# Design

## 10/25

### Slides

#### Design is a process that involves thoughtfulness ([thoughtful design](https://www.youtube.com/watch?v=BBPjKgpODIc))

* + Design can encourage healthy (or unhealthy) habits, foster sustainable lifestyles (or not), and make us feel happy (or frustrated). Thoughtful design takes this important influence into account, and thoughtful designers carefully consider the implications on consumers' behaviour and habits when they craft brands, objects and interfaces. – Kristen Nozell, Ad Age

#### Form

##### ([Key principles: navigation, hierarchy, color](https://www.youtube.com/watch?v=GJN7TemsZtY))

##### Image

* + - Photography
    - Illustration
    - Line and shape
    - Texture
    - ([Logo, photos, illustrations, mascots, 3d renders](https://blog.tubikstudio.com/web-design-basic-types-of-images-web-content/))

##### Color

* + - Hue
    - Value
    - Intensity

##### Typography

* + - Font selection
      * T[he main four types of typography:](https://xd.adobe.com/ideas/principles/web-design/best-modern-fonts-for-websites/)
      * [Serif fonts, Sans-serif fonts, Script fonts, Display fonts](https://xd.adobe.com/ideas/principles/web-design/best-modern-fonts-for-websites/)
    - Type, size, alignment
    - Letter spacing
    - Line spacing
    - Grammar

##### Composition

* + - Rhythm
    - Proportion
    - Structure
    - Variation
    - Balance
    - Boundary
    - Space

#### Context

* + Device
  + Web browser
  + Age of visitor
  + Literacy
  + Geographic location
  + Language(s)
  + Ability

#### Accessibility

* + When we talk about accessible code, what we are really talking about at its core is inclusiveness. The actual process . . . involves rules and standards, tests and tools; but inclusive development is more abstract than that. It’s a shift in thinking . . . Inclusive development means making something valuable, not just accessible, to as many people as we can.” —Carie Fisher

##### Categories of disability

* + - Vision impairment
    - Mobility impairment
    - Auditory impairment
    - Cognitive impairment
* Cultivating a mindful design approach allows you to do more with less

#### **Digital Accessibility**: the practice of ensuring that digital technology, including websites, mobile applications, immersive experiences, digital environments, and mixed reality, can be consumed by anybody or entity, regardless of visual, mobile, cognitive, and auditory abilities.

#### **Universal Design**: Needing accessible digital environments is also necessary for the non-disabled, and/or those compromised due to situations such as a broken limb, pregnancy, or simply juggling tasks and only having one hand to work with. Imagine being in a noisy airport and needing to watch a video concerning your flight and having lost your headphones. In this case, captions would be essential.

#### **W3C**:

* + Access to digital technologies, including to the Internet, is a basic human right, according to the United Nations Convention on the Rights of Persons with Disabilities. Most of the international community has adopted this UN convention and other binding policies.
  + The World Wide Web Consortium (W3C) has created international standards for digital creations, in addition, the W3C Web Accessibility Initiative (WAI) has further developed standards and support materials to help designers, developers, content creators, designers and others in the field understand and implement accessibility.
  + "The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect," says Tim Berners-Lee, W3C Director and inventor of the World Wide Web W3C.

#### Perceivable

* + Provide text alternatives for non-text content
  + Provide captions and other alternatives for multimedia
  + Create content that can be presented in different ways, including by assistive technologies, without losing meaning
  + Make it easier for users to see and hear content

#### Operable

* + Make all functionality available from a keyboard
  + Give users enough time to read and use the content
  + Do not use content that causes seizures or physical reactions
  + Help users navigate and find content
  + Make it easier to use inputs other than the keyboard

#### Understandable

* + Make text readable and understandable
  + Make content appear and operate in predictable ways
  + Help users avoid and correct mistakes

#### **Robust**: maximize compatibility with current and future user tools

* (Some kind of conclusion:) The practice of digital accessibility is deeply rooted in principles of Universal Design. Universal Design is the design and composition of an environment so that it can be accessed, understood, and used to the greatest extent possible by all people regardless of their age, size, ability, or disability. An environment (or any building, product, or service in that environment) should be designed to meet the needs of all people who wish to use it. This is not a special requirement, for the benefit of only a minority of the population. It is a fundamental condition of good design. (The Centre for Excellence in Universal Design.)
* **Accessibility** - the quality of being easy to obtain, use, understand, reach, or enter
* **Web Accessibility** - the inclusive practice of removing barriers that prevent interaction with, or access to, websites by people with disabilities
* **Assistive Technology**: any item, piece of equipment, or product system, whether acquired commercially off the shelf, modified or customized, that is used to increase, maintain or improve functional capabilities of individuals with disabilities
  + e.g. screen readers, pointing devices, switches, alternative keyboards, siri/echo, etc.
  + Alternate access methods: alternate keyboards, single button mice, trackballs/joysticks, head pointing devices, pointing/typing aides, head mice, switches/onscreen keyboards, touch windows, eye gaze
  + Low tech pointing devices: head/chin pointers, styluses, adapted hand pointers, mouth sticks
* 2010 U.S. Census: Nearly 1 in 5 have disability

#### **Web Content Accessibility Guidelines (WCAG) 2.0** defines how to make Web content more accessible to people with disabilities. Accessibility involves a wide range of disabilities, including visual, auditory, physical, speech, cognitive, language, learning, and neurological disabilities. Although these guidelines cover a wide range of issues, they are not able to address the needs of people with all types, degrees, and combinations of disability. **These guidelines also make Web content more usable by older individuals with changing abilities due to aging and often improve usability for users in general.**

* + Four guiding principles of accessibility: Perceivable, Operable, Understandable, Robust
  + Three levels of conformance: A, **AA**, AAA
  + Divided into 63 success criteria

#### Section 508

* + Applies to the federal government
  + Adopted as part of the 2001 Amendment to the Rehabilitation Act
  + Initially based on WCAG 1.0 (May 1999)

#### WCAG 2.0 vs. Section 508

* + (WCAG 2.0 are) more explicit
  + Written in a way that is technology neutral and is therefore directly applicable to a wide range of content types and formats
  + 38 A and AA. Of these, 22 are phrased differently but equivalent in substance to current 508 requirements.
  + Section 508 is in the process of being updated to align with WCAG 2.0 (level AA)
  + WCAG 2.1 - 28 additional; scheduled to be implemented in 2018

#### Spectrum of Disabilities

##### Visual

* + - Low vision
      * Partial sight
      * Poor acuity
      * Tunnel vision
      * Clouded vision
    - Color blindness
    - Blindness
    - Examples of good practice:
      * Images & controls should have equivalent text alternatives
      * Text, images & page layouts can be resized without losing information.
      * Video content has text or audio alternatives, or audio-description track.
      * Text and images have sufficient contrast between foreground and background color.
      * Provide consistent, predictable navigation.
      * Avoid using color alone to identify links or controls.

##### Auditory

* + - Hard of hearing
    - Deafness
    - Examples of good practice:
      * Audio content, including videos, provide captions or transcripts.
      * Media players provide volume controls.
      * Media players provide options to adjust caption text size and colors.
      * No interactions that rely on using voice only

##### Physical

* + - Amputation
    - Arthritis
    - Fibromyalgia
    - Rheumatism
    - Muscular dystrophy
    - Repetitive stress injury
    - Tremors and spasms
    - Quadriplegia
    - Examples of good practice:
      * Provide full keyboard support
        + All links, menu items, controls accessible via keyboard (Tab, Shift+Tab, & Return keys)
        + No keyboard traps
      * Provide sufficient time to complete tasks.
      * Provide consistent, predictable, simple navigation and page functions.
      * Link targets, buttons should be of sufficient size

##### Cognitive

* + - Attention deficit hyperactivity disorder (ADHD)
    - Autism spectrum disorder (ASD)
    - Memory impairments
    - Multiple sclerosis
    - Perceptual or learning disorders
    - Seizure disorders
    - Examples of good practice:
      * Provide simple navigation and page layouts that are easy to understand and use.
      * Avoid, when possible, complex sentences that are difficult to read or unusual words.
      * Avoid moving, blinking, or flickering content. Or provide method to disable.
      * Video, animations, or audio content can be paused or stopped.
      * Simple text is supplemented by images, graphs, or illustrations.

