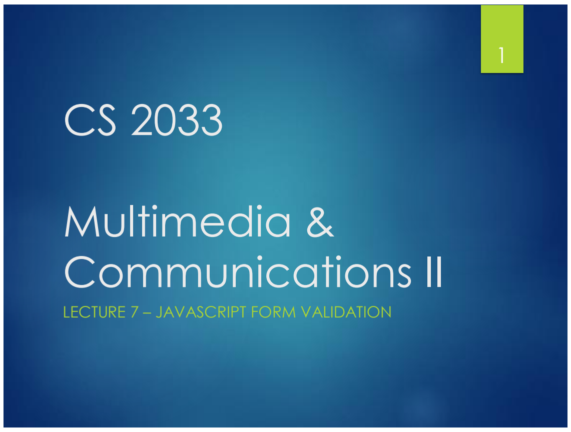
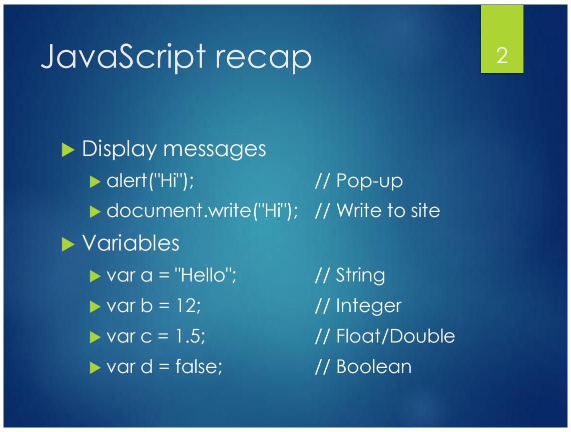


CS 2033

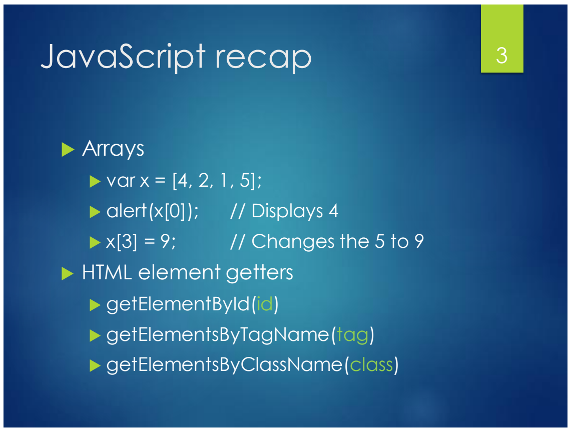
Multimedia & Communications II

LECTURE 7 – JAVASCRIPT FORM VALIDATION



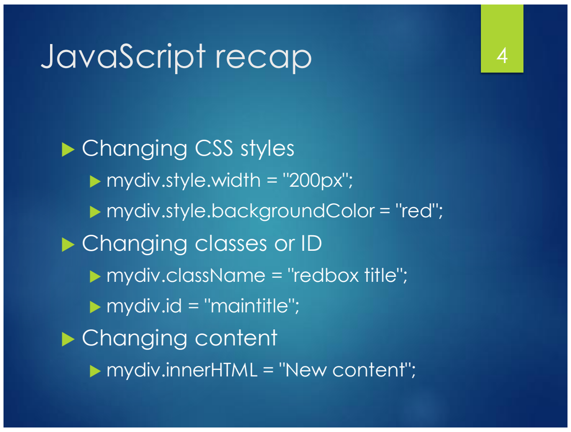
JavaScript recap

- Display messages
  - alert("Hi");   // Pop-up
  - document.write("Hi");   // Write to site
- Variables
  - var a = "Hello";   // String
  - var b = 12;   // Integer
  - var c = 1.5;   // Float/Double
  - var d = false;   // Boolean



