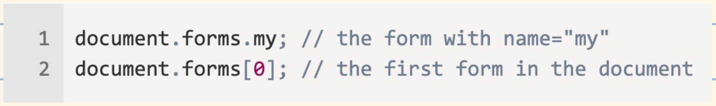
SUBJECT: Form Properties & Methods DATE:

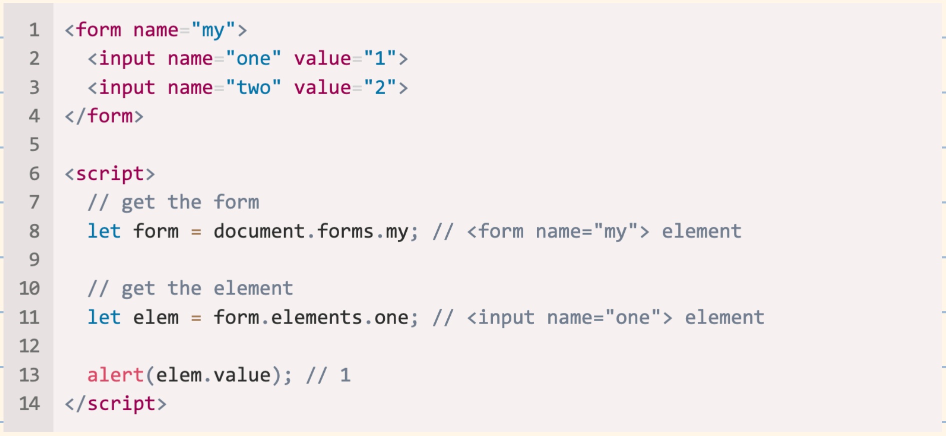
- Forms are members of document.forms

- Able to access a form using either its name or order in the form collection



```
document.forms.my; // the form with name="my"
document.forms[0]; // the first form in the document
```

Example - access using form name & form's input name



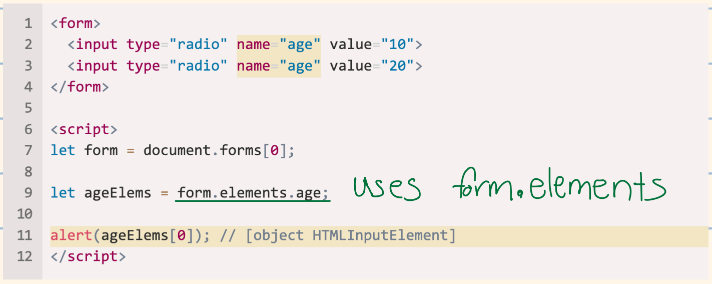
```
<form name="my">
  <input name="one" value="1">
  <input name="two" value="2">
</form>

<script>
  // get the form
  let form = document.forms.my; // <form name="my"> element

  // get the element
  let elem = form.elements.one; // <input name="one"> element

  alert(elem.value); // 1
</script>
```

- Multiple elements can have the same name



```
<form>
  <input type="radio" name="age" value="10">
  <input type="radio" name="age" value="20">
</form>

<script>
let form = document.forms[0];

let ageElems = form.elements.age;

alert(ageElems[0]); // [object HTMLInputElement]
</script>
```

uses form.elements

SUBJECT: Form Properties & Methods DATE:

- Elements w/the same name (continued)

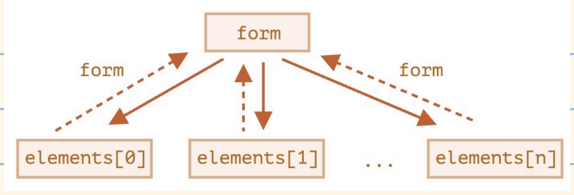
  ⟶ makes form.elements[name] a collection
  ⟶ typical for radio buttons & checkboxes

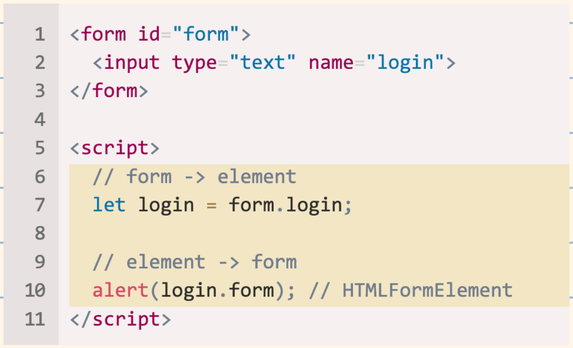
- Can also access elements as form.name (or index)

