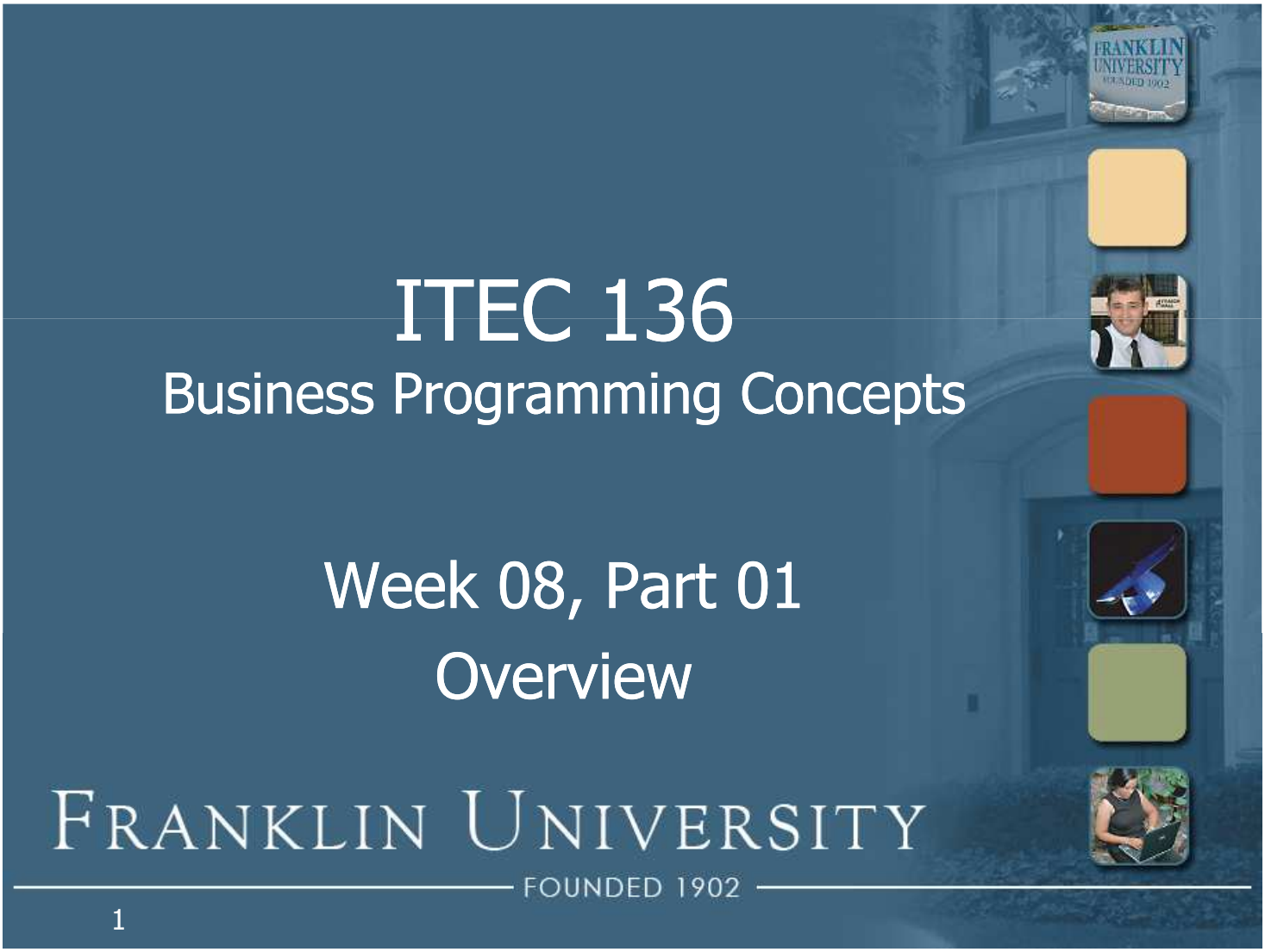


ITEC 136

Business Programming Concepts

Week 08, Part 01

Overview

FRANKLIN UNIVERSITY

FOUNDED 1902

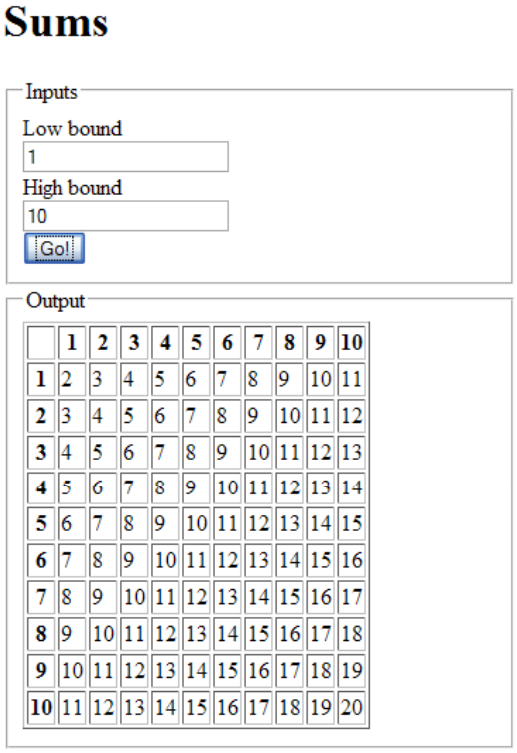
Week 7 Review

- Sentinel controlled loops
- Results controlled loops
- Flag controlled loops
- `break` and `continue` keywords
