CS418/518: Web Programming
Fall 2022

LECTURE14: JAVASCRIPT AND PHP VALIDATION AND ERROR HANDLING

DR. JIAN WU

Courtesy: Dr. Justin Brunelle and Dr. Mohammed Misbhauddin

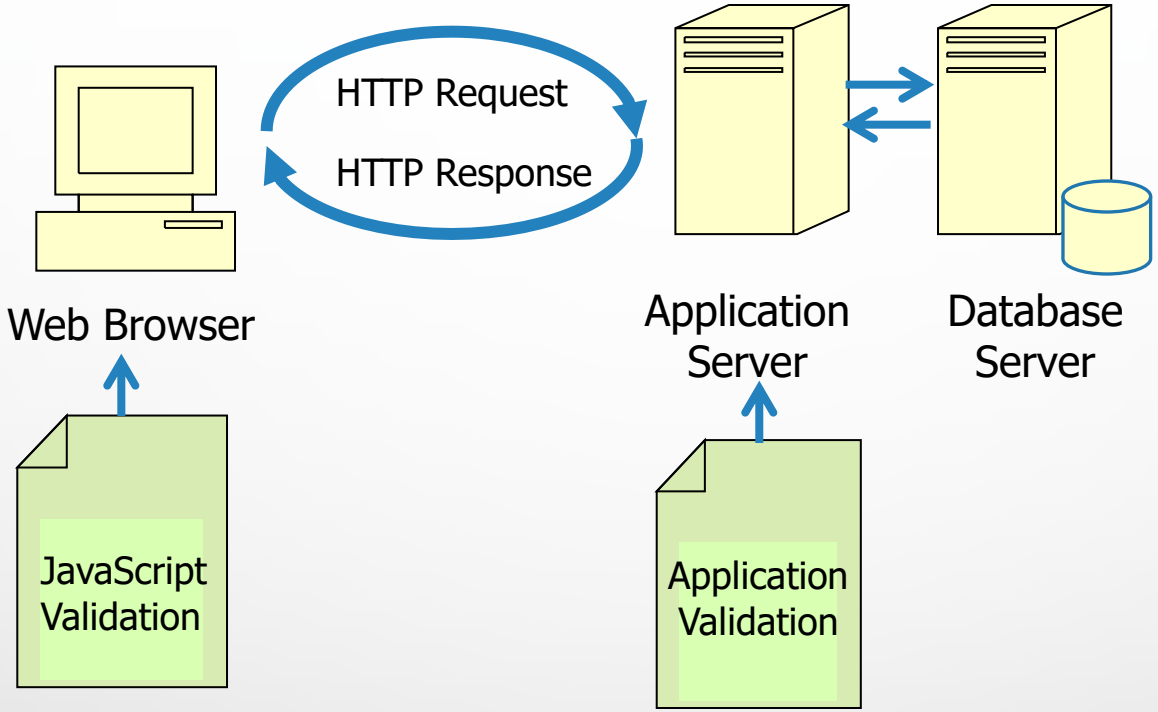
VALIDATING USER INPUT WITH JAVASCRIPT

- Why validating input: you cannot trust any data submitted to your server
  - Users may unintentionally make mistakes
  - Hackers may submit any data of their choosing
- You cannot rely on JavaScript to perform all input validation
  - Some users disable JavaScript
  - Browsers may not support it

JAVASCRIPT FORM VALIDATION

- Before an HTML form is submitted, JavaScript can be used to provide **client-side** data validation
- More user-friendly than **server-side validation**
  - Does not require a server round trip before giving feedback
- If the form is not valid, the form is not submitted until the errors are fixed

CLIENT-SIDE VALIDATION



- JavaScript data validation happens **before** form is submitted
- Server-side application validation happens **after** the form is submitted to the application server

WHAT TO VALIDATE ON A FORM?