##### Speech

#### Accessibility > Compliance: your site can be compliant, yet inaccessible

#### Checking for Accessibility

* + Manual testing
    - Keyboard check
    - Use with screen reader
  + Online tools
    - [WebAIM WAVE](https://wave.webaim.org/)

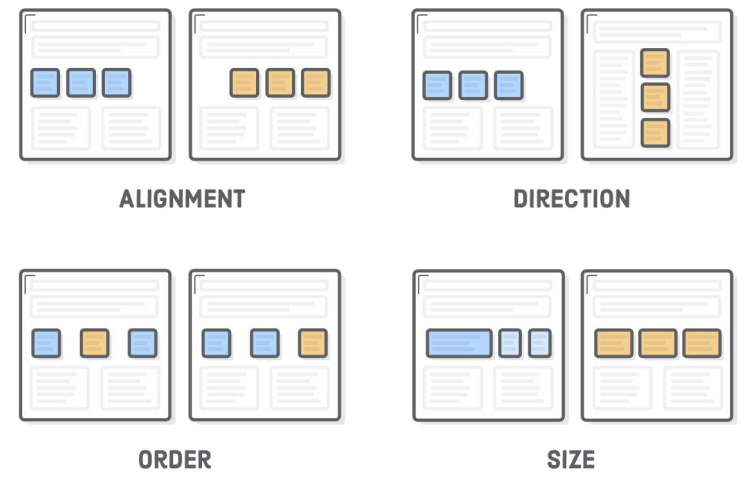
# Web Layout

## 10/30

### Slides

* Flexbox: flex containers + flex items
  + The “Flexible Box” or “Flexbox” layout mode offers an alternative to Floats for defining the overall appearance of a web page. Whereas floats only let us horizontally position our boxes, flexbox gives us complete control over the alignment, direction, order, and size of our boxes.

#### Flexbox vs Grid

* + CSS has always been used to lay out our web pages, but it's never done a very good job of it. First, we used tables, then floats, positioning and inline-block, but all of these methods were essentially hacks and left out a lot of important functionality (vertical centering, for instance).
  + Flexbox helped out, but it's intended for simpler one-dimensional layouts, not complex two-dimensional ones (Flexbox and Grid actually work very well together). Flexbox is generally used for smaller, simpler layouts and Grid for complicated 2-Dimensional ones. Flexbox is also very useful for handling a large amount of different content sizes
* 

#### Flexbox overview: Flexbox uses two types of boxes: flex containers and flex items. The job of a flex container is to group a bunch of flex items together and define how they’re positioned

#### Flex containers

* + The first step in using flexbox is to turn one of our HTML elements into a flex container. We do this with the display property. By giving it a value of flex, we’re telling the browser that everything in the box should be rendered with flexbox instead of the default box model.
  + .menu-container {

display: flex;

…

}

#### Aligning flex items (horizontal) (e.g. justify-content: center;)

* Other values for justify-content are shown below:
  + center, flex-start, flex-end, space-around, space-between
  + Space-around vs space-between
    - (space-between)This value makes it so that the first item is on the start line, the last item is on the end line, and any remaining space is distributed between the items. The space between the flex items is equal, but there is no space before the first item or after the last item.
    - (space-around)This value results in the items having space around them. Unlike space-between, half the space is distributed before the first item and after the last item, and the remaining space is distributed between the items. This means that each item has equal space around it, and the space between items is twice the space before the first item and after the last item.

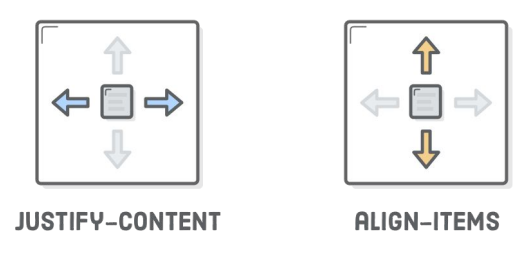
#### Multiple flex items

* + justify-content: space-around;
  + This turns our .menu into a nested flex container, and the space-around value spreads its items out across its entire width.

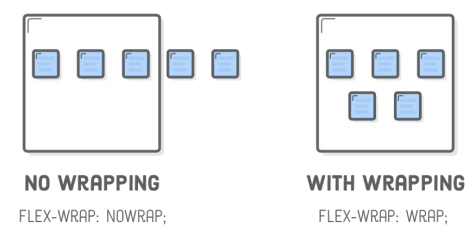
#### Grouping flex items

* + Flex containers only know how to position elements that are one level deep (i.e., their child elements). They don’t care one bit about what’s inside their flex items. This means that grouping flex items is another weapon in your layout-creation arsenal. Wrapping a bunch of items in an extra <div> results in a totally different web page.
  + 

#### Cross-axis (vertical) alignment

* + Flex containers can also define the vertical alignment of their items.
  + 

#### Wrapping flex items as a grid

* + Not only can it render items as a grid - it can change their alignment, direction, order, and size. To create a grid, we need the flex-wrap property.
  + 

#### Individual items - order

* + Adding an order property to a flex item defines its order in the container without affecting surrounding items. Its default value is 0, and increasing or decreasing it from there moves the item to the right or left, respectively.
  + This can be used, for example, to swap order of the .first-item and .last-item elements in our grid.
  + .first-item {

order: 1;

}

.last-item {

order: -1;

}

#### Flex item alignment

* + We can do the same thing with vertical alignment. What if we want that Subscribe link and those social icons to go at the bottom of the header instead of the center? Align them individually! This is where the align-self property comes in. Adding this to a flex item overrides the align-items value from its container:
  + .social,

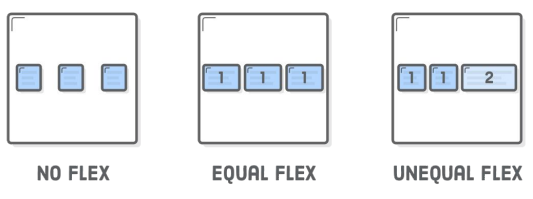
.subscribe {

align-self: flex-end;

margin-bottom: 20px;

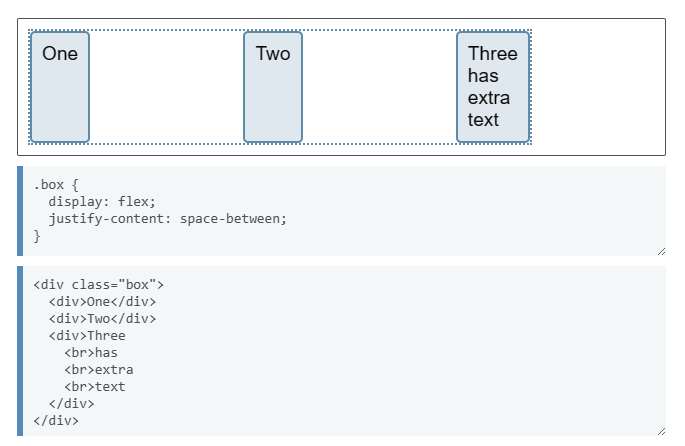
}

#### Flexible items

* + The flex property defines the width of individual items in a flex container. Or, more accurately, it allows them to have flexible widths. It works as a weight that tells the flex container how to distribute extra space to each item. For example, an item with a flex value of 2 will grow twice as fast as items with the default value of 1.
  + 

