builds on that knowledge.

Student Benefit

When students already know selectors and spacing, Bootstrap becomes a productivity tool instead of a mysterious shortcut.

Bootstrap Workflow

Use the framework as a layer on top of strong HTML structure

Core Ideas

- Start with clean semantic HTML
- Add Bootstrap CSS and JavaScript files
- Apply classes to structure and components
- Customize only when the built-in classes are not enough

Practical Rule

Do not start by memorizing dozens of classes. Start by understanding categories: layout, spacing, color, text, and components.

Classroom Flow

Teach one Bootstrap feature, then immediately show the rendered result and explain the class names in normal language.

Bootstrap Starter Structure

A minimal page setup can connect the framework and prepare a layout area

HTML

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <link rel="stylesheet" href="bootstrap.min.css">
</head>
<body>
  <div class="container py-4">
    <h1>Hello, Bootstrap</h1>
  </div>
  <script src="bootstrap.bundle.min.js"></script>
</body>
</html>
```

CSS

```
/* Bootstrap provides the base styling. */
/* Extra custom CSS is optional and can be added later.
*/
```

Practice Prompt

Students should first prove the framework is connected by rendering one heading, one paragraph, and one button cleanly.

Containers

A container creates a responsive content wrapper

Core Ideas

- container creates a centered layout area with responsive width changes
- container-fluid stretches across the full viewport width
- Containers help control readability and alignment
- Most Bootstrap layouts begin with one of these wrappers

Best Beginner Rule

Use container for most page sections first. Move to container-fluid only when the design truly needs full-width behavior.

Visual Result

A container protects content from touching the screen edges too aggressively on large and small devices.

Container Example

A standard container and a fluid container create different layout feelings

HTML

```
<div class="container bg-light p-4 mb-3">
  Centered content area
</div>
<div class="container-fluid bg-secondary text-white p-4">
  Full-width content area
</div>
```

CSS

```
/* Bootstrap classes handle spacing and colors here. */
/* The difference comes from container vs container-fluid. */
```

Practice Prompt

Place the same card inside both wrappers and ask students which version feels calmer for reading text-heavy content.

Bootstrap Utility Snapshot

A compact reference table for the most useful utility groups

CSS CODE	EXPLANATION	USAGE
`container`	Centered responsive wrapper	`<div class="container">...</div>`
`container-fluid`	Full-width wrapper	`<div class="container-fluid">...</div>`
`text-*`	Text color utility	`<p class="text-primary">...</p>`
`bg-*`	Background utility	`<div class="bg-dark text-white">...</div>`
`row / col-*`	Grid layout system	`<div class="row"><div class="col-md-6">`
`table`	Styled table	`<table class="table table-striped">`
`btn`	Button styling	`<button class="btn btn-primary">`
`card`	Card component	`<div class="card">...</div>`

Color Utilities

Bootstrap can style text and backgrounds quickly with utility classes

Core Ideas

- text-* classes change text color
- bg-* classes change background color
- Color utilities speed up emphasis and layout grouping
- Contrast still matters even when the framework supplies the palette

Use With Care

Framework colors are fast, but thoughtful combination still matters. Students should always check readability.

Good Pattern

A dark background often needs text-white. A warning color may need darker text for legibility.

Grid System

Bootstrap divides the layout into a 12-column responsive system

Core Ideas

- A row contains columns
- Columns can change width at different breakpoints
- The grid helps structure pages without writing custom float or width rules
- This is one of Bootstrap's strongest teaching tools

Memory Hook

Think in rows and columns, not in random widths.

Why Students Like It

The grid gives a quick visual reward: a page starts feeling organized very early in the exercise.

Basic Grid Example

Two columns can stack on small screens and sit side by side on larger ones

HTML

```
<div class="container">
  <div class="row g-3">
    <div class="col-md-6">
      <div class="p-3 bg-light border">Column A</div>
    </div>
    <div class="col-md-6">
      <div class="p-3 bg-light border">Column B</div>
    </div>
  </div>
</div>
```

CSS