- Form data that typically are checked by a JavaScript could be:
  - Were required fields left empty?
  - Was a valid e-mail address entered?
  - Was a valid date entered?
  - Was text entered in a numeric field?
  - Were numbers entered in a text field?
  - Did the number entered have a correct range?

onchange Validation

- To force the browser to check each field immediately, we add an **onchange** event to each of the **<input>** tags in the form
- For example: if we wanted to check if the value of a certain text field had a valid e-mail address, we would add this:

```
<input type="text" name="EMail" size="20"
        onchange="emailvalidation(this);" >
```

<form> onsubmit Event

- Your form must have a submit button the user clicks when completing the form
- An **onsubmit** event will be raised and you should put this code in the **<form>** tag
- Call your event handler to go through and test each form field as needed
- If the event handler returns false, the form submission will be cancelled, if true the form will submit, and the form action will be executed

```
<form action="Process.php”  onsubmit="return validate(this);”  >
```

onsubmit Event Handler

- Pass the form object as the **this** parameter
  - `onsubmit="return validate(this);"`
- This function should create variables for each field that needs validation
- Set inputvalid = true to begin with
- Use a series of if statements to perform each validation test, if test fails set inputvalid =false and set error message and/or alert message
- Finally, return inputvalid from the event handler

Required Fields

- What fields on a Web form should be required?
  - Good usability practice suggests the form designer only make the user fill out necessary information
  - Data may be required when sent to a database such as non-null data
- Additional good practice would mark which fields are required on the form
  - Often marked with an *
  - May spell out the “Required” or style differently

GENERAL EVENT HANDLER STRUCTURE

Pseudo-code

```
function validate(form) {
// Set each of needed form variables

var input valid=true;
var message="Please fix the following errors \n";

    if (!testFunction(text)) {
      // something is wrong
      // message =+ "new error \n";
      // validinput= false;
    }
    if (!testFunction2(text)) {
        // something else is wrong
        // message =+ "new error \n";
        // validinput= false;
     }
    // do each validation test
    if(!validinput) {
     alert (message);
    }
return validinput;
}
```

TESTING FOR REQUIRED ENTRY

- Checking a textbox field could be done with a simple test for **text.length == 0**
- Checking a select field to see if an option has been selected use **selectedIndex>0**
  - Place instruction text as the first <option>
- Checking a checkbox **checked==true**
- Checking a radio button need to loop through array and test each **checked==true**
  - May want to always set a default radio button as **selected="selected"**

TESTING FOR VALID INPUT

- If a textbox asks for an email, test the text entered is a valid email
- If a textbox asks for a date, test the text entered is a valid date
  - Very complex to text format and validity
- If a textbox asks for a zipcode, test the text entered is a valid zipcode

REGULAR EXPRESSIONS

- You can use **regular expression** to test your input text
- There are many regular expressions available for common tests
  - U.S. Phone: `/^\(?(\d{3})\)?[- ]?(\d{3})[- ]?(\d{4})$/`
  - Email: `/^[0-9a-zA-Z]+@[0-9a-zA-Z]+[\.]{1}`
    `[0-9a-zA-Z]+[\.]?[0-9a-zA-Z]+$/`
  - Currency: `/^\s*(\+|-)?((\d+(\.\d\d)?)|(\.\d\d))\s*$/`

EXAMPLE REGULAR EXPRESSION

- Test to see if text is a number
- Returns true if number, false otherwise
- Does the text match the pattern?

```
// check if the text is a number
function IsNumber(fData) {
   var reg = /^[0-9]+[\.]?[0-9]+$/;
   return reg.test(fData)
}
```

MULTIPLE VALIDATIONS ???

- A Web form field may need more than one validation test
- For example, a textbox for age:
  - It may be a required field
  - It must be a number
  - It must be in a range 0 - 120
- Order your validation tests from general to most specific

MODULARIZE YOUR FUNCTIONS

- Write a separate function for each type of validation
- Generalize each function
- Place these frequently used validation functions in their own external JavaScript file