### Websites (网页里的不一定重要，我挑了些可能是重点的放进去了，可以忽略)

#### [MDN web docs: CSS Flexible Box Layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout)

* The CSS flexible box layout module defines a CSS box model optimized for user interface design, and the layout of items in one dimension. In the flex layout model, the children of a flex container can be laid out in any direction, and can "flex" their sizes, either growing to fill unused space or shrinking to avoid overflowing the parent. Both horizontal and vertical alignment of the children can be easily manipulated.
* In the following example a container has been set to display: flex, which means that the three child items become flex items. The value of justify-content has been set to space-between in order to space the items out evenly on the main axis. An equal amount of space is placed between each item with the left and right items being flush with the edges of the flex container. You can also see that the items are stretching on the cross axis, due to the default value of align-items being stretch. The items stretch to the height of the flex container, making them each appear as tall as the tallest item.
* 

#### [MDN web docs: CSS Position](https://developer.mozilla.org/en-US/docs/Web/CSS/position)

* The position [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) property sets how an element is positioned in a document. The [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top), [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right), [bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/bottom), and [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left) properties determine the final location of positioned elements.
* [static](https://developer.mozilla.org/en-US/docs/Web/CSS/position#static)
  + The element is positioned according to the normal flow of the document. The [top](https://developer.mozilla.org/en-US/docs/Web/CSS/top), [right](https://developer.mozilla.org/en-US/docs/Web/CSS/right), [bottom](https://developer.mozilla.org/en-US/docs/Web/CSS/bottom), [left](https://developer.mozilla.org/en-US/docs/Web/CSS/left), and [z-index](https://developer.mozilla.org/en-US/docs/Web/CSS/z-index) properties have *no effect*. This is the default value.
* [relative](https://developer.mozilla.org/en-US/docs/Web/CSS/position#relative)
  + The element is positioned according to the normal flow of the document, and then offset *relative to itself* based on the values of top, right, bottom, and left. The offset does not affect the position of any other elements; thus, the space given for the element in the page layout is the same as if position were static.
  + This value creates a new [stacking context](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_positioned_layout/Understanding_z-index/Stacking_context) when the value of z-index is not auto. Its effect on table-\*-group, table-row, table-column, table-cell, and table-caption elements is undefined.
* [absolute](https://developer.mozilla.org/en-US/docs/Web/CSS/position#absolute)
  + The element is removed from the normal document flow, and no space is created for the element in the page layout. The element is positioned relative to its closest positioned ancestor (if any) or to the initial [containing block](https://developer.mozilla.org/en-US/docs/Web/CSS/Containing_block#identifying_the_containing_block). Its final position is determined by the values of top, right, bottom, and left.
  + This value creates a new [stacking context](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_positioned_layout/Understanding_z-index/Stacking_context) when the value of z-index is not auto. The margins of absolutely positioned boxes do not [collapse](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_box_model/Mastering_margin_collapsing) with other margins.
* [fixed](https://developer.mozilla.org/en-US/docs/Web/CSS/position#fixed)
  + The element is removed from the normal document flow, and no space is created for the element in the page layout. The element is positioned relative to its initial [containing block](https://developer.mozilla.org/en-US/docs/Web/CSS/Containing_block#identifying_the_containing_block), which is the viewport in the case of visual media. Its final position is determined by the values of top, right, bottom, and left.
  + This value always creates a new [stacking context](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_positioned_layout/Understanding_z-index/Stacking_context). In printed documents, the element is placed in the same position on *every page*.
* [sticky](https://developer.mozilla.org/en-US/docs/Web/CSS/position#sticky)
  + The element is positioned according to the normal flow of the document, and then offset relative to its *nearest scrolling ancestor* and [containing block](https://developer.mozilla.org/en-US/docs/Web/CSS/Containing_block) (nearest block-level ancestor), including table-related elements, based on the values of top, right, bottom, and left. The offset does not affect the position of any other elements.
  + This value always creates a new [stacking context](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_positioned_layout/Understanding_z-index/Stacking_context). Note that a sticky element "sticks" to its nearest ancestor that has a "scrolling mechanism" (created when overflow is hidden, scroll, auto, or overlay), even if that ancestor isn't the nearest actually scrolling ancestor.
* Description
* Types of positioning
  + A **positioned element** is an element whose computed position value is either relative, absolute, fixed, or sticky. (In other words, it's anything except static.)
* A **relatively positioned element** is an element whose computed position value is relative. The top and bottom properties specify the vertical offset from its normal position; the left and right properties specify the horizontal offset.
* An **absolutely positioned element** is an element whose computed position value is absolute or fixed. The top, right, bottom, and left properties specify offsets from the edges of the element's containing block. (The containing block is the ancestor relative to which the element is positioned.) If the element has margins, they are added to the offset. The element establishes a new block formatting context (BFC) for its contents.
* A **stickily positioned element** is an element whose computed position value is sticky. It's treated as relatively positioned until its containing block crosses a specified threshold (such as setting top to value other than auto) within its flow root (or the container it scrolls within), at which point it is treated as "stuck" until meeting the opposite edge of its containing block.
* Most of the time, absolutely positioned elements that have height and width set to auto are sized so as to fit their contents. However, non-replaced, absolutely positioned elements can be made to fill the available vertical space by specifying both top and bottom and leaving height unspecified (that is, auto). They can likewise be made to fill the available horizontal space by specifying both left and right and leaving width as auto.
* Except for the case just described (of absolutely positioned elements filling the available space):
  + If both top and bottom are specified (technically, not auto), top wins.
  + If both left and right are specified, left wins when direction is ltr (English, horizontal Japanese, etc.) and right wins when direction is rtl (Persian, Arabic, Hebrew, etc.).
* Accessibility concerns
  + Ensure that elements positioned with an absolute or fixed value do not obscure other content when the page is zoomed to increase text size.
  + Scrolling elements containing fixed or sticky content can cause performance and accessibility issues. As a user scrolls, the browser must repaint the sticky or fixed content in a new location. Depending on the content needing to be repainted, the browser performance, and the device's processing speed, the browser may not be able to manage repaints at 60 fps, causing accessibility concerns for people with sensitivities and jank for everyone. One solution is to add [will-change: transform](https://developer.mozilla.org/en-US/docs/Web/CSS/will-change) to the positioned elements to render the element in its own layer, improving repaint speed and therefore improving performance and accessibility.

## 11/1

### Slides

#### Grid layout: header, sidebar, content, footer

* + Grid layouts are fundamental to the design of websites, and the CSS Grid module is the most powerful and easiest tool for creating it

#### Columns & Rows - 2 Dimensions

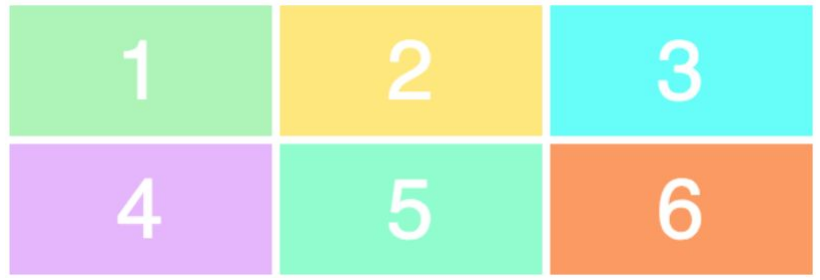
* + .wrapper {

display: grid;

grid-template-columns: 100px 100px 100px;

grid-template-rows: 50px 50px;