- Nested loops
  - Loop variable in outer loop becomes part of the bounds in the inner loop

FRANKLIN UNIVERSITY

Week 7 Review

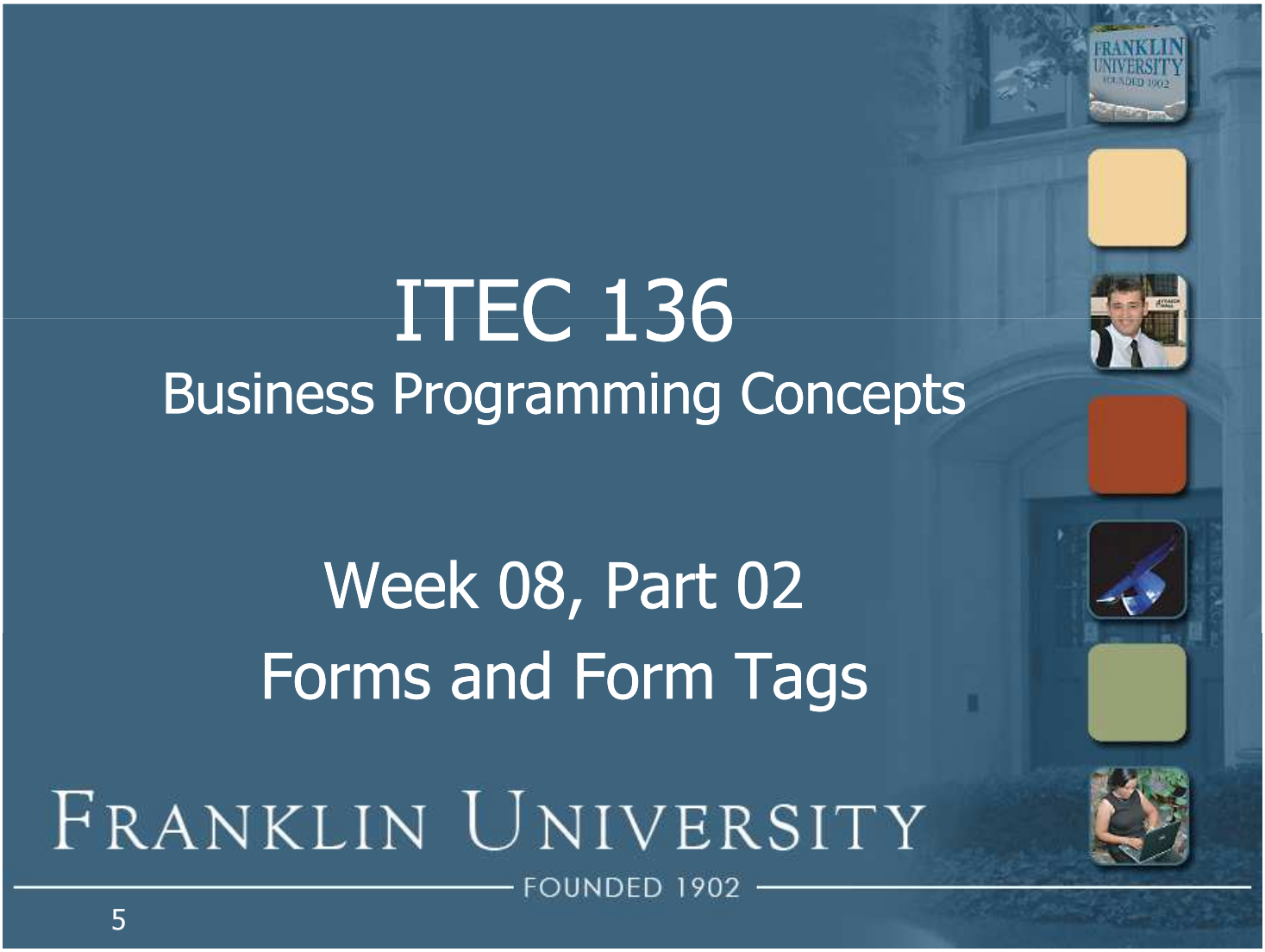
- Problem:
  - Print a sums table like the following. You should input the left and right bounds and validate them.



