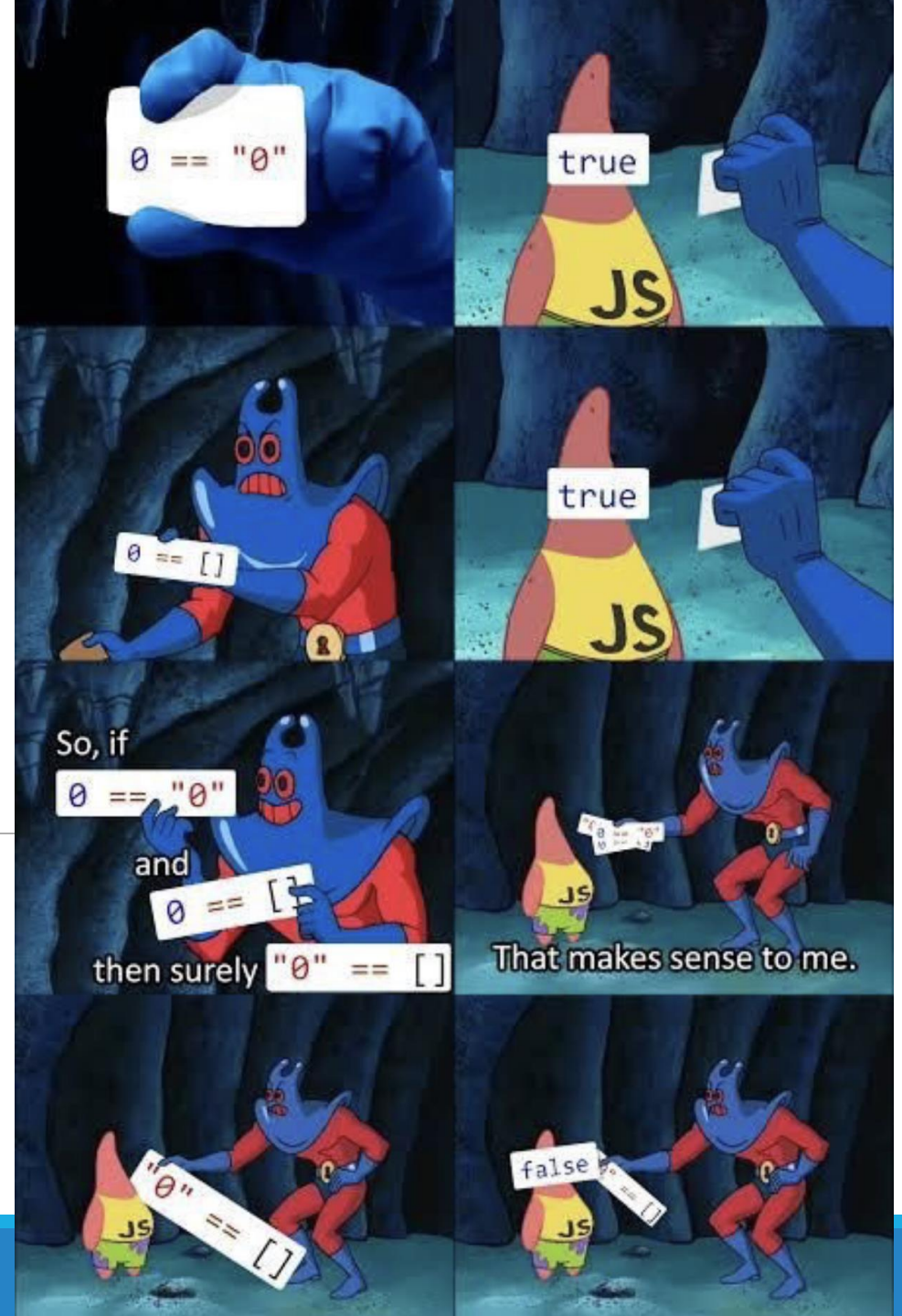


# CSc 337

## LECTURE 23: REGULAR EXPRESSIONS



# What is form validation?

---

- **validation:** ensuring that form's values are correct
- some types of validation:
  - preventing blank values (email address)
  - ensuring the type of values
    - integer, real number, currency, phone number, Social Security number, postal address, email address, date, credit card number, ...
  - ensuring the format and range of values (ZIP code must be a 5-digit integer)
  - ensuring that values fit together (user types email twice, and the two must match)

# A real form that uses validation



[Cancel](#)



Some of the information you entered is missing or incorrect. Please check all highlighted messages below.