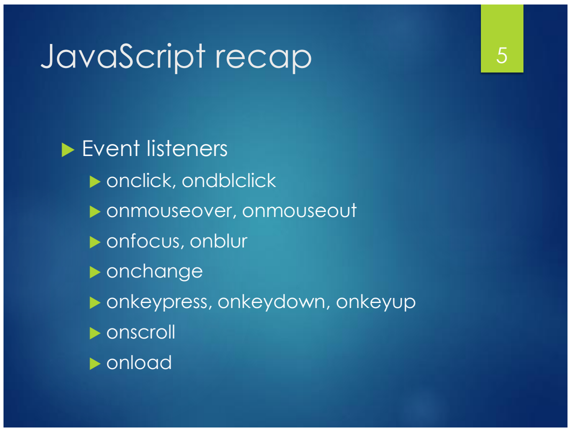
JavaScript recap

- Arrays
  - var x = [4, 2, 1, 5];
  - alert(x[0]);   // Displays 4
  - x[3] = 9;   // Changes the 5 to 9
- HTML element getters
  - getElementById(id)
  - getElementsByTagName(tag)
  - getElementsByClassName(class)



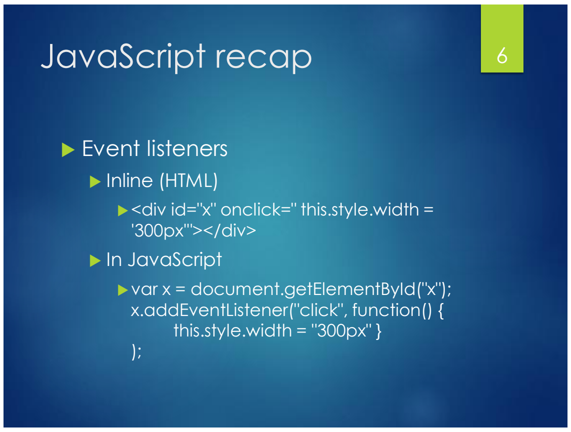
JavaScript recap

- Changing CSS styles
  - mydiv.style.width = "200px";
  - mydiv.style.backgroundColor = "red";
- Changing classes or ID
  - mydiv.className = "redbox title";
  - mydiv.id = "maintitle";
- Changing content
  - mydiv.innerHTML = "New content";



JavaScript recap

- Event listeners
  - onclick, ondblclick
  - onmouseover, onmouseout
  - onfocus, onblur
  - onchange
  - onkeypress, onkeydown, onkeyup
  - onscroll
  - onload

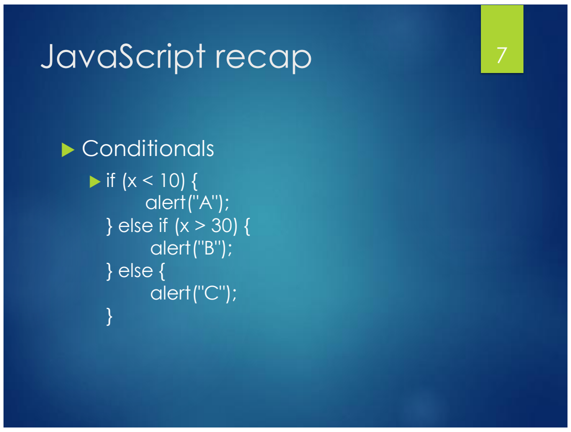


JavaScript recap

- Event listeners
  - Inline (HTML)

    -
      ```
      <div id="x" onclick=" this.style.width =
      '300px'"></div>
      ```
  - In JavaScript

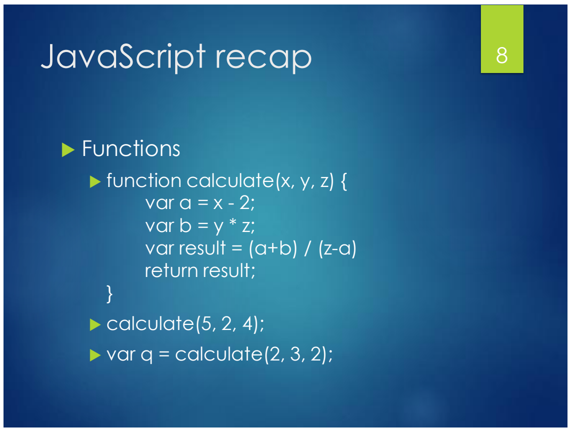
    -
      ```
      var x = document.getElementById("x");
      x.addEventListener("click", function() {
              this.style.width = "300px" }
      );
      ```