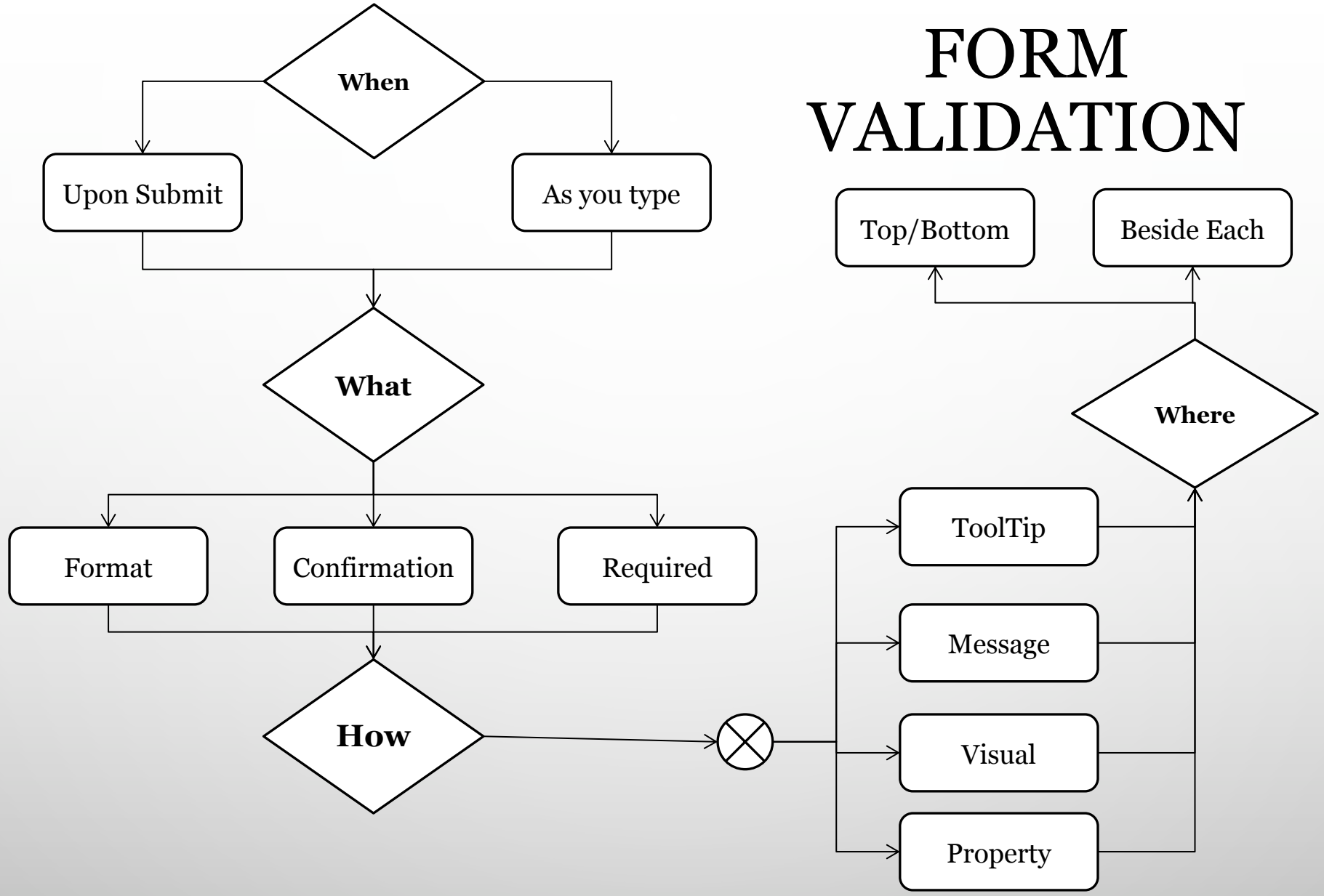
FINE TUNE ERROR MESSAGES

- Help users to successfully complete the Web form
- Provide hints on formatting by the fields on the form rather than wait for a submission error
- Be professional and helpful with the error messages
- Try to provide both error message by each field and summary text in an alert box

TESTING THE WEB FORM

- First test that required fields must be provided
- Then test fields that need valid input such as:
  - Phone number
  - Email address
  - Dates
- Make sure error messages are appropriate and specific for each error