  ⟶ Shorter than form.element.name notation
  ⟶ <u>be careful!</u> if an element is accessed this way, but then its name gets changed, it is made available under the old & new name

Backreference: element.form

- For any element, the form is available as element.form





```
<form id="form">
  <input type="text" name="login">
</form>

<script>
  // form -> element
  let login = form.login;

  // element -> form
  alert(login.form); // HTMLFormElement
</script>
```

SUBJECT: Form Elements DATE:

input and textarea

- values can be accessed as input.value (string)
  or
  input.checked
  (for checkboxes & radio buttons)

- <textarea>... </textarea> values cannot be accessed using textarea.innerhtml

  → it only stores the initial html — not the current value

Select and option

- The 3 important select properties:

  1. Select.options → the collection of <option> subelements
  2. Select.value → the value of the currently selected option
  3. Select.selectedIndex → the number of the currently selected option

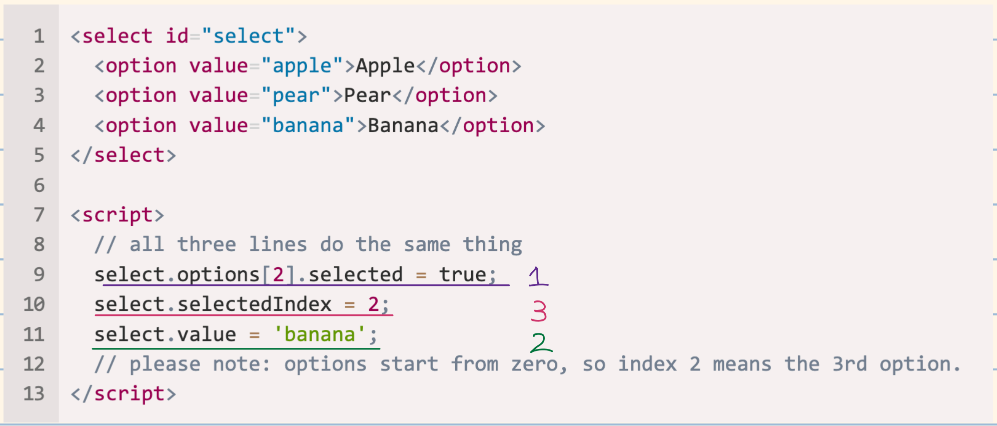
SUBJECT: Form Elements DATE:

- Each of the 3 select options provides a way to set a value for <select>

  1. Find the corresponding option element among select.options
     → set option.selected to true

  2. Set select.value to a known new value

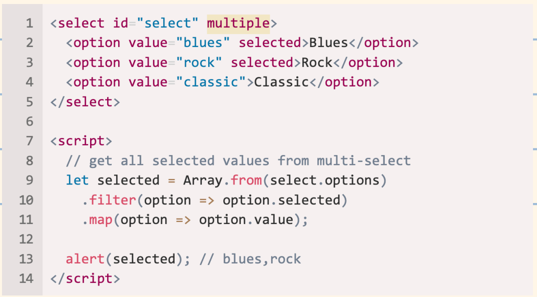
  3. Set select.selectedIndex to a known new option number



```
<select id="select">
  <option value="apple">Apple</option>
  <option value="pear">Pear</option>
  <option value="banana">Banana</option>
</select>

<script>
  // all three lines do the same thing
  select.options[2].selected = true;   1
  select.selectedIndex = 2;            3
  select.value = 'banana';             2
  // please note: options start from zero, so index 2 means the 3rd option.
</script>
```

- Select allows for multiple options to be selected using the multiple attribute (is rarely used)



```
<select id="select" multiple>
  <option value="blues" selected>Blues</option>
  <option value="rock" selected>Rock</option>
  <option value="classic">Classic</option>
</select>

<script>
  // get all selected values from multi-select
  let selected = Array.from(select.options)
    .filter(option => option.selected)
    .map(option => option.value);

  alert(selected); // blues,rock
</script>
```

SUBJECT: Creating an Option Element DATE:

- Can create an option element using document.createElement('option')

- Can also use the new Option syntax:

```
option = new Option(text, value, defaultSelected, selected);
```

⟶ text : the text inside the option

⟶ value : the option value

⟶ defaultSelected : if true, then the selected attribute is created

- Sets the HTML attribute... which can be accessed via option.getAttribute('selected')

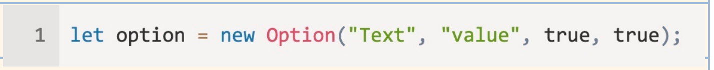
⟶ Selected : sets whether the option is selected or not