	1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10	11
2	3	4	5	6	7	8	9	10	11	12
3	4	5	6	7	8	9	10	11	12	13
4	5	6	7	8	9	10	11	12	13	14
5	6	7	8	9	10	11	12	13	14	15
6	7	8	9	10	11	12	13	14	15	16
7	8	9	10	11	12	13	14	15	16	17
8	9	10	11	12	13	14	15	16	17	18
9	10	11	12	13	14	15	16	17	18	19
10	11	12	13	14	15	16	17	18	19	20

Week 8 Overview

- Outcomes
  - Create and validate forms.
  - Describe and use form-based events.



Forms and Form Tags

- The `<form>` tag
  - Groups a set of input fields into a bundle that can be submitted to a server.

```
<body>
  <form id="myForm" name="myForm">
    <!-- form child elements go in here -->
  </form>
</body>
```

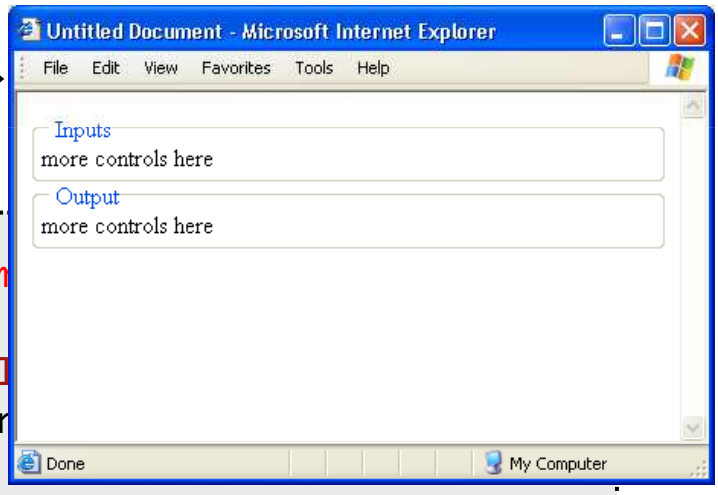
Forms and Form Tags

- The `<fieldset>` tag
  - *Visually* groups form elements together

```
<form id="myForm" name="myForm">
  <fieldset>
    <legend>Inputs</legend>
    more controls here
    </fieldset>
  <fieldset>
    <legend>Output</legend>
    more controls here
  </fieldset>
</form>
```


Forms and Form Tags

- The `<fieldset>`
  - *Visually* groups

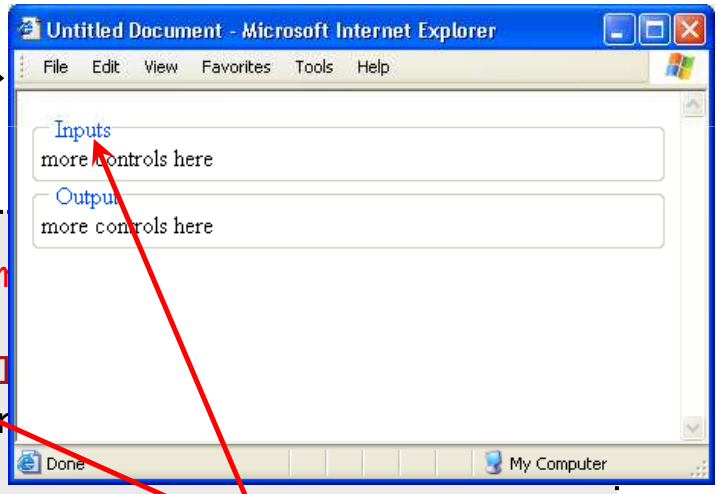



```
<form id="myForm" nam
  <fieldset>
    <legend>Inputs</l
    more controls her
    </fieldset>
  <fieldset>
    <legend>Output</legend>
    more controls here
  </fieldset>
</form>
```

Forms and Form Tags

- The `<fieldset>`
  - *Visually* groups

```
<form id="myForm" nam
  <fieldset>
    <legend>Inputs</l
    more controls her
    </fieldset>
  <fieldset>
    <legend>Output</legend>
    more controls here
  </fieldset>
</form>
```



"legend" labels the entire fieldset and everything in it.

