SQL AND MORE EVENT VALIDATION

Database and front end tricks

Announcements

- Demo Tuesday, April 1
  - Sprint 1
    - Each person chooses a story
    - The team presents the results to the TA
    - The TA grades the results

Logging in Users

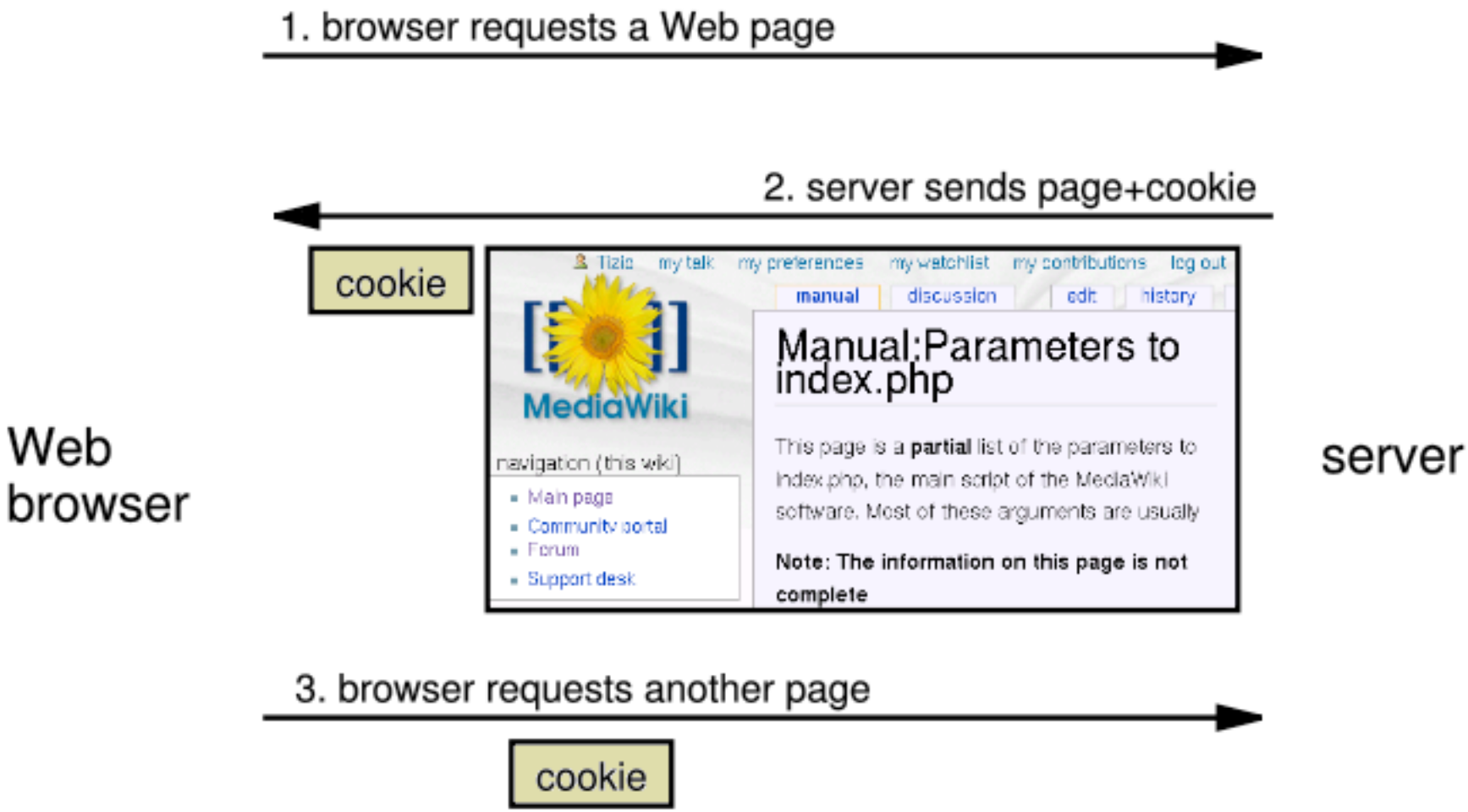
Tasks to be done

1. Index page with login and new user forms
    1. Create a new session if needed
2. landing page for login (i.e. home page)
    1. retrieve user name and password
    2. check database
        1. add name $_SESSION[“name”] if valid
        2. return to index.php if not