}

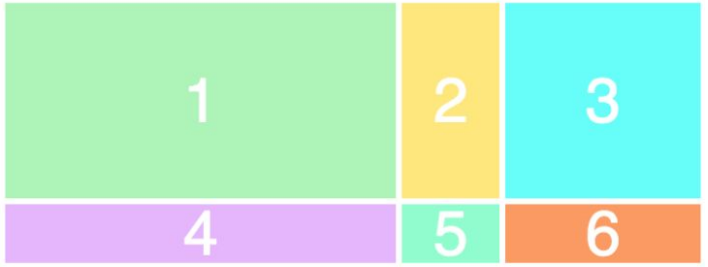
* + As we’ve written three values for grid-template-columns, we’ll get three columns. We’ll get two rows, as we’ve specified two values for the grid-template-rows. The values dictate how wide we want our columns to be (100px) and how tall we’d want our rows to be (50px).
  + 
  + .wrapper {

display: grid;

grid-template-columns: 200px 50px 100px;

grid-template-rows: 100px 30px;

}

* + 

#### Position & size: to position and resize the items we’ll target them and use the grid-column and grid-row properties

* + .item1 {  
     grid-column-start: 1;

grid-column-end: 4;

}

* + What we’re saying here is that we want item1 to start on the first grid line and end on the fourth column line. In other words, it’ll take up the entire row

#### Static vs. Fluid Layouts

* + A **static** page layout (sometimes called a “**fixed**” layout or “fixed width” layout) uses a preset page size and does not change based on the browser width. In other words, the page layout might have a permanent width of 960 pixels no matter what. This is how web pages were traditionally built for many years until modern influences like media queries and responsive design were introduced around the start of the 2010s. Different devices will treat a static page layout in various ways, so the rendered page could be slightly unpredictable. For example, on a desktop browser, if the window is too small horizontally, then the page will be cut off and horizontal scroll bars will be displayed. On a mobile device like an iPhone, the page will be scaled automatically, allowing the user to zoom in on points of interest (provided that no metatags override this default behavior). When new websites are built, most of them don’t opt for a static layout, because it means that the mobile experience will require a separate website. There are major exceptions, such as the online Apple.com store, but Apple is a unique case because a selling point of their mobile devices is that they can view static layouts. In other words, Apple doesn’t seem to be adopting responsive design just yet
  + A **liquid** page layout (sometimes called “**fluid**” or “fluid width”) uses relative units instead of fixed units. Typically a liquid layout will use percentages instead of pixels, but any relative unit of measurement will work, such as ems. A liquid layout often will fill the width of the page, no matter what the width of the browser might be. Liquid layouts don’t require as much thought as a responsive design, but there are some major drawbacks at very large or very small browser widths. If the browser is very wide, some content might be stretched too far. On large screens, a single paragraph might run across the page on a single line. Conversely, a multi-column layout on a small screen could be too crowded for the content.

### Websites

#### [MDN web docs: CSS Grid Layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout)

#### [MDN web docs: Firefox CSS Grid Inspector](https://developer.mozilla.org/en-US/docs/Tools/Page_Inspector/How_to/Examine_grid_layouts)

# Responsive Design

## 11/6

### Slides

#### A Dao of Web Design

* + “The control which designers know in the print medium, and often desire in the web medium, is simply a function of the limitation of the printed page. We should embrace the fact that the web doesn’t have the same constraints, and design for this flexibility. But first, we must ‘accept the ebb and flow of things.’” —John Allsopp, “A Dao of Web Design”

#### Responsive Design

* + Responsive Design Mobile traffic is as (if not more) relevant as desktop traffic now.
  + We should build for the type of screens that will be used to access our sites.
  + Responsive web design uses “media queries” to figure out what resolution of device it’s being served on.
  + Flexible images and fluid grids size correctly to fit the screen.
  + Responsive web design is design for flexibility

#### Foundations of Responsive Design

* + Flexible grids (fluid layouts)
  + Media queries
  + Flexible, responsive images

#### Media Queries

* + Features you can include in a media query include : width, height, device-width, device-height, orientation, aspect-ratio, device-aspect-ratio, color, color-index, monochrome, resolution, scan grid
  + Most of the above can be combined with min- and max- prefixes.
  + The most common media queries assess min-width and max-width.
  + Media queries can be used to load an alternate style sheet or, more commonly, to offer alternate styles within an existing style sheet.

#### Media Query Syntax

* + CSS media queries use the @media rule followed by two optional values: “only” or “not”
  + “only” screens out older browsers from reading the rest of the query.
  + “not” negates the result: “not screen“ means everything except screen-based media.
  + A feature: value pair, enclosed by parentheses, comprises the essence of the media query.
  + Media features that can be assessed are predefined.
  + Multiple feature: value pairs can be combined with “and“

#### CSS Rule Set

* + body {

background-color: orange;

}

#### CSS Positioning

* + The CSS position property specifies the type of positioning used for an element on a page
  + Static: Default document flow
  + Absolute: Element is positioned relative to its first positioned (not static) parent element
  + Fixed: Element is positioned relative to the browser window
  + Relative: Element is positioned relative to its normal position
  + Sticky: Positioned based on the user’s scroll position

#### CSS Rule Set with a Media Query

* + @media only screen and (min-width: 480px) {

body {

background-color: orange;

}

}

* Common media query device break points
  + 

### Websites

#### [MDN web docs: Using CSS custom properties (variables)](https://developer.mozilla.org/en-US/docs/Web/CSS/Using_CSS_custom_properties)

#### [MDN web docs: Responsive images](https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Responsive_images)

## 11/8

### Slides

* Same as 11/6

### Websites

#### [Mobile First Design Strategies](https://xd.adobe.com/ideas/process/ui-design/what-is-mobile-first-design/)

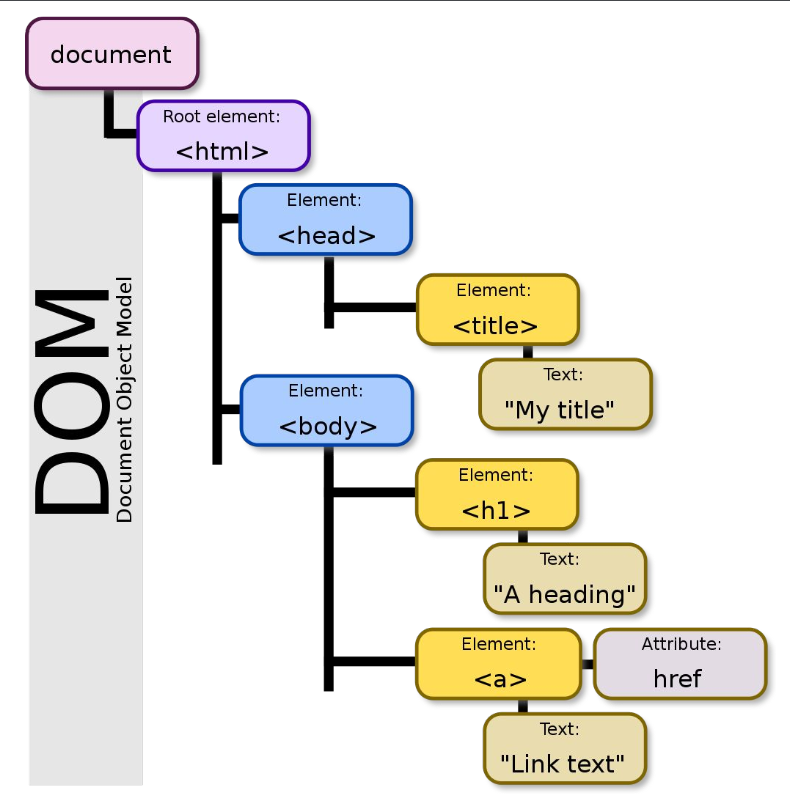
* [Mobile-First Design: The article discusses the importance of mobile-first design, which prioritizes creating user experiences for mobile devices before designing for larger screens1](https://edgeservices.bing.com/edgesvc/chat?udsframed=1&form=SHORUN&clientscopes=chat,noheader,udsedgeshop,channelstable,&shellsig=4841ac814f1becc8eded5a4edb4ded0a386c3711&setlang=en-US&lightschemeovr=1#sjevt%7CDiscover.Chat.SydneyClickPageCitation%7Cadpclick%7C0%7C83541a06-817e-43e2-8e5e-5ae7fa6f65fc%7C%7B%22sourceAttributions%22%3A%7B%22providerDisplayName%22%3A%22Mobile-fir...%22%2C%22pageType%22%3A%22html%22%2C%22pageIndex%22%3A1%2C%22relatedPageUrl%22%3A%22https%253A%252F%252Fxd.adobe.com%252Fideas%252Fprocess%252Fui-design%252Fwhat-is-mobile-first-design%252F%22%2C%22lineIndex%22%3A10%2C%22highlightText%22%3A%22Mobile-first%20design%20is%20a%20design%20philosophy%20that%20aims%20to%20create%20better%20user%20experiences%20by%20beginning%20the%20design%20process%20with%20mobile%20devices%20in%20mind%20first%2C%20often%20prioritizing%20the%20smallest%20of%20screens.%22%2C%22snippets%22%3A%5B%5D%7D%7D).
* User Experience Focus: It emphasizes focusing on the core functions and essential UX components, as mobile users are expected to surpass 72.5% by 2025.
* Design Principles: The page outlines five key mobile-first design principles, including prioritizing users, maintaining visual hierarchy, simplicity, bold CTAs, and considering site loading speed.
* Design Process: It also describes the mobile-first design process, from taking inventory of content to testing on real devices to ensure usability and performance.