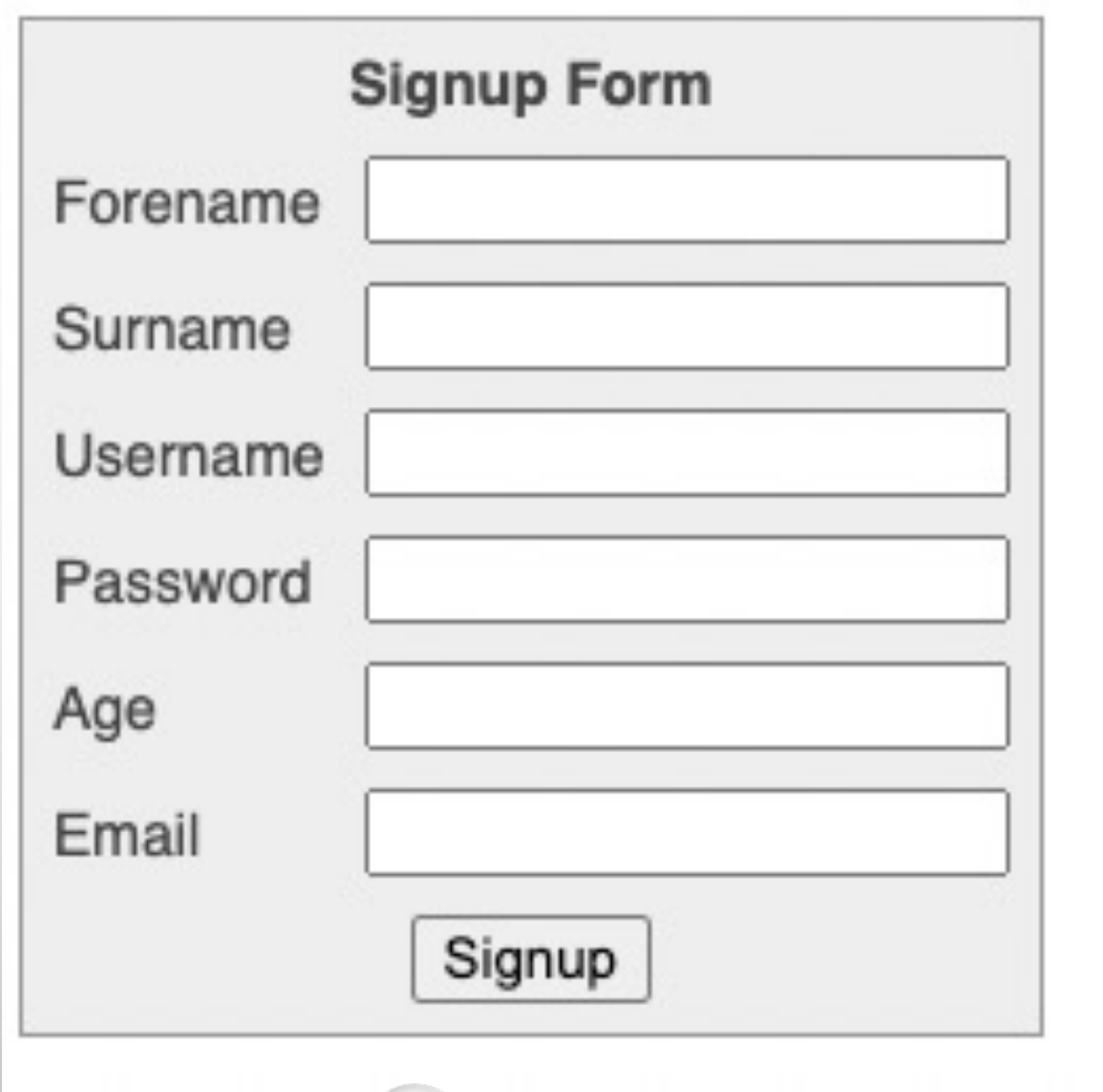
FORM VALIDATION



EXAMPLE: VALIDATE.HTML

WHAT DOES THE FORM LOOK LIKE

- This form will display correctly but will not self-validate
- Functions are not implemented



EXAMPLE: VALIDATE.HTML – 1ST PART

```
<script>
      function validate(form)
      {
        fail  = validateForename(form.forename.value)
        fail += validateSurname(form.surname.value)
        fail += validateUsername(form.username.value)
        fail += validatePassword(form.password.value)
        fail += validateAge(form.age.value)
        fail += validateEmail(form.email.value)
        if   (fail == "")   return true
        else { alert(fail); return false }
      }
</script>
```

Each function either returns an empty string if a field validates, or an error message if it fails.