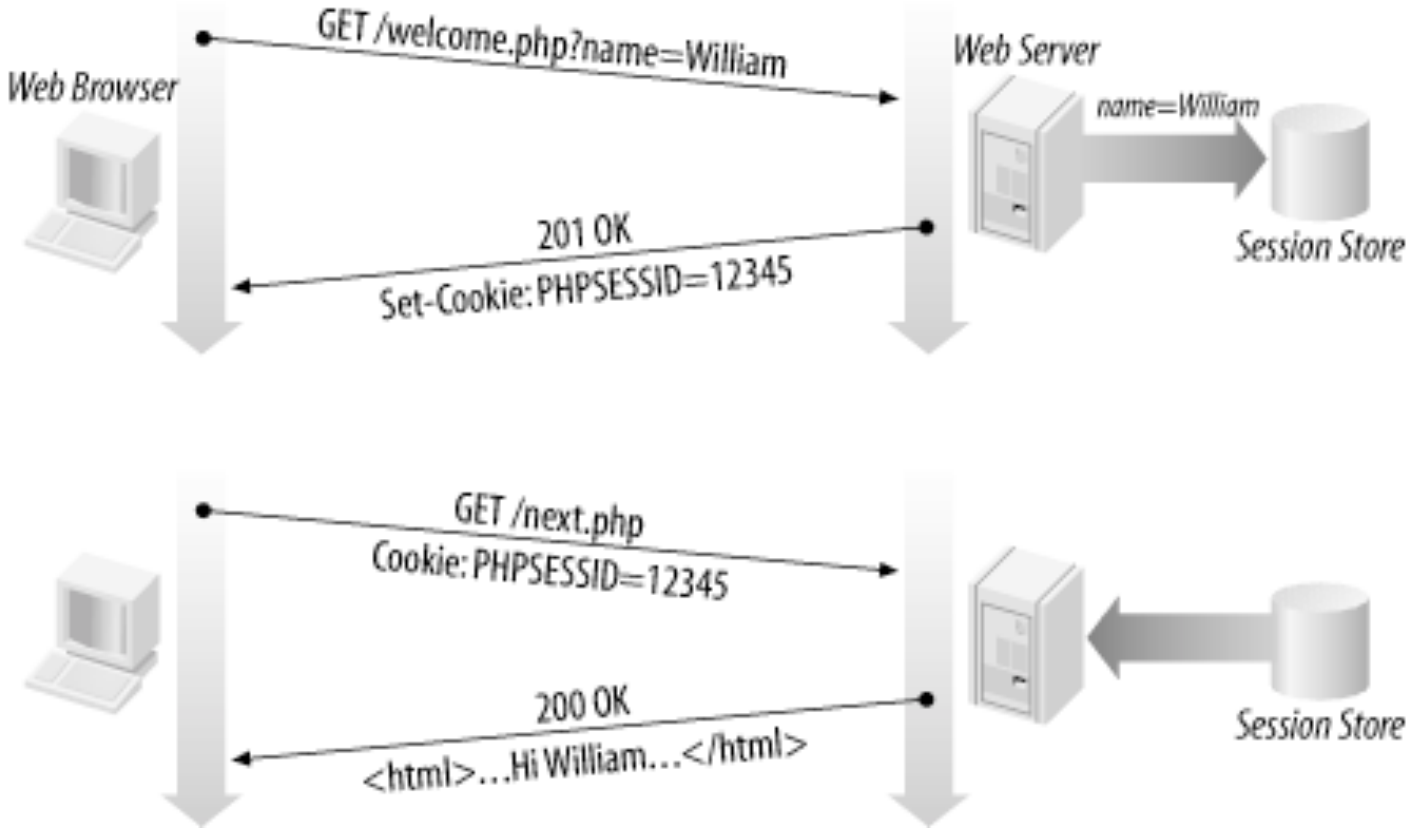
Tasks to be done

3. landing page for new user
    1. add new user to database

How cookies are sent



How sessions are established



functions.php start a session


```
session_save_path("/var/www/html/nat/login/sessions/");
if (!isset($_SESSION)) { session_start(); }
```