# 

# Javascript

* Three layers: HTML - structure, CSS - style/presentation, JavaScript - behavior
* JavaScript:
  + A programming language for creating interactivity and functionality in web browsers
  + Invented by Brendan Eich and introduced by Netscape in 1995
  + the name "JavaScript" was attempted to ride this popularity.
* ECMAScript 2015 (ES6) - latest ver., also second major revision to JavaScript
  + The let keyword
    - let var1; //let decalres a variable with block scope
  + The const keyword
    - const var2 = 200; //const declares a variable whose value cannot change
  + Arrow functions
  + Promises
* Front-end language
  + Because usually rendered in the web browser rather than server=>considered as “front-end” language
  + Javascript code is interpreted and executed by a browser's “javascript engine”, and there are different javascript engines for different browser
* Capability
  + Robust programming language
  + Core function: modifying HTML and CSS, (page content and style) communicating with the server and storing data
* Application
  + Targets HTML elements to do sth with them
  + Ways to incorporate JavaScript to HTML
    - Inline JavaScript
      * <!DOCTYPE html>
      * <html>
      * <head>
      * <title>Inline JavaScript</title>
      * </head>
      * <body>
      * <h1>Inline JavaScript Example</h1>
      * <button onclick="alert('Hello, Inline JS!')">Click me</button>
      * </body>
      * </html>
    - Embedded Javascript
      * <!DOCTYPE html>
      * <html>
      * <head>
      * <title>Internal JavaScript</title>
      * <script>
      * function greet() {
      * alert('Hello, Internal JS!');
      * }
      * </script>
      * </head>
      * <body>
      * <h1>Internal JavaScript Example</h1>
      * <button onclick="greet()">Click me</button>
      * </body>
      * </html>
    - External
      * <!DOCTYPE html>
      * <html>
      * <head>
      * <title>External JavaScript</title>
      * <script src="external.js"></script>
      * </head>
      * <body>
      * <h1>External JavaScript Example</h1>
      * <button onclick="externalGreet()">Click me</button>
      * </body>
      * </html>
  + External and embedded are preferable for their separation of content and behavior
  + difference between them
    - Embedded/internal JavaScript is placed between <script> tags and is part of the HTML document. It can include larger blocks of code and is not limited to a single HTML element's event.
    - Inline JavaScript is tied specifically to an attribute of an HTML element and is usually just a single line or function call.
    - External JavaScript is kept in separate .js files and is included in the HTML document through a reference in the <script src="..."> tag.

Dom Structure (Document Object Model Manipulation)



**Document Object Model (DOM):**

a cross-platform and language-independent interface

treats an XML or HTML document as a tree structure wherein each node is an object representing a part of the document

DOM methods allow programmatic access to the tree; with them one can change the structure, style or content of a document.

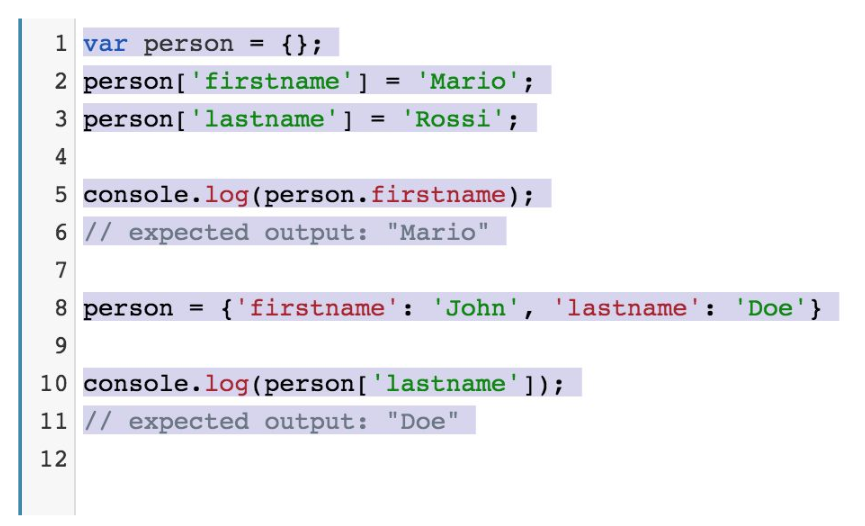
Designed to be independent of any particular programming language, making the structural representation of the document available from a single, consistent API.

implementations of the DOM can be built for any language

* There are four main types of nodes.
* The Document node, which represents the entire page
* Element nodes, which represent individual HTML tags
* Attribute nodes, which represent attributes of HTML tags, such as a class
* Text nodes, which represents the text within an element, such as the content of a tag
* We talk about the relationship between element nodes as “parents,” “children,” and “siblings.”

**Property accessors:**

provide access to an object's properties by using the dot notation or the bracket notation



**JavaScript HTML DOM Elements:**

document.getElementById(" ");

document.getElementsByTagName(" ");

document.getElementsByClassName(" ");

document.querySelectorAll(" "); (CSS selector)

The following **HTML objects** (and object collections) are also accessible:

● document.anchors

● document.body

● document.documentElement

● document.embeds

● document.forms

● document.head

● document.images

● document.links

● document.scripts

● document.title

**DOM Queries/Query: JS method that find elements in the DOM tree**

* May return to one element or “node list”
* Which query use depends on what you want to do and the scope of browser support required
* Method return a single element node:
  + getElementById()
  + querySelector()
* **Return one or more elements as a node list:**
  + getElementsByClassName()
  + getElementsByTagName()
  + querySelectorAll()
* **Examples:**

****

****

**UI events:**

Load

unload error

resize

scroll

**Keyboard events:**

keydown

keyup

keypress

**Mouse Events:**

click

dblclick

mousedown

mouseup

mousemove

mouseover

mouseout

**Focus Events:**

focus

blur

**Form Events:**

input

change

submit

reset

cut

copy

paste

select

**Mutation Events:**

DOMSubtreeModified

DOMNodeInserted

DOMNodeRemoved

DOMNodeInsertedIntoDocument

DOMNodeRemovedFromDocument

**Touch Events:**

touchstart

touchmove

touchend

touchcancel

**Binding:**

Specifying which event will trigger the response is also known as "binding."

