Forms

It's traditional to send a form to a server where a backend language processes it(PHP). But now all the cool kids process them on the frontend.

Basic Controls:

1. input field – **with name attribute** 'searchInput' **to access info inside it**
2. button

Attributes

1. **!!**name – 'search' **describes the form purpose**
2. action – '/search' **the url that the form will be submitted to for server-side processing**

DOM access to forms

document.forms – **returns HTML collection of all forms in document by order of appearance**

To access form in this collection:
`const form = document.forms[0];`

Using getElements:

document.getElementsByTagname('form')[0]

!!Using name attribute:

document.forms.search **BUT Better to do** `const form = document.forms['search'];`

By Method:

form.elements – **returns HTML collection of all elements currently in form:**
`const [input,button] = form.elements;`
input
button

!!By Form Control name attribute:

form.searchInput **BUT Better to do** `const input = form['searchInput']`

Submitting a form

Automatic Submission:

Methods: form.submit()- submits form but doesn't trigger submit event

form.action = '/an/other.url'; - used to set the action attribute of a form, so it's sent to a different URL to be processed on the server.

Manual Submission:

<input type='submit' value='Submit'>

+

Button Element: **OR** **Input Element W/ image:**

<button type='submit'>Submit</button> <input type='image' src='button.png'>

Resetting a form

Set HTML to initial values

Method: form.reset() – reset form controls to intital values

Using button

<button type='reset'>Reset</button>

Form Events

```
const input = form.elements.searchInput;
```

focus – when a cursor clicks, taps or navigates to the element

```
input.addEventListener('focus', ()=>alert('focused'), false);
```

blur – when cursor moves the focus away from element

```
input.addEventListener('blur', () => alert('blurred'),  false);
```

change – when cursor moves away from element after it's changed

```
input.addEventListener('change', () => alert('changed'), false);
```

submit – sends the contents to the server. JS can intercept the form before it's sent with event listener:

```
form.addEventListener ('submit', search, false);

function search() { alert(' Form Submitted'); }
```

Stop Form submitting itself

preventDefault(): by adding a submit eventlistener to a button, you can use this in a function:

```
function search(event) {
    alert('Form Submitted');
    event.preventDefault(); }
```

Input Values

Showing User what they searched for – input.value

Set the input value - input.value = 'Search Here';

Showing text in the input field

How to hide text value when users click input box – use focus and blur events

When user clicks on field – focus:

```
input.addEventListener('focus', function(){
if (input.value==='Search Here') { input.value = '' }}, false);
```

When user leaves field blank and clicks away– blur:

```
input.addEventListener('blur', function(){
if (input.value===' ') { input.value = 'Search Here ' }}, false);
```

HTML5 input attributes

placeholder – this type of text isn't a value of the input field:

```
<input type='text' name='search-box' placeholder='Search Here'>
```

autofocus – gives focus to element on page load:

```
<input type='text' name='search-box' autofocus>
```

!! You can do the same thing in JS !! - input.focus();

maxlength – limits num of characters entered into field:

```
<input type='text' name='search-box' maxlength="32">
```

Form Controls

input, select, textarea, button

text – default input type, not necessary to use but recommended to explicitly show intention & readability:

```
<input type="text" name="heroName" />
```

password – characters are concealed:

```
<input type="password" name="password" />
```

checkbox – allows multiple options to be checked (true) or left unchecked (false). Give all related checkbox elements the same 'name' property. They are accessible as an HTML Collection- **form.powers** :

```
<input type='checkbox' value='Strength' name='powers'>
<input type='checkbox' value='Flight' name='powers'>
```

!! You can do the same thing in JS !! : hero.powers[0].checked = true;

checked attribute can be initially set:

```
<input type='checkbox' value='Flight' name='powers' checked>
```

checking if each check box is checked – if checked, the value attribute is set to powers:

```
for (let i=0; i < form.powers.length; i++) {
    if (form.powers[i].checked) {
    hero.powers.push(form.powers[i].value);} }
```

radio – allow only one choice for multiple options. Give all related radio elements the same 'name'. They are accessible as an HTML Collection- **form.category**:

```
<input type='radio' name='category' value='Hero'>
<input type='radio' name='category' value='Villain'>
```

!! You can do the same thing in JS !! : form.type[2].checked = true;

set category value: hero.category = form.category.value;

can be initially set: <input type='radio' name='category' value='Villain' checked>

hidden – not displayed, but contain value submitted with form. Used to send info user already provided. **DON'T USE FOR SENSITIVE DATA SINCE VISIBLE IN HTML.**

file – used to upload files, provides button for users to select from file system.

number, tel, color, date, email – these fields validate automatically; functionality same ↑

select drop-down list - allow selection of one or more options from list of values.Add 'multiple' if more than one option selected.Add 'selected' to set initial value:

```
<select name='city' id='city'>
<option value='' selected>Choose a City</option>
<option value='Metropolis'>Metropolis</option></select>
```

find what the user selected - city.selectedIndex

use this to access text in selected option - city.options[city.selectedIndex].text

text areas – allow longer pieces of text to be entered. Same as input field. Initial value text can be placed between tags. rows and col can be set:

```
<textarea name='origin' rows='20' cols='60'>Born as  Kal-El on the planet Krypton...</textarea>
```

buttons – the default type is 'submit'. another type is 'reset'. also there is type 'button'.

Validation

Types of validation

A required field is completed	An email address is valid
A number is entered when numerical data is required	A password is at least a minimum number of characters

required – add 'required' to field that you want required

custom JS validation – Validates first letter in input

```
form.addEventListener('submit',validate,false);
function validate(event) {
    const firstLetter = form.heroName.value[0];
    if (firstLetter.toUpperCase() === 'X') {
        event.preventDefault();
        alert('Your name is not allowed to start with X!');} }
```

instant feedback – add an eventlistener to input when key is pressed 'keyup'. Insert feedback in label with an 'error' class

disable submit button – add 'disabled' to element:

```
function disableSubmit(event) {
    if(event.target.value === ''){
        document.getElementById('submit').disabled = true; } else {
        document.getElementById('submit').disabled = false; }}
```