{POWER.CODERS}

# HTML best practice

# Contents

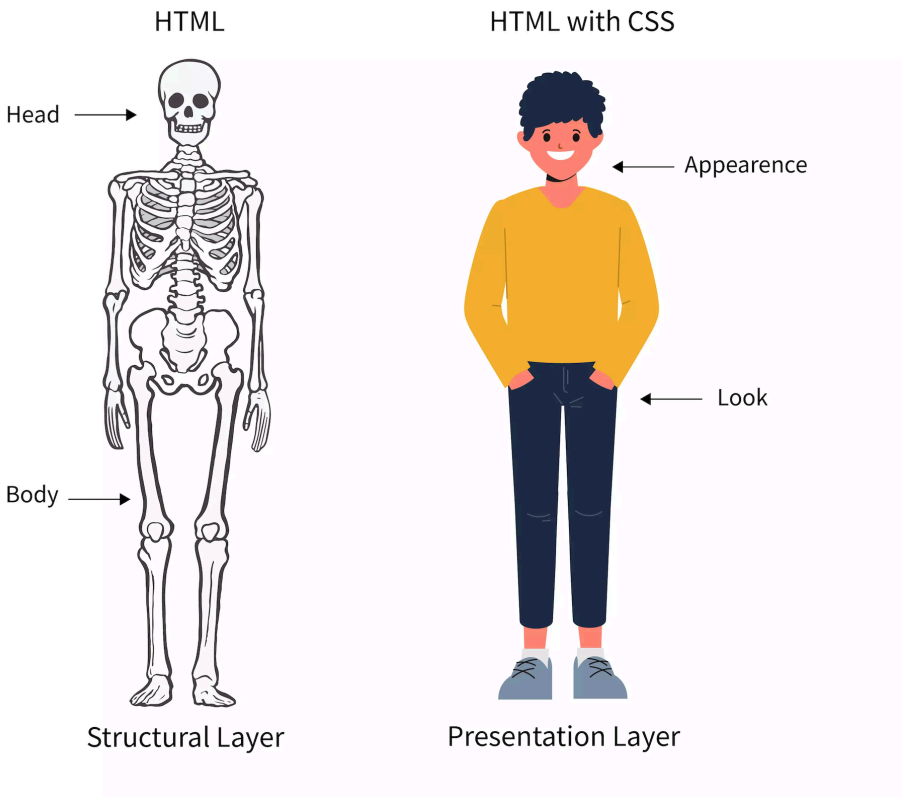
# History and terminology

# HTML was invented in 1991 by Tim Berners-Lee

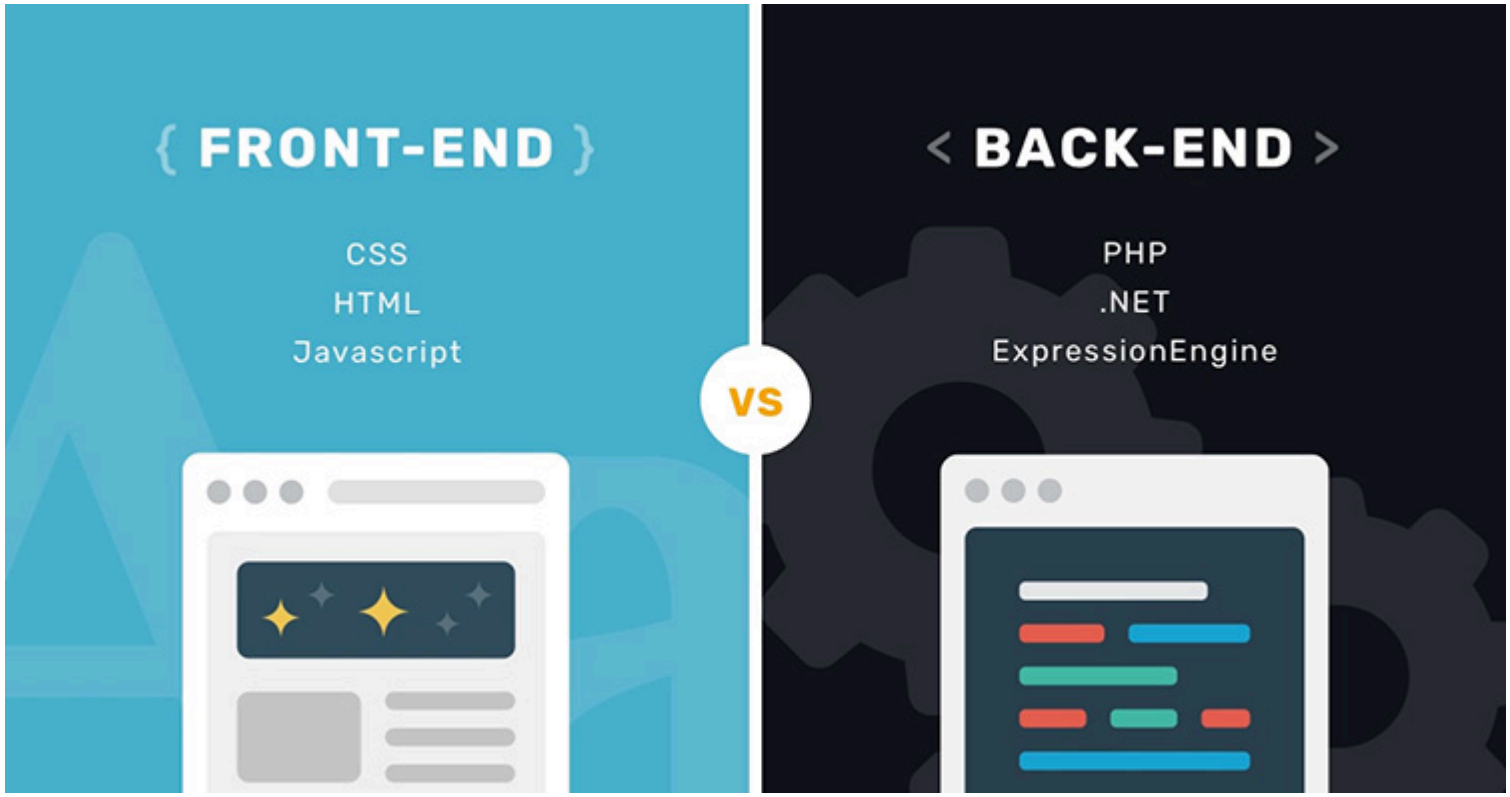
# History of HTML

- HTML is a standardized markup-language that allows us to structure documents to be processed by the browser. The standard contains a set of markups / tags that all browsers recognize.
- HTML started out with 18 different elements, but as it became the official web standard it grew to more than 140 elements.
- The HTML specifications are now maintained and developed by the W3C (World Wide Web Consortium)
- HTML 5 in 2014

# HTML vs CSS


HTML
HTML with CSS
Head
Body
Appearence
Look
Structural Layer
Presentation Layer

# Front-end vs Back-end



# Terms

- **Web design:** The process of planning, structuring and creating a website
- **Web development:** The process of programming dynamic web applications

- **Web development Front-end:** The outwardly visible elements of a website or application
- **Web development Back-end:** The inner workings and functionality of a website or application.

# Quiz

# Quiz

## What is HTML?

- **H**yper**T**ext **M**arkup **L**anguage
- HTML is composed of **tags** that together provide a **blueprint** for a webpage.
- **Hypertext** is a text enriched with hyperlinks.
- **Markup language** uses tags to define the page layout and elements within the page. It is human-readable.

# Quiz

## What are a few different HTML tags?

# Quiz

**Which tag is used to create a link to another page?**

1. `<l>`
2. `<link>`
3. `<a>`
4. `<p>`

# Quiz

**What is a container element compared to a stand alone element?**

## Container Element

- contains other elements (nested elements) or content, e.g. a paragraph `<p></p>` contains text

## Stand Alone Element

- An element that cannot contain anything else, like `<br>` and `<img>`.

# Quiz

**What are the two tags that nest directly within the `<html>` tags?**

- `<head>`
- `<body>`

# Quiz

**What is a HTML comment?**

`<!-- Document Content -->` is only visible in the source code.

Comments can be used to organize your code into sections so you (or someone else) can easily understand your code. It can also be used to 'comment out' large chunks of code to hide it from the browser.

# Quiz

**What is a relative path versus an absolute path?**

If a file is part of the same web site, then a **relative URL** can be used. This can be only the name of the file.

If the file is located on another website, an **absolute URL** must be used. Absolute URLs contain the entire domain name and path.