Index.php login form


```
<div id="login">
    <form action="home.php" method="post">
        <legend>Login</legend>
        <input type="text" name="username" value="User Name"/> <br/>
        <input type="password" name="password" value="Password"/> <br/>
        <input type="submit" value="Login"/>
    </form>
  </div>
```

home.php: log in or new userq


```
<?php
    $name = $_REQUEST["name"];
    $email = $_REQUEST[”email"];
    $uname = $_REQUEST["username"];
    $pw = $_REQUEST["password"];
    include("functions.php");
    ?>
```

```
<?php
    if (isset($_REQUEST["name"])) {
      new_user;
    } else {
      log_in($uname, $pw);
    }
    ?>
```

functions.php log_in


```
function log_in($uname, $password) {
  $db = open_users_database();
  $quoted_uname = $db->quote($uname);
  if (is_password_correct($uname, $password)) {
    $rows = $db->query("SELECT name FROM users WHERE username = $quoted_uname");
    if ($rows) {
      foreach ($rows as $row) {
        $name = $row["name"];
        $_SESSION["name"] = $name;
      }
}
  } else {
    redirect("index.php", "Incorrect password or username");
  }
}
```

functions: open_users_database


```
function open_users_database() {
  try {
    $db = new PDO("mysql:dbname=login;localhost","nat",”<mypassword>");
  } catch (PDOException $e) {
    print "Error!: " . $e->getMessage() . "<br/>";
    die();
  }
  return $db;
}
```

functions.php: is_password_correct


```
function is_password_correct($name, $password) {
  $db = open_users_database();
  $name = $db->quote($name);
  $rows = $db->query("SELECT password FROM users WHERE username = $name");
  if ($rows) {
    foreach ($rows as $row) {
      $correct_password = $row["password"];
      return $password === $correct_password;
    }
  } else {
    return FALSE;   # user not found
  }
}
```

functions.php: redirect


```
function redirect($url, $flash_message = NULL) {
  if ($flash_message) {
    $_SESSION["flash"] = $flash_message;
  }
  print $_SESSION["flash"];
  header("Location: $url");
  die();
}
```

Standup

Discuss questions with your Scrum Team

Quiz

Quiz

1. In PHP, connect to a database with:
   a. $d = new PDO(…)
   b. $d = connect(…)
   c. $d = new connection(…)
   d. $d = new DB(…)
2. In PHP, get information with:
   a. $r = $d->Select(…)
   b. $r = $d.Select(…)
   c. $r = Select($d, …)
   d. $r = $d->query(…)
3. In PHP, access the only 'name' with:
   a. $i = $r[0]["name"]
   b. $i = $r[0]
   c. $i = $r->data["name"]
   d. foreach ($i in r) {$i ["name"]}
4. In PHP, sanitize db inputs with:
   a. $s = '$input'
   b. $s = quote ($input)
   c. $s = $d->quote($input)
   d. $s = $d->sanitize($input)

And the answer is …

Quiz

1. In PHP, connect to a database with: a. $d = new PDO(…)

e.g., $d = new PDO(“mysql:dbname=imdb_small” , “jessica” , “guinness");

2. In PHP, get information with: d. $r = $d->query(…)

e.g., $r = $d->query("SELECT * FROM actors WHERE last_name LIKE 'Del%'");

3. In PHP, access the first ‘name’ with: d. foreach ($i in r) {$i [“name”]}

e.g. <?php foreach($rows as $row) {

echo ‘ First name:’ . $row[“first_name"] ?> . ‘’;

} ?>

4. In PHP, sanitize db inputs with: b. $s = quote ($input);

e.g., $s = quote($name);

SQL

Databases on Betaweb

SQL

- 13.1: Database Basics
- 13.2: SQL
- 13.3: Multi-table Queries
- **13.4: Databases and PHP**
- Not in book: Database Design