✳ Either set both defaultSelected & selected to true/false, or nothing (they will both be set to false by default)

SUBJECT: Creating an Option Element DATE:

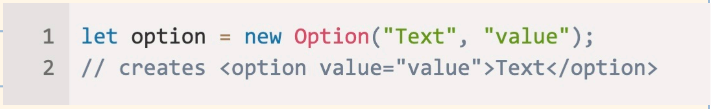
creating the same option

with selected + defaultSelected



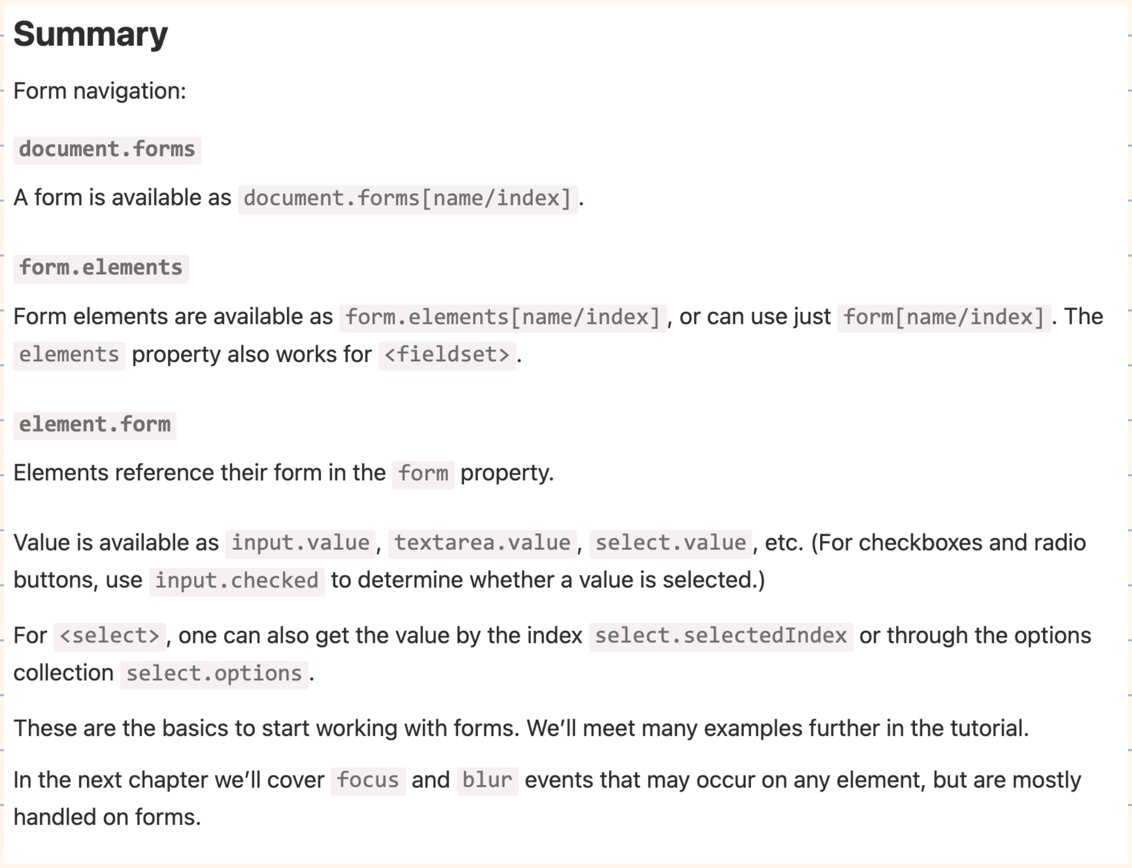
```javascript
let option = new Option("Text", "value", true, true);
```

without selected + defaultSelected



```javascript
let option = new Option("Text", "value");
// creates <option value="value">Text</option>
```

Option element properties ⟶ option.selected
option.index
option.text



Summary

Form navigation:

`document.forms`

A form is available as `document.forms[name/index]`.

`form.elements`

Form elements are available as `form.elements[name/index]`, or can use just `form[name/index]`. The `elements` property also works for `<fieldset>`.

`element.form`

Elements reference their form in the `form` property.

Value is available as `input.value`, `textarea.value`, `select.value`, etc. (For checkboxes and radio buttons, use `input.checked` to determine whether a value is selected.)

For `<select>`, one can also get the value by the index `select.selectedIndex` or through the options collection `select.options`.

These are the basics to start working with forms. We'll meet many examples further in the tutorial.

In the next chapter we'll cover `focus` and `blur` events that may occur on any element, but are mostly handled on forms.