There are three different ways to bind an event to an element.

·HTML event handlers

·DOM event handlers

·DOM event listeners

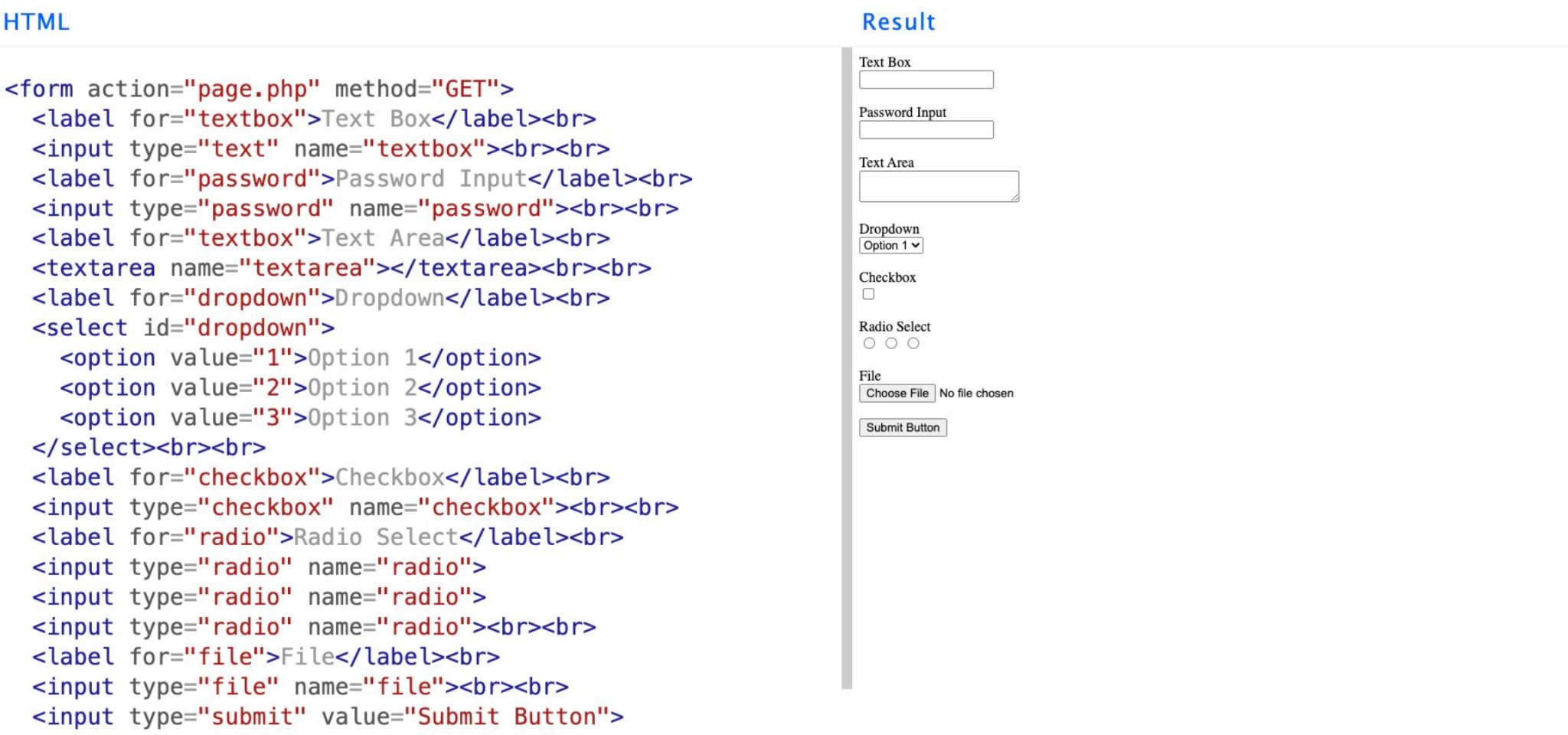
Event handling

1. Select an element for the script to respond to.

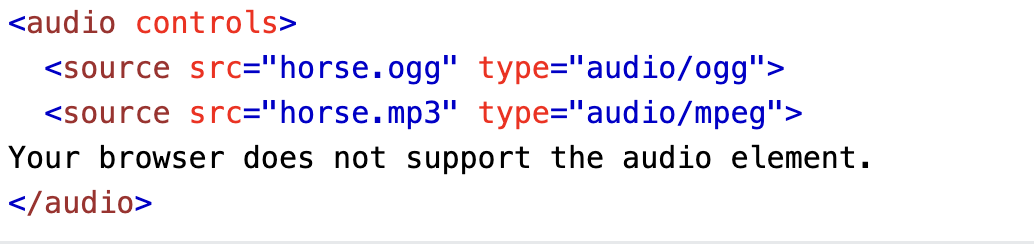
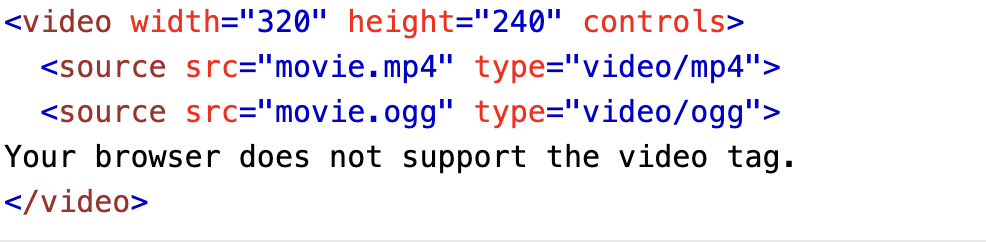
2. Specify which event will trigger the response.