# Examples

```
<!-- Relative URLs -->
<a href="image-gallery.html">Image Gallery</a>
<a href="blog/first-blog-entry.html">My First Blog Entry</a>
<a href="../image-gallery.html">Back to Image Gallery</a>

<!-- Absolute URLs -->
<a href="https://www.my-colleague.com/blog.html">Blog of a Colleague</a>
```

- Inside the same folder we just use the filename, for example `portrait.jpg`.
- Two dots (`..`) refer to the parent directory.
- If we want to start in the root directory we add an `/` before the path of the file, for example `/portrait.jpg`.

# Quiz

## What does a complete link (anchor) element look like?

```
<a href="https://google.com/" target="_blank">This goes to google</a>
```

# Quiz

## What does block-level and inline elements mean?

### Block-level elements

- start in a new line
- take up the full width of the page
- e.g. `<h1-6>`, `<p>`, `<br>`

### Inline elements

- do not start in a new line
- only take up the necessary width
- e.g. `<a>`, `<em>`, `<strong>`

# Quiz

**What is an attribute? Explain and list some examples.**

- An attribute provides additional information about the HTML element
- It is placed inside an opening tag, before the right ankle bracket
- Examples: `class`, `id`, `style`, `src`, `href`, ...

# Quiz

**What is a HTML entity?**

- **special** characters: like accent marks and German umlaut, e.g. `ü`
- **invisible** characters: like non-breaking spaces, e.g. ` `
- **reserved** characters: which would be interpreted as HTML code, e.g. `<`

# Often used entities

- non-breakable space = ` `
- – = `&ndash;`
- — = `&mdash;`
- © = `&copy;`
- ¼ = `&frac14;`
- ½ = `&frac12;`
- ¾ = `&frac34;`
- « = `&laquo;`
- » = `&raquo;`
- ä = `&auml;`
- Ü = `&Uuml;`
- é = `&eacute;`
- è = `&egrave;`
- ← = `&leftarrow;`
- ↑ = `&uparrow;`
- → = `&rightarrow;`
- ↓ = `&downarrow;`

More entities

# Project best practice

# Defintion of best practice

A **method** or **technique** that has been **generally accepted as superior** to any alternatives because it produces results that are superior to those achieved by other means or because it has become **a standard way of doing things**.

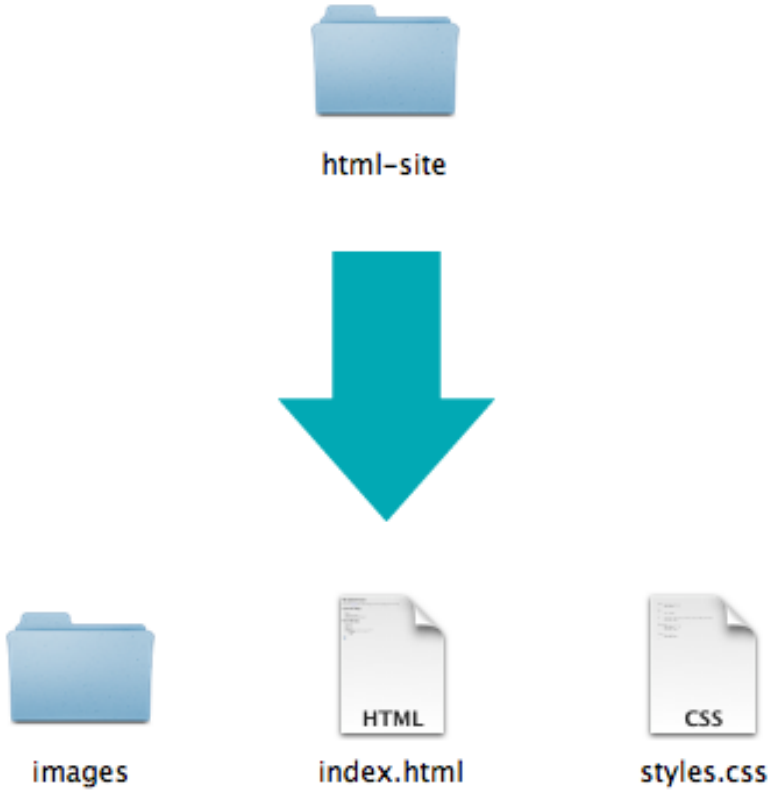
# Folder Structure

All the files for your site should be stored within the same folder.