JavaScript recap

- Conditionals

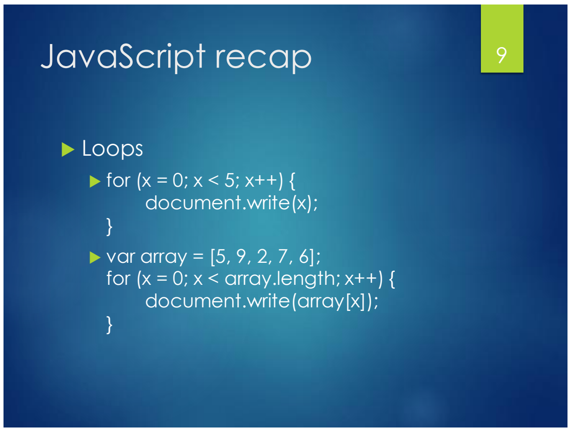
  -
    ```
    if (x < 10) {
          alert("A");
    } else if (x > 30) {
          alert("B");
    } else {
          alert("C");
    }
    ```



JavaScript recap

- Functions

  -
    ```
    function calculate(x, y, z) {
          var a = x - 2;
          var b = y * z;
          var result = (a+b) / (z-a)
          return result;
    }
    ```
  - calculate(5, 2, 4);
  - var q = calculate(2, 3, 2);

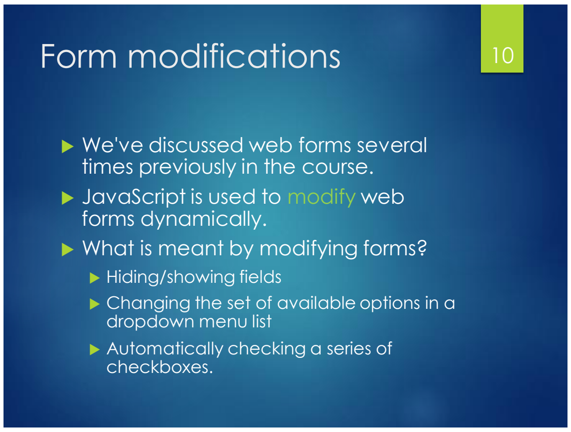


JavaScript recap

- Loops

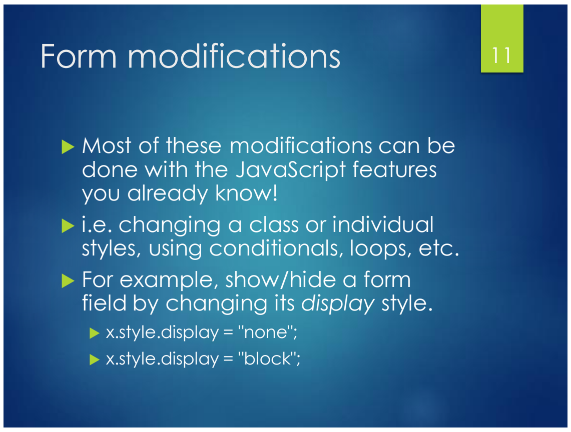
  -
    ```
    for (x = 0; x < 5; x++) {
          document.write(x);
    }
    ```

  -
    ```
    var array = [5, 9, 2, 7, 6];
    for (x = 0; x < array.length; x++) {
          document.write(array[x]);
    }
    ```