PHP Database query through PDO


```
name = new PDO("dbprogram:dbname=database;host=server", username, password);
$name->query("SQL query");

# connect to world database on local server
$db = new PDO("mysql:dbname=world;host=localhost", "traveler", "packmybags");
$db->query("SELECT * FROM countries WHERE population > 100000000;")
```

- PDO database library lets you connect to many different database programs
  - replaces older, less versatile functions like mysql_connect
- PDO object's query function returns rows that match a query

Result rows: query


```
$db = new PDO("mysql:dbname=world;host=localhost", "traveler", "packmybags");
$rows = $db->query("SELECT * FROM countries WHERE population >
100000000;");
foreach ($rows as $row) {
  do something with $row;
}
```

- query returns all result rows
  - As an associative array of [column name -> value]
  - E.g, $row["population"] is row's population column

A complete example


```
$db = new PDO("mysql:dbname=imdb_small", "jessica", "guinness");
$rows = $db->query("SELECT * FROM actors WHERE last_name LIKE 'Del%'");
foreach ($rows as $row) {
  ?>
  <li> First name: <?= $row["first_name"] ?>,
       Last name:  <?= $row["last_name"]  ?> </li>
  <?php
}
```

- First name: Benicio, Last name: Del Toro
- First name: Michael, Last name: Delano
- …

PDO object methods

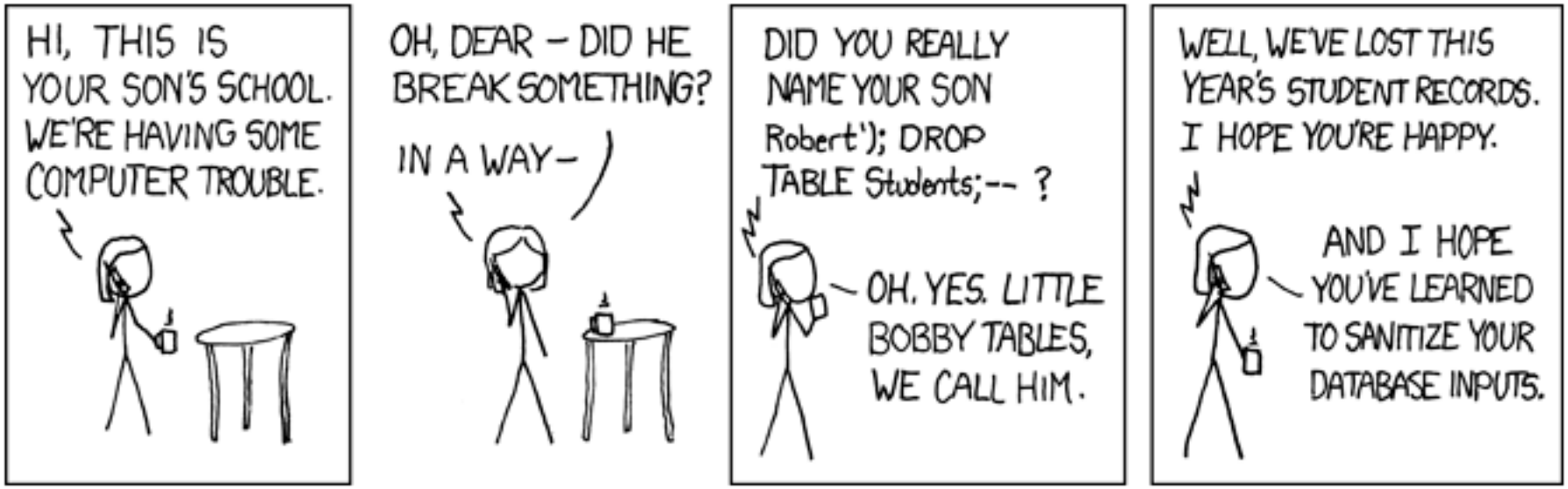
- query: performas a SQL SELECT query on the database
- exec: performs a SQL query that modifies the database (INSERT, DELETE, UPDATE, etc.)
- getAttribute: gets DB connection properties
- setAttribute: sets DB connection properties
- quote: encodes a value for use within a query