This includes:

- HTML Files
- CSS Files
- Images
- Script files
- Anything else that will appear on your site

Note: File names should not include spaces or special characters. File names ARE case sensitive.



# Naming Folders + Files

- Name your file `index.html`
- In file and folder names, only use lowercase letters, numbers, hyphens/dashes.
- File names are usually case sensitive: `INDEX.html` vs. `index.html`
- Use the right extension: `.html` vs `.css` vs `.js`

# Tips + Shortcuts

After each opening tag, the next element should be indented with a `tab` for better overview. Make sure you follow this habit.

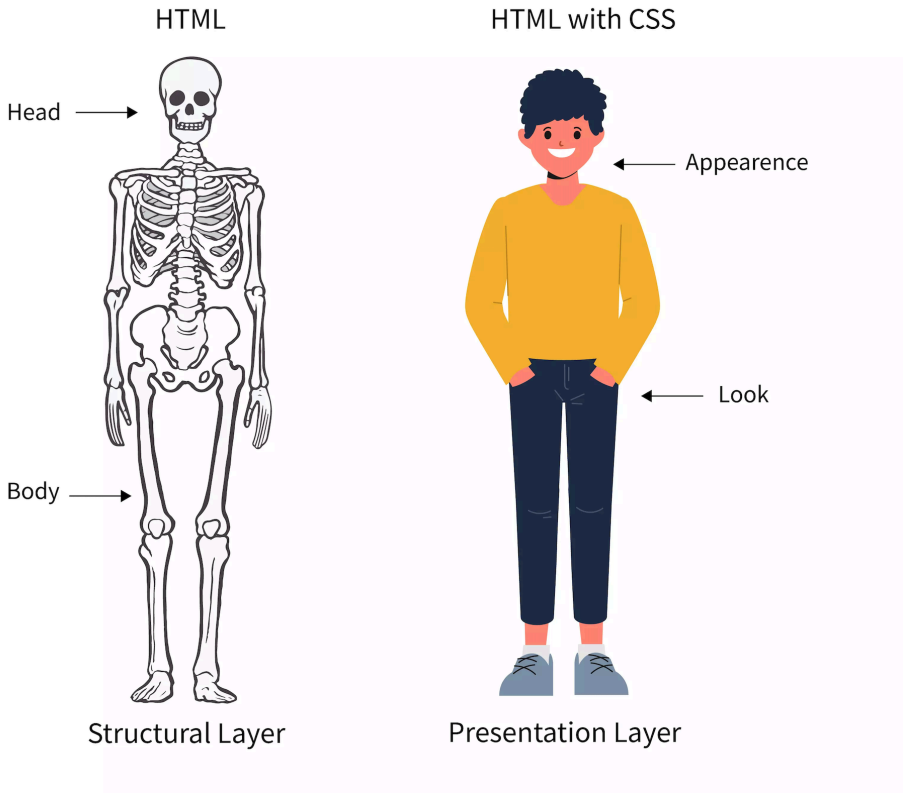
| Action | Windows | Mac |
| --- | --- | --- |
| VSCode, save | `Ctrl` + `s` | `⌘` + `s` |
| VSCode, undo | `Ctrl` + `z` | `⌘` + `z` |
| Switch apps | `Alt` + `Tab` | `⌘` + `Tab` |
| Chrome, reload | `Ctrl` + `r` | `⌘` + `r` |

# A WELL DEVELOPED WEBSITE

- has **valid** code
- is **fast** and **performant**
- is **searchable** by humans and bots
- is **accessible** for disabled people

# Anatomy of a Website

**Your Content**
**+ HTML**: Structure
**+ CSS**: Presentation
**= Your Website**



# SEMANTICS

> I've got nearly 15 years under my belt as web developer and I still question my use of HTML elements.
>
> —**Stephanie Eckles**

# Semantics

# Semantic Markup

**Definition semantics:**

The correct interpretation of the meaning of a word or sentence.

**Definition semantic web:**

Many HTML tags have semantic meaning. That is, the element itself conveys some information about the type of content contained between the opening and closing tags.

# An example

**The something thing jumps over the something thing.**

What do you see in your mind?

Every time you use a `div` or `span`, you tell the browser: **"There's a thing"**.

Non-graphical browsers, screen readers and bots do not see how the website looks like. They need the tags to understand the meaning of the content.