The validate function returns true (form will be submitted) or false (form will not be submitted).

- For more see: https://www.w3schools.com/js/js_validation.asp

VALIDATE.HTML: 2ND PART

```
<table class="signup" border="0" cellpadding="2"
               cellspacing="5" bgcolor="#eeeeee">
      <th colspan="2" align="center">Signup Form</th>
      <form method="post" action="adduser.php" onsubmit="return validate(this)">
        <tr><td>Forename</td>
          <td><input type="text" maxlength="32" name="forename"></td></tr>
        <tr><td>Surname</td>
          <td><input type="text" maxlength="32" name="surname"></td></tr>
        <tr><td>Username</td>
          <td><input type="text" maxlength="16" name="username"></td></tr>
        <tr><td>Password</td>
          <td><input type="text" maxlength="12" name="password"></td></tr>
        <tr><td>Age</td>
          <td><input type="text" maxlength="3"  name="age"></td></tr>
        <tr><td>Email</td>
          <td><input type="text" maxlength="64" name="email"></td></tr>
        <tr><td colspan="2" align="center"><input type="submit"
  value="Signup"></td></tr>
</form>
</table>
```

Using onSubmit, you can cause a function of your choice to be called when a form is submitted.

That function can perform some checking and return a value of either true or false to signify whether the form should be allowed to be submitted.

The “this” parameter is the current object (form)

VALIDATE.HTML: 3RD PART

```
<script>
        function validate(form)
        {…} // the body of the validate() function
        function validateForename(field) {
                return (field == "") ? "No Forename was entered.\n" : ""
        }
        function validateSurname(field) {
                return (field == "") ? "No Surname was entered.\n" : ""
        }
        function validateUsername(field) {
                if (field == "") return "No Username was entered.\n"
                else if (field.length < 5)
                        return "Usernames must be at least 5 characters.\n"
                else if (/[^a-zA-Z0-9_-]/.test(field))
                        return "Only a-z, A-Z, 0-9, - and _ allowed in Usernames.\n"
                return ""
        }
```

Even if the user entered spaces in this field, it would be accepted.

VALIDATE.HTML: 4TH PART

```
function validatePassword(field) {
        if (field == "")
                return "No Password was entered.\n"
        else if (field.length < 6)
                return "Passwords must be at least 6 characters.\n"
        else if (!/[a-z]/.test(field) || ! /[A-Z]/.test(field) || !/[0-9]/.test(field))
                return "Passwords require one each of a-z, A-Z and 0-9.\n"
        return ""
}
```

must have at least one each of a lowercase, uppercase, and numerical character

VALIDATE.HTML: 5TH PART

NaN=Not a Number: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/NaN

```
function validateAge(field) {
        if (field == "" || isNaN(field)) return "No Age was entered.\n"
        else if (field < 18 || field > 110)
                return "Age must be between 18 and 110.\n"
        return ""
}
function validateEmail(field) {
        if (field == "") return "No Email was entered.\n"
        else if (!((field.indexOf(".") > 0) && (field.indexOf("@") > 0)) || /[^a-zA-Z0-9.@_-]/.test(field))
                return "The Email address is invalid.\n"
        return ""
}
</script>
```

ensure there is a period (.) somewhere after the first character of the field and an @ symbol appears somewhere after the first character

PUT THEM TOGETHER

- validate.html
  - Containing only the forms
  - Include <script src="validate_functions.js"></script> in the <head>
- validate_functions.js
  - Containing only actions
  - Saved under the same directory at validate.html

REGULAR EXPRESSIONS

- Regular expressions (Regex) are used for matching patterns in text strings
- Every regular expression must be enclosed in slashes.