Including variables in a query


```
# get query parameter for name of movie
$title = $_GET["movietitle"];
$rows = $db->query("SELECT year FROM movies WHERE name = '$title'");
```

- Risky: do not directly include variables in a query because they might contain illegal characters or SQL syntax

Exploits of a Mom (xkcd.com)



Quoting variables


```
# get query parameter for name of movie
$title = $_GET["movietitle"];
$title = $db->quote($title);
$rows = $db->query("SELECT year FROM movies WHERE name = $title");
```

- call PDO's quote method on any variable to be inserted
- quote escapes any illegal chars and surrounds the value with ' quotes
- prevents bugs and security problems in queries containing user input

Database/query errors


```
$db = new PDO("mysql:dbname=imdb_small", "jessica", "guinness");
$rows = $db->query("SEEELECT * FROM movies WHERE year = 2000");   # FALSE
```

- database commands can often fail (invalid query; server not responding; etc.)
- normally, PDO commands fail silently by returning FALSE or NULL
- but this makes it hard to notice and handle problems

Exceptions for errors


```
$db = new PDO("mysql:dbname=imdb_small", "jessica", "guinness");
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$rows = $db->query("SEEELECT * FROM movies WHERE year = 2000");   # kaboom!
```

- using setAttribute, you can tell PDO to throw (generate) a PDOException when an error occurs
- the exceptions will appear as error messages on the page output
- you can catch the exception to gracefully handle the error

Catching an exception


```
try {
  statement(s);
} catch (ExceptionType $name) {
  code to handle the error;
}
```

- a try/catch statement attempts to run some code, but if it throws a given kind of exception, the program jumps to the catch block and runs that code to handle the error

Example with error checking


```
try {
  $db = new PDO("mysql:dbname=imdb_small", "jessica", "guinness");
  $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $rows = $db->query("SEEELECT * FROM movies WHERE year = 2000");
  foreach ($rows as row) { ... }
} catch (PDOException $ex) {
  ?>
  <p>Sorry, a database error occurred. Please try again later.</p>
  <p>(Error details: <?= $ex->getMessage() ?>)</p>
  <?php
}
```

PDOStatement methods

- The $rows returned by PDO's query method is an object of type PDOStatement with these methods:
  - columnCount() number of columns in the results
  - fetch() return the next row from the results
  - fetchColumn(*number*) return the next column from the results
  - rowCount() number of rows returned by the query

```
if ($db->rowCount() > 0) {
  $first_row = $db->fetch();
  ...
}
```

Database Design

- 13.1: Database Basics
- 13.2: SQL
- 13.3: Multi-table Queries
- 13.4: Databases and PHP
- **Not in book: Database Design**

Database design principles

- database design : the schema for a database
- database schema: description of table columns and constraints
- some database design principles:
  - keep it simple, stupid (KISS)
  - provide a unique identifier for every row
  - eliminate redundancy
    - redundancy allows inconsistency
  - favor integers for comparisons and repeated values

First database design

name	email	course	teacher	grade
Bart	bart@fox.com	Computer Science 142	Krabappel	B-
Bart	bart@fox.com	Computer Science 143	Hoover	C
Milhouse	milhouse@fox.com	Computer Science 142	Krabappel	B+
Lisa	lisa@fox.com	Computer Science 143	Hoover	A+
Lisa	lisa@fox.com	Computer Science 190M	Stepp	A+
Ralph	ralph@fox.com	Informatics 100	Krabappel	D+

- good: simple, can see all data in one place
- bad: redundancy (name, email, course repeated)
- bad: searches require string comparisons
- bad: no unique identifier for each row

Improved database design

students	id	name	email
	123	Bart	bart@fox.com
	456	Milhouse	milhouse@fox.com
	888	Lisa	lisa@fox.com

teachers	id	name
	1234	Krabappel
	5678	Hoover

courses	id	name	teacher_id
	10001	Computer Science 142	1234
	10002	Computer Science 143	5678

grades	student_id	course_id	grade
	123	10001	B-
	123	10002	C
	456	10001	B+
	888	10002	A+