3. Run code specific to that event.

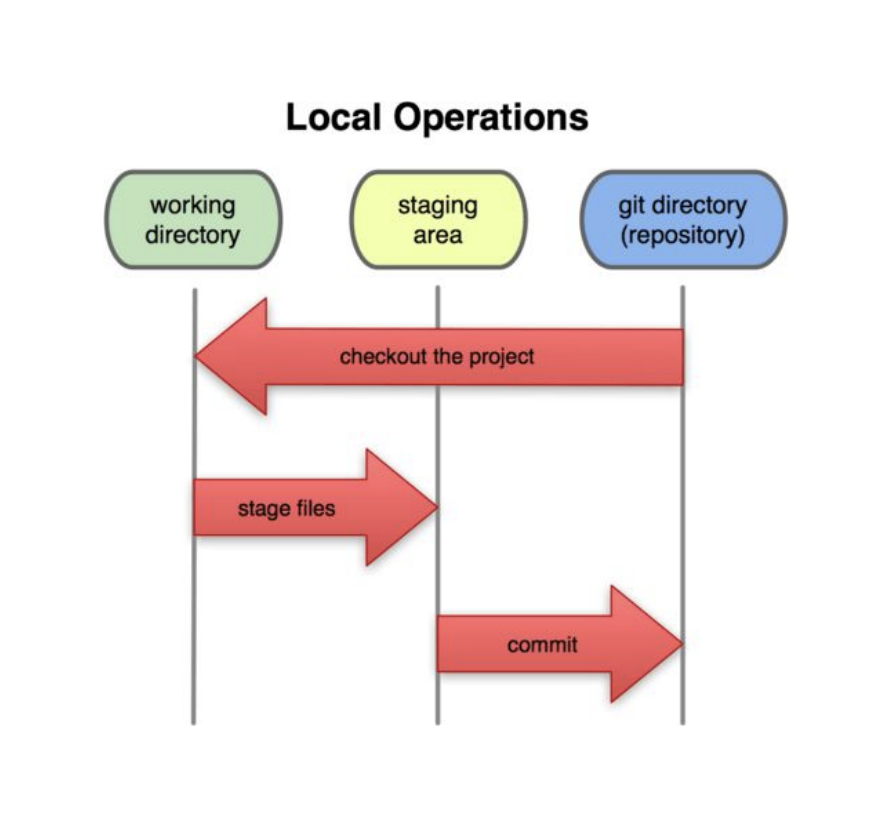
# Web form

* Gathering information from users.
* The HTML<form>element is used to define a form for getting user input.
* It can be used to collecting data
* A variety of form elements are used to provide an interface for the input
* These form elements include text fields, checkboxes, dropdown menus, and buttons.
* 
  + The <br> tag in HTML is used for inserting a line break in the text. It's a self-closing tag, which means it doesn't need an end tag and is written simply as <br> or <br />.
  + Forms always begin with the <form> element.
  + The <form> element's action attribute specifies how the form will be processed.
  + The < input >element is used for various kinds of user input.
  + The<input>element's type attribute determines what kind of input is received from users.
  + Each < input >element must also have a name attribute and value in order for the data to be sent.
* Form validation
  + Before form data gets sent, it’s important to validate the input.
  + You may want to make certain form fields required.
  + You probably want to make sure that certain fields are completed properly.
  + You should also verify that malicious code is not sent along with form input.
  + Form validation can be done client-side, server-side, or both.
  + Often this data is sent to a php file on the server for processing via a http get or post requests (security and functionality)
  + 
  + Normally, forms are sent to the server to be processed.
  + This requires a server-side application written in a back-end language.
  + In HTML, the <input> element is used to create interactive controls for web-based forms to collect data from the user. Depending on the type attribute, an <input> element can render as many different controls
  + 

# Web Audio and Video

* Sound
  + Sound consists of pressure waves moving through air.
  + Without air, there is no sound.
  + Our ears are sensitive to pressure waves and transmit these signals to the brain.
* Sound recording
  + When sound is recorded, acoustic waves are converted to electrical waves.
  + A microphone consists of a small membrane that vibrates.
  + Movements of the membrane are translated into electrical signals.
  + Higher pressure typically corresponds to higher voltage.
* Digital audio
  + An audio signal is an analog (continuous) format.
  + The electrical waves must be converted to digital information for computational processing.
  + Digital recording is accomplished with an analog-to-digital converter (ADC).
  + The ADC captures a snapshot of the electric voltage on an audio line and represents it as a digital number.
  + Capturing the voltage thousands of times per second creates a good approximation of the original audio.
* Digital audio playback
  + All computers must give us analog signals to be useful.
  + The screen converts digital information to light.
  + The digital-to-analog converter (DAC) takes the sample and sets a certain voltage on the analog outputs to recreate the signal.
  + This voltage is conveyed to the speakers which create pressure waves in the air.
* Digital Video
  + Analog video uses a continuous electrical signal to capture footage on magnetic tape. Examples: VHS or Hi-8.
  + In theory, analogue video can capture more information leading than digital video but dv is now so widely implemented it would be rare to find analogue video still being used.
  + Digital Video is encoded in binary information and can be reproduced infinitely with no quality loss.
  + mp4 , mov , avi , mov , .wmv are some formats found on the web
* VIDEO CODECS + FORMATS FOR THE WEB
  + Video codecs are software or hardware tools that encode and decode digital video.
  + RECOMMENDED: MP4 using H264 Codec(Widely used for streaming video on the internet and for video conferencing.)
  + Can also use Quicktime for .mov if an alpha channel is needed(An alpha channel is a concept used in digital graphics and refers to the channel that controls the transparency of an image. It's essentially an 8-bit layer in a graphics file format that represents the opacity level of each pixel within an image. The alpha channel is often stored alongside the standard color channels: red, green, and blue (RGB).)
* HTML Audio and Video
  + HTML5 supports audio and video natively in the web browser
  + For years, it was necessary to rely on a third party to deliver this kind of content.
  + Now we can use the <audio> and <video> tags
  + the <audio> and <video> tags use src attribute or the <source> tag to specify one or more media resources.
  + 
  + 
* HTML Inline Frames
  + Another way to embed media on a web page is with the HTML inline frame element: <iframe>
  + An inline frame represents a nested browsing context, embedding another HTML page into the current one.
  + Embedding all or part of one web page into another is way to present content on a website.

# Version Control

* A system that records changes to a file or set of files over time so that you can recall specific versions later
* Commonly used for software source code but any type of file can be placed under version control
* A Version Control System (VCS) allows you to:
  + Revert files back to a previous state
  + Review changes made over time
  + Collaborate more efficiently
  + Maintain project backups
* Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members
* Centralized Version Control
  + Centralized Version Control Systems (CVCS) are systems that use a central server to store all files and enable team collaboration. The centralization means that there is a single "source of truth" for the entire project, and developers can commit their changes to this central server. Other team members can then update their local copies with changes made by their colleagues.
  + Centralized Version Control Systems were developed to allow collaboration with developers on other systems.
  + With a CVCS, a single server contains all the versioned files, and clients “check out” files from that central place
  + For many years, this has been the standard for version control
  + The downside of centralized version control is the vulnerability of having the entire history of a project in one place.
* Distributed Version Control
  + Distributed Version Control Systems (DVCS) operate under the principle that instead of a single, central repository that acts as the "source of truth," each contributor has a full-fledged, local copy of the repository, including all the history and branches. This approach decentralizes the version control process, offering numerous advantages over centralized version control systems (CVCS).
  + With Distributed Version Control Systems, clients don’t just check out the latest snapshot of files, they fully mirror the entire history of the project.
  + If a server dies, anyone with a copy of all the versioned files can restore it to the server.
  + Every checkout is really a full backup of all the data.
  + You can also collaborate with different groups of people in different ways simultaneously within the same project.
* Git History
  + Git was was created in 2005 by Linus Torvalds and the Linux development community for Linux kernel maintenance
  + Linux is an open source operating system project of fairly large scope
  + Its goal was to be a fully distributed VCS with a simple design, support for non-linear development, and the ability to handle large projects efficiently
* Git Basics
  + Git thinks of its data like a set of snapshots of a mini file system.
  + Every time you save the state of your project, it basically takes a picture of what all your files look like then and stores a reference to that snapshot.
  + To be efficient, if files have not changed, Git doesn’t store the file again—just a link to the previous identical file it has already stored.
  + This makes Git more like a mini file system with some powerful tools built on top of it.
* Git States
  + Git has three main states that your files can reside in: modified, staged, and committed.
  + Modified means that you have changed the file but have not committed it to your database yet.
  + Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
  + Committed means that the data is safely stored in your local database.
  + 
* Git Workflow
  + Modify files in your working directory.
  + Stage the files, adding snapshots of them to your staging area.
  + Commit changes, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory
* GitHub
  + GitHub is a web-based hosting service that uses the Git VCS.
  + The site also provides social networking functionality such as feeds, followers, wikis, and statistics.
  + The company was founded in 2008 and is located in San Francisco.
  + In addition to computer programmers, architects, musicians, municipal governments, and academics are among its users.