```
/* col-md-6 means each column uses half width from the
md breakpoint upward. */
/* On smaller screens, each column naturally becomes
full width. */
```

Practice Prompt

Resize the page and have students explain why the columns stack first and align later.

Breakpoint-Based Columns

Grid classes become more expressive when breakpoints are combined

Core Ideas

- col-12 creates full width on all screens
- col-sm-6 splits content earlier
- col-md-4 can create three columns from medium screens upward
- Mixed breakpoint classes help design stable transitions

Common Pattern

Use full-width content first, then reduce column width only when the screen has enough room.

Design Reminder

A denser grid is not automatically better. Content must still feel readable and tappable.

Typography Utilities

Bootstrap can speed up headings, text emphasis, and alignment

Core Ideas

- Display headings create stronger visual hierarchy
- Lead paragraphs help highlight introductory text
- Text alignment utilities handle left, center, and end alignment
- Utility classes can quickly improve hierarchy without custom CSS

Good Use

Use utility classes to accelerate routine text styling, then refine only where the design truly needs custom behavior.

Teaching Tip

Compare plain text with a version that uses lead text and hierarchy utilities so the value feels obvious.

Table Classes

Bootstrap tables improve clarity with very little code

Core Ideas

- The table class adds baseline styling
- striped and hover variants improve scanning
- bordered tables can clarify dense data
- Framework classes save time while keeping the structure semantic

Student Reminder

Tables are still for data, not for whole-page layout.

Useful Combination

table table-striped table-hover often creates a clean classroom example quickly.

Bootstrap Table Example

A small course table becomes clearer with a few framework classes

HTML

```
<table class="table table-striped table-hover table-bordered">
  <thead class="table-dark">
    <tr><th>Week</th><th>Topic</th><th>Status</th></tr>
  </thead>
  <tbody>
    <tr><td>9_2</td><td>Responsive Design</td><td>Ready</td></tr>
    <tr><td>10</td><td>Flexbox</td><td>Next</td></tr>
  </tbody>
</table>
```

CSS

```
/* Bootstrap handles striping, hover, borders, and header contrast. */
```

Practice Prompt

Ask students to change only the table classes and compare which version is easiest to scan.

Border Utilities

Borders can be controlled with reusable classes instead of custom CSS

Core Ideas

- Border utilities can add, remove, or recolor borders
- Rounded utilities can soften corners quickly
- These classes are useful for cards, thumbnails, and layout grouping
- Small visual changes can improve component clarity a lot

Design Advice

Use borders to support structure. Avoid surrounding every element with equally strong outlines.

Classroom Prompt

Show three boxes with different border treatments and ask which one feels most balanced.

Hero Sections and Jumbotron-Style Layouts

Large attention areas are useful for announcements, introductions, and onboarding

Core Ideas

- Older Bootstrap lessons often call this a jumbotron-style section
- The main idea is still valid: a large content block that attracts attention
- It usually includes a strong heading, supporting text, and one call to action
- Spacing, contrast, and hierarchy matter more than the name

Teaching Choice

Focus students on the design purpose of the component rather than only the historical class name.

Student Output

A good hero section should feel readable, not just big.

Alert Components

Alerts communicate status and feedback quickly

Core Ideas

- Alerts can signal success, warning, information, or error states
- They are useful for form feedback, system messages, or short notices
- Strong color and spacing help them stand out immediately
- Good alert text stays short and direct

Main Rule

An alert should communicate one clear message, not become a paragraph container.

Examples

Success, warning, danger, info, and secondary styles cover many student exercises effectively.

Buttons

Buttons are one of the fastest ways to see Bootstrap utility in action

Core Ideas

- btn creates the base button style
- Variant classes such as btn-primary or btn-success change emphasis
- Size classes help create hierarchy
- Button groups and alignment can organize actions more clearly

Strong Principle

One screen should usually have one primary action. Too many equally strong buttons create confusion.

Student Prompt

Ask which button should lead the user and style that one with the strongest visual emphasis.

Badges

Badges are small labels that add quick context

Core Ideas

- Badges can show counts, status, or category labels
- They often appear near headings, buttons, or cards
- Their small size means contrast and spacing matter a lot
- They are useful for portfolios, dashboards, and announcements

Good Use Cases

New, Draft, 3 items, Updated, Beginner, Advanced

Design Note

A badge should support the main content, not compete with it.

Cards

Cards are reusable content containers for repeated information blocks

Core Ideas

- Cards can hold images, titles, text, metadata, and actions
- They are useful for courses, products, profiles, and articles
- Consistent spacing inside the card is essential
- Cards make component thinking visible to students very quickly

Why They Matter

Cards combine layout, typography, actions, and hierarchy in one manageable teaching example.

Student Target

Students should be able to explain why the card feels organized, not only list the classes used.

Bootstrap Card Example

A course card combines structure, utility classes, and one action button

HTML