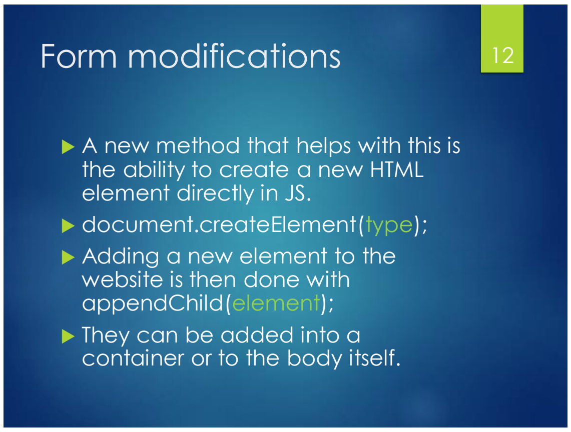
Form modifications

- We've discussed web forms several times previously in the course.
- JavaScript is used to modify web forms dynamically.
- What is meant by modifying forms?
  - Hiding/showing fields
  - Changing the set of available options in a dropdown menu list
  - Automatically checking a series of checkboxes.



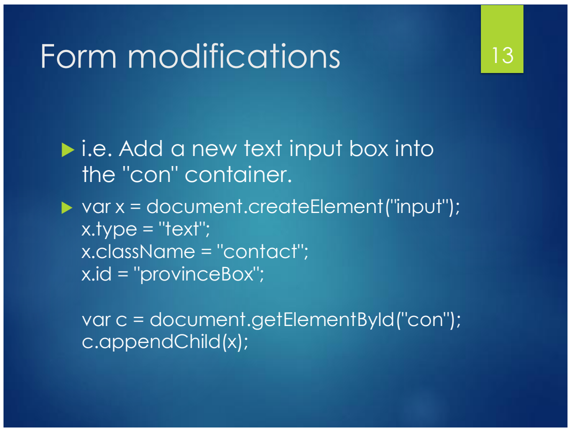
Form modifications

- Most of these modifications can be done with the JavaScript features you already know!
- i.e. changing a class or individual styles, using conditionals, loops, etc.
- For example, show/hide a form field by changing its *display* style.
  - x.style.display = "none";
  - x.style.display = "block";



Form modifications

- A new method that helps with this is the ability to create a new HTML element directly in JS.
- document.createElement(type);
- Adding a new element to the website is then done with appendChild(element);
- They can be added into a container or to the body itself.

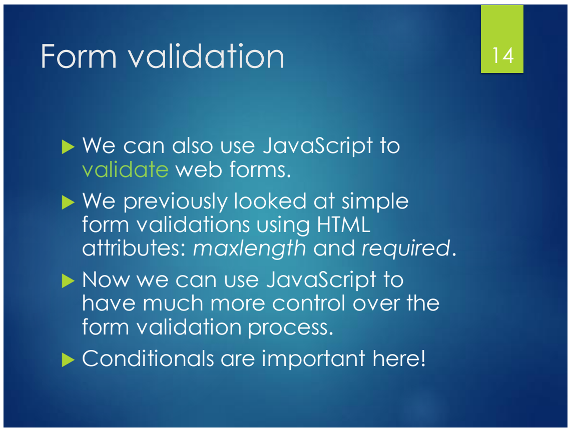


Form modifications

13

- i.e. Add a new text input box into the "con" container.

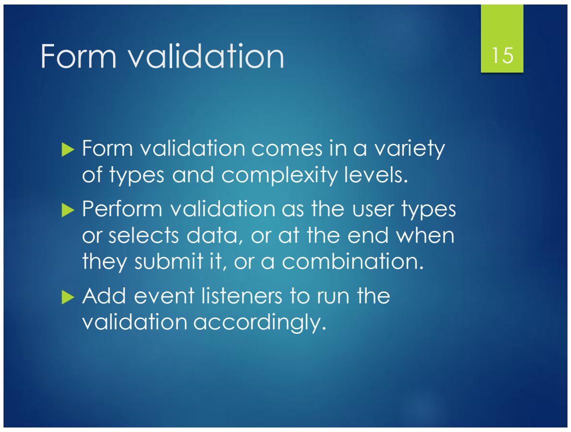
-
  ```
  var x = document.createElement("input");
  x.type = "text";
  x.className = "contact";
  x.id = "provinceBox";

  var c = document.getElementById("con");
  c.appendChild(x);
  ```