- ⚠ Please enter Last Name using letters, apostrophes or dashes.
- ⚠ Enter a valid date for Date of Birth.
- ⚠ Please enter a valid e-mail address.

## Personal Info

First Name:

Last Name:

Date of Birth:

E-mail Address:

Secure Site

Questions? Call us:

(800) 788-7000

- Identify yourself by your:
- Account Number
  - ATM/Debit Card
  - Credit Card

# Client vs. server-side validation

---

Validation can be performed:

- **client-side** (before the form is submitted)
  - can lead to a better user experience, but not secure (why not?)
- **server-side** (in PHP code, after the form is submitted)
  - needed for truly secure validation, but slower
- both
  - best mix of convenience and security, but requires most effort to program

# An example form to be validated

```
<div>
  City:  <input name="city" /> <br />
  State: <input name="state" size="2" maxlength="2" /> <br />
  ZIP:   <input name="zip" size="5" maxlength="5" /> <br />
  <input type="submit" />
</div>
```

City:

State:

ZIP:

output

- Let's validate this form's data on the server...

# Regular expressions

---

```
/^[a-zA-Z_\-]+@(( [a-zA-Z_\-])+\.)+[a-zA-Z]{2,4}$/
```

- **regular expression** ("regex"): a description of a pattern of text
  - can test whether a string matches the expression's pattern
  - can use a regex to search/replace characters in a string
- regular expressions are extremely powerful but tough to read (the above regular expression matches email addresses)
- regular expressions occur in many places:
  - Java: Scanner, String's split method (CSE 143 sentence generator)
  - supported by PHP, JavaScript, and other languages
  - many text editors (TextPad) allow regexes in search/replace
  - The site [Rubular](#) is useful for testing a regex.

# Basic regular expressions

---

```
/abc/
```

- in Node, regexes are strings that begin and end with /
- the simplest regexes simply match a particular substring
- the above regular expression matches any string containing "abc":
  - YES: "abc", "abcdef", "defabc", "=".abc.=" , ...
  - NO: "fedcba", "ab c", "PHP", ...

# Wildcards: .

---

- A dot `.` matches any character except a `\n` line break
  - `/.oo.y/` matches "Doocy", "goofy", "LooNy", ...
- A trailing `i` at the end of a regex (after the closing `/`) signifies a case-insensitive match
  - `/all/i` matches "Allison Obourn", "small", "JANE GOODALL", ...



# Special characters: |, (), \

---

- | means *OR*
  - `/abc|def|g/` matches "abc", "def", or "g"
  - There's no *AND* symbol. Why not?
- () are for grouping
  - `/(Homer|Marge) Simpson/` matches "Homer Simpson" or "Marge Simpson"
- \ starts an escape sequence
  - many characters must be escaped to match them literally: `/\ $ . [ ] ( ) ^ * + ?`
  - `/<br \>/` matches lines containing `<br />` tags

# Quantifiers: \*, +, ?

---

- \* means 0 or more occurrences
  - `/abc*/` matches "ab", "abc", "abcc", "abccc", ...
  - `/a(bc)*/` matches "a", "abc", "abcbc", "abcbcbc", ...
  - `/a.*a/` matches "aa", "aba", "a8qa", "a!?xyz\_\_9a", ...
- + means 1 or more occurrences
  - `/Hi!+ there/` matches "Hi! there", "Hi!!! there", ...
  - `/a(bc)+/` matches "abc", "abcbc", "abcbcbc", ...
- ? means 0 or 1 occurrences
  - `/a(bc)?/` matches "a" or "abc"