Improvements

- normalizing: splitting tables to improve structure / redundancy (linked by unique IDs)
- **primary key**: a column guaranteed to be unique for each record (e.g. Lisa Simpson's ID 888)
- **foreign key**: a column in table A storing a primary key value from table B
  - (e.g. records in grades with student_id of 888 are Lisa's grades)

More Events and Validation

Page/window events

name	description
load	the browser loads the page
unload	the browser exits the page
resize	the browser window is resized
contextmenu	the user right-clicks to pop up a context menu
error	an error occurs when loading a document or an image

Page/window events


```
// best way to attach event handlers on page load
window.onload = function() { ... };
      document.observe("dom:loaded", function() {
      $("orderform").observe("submit", verify);
});
```

JS

Form events

event name	description
submit	form is being submitted
reset	form is being reset
change	the text or state of a form control has changed

```
window.observe("load", function() {
	$("orderform").observe("submit", verify);
});
function verify(event) {
	if ($("zipcode").value.length < 5) {
		event.stop(); // cancel form submission unless
	} // zip code is 5 chars long
}
```

JS

Prototype and forms


```
$("id")["name"]                                                    JS
```

- gets parameter with given name from form with given id

```
$F("id")                                                           JS
```

- $F returns the value of a form control with the given id

```
var name = $F("username");
if (name.length < 4) {
        $("username").clear();
        $("login").disable();
}                                                                  JS
```

Prototype and forms

- other form control methods:

activate	clear	disable	enable
focus	getValue	present	select

Client-side validation code


```
<form id="exampleform" action="http://foo.com/foo.php">
```
HTML

```
window.onload = function() {
	$("exampleform").onsubmit = checkData;
};
function checkData(event) {
	if ($("city").value == "" || $
("state").value.length != 2) {
		Event.stop(event);
		alert("Error, invalid city/state."); // show
error message
	}
}
```
JS

- forms expose onsubmit and onreset events
- to abort a form submission, call Prototype's Event.stop on the event

Regular expressions in JavaScript

- `string.match(regex)`
  - if string fits the pattern, returns the matching text; else returns null
  - can be used as a Boolean truthy/falsey test:

```
var name = $("name").value;
if (name.match(/[a-z]+/)) { ... }
```

- an i can be placed after the regex for a case-insensitive match
  - name.match(/Xenia/i) will match “xenia", “XeNiA", ...

Replacing text with regular expressions

- `string.replace(regex, "text")`
  - replaces the first occurrence of given pattern with the given text
  - `var str = “Xenia Mountrouidou";`

    `str.replace(/[a-z]/, "x") returns " Xxnia Mountrouidou"`
  - returns the modified string as its result; must be stored

    `str = str.replace(/[a-z]/, "x")`
- a g can be placed after the regex for a global match (replace all occurrences)
  - `str.replace(/[a-z]/g, "x")` returns “Xxxxx Mxxxxxxxxxxx"

Keyboard/text events

name	description
keydown	user presses a key while this element has keyboard focus
keyup	user releases a key while this element has keyboard focus
keypress	user presses and releases a key while this element has keyboard focus
focus	this element gains keyboard focus
blur	this element loses keyboard focus
select	this element's text is selected or deselected)

Key event objects

property name	description
keyCode rototype's key code constants	ASCII integer value of key that was pressed (convert to char with String.fromCharCode)
altKey, ctrlKey, shiftKey	true if Alt/Ctrl/Shift key is being held

Key event objects

Event.KEY_BACKSPACE	Event.KEY_DELETE	Event.KEY_DOWN	Event.KEY_END
Event.KEY_ESC	Event.KEY_HOME	Event.KEY_LEFT	Event.KEY_PAGEDOWN
Event.KEY_PAGEUP	Event.KEY_RETURN	Event.KEY_RIGHT	Event.KEY_TAB
Event.KEY_UP			

- issue: if the event you attach your listener to doesn't have the focus, you won't hear the event
  - possible solution: attach key listener to entire page body, outer element, etc.