Form validation

14

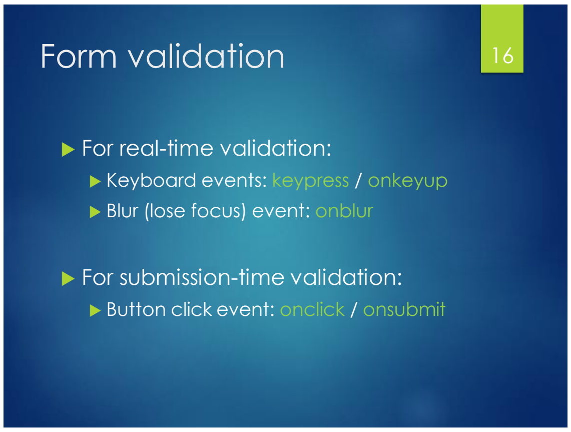
- We can also use JavaScript to validate web forms.
- We previously looked at simple form validations using HTML attributes: *maxlength* and *required*.
- Now we can use JavaScript to have much more control over the form validation process.
- Conditionals are important here!



Form validation

15

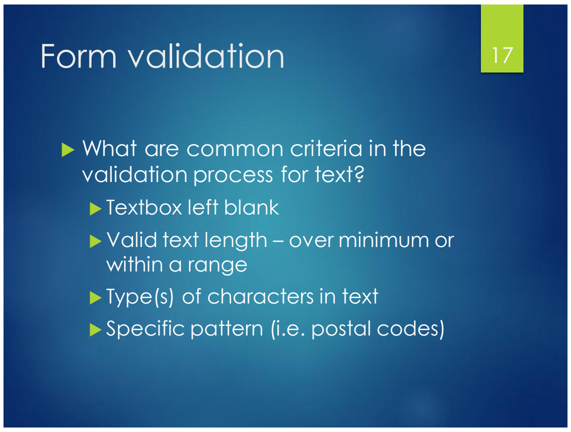
- Form validation comes in a variety of types and complexity levels.
- Perform validation as the user types or selects data, or at the end when they submit it, or a combination.
- Add event listeners to run the validation accordingly.



Form validation

16

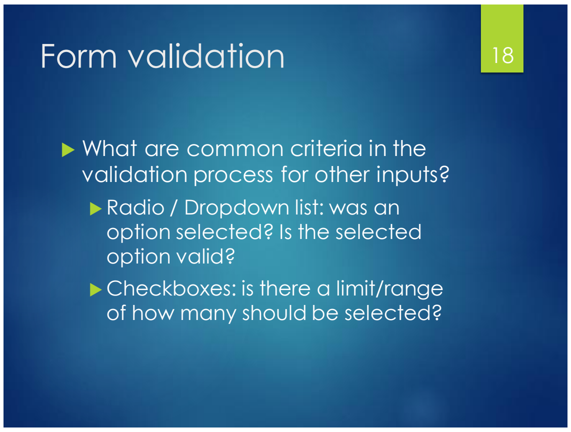
- For real-time validation:
  - Keyboard events: keypress / onkeyup
  - Blur (lose focus) event: onblur
- For submission-time validation:
  - Button click event: onclick / onsubmit



Form validation

17

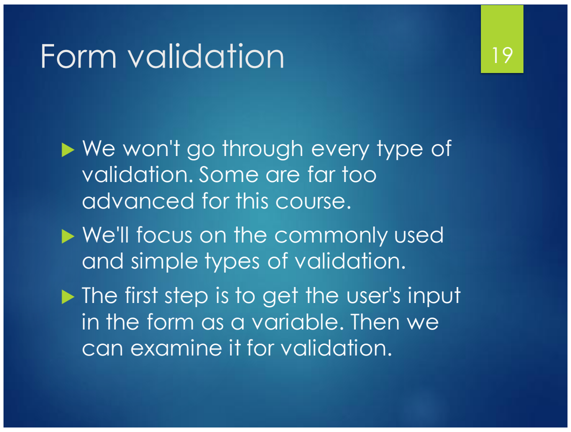
- What are common criteria in the validation process for text?
  - Textbox left blank
  - Valid text length – over minimum or within a range
  - Type(s) of characters in text
  - Specific pattern (i.e. postal codes)