Forms and Form Tags

- The `<label>` and `<input>` tags
  - `<label>` associates text with an input element (clicking the text focuses the input element).
  - `<input>` has attributes that identify the type of input (text, submit, button, radio, file, etc.)

Forms and Form Tags

- The `<label>` and `<input>` tags

```
<form id="myForm" name="myForm">
  <fieldset>
    <legend>Inputs</legend>
    <label for="name">Name:</label>
    <input type="text" name="name" id="name" />
    <br />
  </fieldset>
  <fieldset>
    <legend>Output</legend>
    more controls here
  </fieldset>
</form>
```


Forms and Form Tags

- The `<label>` and `<input>` tags

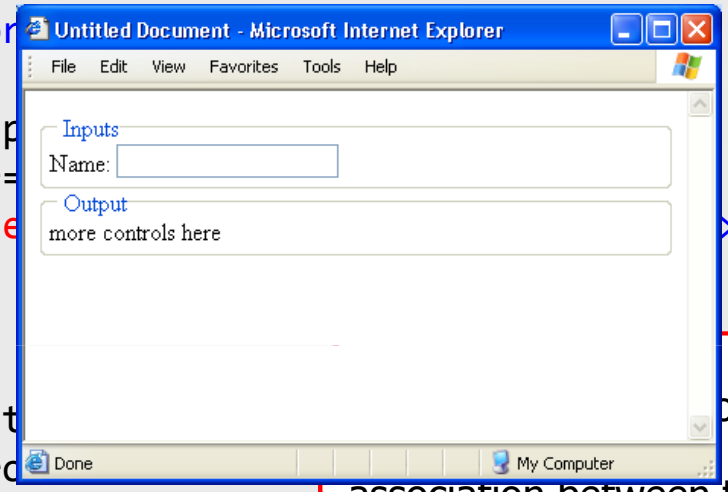
```
<form id="myForm" name="myForm">
  <fieldset>
    <legend>Inputs</legend>
    <label for="name">Name:</label>
    <input type="text" name="name" id="name" />
    <br />
  </fieldset>
  <fieldset>
    <legend>Output</legend>
    more controls here
  </fieldset>
</form>
```

"for" and "id" need to match to make the association between the label and input control

Forms and Form Tags

- The `<label>` and `<input>` tags

```
<form id="myFor
  <fieldset>
    <legend>Inp
    <label for=
    <input type
    <br />
  </fieldset>
  <fieldset>
    <legend>Out
    more contro
  </fieldset>
</form>
```



association between the label and input control

Forms and Form Tags

- The `<textarea>` tag
  - Used for multi-line input or simple console-based output.

Forms and Form Tags

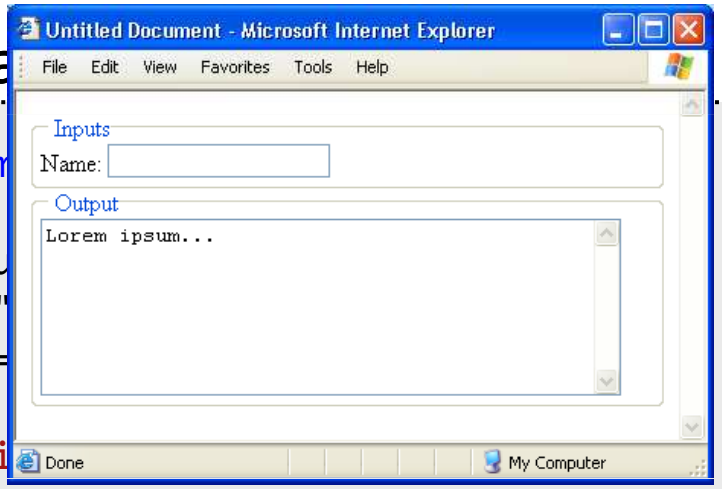
- The `<textarea>` tag

```
<form id="myForm" name="myForm">
  <fieldset>
    <legend>Inputs</legend>
    <label for="name">Name:</label>
    <input type="text" name="name" id="name" />
    <br />
  </fieldset><fieldset>
    <legend>Output</legend>
    <textarea id="output" name="output" rows="7"
      cols="45">Lorem ipsum...</textarea>
  </fieldset>
</form>
```


Forms and Form Tags

- The `<texta`

```
<form id="myForm
  <fieldset>
    <legend>Inpu
    <label for="
    <input type=
    <br />
  </fieldset><fi
    <legend>Output</legend>
    <textarea id="output" name="output" rows="7"
      cols="45">Lorem ipsum...</textarea>
  </fieldset>
</form>
```



Forms and Form Tags

- The `<select>` tag
  - Can also use multiple-selection lists as well with the "multiple" and "size" attributes.
  - Combine with `<option>` to create drop-down lists of elements to select.