# Web Hosting and Domain Names

* Domain names
  + Domain names serve as a more memorable reference to Internet resources.
  + Domain names are used to identify Internet Protocol (IP) addresses.
  + An IP address is an identifier for a node—a computer or device on a network.
  + 
* Top level domain
  + Every domain name has a suffix that indicates which top level domain (TLD) it belongs to.
  + Top-level domains today are grouped as follows:
    - Generic top-level domains
    - Infrastructure top-level domain
    - Country-code top-level domains
    - Sponsored top-level domain
  + Generic top-level domains
    - Generic top-level domains initially consisted of:
      * GOV: Government agencies
      * EDU: Educational institutions
      * ORG: Nonprofit organizations
      * MIL: Military
      * COM: Commercial business
      * NET: Network organizations
    - Some of these, such as .com and .net, are no longer restricted to their original intended usage.
    - More generic TLDs have since been added and are being added today.
    - New top-level domains: <https://www.domain.com/domains/new-domain-extensions>
  + Selecting a Domain Name
    - When you register a domain name, you are not its owner, rather you have the exclusive right to use it.
    - Some factors to consider when selecting a domain name:
      * Relevance to site
      * Communicability
      * Availability
  + Web Hosting
    - A web hosting service allows individuals and organizations to make their website accessible to others.
    - The host usually provides storage space on a server as well as Internet connectivity.
    - Theoretically, any computer can serve as a web host, but it needs to always be on and implement measures for security and stability.
  + Selecting a web hosting
    - Dedicated vs. shared server space Host
    - Disk space •Bandwidth (data transfer)
    - Up time (reliability)
    - Overage
    - Extras: databases, mailboxes, and types of customer support
  + Search Engine Optimization
    - Search engine optimization (SEO) is the process of making your site easy for others to locate.
    - The more thoughtfully and selectively you add keywords to your pages, the better your search rankings.
    - There are several factors that help your website to rise in search results.
    - On-Page Techniques of SEO:
      * This involves identifying and implementing keywords in seven particular places in your page
      * 1. Page title
      * 2. URL
      * 3. Headings
      * 4. Text
      * 5. Link text
      * 6. Image alt text
      * 7. Page descriptions
    - Off-Page Techniques of SEO
      * Search engines also look at the number of other sites that link to yours to determine search ranking
      * This is especially so when the content of a referring site is similar to yours
      * It’s ideal when the words that appear in links to your site also appear in the text of the page that the site links to.
      * Finally, as more people visit your site, the search ranking will also improve.
  + Website Analytics
    - Once people start visiting your site, it’s helpful to know!
    - Analytics tools allow you to observe data about the traffic your site receives.
    - This can include the following information:
      * Number of visits
      * Geographic location of visitors
      * Time spent on pages
      * Referring web page
      * Browser information
      * Real-time activity
  + OPTIMIZING SEO
    - **1. Use a Program That Integrates With Google Analytics to Track Metrics** The best step that you can take in improving your SEO is to understand Google Analytics. Google Analytics is a helpful tool for understanding your website, so you can make intelligent data-based decisions. Metrics will not only help you understand how you are ranking but also your audience, so you can better curate content for them.
      * What browsers they’re using: Google Analytics can provide you with powerful data about which browsers your customers are using. This will allow you to improve their experience by focusing web design on the most-used browsers.
      * The most popular devices used to visit your site: Similarly to knowing the browsers that your customers use, knowing what devices they use can be helpful. Certain sites have more mobile users than desktop users or vice versa. Understanding your website’s metrics will allow you to create content tailored to the most commonly used screen size, enhancing user experience.
      * Referral traffic: Google Analytics also provides you with data about where your website viewers are coming from. You can use this data to see which social media platforms your viewers are using to get to your site. This information is helpful when you pay for social media advertising campaigns
      * Understand your competition: Google Analytics also offers insights into competitor traffic. This data can be useful when deciding how to improve your content and outrank competitors.
    - **Identify Low-Performing Pages and Refresh Them With New Content People Want to Read**
    - **Ensure Your Content Is Created Around a Primary Keyword and Relevant Secondary Keywords**
      * Researching keywords for your content not only helps to develop the framework of your piece; it also enables you to understand what your audience wants to read. Understanding what keywords are best for your target audience and content type can help you build a content strategy to boost SEO
      * Google-friendly writing is dependent on demonstrating a balance between keywords and everyday language. This means that your content should be written in such a way that it flows naturally. Keywords should naturally come up in your content so that you don’t have to stuff your content at the last minute. One way to do this is by identifying semantically related keywords to your primary keyword target.
    - **Diversify Your Backlink Portfolio**
      * Even if you follow all of the tips related to on-page technical SEO, it still won’t guarantee to make the front page of Google. A big part of SEO deals with backlinks and whether you’re generating backlinks from high-authority sites.
      * Backlink diversity come can come from two sources, specifically:
      * The type of backlink: Generally, your backlink will be a dofollow or nofollow, with a dofollow carrying more weight.
      * A site where the backlink originates: If you’re promoting your content, for instance, and targeting publishers to run a story, the site that links back to your content would be the source of your backlink.
      * A diverse backlink portfolio signals to Google that your site is an authoritative source and that you’re generating links in a natural way versus relying on outdated black hat or other spam tactics
    - **Use Effective Header Tags to Target Google Featured Snippets**
      * When developing content, you want to be mindful of how you structure your content on the page. Every page should have content organized logically, with the most important information at the top of the page. In fact, studies have found that 80% of readers spend most of their time looking at the content at the top of the page.
      * To get the most out of the keywords you’re targeting, consider adding jump links to the top of your page. This not only makes for a more enjoyable user experience, but it also allows you to use your header tags in more creative ways to go after Google Featured Snippets.
      * Strategically using header tags helps search engines understand your content better. Identifying headers when writing your content helps search engines when analyzing your content. Headers let search engines know what your content is about, and if you insert keywords into headers, the search engines will have an even better understanding of your content.
      * Headers also help your content to be skimmable for readers, which keeps them engaged.
    - **6. Remember to Optimize Images**
      * Often, image optimization is overlooked in terms of boosting SEO.
      * There are also a ton of benefits to using visual content, including
      * Helps break up a wall of text
      * Aligns with Google’s future goal of visual searching
      * Gives better context to your webpage’s content
      * Allows readers to better comprehend written content
      * Like written content, visual content can leverage better SEO. Image optimization helps your content rank higher on Google Image Search while also improving your site’s visibility.
      * One of the best ways to improve your SEO is by optimizing your alt text. The best way to tackle writing alt text is not to overthink it too much. Alternatively, you should consider the content of the image. Writing good alt text is essentially being able to describe what the image shows
      * When using alt text, follow these rules:
        + Descriptive: All alt text should describe the image in as much detail as you can. Information should describe the image itself and provide context as to how it relates to the content. When you add specific details regarding the image, this will help the image rank in Google Image Search.
        + ● Distinctive: One important rule to follow is that you should not use your content keywords as the alt tag for every image on the page. Instead, the alt text should describe the contents of the image and how it relates to the content; however, it should not repeat the contents of the page overall.
        + ● Relevant: Just like keywords, the alt text should be used sparingly. This means that you should only use alt tags to explain the image and its relation to the content.
    - **7. Make Sure Your Site Is Mobile-Friendly**
      * Recent studies indicate that over 60% of Google searches come from mobile devices. For this reason, it’s important to ensure that your website is optimized for them. Google rewards websites that are mobile-optimized.
    - **8. Use Clean URLs and Meta Descriptions**
      * Your URL and meta description are key factors in helping Google understand what your content is about. Although there is no hard and fast rule about the length of either of these to improve your rankings, the goal should be to make them both as concise and clear as possible while including your target keywords.
      * Below are some best practices for keeping your URL and meta description as clean as possible:
        + Use a consistent structure that keeps future posts in mind: Whether you’re organizing specific content to one area of your site or creating similar content down the road, you’ll want to keep your URL and meta description structure somewhat similar for related content.
        + ● Avoid superfluous language: For both, avoid fluffy language. Each should be short and to the point.
        + ● Make them keyword-rich: Again, be sure that your URL and meta description include the keyword or keywords you’re targeting.
    - **Decrease Load Times on Top-Performing Pages**
      * Slow loading times are one of the primary causes of high bounce rates. According to Google, “speed equals revenue,” which essentially means that slow loading times increase the odds that your site visitors are going to leave your website page. For this reason, your SEO ranking is dependent on the speed of your website.