Form validation

18

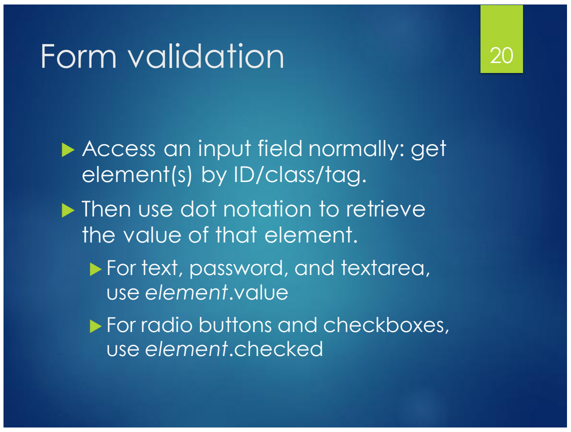
- What are common criteria in the validation process for other inputs?
  - Radio / Dropdown list: was an option selected? Is the selected option valid?
  - Checkboxes: is there a limit/range of how many should be selected?



Form validation

19

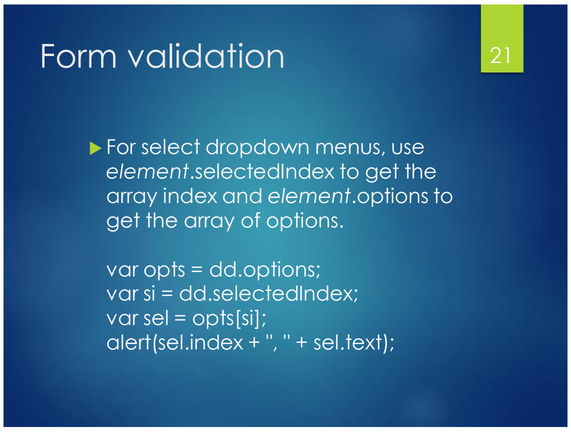
- We won't go through every type of validation. Some are far too advanced for this course.
- We'll focus on the commonly used and simple types of validation.
- The first step is to get the user's input in the form as a variable. Then we can examine it for validation.



Form validation

20

- Access an input field normally: get element(s) by ID/class/tag.
- Then use dot notation to retrieve the value of that element.
  - For text, password, and textarea, use *element*.value
  - For radio buttons and checkboxes, use *element*.checked

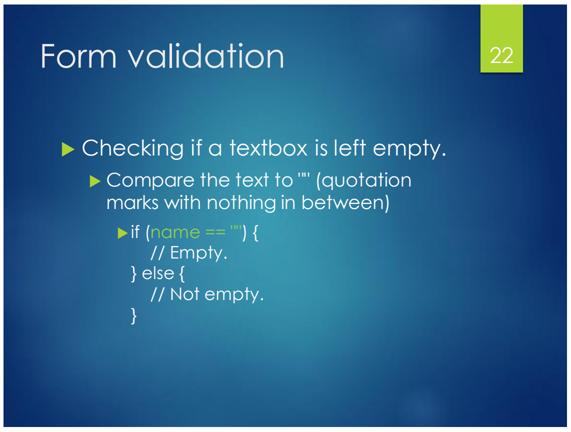


Form validation

21

- For select dropdown menus, use *element*.selectedIndex to get the array index and *element*.options to get the array of options.

```
var opts = dd.options;
var si = dd.selectedIndex;
var sel = opts[si];
alert(sel.index + ", " + sel.text);
```