Forms and Form Tags

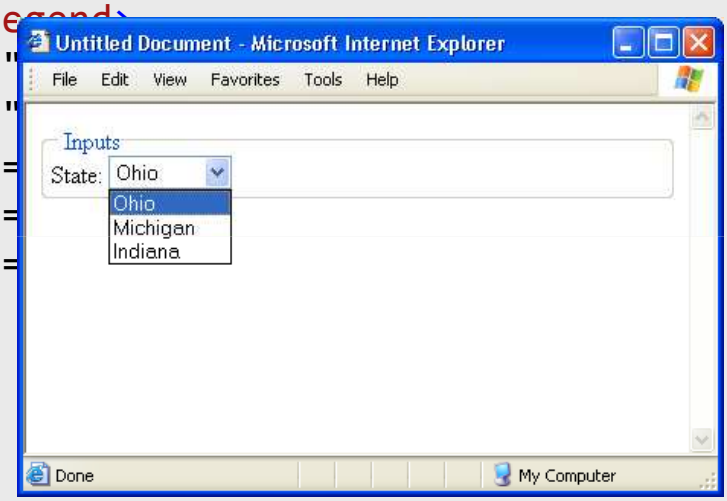
- The `<select>` tag

```
<form id="myForm" name="myForm">
  <fieldset>
    <legend>Inputs</legend>
    <label for="state">State:</label>
    <select id="state">
        <option value="OH">Ohio</option>
        <option value="MI">Michigan</option>
        <option value="IN">Indiana</option>
    </select>
    <br />
  </fieldset>
</form>
```

Forms and Form Tags

- The `<select>` tag

```
<form id="myForm" name="myForm">
  <fieldset>
    <legend>Inputs</legend>
    <label for="state"
    <select id="state"
        <option value=
        <option value=
        <option value=
    </select>
    <br />
  </fieldset>
</form>
```



Forms and Form Tags

- Radio buttons
  - Used to select 1 from many
  - Use the <input> tag with attribute "type" set to "radio"
  - Buttons are grouped based on the "name" attribute

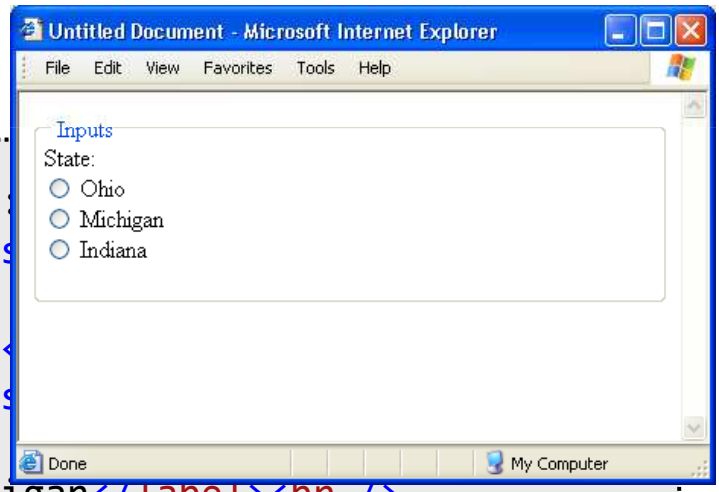
Forms and Form Tags

- Radio buttons

```
<label for="state">State:</label><br />
<input type="radio" id="state1" name="state"
     value="OH" />
<label for="state1">Ohio</label><br />
<input type="radio" id="state2" name="state"
     value="MI" />
<label for="state2">Michigan</label><br />
<input type="radio" id="state3" name="state"
     value="IN" />
<label for="state3">Indiana</label><br />
```


Forms and Form Tags

- Radio buttons




```
<label for="state">State:
<input type="radio" id="s
     value="OH" />
<label for="state1">Ohio<
<input type="radio" id="s
     value="MI" />
<label for="state2">Michigan</label><br />
<input type="radio" id="state3" name="state"
     value="IN" />
<label for="state3">Indiana</label><br />
```