# Now imagine

**The quick brown fox jumps over the lazy dog.**


= English-language pangram (contains all letters of the English alphabet)

# Semantic markup

- requires that HTML elements are used according to their **intended purpose**.
- requires the **separation of content and presentation**.

# Why be semantic?

- more searchable content
- better search engine ranking
- less clutter of meaningless HTML
- less errors in source code and more performant
- better accessible for assistive technologies, like screen readers

# Better searchable?

**Search Engine Optimization** (SEO) is a huge and important topic in web development. There are specialists in SEO asking for a lot of money to optimize existing websites.

Semantic HTML and valid code is the base for good SEO. It makes all other SEO measures so much easier.

# Better accessible?

The goal of **Accessibility** is that a person with a disability can

- **acquire** the same information
- **engage** in the same interactions
- **enjoy** the same services

as a person without a disability.

# Best practice

Use `span` and `div` if you need HTML tags **only for presentation** purposes.

Try to avoid them in general. Always ask yourself, **"is there a tag with better semantic meaning?"**

# Quick word to history

When you google **semantic web**, you will find the term often in relation with **HTML 5**.

A lot of new semantic tags were introduced in 2014 with HTML 5, but HTML was from the beginning a **language with semantic tags and meaning**.

# Semantic HTML

**Always use the tags best describing the content:**

- `<h1>` for headlines
- `<h2-6>` for sub headlines
- `<ul>` for unordered lists
- `<ol>` for ordered lists
- `<table>` for complex data

# Semantic Block-level tags

- `<header>` for introduction of the page, e.g. logo, page headline, content summary
- `<nav>` as container for navigational elements, links to access content within the website
- `<main>` for the main and most important content of a page
- `<section>` for thematic grouping of content, e.g. news, team
- `<article>` is independent, self-contained content, e.g. news entry, blog entry
- `<aside>` for less important content, e.g. in a sidebar or overlay
- `<footer>` for information at the end of document, e.g. copyright, address, author, legal info



Full list on MDN

# Headlines

`<h>`-tags can be within different block-level elements, e.g. header, main, article, section, footer.
Use **hierarchical order** of headlines, e.g.:

- `<h1>` in the `<header>`
- `<h2>` in the `<section>`
- `<h3>` in the `<article>`

Each article should contain at least 1 headline.
Each page should only have 1 `<h1>`-tag.

**Don't leave a number out, e.g. never write first `<h1>`, then `<h3>`.**

# Section in articles

You can group articles in a section, e.g. news items within the section **news**.

You can also have several sections within one article, if needed.

Can you think of an example?

# Boilerplate

# What is a boilerplate?

The word "**boilerplate**" means standardized pieces of text for use as clauses in contracts or as part of a computer program.

An HTML boilerplate will contain the most common elements of a page as a sample that can be cloned and used as a starting point for a project.

# HTML BOILERPLATE

Let's code our own boilerplate.
What should it contain?