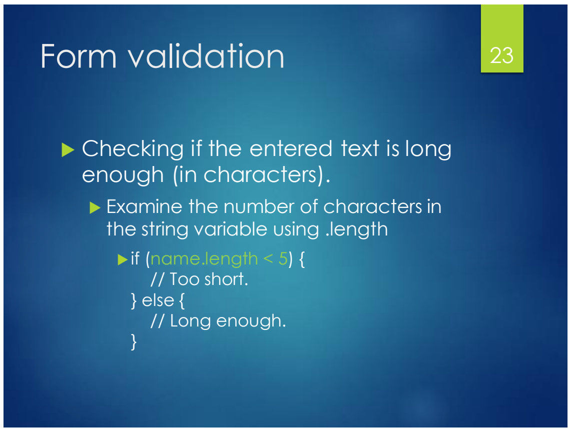


Form validation

22

- Checking if a textbox is left empty.
  - Compare the text to "" (quotation marks with nothing in between)

```
if (name == "") {
    // Empty.
} else {
    // Not empty.
}
```

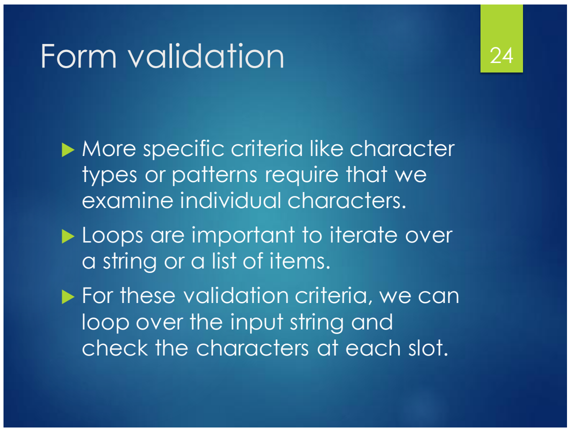


Form validation

23

- Checking if the entered text is long enough (in characters).
  - Examine the number of characters in the string variable using .length

```
if (name.length < 5) {
    // Too short.
} else {
    // Long enough.
}
```



Form validation

24

- More specific criteria like character types or patterns require that we examine individual characters.
- Loops are important to iterate over a string or a list of items.
- For these validation criteria, we can loop over the input string and check the characters at each slot.

Form validation

- Checking the character types within a string can be complex.
- One basic option to check if the entire string is a number or not is with the built-in isNaN() function (checks if value is Not a Number).
- isNaN(34) = isNaN(2.5) = false
- isNaN("abc") = isNaN("B7") = true

Form validation

- Before we continue with the form validation, let's look more at strings.
- Strings are just arrays of characters; only one character can be placed in each slot. Recall that positions start at 0 from the leftmost slot.
- var course = "CS2033";

C	S	2	0	3	3
0	1	2	3	4	5

- var msg = "HELLO WORLD";

H	E	L	L	O		W	O	R	L	D
0	1	2	3	4	5	6	7	8	9	10

Form validation

- Like other arrays, access each of the characters with a for-loop.
- var msg = "HELLO WORLD";

```
for (var i = 0; i < msg.length; i++) {
    // Examine character at position i
}
```

Form validation

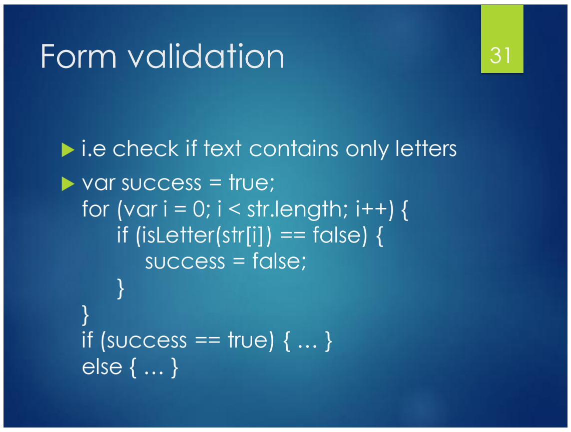
- Examining a character usually means comparing it to another value or a range of values.
- One option is to get the value directly from the string at position i and use that value for the analysis.
  - var char = msg[i];
    if (char == "W") {
    }

Form validation

- Instead of getting the character value itself in the loop, you could get its ASCII code for analysis.
  - var code = msg.charCodeAt(i);
  - if (code >= 65 && code <= 90) {
    }
  - Look up ASCII code charts for the ranges (65 to 90 is capital letters).