Forms and Form Tags

- Checkboxes
  - Used to select many from many
  - Use the <input> tag with attribute "type" set to "checkbox"
  - Buttons are grouped based on the "name" attribute

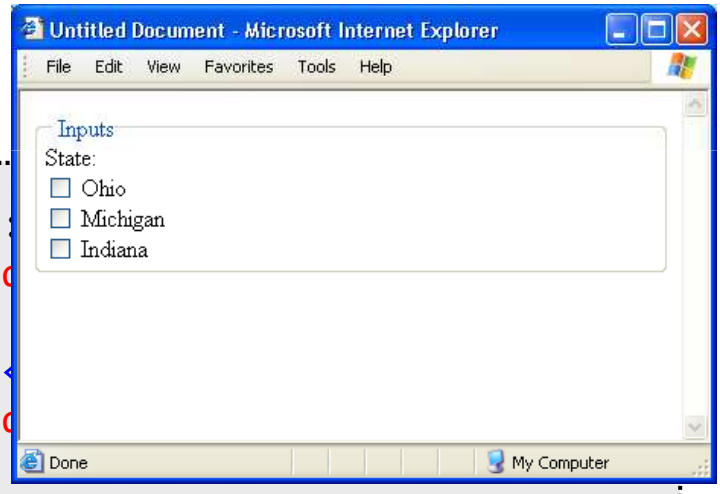
Forms and Form Tags

- Checkboxes

```
<label for="state">State:</label><br />
<input type="checkbox" id="state1" name="state"
    value="OH" />
<label for="state1">Ohio</label><br />
<input type="checkbox" id="state2" name="state"
    value="MI" />
<label for="state2">Michigan</label><br />
<input type="checkbox" id="state3" name="state"
    value="IN" />
<label for="state3">Indiana</label><br />
```

Forms and Form Tags

- Checkboxes




```
<label for="state">State
<input type="checkbox" id
    value="OH" />
<label for="state1">Ohio<
<input type="checkbox" id
    value="MI" />
<label for="state2">Michigan</label><br />
<input type="checkbox" id="state3" name="state"
    value="IN" />
<label for="state3">Indiana</label><br />
```


ITEC 136

Business Programming Concepts

Week 08, Part 03

Form Processing

FRANKLIN UNIVERSITY
FOUNDED 1902

Form Processing

- Responding to events
  - Can respond to many kinds of events, such as clicks, focus, blur, change, keypress, etc.
  - Use the "onXXX" attributes within the form element tags to execute code in response to an event (i.e. `onclick="doSomething()"`)

Form Processing

- Reading an input field's value
  - Use `document.getElementById()` to get the field, then read the value, then convert the value into some useful form

Form Processing

- Reading an input field's value

```
function getValueFromField(fieldId, conversionFunction)
{
    var element = document.getElementById(fieldId);
    if (element == null) {
        return null;
    }
    if (conversionFunction != undefined) {
        return conversionFunction(element.value);
    }
    return element.value;
}
```


Form Processing

- Reading an input field's value

```
function getValueFromField(fieldId, conversionFunction)
{
    var elemen var myInteger = getValueFromField(
    if (elemen   "field1", parseInt);
        return var myFloat = getValueFromField(
    }            "field2", parseFloat);
    if (conver var myString = getValueFromField(
        return   "field3");
    }
    return element.value;
}
```

Form Processing

- Reading an input field's value

```
function getValueFromField(fieldId, conversionFunction)
{
    var elemen
    if (elemen
        return
    }
    if (conver
        return
    }
    return element.value;
}
```

```
var myInteger = getValueFromField(
    "field1", parseInt);
var myFloat = getValueFromField(
    "field2", parseFloat);
var myString = getValueFromField(
    "field3");
```

These strings match the "id" attribute of the input element.