# One possibility

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Boilerplate</title>
  </head>
  <body>
    <header>
      <h1></h1>
      <nav>
      </nav>
    </header>
    <main>
      <p></p>
    </main>
```

# Examples of HTML boilerplates

- htmlshell.com
- html5boilerplate.com
- sitepoint.com

Check them out. Most of them are more extensive than our example. Keep in mind: start small and add more if you need it.

# Resources and online material

- Handout: HTML reference sheet
- When should alt tags be blank?
- HTML outliner
- W3C validator
- Free accessibility course by Google

# Resources and online material

- Why you should choose `article` over `section`
- HTML 5 Doctor
- Semantics in HTML 5
- An introduction to the semantic web
- The ADA checklist 2019

# Appendix: More about domains

# Domain

Anatomy of domain names:
subdomain.domain.topleveldomain

- powercoders.org
- www.gmail.com
- calendar.google.com

# TOPLEVELDOMAIN (TLD)

The most common TLDs are:

- `.com`: "Commercial entities"
- `.net`: "Networking companies"
- `.org`: "Non-profits"

As the use of these TLDs isn't restricted, they are used more flexibly than as originally intended.

# Country-code TLDs

Each country has its own TLD (ccTLD), and they can decide who can register for domains with that TLD. Some also specify a set of "second-level-domain" for the TLDs.

- `.ch`
- `.fr`
- `.it`
- `.de`
- `.uk` / `.co.uk`
- ...

# New gTLDs Program

Since 2013 new generic TLDs were introduced.
Over a thousand new gTLDs:

- thematic: `.hotel` `.bank`
- geographic: `.florida` `.berlin`
- product-related: `.toys` `.yoga`
- specific: `.tech` `.design`
- individual: `.codes` `.ninja`
- ...

# What TLD should you get?

Most users expect to type in a `.com` or `.ch` domain, so you should always try to reserve that name (though it's often taken).

If you're worried about competitors, you should purchase related TLDs (.info, .net, .biz).

If you target a particular geographic market, it makes sense to get the ccTLD.

For your own portfolio one of new gTLDs might be best, e.g. `.codes` or `.dev`.

# Domain names

The domain name must follow these rules:

- No spaces or underscores
- Dashes and numbers are OK

The domain name is not case-sensitive - google.com = GOOGLE.com.

# How to choose a domain name?

A good domain name follows these guidelines:

- Represents your name or business name
- Short and memorable
- Easy to say aloud
- Easy to spell without mistakes
- Uses ASCII characters
- Doesn't infringe other's copyright

But good domain names are often taken. You can try using hyphens (experts-exchange.com) or making a TLD form part of the name (del.icio.us) to make it more likely the name will be available.

# Subdomain

Once you own a domain name, you can make subdomains for different aspects of your product/company, e.g.

- www.twitter.com
- search.twitter.com
- api.twitter.com

The www-Subdomain is the most common. If you type in a domain, e.g. google.com, it automatically refers to the www-subdomain. Try it yourself.

# Domain registrars

You must use a "domain registrar" to purchase a domain.
Just google "register domain" to find a registrar.

Worldwide the most popular domain registrar is GoDaddy.

I register my domains with Metanet or Infomaniak.

# Interactive HTML

# Interactive Web

One key success factor of any website and web app is **user engagement**.
The more a user engages with a website and the more he interacts, the more interested he is.

# Forms

# HTML FORMS

HTML forms are used to acquire user input. These types of interaction include

- filling out a contact form / entering personal information
- signing up and logging into websites
- filtering content (by using checkboxes or dropdowns)
- performing a search
- uploading files

# The 5 components of forms

- Input fields
- Field labels
- Structure
- Action buttons
- Feedback

# Input fields

# Form controls

HTML provides different interactive form controls:

- `<input>` for single-line text, radio buttons and checkboxes
- `<textarea>` for multi-line text
- `<select>` for dropdowns

# `<form>`

For form controls to work they need to be nested inside of a form-tag `<form></form>`. Two attributes are required:

- `action` contains an address that defines where the form information will be sent
- `method` can be either GET or POST and defines how the form information will be sent

Optional attributes are:

- `novalidate` disables the native browser form validation
- `autocomplete`
- `name`
- `enctype` is usually only added when files need to be uploaded

# `<input>`

There are different types of input fields, depending on semantics

- `<input type="password">` for passwords
- `<input type="number">` for numbers
- `<input type="email">` for email address
- `<input type="url">` for url
- `<input type="tel">` for number
- `<input type="search">` for submitting a search keyword

# Native date pickers

Even more input fields

```
> <input type="time">
> <input type="date">
> <input type="week">
> <input type="month">
> <input type="datetime-local">
```

# And more...

```
> <input type="range">
> <input type="color">
> <input type="file">
> <input type="hidden">
> <input type="image">
```

# What changes?

Depending on the attribute "**type**", browsers display different UI and use different validation.

Let's try it out.

# Most common attributes

# Field labels

# <label>

The attribute `for` corresponds with the `id` of the input field.

```
<form action="do_something.html" method="post">
    <label for="fname">First name:</label>
    <input type="text" id="fname" name="fname" required>
</form>
```

# STRUCTURE

# Structure

This includes the order of fields, the form’s appearance on the page, and the logical connections between different fields.

# <fieldset>

Long forms with a lot of input fields can become unreadable and very user-unfriendly.

The `<fieldset>` tag allows you to group form fields that share the same purpose.

Label the group with a nested `<legend>` tag.

Best practice: Only use the minimum of mandatory form fields. The higher the number of information to fill in, the more likely it is that users stop filling out the form.

# Example