Form validation

- When using loop-based analysis, create a Boolean flag for "success".
- Default value depends on situation.
- Change its value to true or false as needed in the loop.
- At the end, check its final value to see if the overall string is valid or invalid.

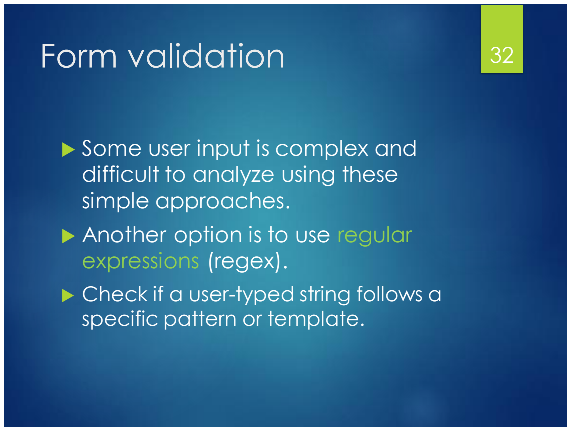


Form validation

31

- i.e check if text contains only letters
- var success = true;

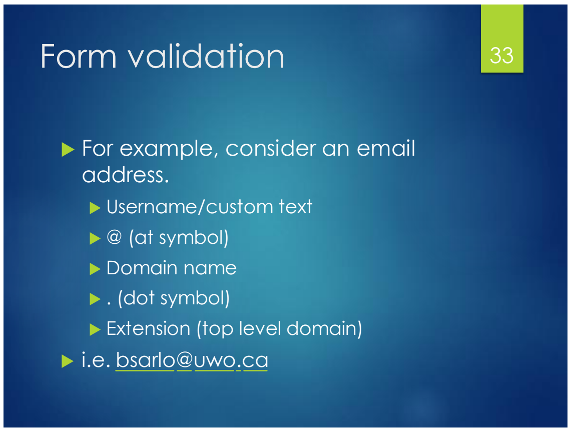
```
for (var i = 0; i < str.length; i++) {
      if (isLetter(str[i]) == false) {
          success = false;
      }
}
if (success == true) { … }
else { … }
```



Form validation

32

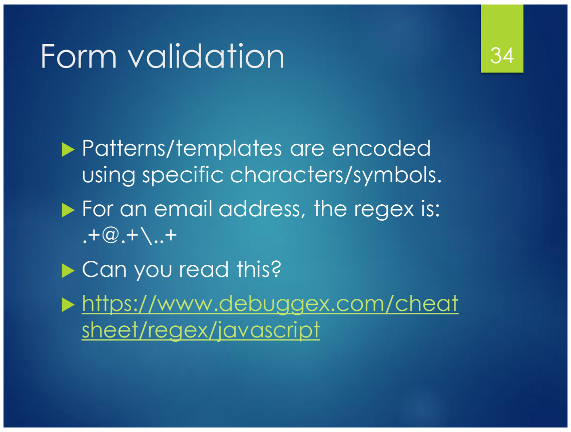
- Some user input is complex and difficult to analyze using these simple approaches.
- Another option is to use regular expressions (regex).
- Check if a user-typed string follows a specific pattern or template.



Form validation

33

- For example, consider an email address.
  - Username/custom text
  - @ (at symbol)
  - Domain name
  - . (dot symbol)
  - Extension (top level domain)
- i.e. bsarlo@uwo.ca



Form validation

34

- Patterns/templates are encoded using specific characters/symbols.
- For an email address, the regex is: `.+@.+\..+`
- Can you read this?
- https://www.debuggex.com/cheatsheet/regex/javascript