Form Processing

- Field validation
  - Can use "if/else" statements to check values and ranges

Form Processing

- Field validation
  - Can use "if/else" statements to check

```
function validateNonEmpty(elementId)
{
    var elt = document.getElementById(elementId);
    if (elt.value == "" || elt.value == null)
    {
        addError(elementId + " may not be empty.");
        return false;
    }
    return true;
}
```


Form Processing

- Field validation
  - Can use "if/else" statements to check values and ranges
  - Frequently want the field to match a particular pattern (i.e. zip codes)
    - Use a *regular expression* to do pattern matching

Form Processing

- Field validation
  - Regular expressions – an entire data

```
function validateIsZipcode(elementId)
{
    if (validateNonEmpty(elementId)) {
        var elt = document.getElementById(elementId);
        var pattern = /^\d{5}([\-]\d{4})?$/;
        if (!pattern.test(elt.value)) {
            addError(element + " is not valid.");
        }
    }
}
```


Form Processing

- Field validation
  - Regular expressions – an entire data

```
function validateIsZipcode(elementId)
{
    if (validateNonEmpty(elementId)) {
        var elt = document.getElementById(elementId);
        var pattern = /^\d{5}([\-]\d{4})?$/;
        if (!pattern.test(elt.value)) {
            addError(element + " is not valid.");
        }
    }
}
```

This apparent gobbledygook is a pattern that can be used to test strings. Called a "regular expression."

Form Processing

- Field validation
  - Regular expressions – an entire data type and language in itself!
    - See pages 78-84 of "Learning JavaScript"

ITEC 136

Business Programming Concepts

Week 08, Part 04

Miscellaneous

FRANKLIN UNIVERSITY

FOUNDED 1902

Dynamic HTML

- Changing content on the page
  - Create a `<div id="foo"></div>` region
  - Use `document.getElementById("foo")` to get access to the division
  - Use the "`innerHTML`" attribute to overwrite the contents of the division

Dynamic HTML

- Changing content on the page

```
<body>
  <input type="button" onclick="fillDiv('output')"
      value="Fill!" />
  <input type="button" onclick="clearDiv('output')"
      value="Clear!" />
  <div id="output"></div>
</body>
```

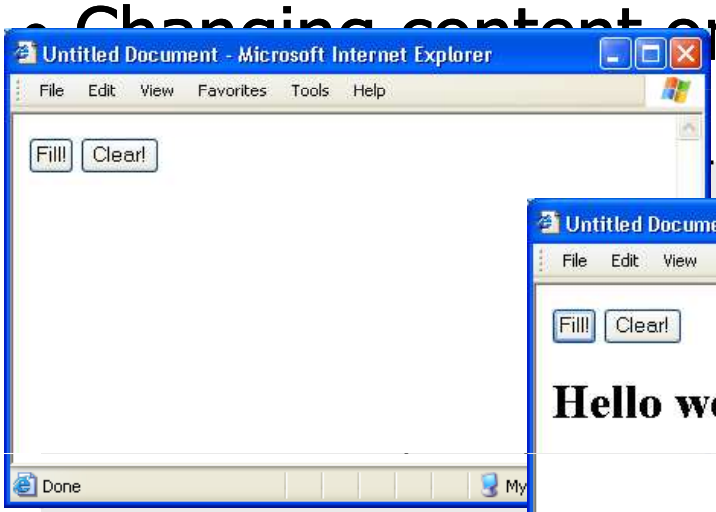
Dynamic HTML

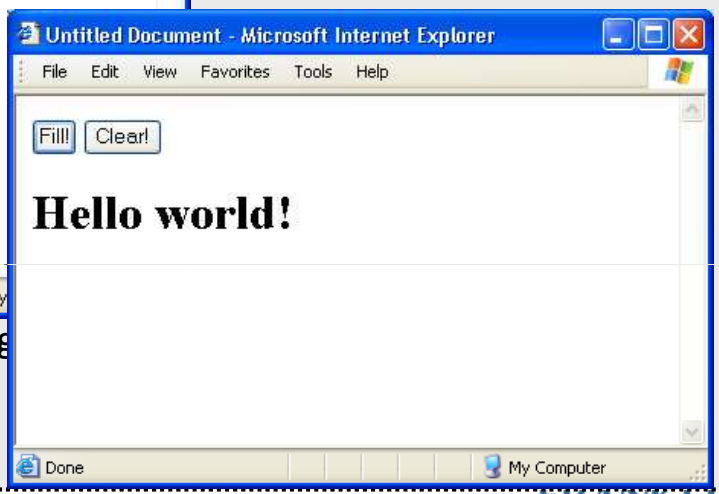
- Changing content on the page

```
function fillDiv(divId)
{
    var div = document.getElementById(divId);
    div.innerHTML = "<h1>Hello world!</h1>"
}
function clearDiv(divId)
{
    var div = document.getElementById(divId);
    div.innerHTML = "";
}
```


