{POWER.CODERS}

Interactive HTML

Agenda

Today we will learn about

- forms
- links
- media

HTML RECAP

Web semantics: Always look for the HTML tags which best explain the content they define. It is the foundation for:

HTML RECAP

Web semantics: Always look for the HTML tags which best explain the content they define. It is the foundation for:

- valid code
- performance
- searchable content
- accessible content

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

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

The 5 components of forms

- Input fields

The 5 components of forms

- Input fields
- Field labels

The 5 components of forms

- Input fields
- Field labels
- Structure

The 5 components of forms

- Input fields
- Field labels
- Structure
- Action buttons

The 5 components of forms

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

Input fields

Form controls

HTML provides different interactive form controls:

Form controls

HTML provides different interactive form controls:

- `<input>` for single-line text, radio buttons and checkboxes

Form controls

HTML provides different interactive form controls:

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

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:

`<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

`<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

`<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:

`<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

`<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`

`<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`

`<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>`

There are different types of input fields, depending on semantics

- `<input type="password">` for passwords

`<input>`

There are different types of input fields, depending on semantics

- `<input type="password">` for passwords
- `<input type="number">` for numbers

`<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>`

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>`

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>`

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

Native date pickers

Even more input fields

```
> <input type="time">
```

Native date pickers

Even more input fields

```
> <input type="time">
> <input type="date">
```

Native date pickers

Even more input fields

```
> <input type="time">
> <input type="date">
> <input type="week">
```

Native date pickers

Even more input fields

```
> <input type="time">
> <input type="date">
> <input type="week">
> <input type="month">
```

Native date pickers

Even more input fields

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

Native date pickers

Even more input fields

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

And more...

And more...

```
> <input type="range">
```

And more...

```
> <input type="range">
> <input type="color">
```

And more...

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

And more...

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

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.

What changes?

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

Let's try it out.

Most common attributes

Most common attributes

Most common attributes

Most common attributes

Most common attributes

Most common attributes

Most common attributes

Most common attributes

Field labels

<label>

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

<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.

<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.

<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.

<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).

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

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

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.

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.

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

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.".

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

Form best practice: Usability

- Offer automatic field focus

Form best practice: Usability

- Offer automatic field focus
- Explain why you need sensitive data

Form best practice: Usability

- Offer automatic field focus
- Explain why you need sensitive data
- Minimize the total number of fields

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

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

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

Form best practice: Accessibillity

- Put tab indices on your inputs: `tabindex="0"`

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)

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"`

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

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

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

Form best practice: Mobile

- Provide matching keyboard: `e.g. type="tel"`

Form best practice: Mobile

- Provide matching keyboard: `e.g. type="tel"`
- Avoid dropdown menus

Form best practice: Mobile

- Provide matching keyboard: `e.g. type="tel"`
- Avoid dropdown menus
- Top-align labels

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

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:

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

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

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.

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

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.

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?

What is a media element?

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

What is a media element?

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

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.

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

Media formats

- A **WebM** container usually packages Ogg Vorbis audio with VP8/VP9 video. This is supported mainly in **Firefox** and **Chrome**.

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**.

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.

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)

Group Exercise

1. Create a form with validation and 5 required form fields, each another input type: email, phone, number, color, date
2. Look at the UI and validation of each field in at least 2 different browsers, e.g. Google Chrome and Mozilla Firefox.
3. What do you notice? Describe in a few sentences differences as well as similarities.

Exercise

Create the semantic HTML of this form layout.

Make sure, the code is valid. Is there something you would optimize according to best practice? Add it in the comments what you would change and why.

Use git for regular commits to your github repo.

Exercise / Live Coding

Create the semantic HTML of this form layout.

Make sure, the code is valid.

Use git for regular commits to your github repo.