# More quantifiers: {min,max}

---

- $\{min,max\}$  means between *min* and *max* occurrences (inclusive)
  - `/a(bc){2,4}/` matches "abcbc", "abcbcbc", or "abcbcbcbc"
- *min* or *max* may be omitted to specify any number
  - $\{2,\}$  means 2 or more
  - $\{,6\}$  means up to 6
  - $\{3\}$  means exactly 3

# Practice exercise

---

- When you search Google, it shows the number of pages of results as "o"s in the word "Google". What regex matches strings like "Google", "Goooogle", "Goooooogle", ...? ([try it](#)) ([data](#))
- Answer: `/Goo+gle/` (or `/Go{2,}gle/`)

# Anchors: ^ and \$

---

- ^ represents the beginning of the string or line;  
\$ represents the end
  - /Jess/ matches all strings that contain Jess;  
/^Jess/ matches all strings that *start with* Jess;  
/Jess\$/ matches all strings that *end with* Jess;  
/^Jess\$/ matches the exact string "Jess" only
  - /^Alli.\*Obourn\$/ matches "AlliObourn", "Allie Obourn", "Allison E Obourn", ...  
but NOT "Allison Obourn stinks" or "I H8 Allison Obourn"
- (on the other slides, when we say, /PATTERN/ matches "text", we really mean that it matches any string that contains that text)

# Character sets: []

---

- [] group characters into a **character set**; will match any single character from the set
  - `/[bcd]art/` matches strings containing "bart", "cart", and "dart"
  - equivalent to `/(b|c|d)art/` but shorter
- inside [], many of the modifier keys act as normal characters
  - `/what[!*?]*/` matches "what", "what!", "what?\*!", "what??!", ...
- What regular expression matches DNA (strings of A, C, G, or T)?
  - `/[ACGT]+/`

# Character ranges: [start-end]

---

- inside a character set, specify a range of characters with -
  - `/[a-z]/` matches any lowercase letter
  - `/[a-zA-Z0-9]/` matches any lower- or uppercase letter or digit
- an initial `^` inside a character set negates it
  - `/[^abcd]/` matches any character other than a, b, c, or d
- inside a character set, `-` must be escaped to be matched
  - `/[+\-]?[0-9]+/` matches an optional `+` or `-`, followed by at least one digit

# Practice Exercises

---

What regular expression matches letter grades such as A, B+, or D- ? ([try it](#)) ([data](#))

What regular expression would match UA Student ID numbers? ([try it](#)) ([data](#))

What regular expression would match a sequence of only consonants, assuming that the string consists only of lowercase letters? ([try it](#)) ([data](#))



# Escape sequences

---

- special escape sequence character sets:
  - `\d` matches any digit (same as `[0-9]`); `\D` any non-digit (`[^0-9]`)
  - `\w` matches any word character (same as `[a-zA-Z_0-9]`); `\W` any non-word char
  - `\s` matches any whitespace character ( , `\t`, `\n`, etc.); `\S` any non-whitespace
- What regular expression matches names in a "Last, First M." format with any number of spaces?
  - `/\w+,\s+\w+\s+\w\./`

# Regular expressions in HTML forms

---

How old are you?

```
<input type="text" name="age" size="2" pattern="[0-9]+" title="an integer" />
```

HTML

How old are you?

output

- HTML5 adds a new [pattern attribute](#) to input elements
- the browser will refuse to submit the form unless the value matches the regex

# Regular expressions in JavaScript

---

```
var str = "The rain in SPAIN stays mainly in the plain";  
var res = str.match(/ain/g);
```

- The match function takes a regular expression as a parameter and returns the matches as an Array object
- The `g` option means global match
  - without it you will only get the first match in the string
- Note that there are no quotes around the regex

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.

OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!



BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!



IT'S HOPELESS!

EVERYBODY STAND BACK.



I KNOW REGULAR EXPRESSIONS.