Dynamic HTML

- Changing content on the page

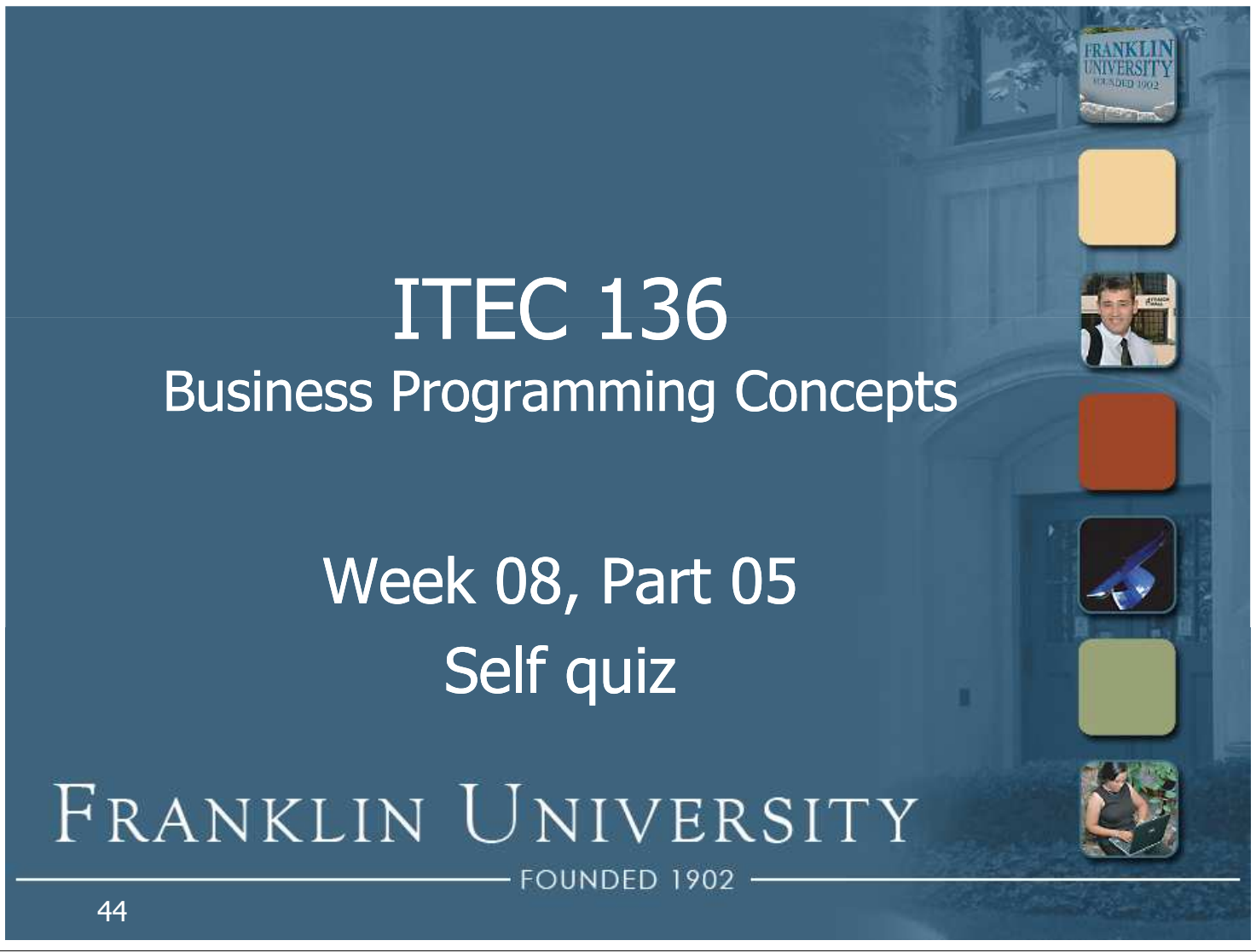





```
    var div = document.g
    div.innerHTML = "";
}
```

Questions?

FRANKLIN UNIVERSITY



ITEC 136

Business Programming Concepts

Week 08, Part 05

Self quiz

FRANKLIN UNIVERSITY

FOUNDED 1902

Self Quiz

- What tags permit you to create radio buttons? Checkboxes? Drop-down selects? What events do these controls generate?
- How do you programmatically access/change the values of these controls through Javascript?

FRANKLIN UNIVERSITY

ITEC 136

Business Programming Concepts




Week 08, Part 06

Upcoming deadlines

FRANKLIN UNIVERSITY

FOUNDED 1902

Upcoming Deadlines

- Homework 7 – due March 2
- Preclass exercise 9 – due March 2