- Metacharacters (wildcard characters)
  - *: The text you're trying to match may have any number of the preceding characters—or none at all
  - .: match anything except a newline
  - ^: If it appears at the beginning of the regular expression, the expression has to appear at the beginning of a line of text; otherwise, it doesn't match.
  - $: If it appears at the end of the regular expression, the expression has to appear at the end of a line of text.
  - See https://www.w3schools.com/jsref/jsref_obj_regexp.asp for more metacharacters
- What if you want to match a metacharacter?
  - Use the escape character "\"

REGEX EXAMPLES – 1ST PART

- String: The difficulty of classifying Le Guin's works
- Regex: /Le *Guin/
  - Match `LeGuin`, as well as `Le` and `Guin` separated by any number of spaces.
- Regex: /Le +Guin/
  - Matches a text span with `Le` and `Guin` separated by at least one space
- Regex: /^Le *Guin$/
  - Matches a line that has "Le Guin" and nothing else

- String: an html file
- Regex: /<.*>/
  - Looks for an HTML tag, which start with < and end with >.
  - Match anything that lies between < and >, even if there's nothing.
  - <>, , <br>, <a href=www.mozilla.org>, <h1>Introduction</h1>
- Regex: /5\.0/
  - Match the floating-point number 5.0

what if you don't want to match <>?
what if you just want to match a tag containing a single character?

GROUPING THROUGH PARENTHESES

- Parentheses mean “treat this as a group when you apply something such as a plus sign.
- Example: match the following
  - 1,000
  - 1,000,000
  - 1,000,000,000
  - 1,000,000,000,000
- Regex: /1(,000)+ /
- Note: 1,00,000 and 1,000,00 won’t match (why?)

Note: there is a space after the + character indicates that the match must end when a space is encountered

CHARACTER CLASS AND RANGES BY SQUARE BRACKETS []

- Character class: if any of those characters in the bracket appears, the text matches. You can you put a list of things that can match.
- Example: to match grey and gray, you can use
- Regex: /gr[ae]y/

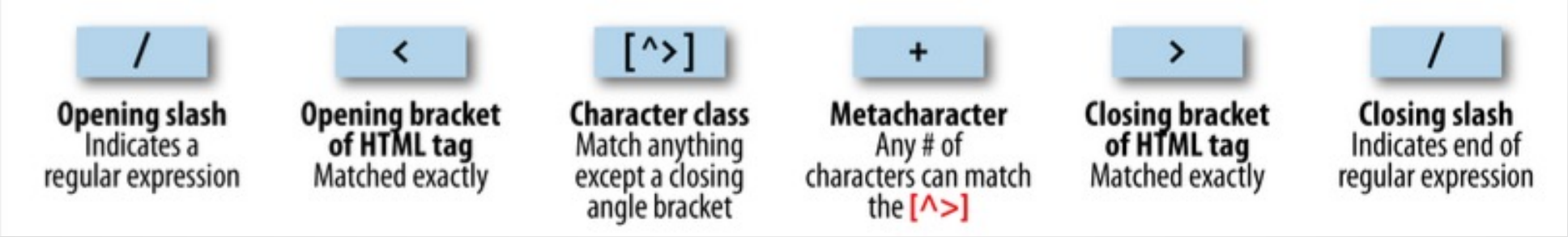
- Range: character or digit ranges
- Example: to match any single digit
- Regex: /[0-9]/ Or use /\d/

NEGATION BY A CARET ^

- Match any characters except the following
- Example: find instances of Yahoo that lack the following exclamation point
- Regex: /Yahoo[^!]/