```
<div class="card shadow-sm" style="max-width: 22rem;">
  <div class="card-body">
    <span class="badge bg-primary mb-2">Week 10</span>
    <h5 class="card-title">Responsive Design Lab</h5>
    <p class="card-text">Learn media queries, layout
shifts, and Bootstrap basics.</p>
    <a href="#" class="btn btn-outline-primary">Open
Task</a>
  </div>
</div>
```

CSS

```
/* The card, badge, shadow, spacing, and button classes
work together. */
```

Practice Prompt

Ask students to restyle this card for a dark section without losing readability.

Collapse and Accordion Patterns

Expandable components help hide detail until the user asks for it

Core Ideas

- Collapse keeps a page cleaner when details are optional
- Accordion layouts help organize FAQ or step-based content
- Students should use these patterns when information can be progressively revealed
- Overusing hidden content can make pages harder to scan

Best Use

Frequently asked questions, extra details, and multi-step explanations are good fits.

Design Reminder

Collapsed content still needs a clear label so users know what they are opening.

Accordion Example

A small FAQ block shows how expandable content can stay organized

HTML

```
<div class="accordion" id="faq">
  <div class="accordion-item">
    <h2 class="accordion-header" id="q1">
      <button class="accordion-button" type="button"
data-bs-toggle="collapse" data-bs-target="#a1">What is
responsive design?</button>
    </h2>
    <div id="a1" class="accordion-collapse collapse
show" data-bs-parent="#faq">
      <div class="accordion-body">It adapts layout and
readability to different devices.</div>
    </div>
  </div>
</div>
```

CSS

```
/* This example assumes the Bootstrap JavaScript bundle
is connected. */
```

Practice Prompt

Change the question text and add a second item so students see the repeating structure clearly.

Responsive Navigation Bars

Navigation must stay usable when horizontal space becomes limited

Core Ideas

- Responsive navbars compress and expand based on screen width
- Brand area, menu links, and the toggle button must stay clear
- Navigation should not overwhelm the first screen on mobile
- Students should understand both the layout and the user experience goal

Main Goal

A navbar should stay findable, readable, and easy to tap on every device size.

Teaching Tip

Have students compare a crowded fixed menu with a responsive one so the problem becomes concrete.

Navbar Example

A responsive navbar expands on larger screens and collapses on smaller ones

HTML

```
<nav class="navbar navbar-expand-md navbar-dark bg-dark fixed-top">
  <div class="container">
    <a class="navbar-brand" href="#">Course Hub</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#menu">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="menu">
      <ul class="navbar-nav ms-auto">
        <li class="nav-item"><a class="nav-link" href="#">Home</a></li>
      </ul>
    </div>
  </div>
</nav>
```

CSS

```
/* Keep enough top spacing in the page body when using a fixed-top navbar. */
```

Practice Prompt

Ask students what changes when the screen becomes narrow: which part hides, which part stays visible, and why.

Form Classes

Bootstrap can quickly make forms cleaner and easier to scan

Core Ideas

- form-control styles inputs, textareas, and selects
- Grouping labels and fields improves readability
- Good form layout supports both desktop and mobile use
- Clear spacing and error feedback matter as much as color

Student Reminder

A form is not finished when it only looks good. It must also feel easy to complete.

Good Habit

Keep labels visible and keep the action button clearly separated from the fields.

Form Example

A compact contact form uses standard framework classes

HTML

```
<form class="p-3 border rounded bg-light">
  <div class="mb-3">
    <label class="form-label">Full Name</label>
    <input type="text" class="form-control">
  </div>
  <div class="mb-3">
    <label class="form-label">Message</label>
    <textarea class="form-control" rows="3"></textarea>
  </div>
  <button class="btn btn-primary">Send</button>
</form>
```

CSS