```
<form action="do_something.html" method="post">
  <fieldset id="personal-info">
    <legend>Personal Info</legend>
    <label for="fname">First name:</label>
    <input type="text" id="fname" name="fname" required>
  </fieldset>
</form>
```

# Action buttons

# Action buttons

The form will have at least one call to action (the button that triggers data submission).

- `<input type="button">` for creating a button without default action
- `<input type="submit">` for submitting a form
- `<input type="reset">` for resetting a form

# FEEDBACK

# Feedback

It is very important to give users feedback in their form entry as well as submission.

- **Instant feedback:** Give feedback ASAP, e.g. for field validation when you leave the field.
- **Positive feedback:** The field was validated, the form successfully submitted
- **Negative feedback:** The field could not be validated, the form not submitted. Make sure that the feedback is useful and specific. Not "There was an error", but "The number you've provided is incorrect, it has to be between 0 and 100.".

Always validate your forms on both client and server.

# Form best practice: Usability

- Offer automatic field focus
- Explain why you need sensitive data
- Minimize the total number of fields
- Clearly distinguish optional from mandatory fields
- Size fields accordingly
- Provide "show password" option

# Form best practice: Accessibillity

- Put tab indices on your inputs: `tabindex="0"`
- Use patterns where appropriate:
  `pattern="^#?([a-fA-F0-9]{6}|[a-fA-F0-9]{3})$"`
  (hex-color)
- Use placeholders
  `placeholder="e.g. +41 78 123 45 67"`
- Write clear and concise labels
- Don't use placeholder text as labels
- Don't slice fields (e.g. birthday)

# Form best practice: Mobile

- Provide matching keyboard: `e.g. type="tel"`
- Avoid dropdown menus
- Top-align labels
- Make sure your font-size for the form is min. 16px
- Make your buttons finger-friendly

# Links


Hyperlinks are the main tool for interacting and engaging. You can and should use them:

- within a navigation
- in link lists
- in call-to-actions
- inline

Make sure to use keywords in your link. Instad of *Read more* use *Read more about Powercoders*.

# Anchors

Next to relative and absolute paths it is also possible to use an `<a>`-tag as **anchors**.

**Anchors** are specific elements you can define by giving them an id.

You can then link within a page not to the top, but to a specific **element (anchor).**

# EXAMPLE ANCHOR

```
<a href="#subtitle2">This goes to the element with the id subtitle2</a>

<h2 id="subtitle2">This is my second subtitle</h2>

<a href="/blog/#subtitle2">
  This goes to the element with the id subtitle2 on the page "blog"
</a>
```

# Links vs Buttons

- **Links** change the URL, e.g. going to another document, move down the page to an anchor, etc.
- **Buttons** perform an action, e.g. show/hide content, submit forms, etc.

Use it semantically correct, don't surprise the user, you can style links as buttons and vice verse, as long as the user always knows what to expect.

# Media

# What is a media element?

- `<audio>` to embed sound content in a website
- `<video>` to embed video content in a website

These interactive tags work with a **Javascript API** modern browsers have natively integrated.

API = Application Programming Interface to allow one application to access features and data from the OS (operating system), another service or application.

# Audio

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
  <p>Your browser doesn't support HTML5 audio.
     Here is a <a href="horse.mp3">link to the audio</a> instead.</p>
</audio>
```

# <audio>

Check all attributes and possibilities on:

- MDN
- w3schools

# <video>

```
<video poster="horse.png" controls>
  <source src="horse.mp4" type="video/mp4">
  <source src="horse.webm" type="video/webm">
  <p>Your browser doesn't support HTML5 video.
  Here is a <a href="horse.mp4">link to the video</a> instead.</p>
</video>
```

# Media formats

- A **WebM** container usually packages Ogg Vorbis audio with VP8/VP9 video. This is supported mainly in **Firefox** and **Chrome**.
- An **MP4** container often packages AAC or MP3 audio with H.264 video. This is supported mainly in **Internet Explorer** and **Safari**.
- The older **Ogg** container tends to go with Ogg Vorbis audio and Ogg Theora video. This was supported mainly in **Firefox** and **Chrome**, but has basically been superseded by the better quality WebM format.

These formats compress the video and audio data into manageable files. Browsers use different codecs to convert the compressed data back.

# Useful links

- Can I use ...
- Miro Video Converter
- Dive into HTML5
- Video playback on the web (Part 1)
- Video playback on the web (Part 2)