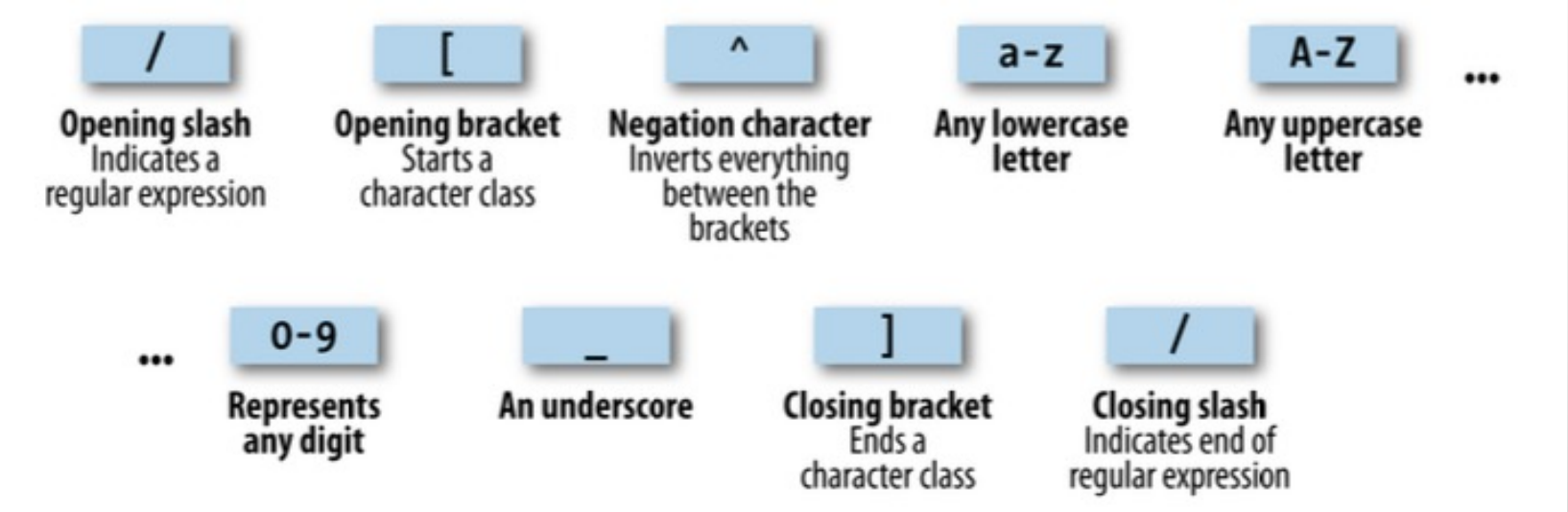
A MORE COMPLICATED EXAMPLE

- How to design a regex so that it avoids going past the end of a single tag, but still matches tags such as and as well as tags with attributes such as <a href="www.mozilla.org">
- Regex: /<[^>]+>/



GOING BACK TO validateUsername()

- /[^a-zA-Z0-9_-]/



USING REGULAR EXPRESSIONS IN JAVASCRIPT

- test: tells you whether its argument matches the regular expression.
- replace: generates a new string as a return value; it does not change the input.
- Example:
  - Returns true to let us know that the word cats appears at least once somewhere within the string
  - document.write(/cats/i.test("Cats are funny. I like cats."))
  - i: case-insensitive. By default, case sensitive.
  - document.write("Cats are friendly. I like cats.".replace(/cats/gi,"dogs"))
- Example:
  - Replaces both occurrences of the word cats with the word dogs, printing the result
  - document.write("Cats are friendly. I like cats.".replace(/cats/gi,"dogs"))
  - g: global search to find all occurrences

REDISPLAYING A FORM AFTER VALIDATION

- In validate.html, we pass the input to adduser.php, but only if JavaScript validates the fields or if JavaScript is disabled or unavailable.

```
<table class="signup" border="0" cellpadding="2"
              cellspacing="5" bgcolor="#eeeeee">
      <th colspan="2" align="center">Signup Form</th>
      <form method="post" action="adduser.php" onsubmit="return validate(this)">
        <tr><td>Forename</td>
          <td><input type="text" maxlength="32" name="forename"></td></tr>
        <tr><td>Surname</td>
          <td><input type="text" maxlength="32" name="surname"></td></tr>
        <tr><td>Username</td>
          <td><input type="text" maxlength="16" name="username"></td></tr>
        <tr><td>Password</td>
          <td><input type="text" maxlength="12" name="password"></td></tr>
        <tr><td>Age</td>
          <td><input type="text" maxlength="3"  name="age"></td></tr>
        <tr><td>Email</td>
          <td><input type="text" maxlength="64" name="email"></td></tr>
        <tr><td colspan="2" align="center"><input type="submit"
  value="Signup"></td></tr>
</form>
</table>
```

REDISPLAYING A FORM AFTER VALIDATION

- If JavaScript is disabled, PHP should do the sanity check. See the code in adduser.php.
- Note the section called Heredoc starting with <<<_END and end with _END

SUMMARY

- JavaScript can be used for client-side data validation of a Web form
- Modularize validation functions to be reusable for future work
- Begin by making form user-friendly with instructions and hints
- Provide helpful error messages