```
/* Spacing utilities such as mb-3 help the form breathe
without custom CSS. */
```

Practice Prompt

Let students add a select field and one checkbox group so they practice more than one control type.

Carousel and Slider Components

A slider can present images or highlights, but it should be used carefully

Core Ideas

- Carousels can rotate featured content or images
- They can add movement and variety to the page
- Overuse can reduce clarity and control
- Students should use them when the content really benefits from sequential presentation

Use Thoughtfully

A carousel is useful for a gallery or highlights section, but weak for critical information users must not miss.

Accessibility Reminder

Motion, timing, and control buttons should stay understandable to the user.

Carousel Example

A basic carousel can cycle through featured images or messages

HTML

```
<div id="gallery" class="carousel slide">
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="slide-1.jpg" class="d-block w-100"
alt="Slide 1">
    </div>
    <div class="carousel-item">
      <img src="slide-2.jpg" class="d-block w-100"
alt="Slide 2">
    </div>
  </div>
  <button class="carousel-control-prev" type="button"
data-bs-target="#gallery" data-bs-
slide="prev"></button>
  <button class="carousel-control-next" type="button"
data-bs-target="#gallery" data-bs-
slide="next"></button>
```

CSS

```
/* The JavaScript bundle is required for the
interactive controls. */
```

Practice Prompt

Discuss whether the content would actually be clearer as cards or a static gallery before choosing the carousel.

Modal Components

Modals bring focused content above the main page without leaving the current screen

Core Ideas

- Modals can show details, forms, confirmation prompts, or extra media
- They are useful when users need a temporary focused layer
- They should not replace full pages when the content is long or complex
- Clear close actions are essential

Best Use Cases

Product details, short forms, quick confirmation messages, and optional extra information work well in modals.

Design Caution

If the content needs a lot of scrolling or many decisions, a dedicated page may be better than a modal.

Modal Example

A button opens a small detail window without navigating away

HTML

```
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#details">
  Open Details
</button>
<div class="modal fade" id="details">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Product Details</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal"></button>
      </div>
      <div class="modal-body">Short focused content goes here.</div>
    </div>
  </div>
</div>
```

CSS

```
/* Modals also depend on the Bootstrap JavaScript bundle. */
```

Practice Prompt

Ask students whether the content belongs in a modal, an accordion, or a full page and make them justify the choice.

Icons

Icons add quick meaning when they are used with restraint

Core Ideas

- Icons can support navigation, alerts, actions, and feature labels
- They work best when paired with text, not when they replace meaning entirely
- Consistent icon style helps the interface feel more coherent
- Students should understand icons as support, not decoration alone

Good Rule

If an icon becomes unclear without text, the interface may be asking too much from the symbol alone.

Simple Workflow

Use a lightweight icon set consistently and avoid mixing many unrelated visual styles.

Guided Practice Flow

A short in-class build ties together responsiveness and Bootstrap

Core Ideas

1. Create a clean HTML page shell
2. Add a container and a responsive grid
3. Build one hero section and one card row
4. Add a navbar and one form block
5. Improve the layout with a media query or Bootstrap breakpoint classes

Practice Goal

Students should move from random assembly to intentional page planning: wrapper, hierarchy, components, and responsive behavior.

Assessment Lens

Look for readable structure, calm spacing, responsive behavior, and clear class choices rather than flashy output alone.

Mini Project Brief

A student-friendly assignment that extends today's lesson

Core Ideas

- Build a small course page, event page, or product highlight page
- Use a responsive wrapper and a grid section
- Include at least four Bootstrap components or utility groups
- Keep the design readable on both mobile and desktop widths
- Write a short explanation of the most important layout decisions

Minimum Requirements

Navbar, one content section, one card or table section, one form or modal, and visible responsive behavior.

Quality Standard

A good project looks calm, clear, and intentional. It should not feel like disconnected components pasted onto one page.

Summary

Responsive logic and framework literacy make CSS work faster and stronger

Core Ideas

- Responsive design protects readability across devices
- Media queries describe when a visual change should happen
- Frameworks accelerate common layout and UI work
- Bootstrap gives students a practical component toolkit
- Strong results still depend on good HTML structure and design judgment

Final Message

Frameworks are most useful when students already understand the CSS ideas underneath them.

Next Step

Students are now ready for more confident component building, layout refinement, and project-oriented front-end work.