CORSO DI LAUREA IN INFORMATICA

# Tecnologie Software per il Web

## REGULAR EXPRESSIONS

a.a. 2020-2021

## Regular expressions

- A **regular expression** is an object that describes a pattern of characters

- *Regular expressions are used to perform "pattern-matching" and "search-and-replace" functions on text*

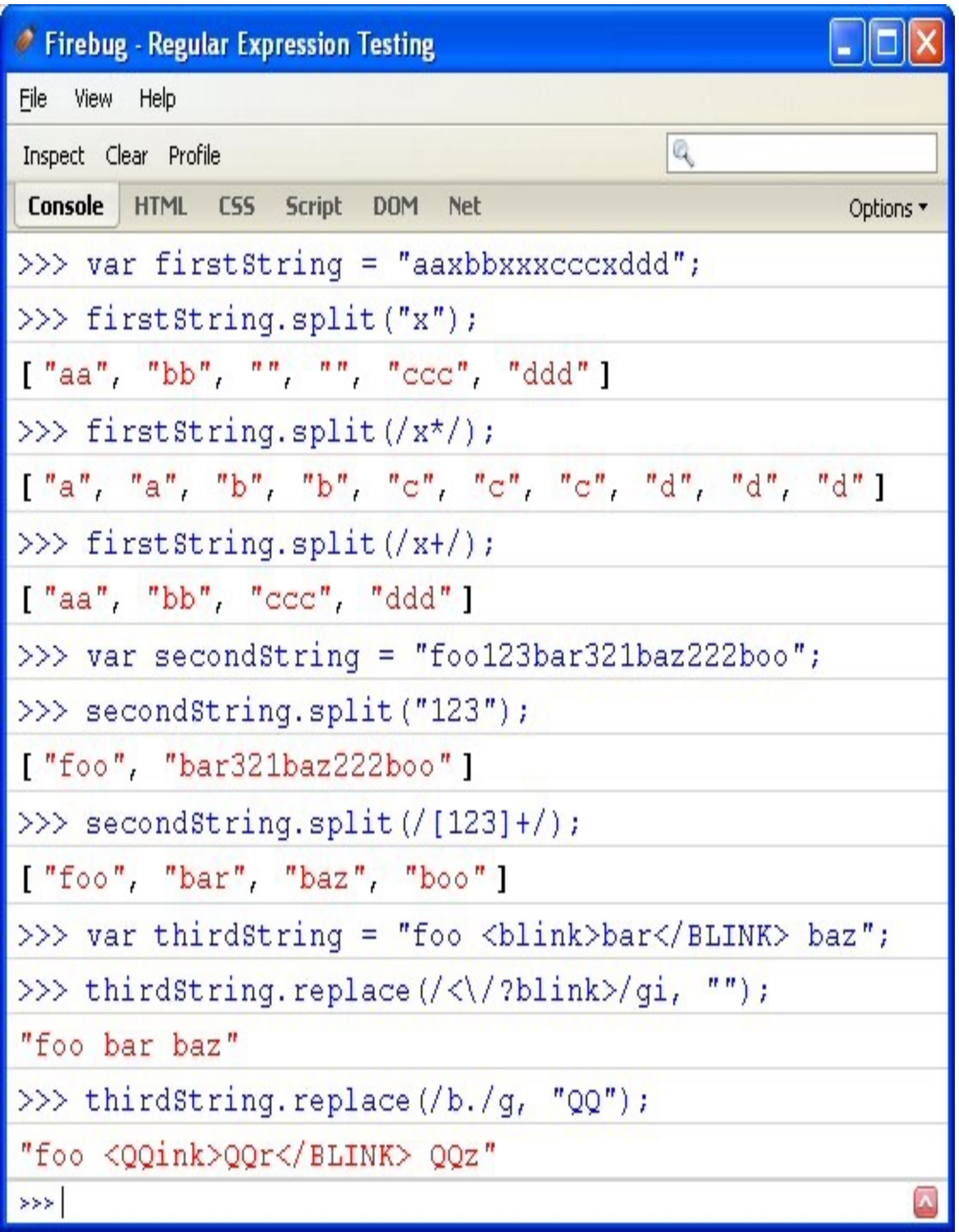
# Regular expressions: overview

- You specify a regex with /**pattern**/
  - *Not* with a string as in java and many other languages
- Most special characters same as in java/unix/perl
  - ^, $, . – beginning, end of string, any one char (except newline and line terminator)
  - \ – escape what would otherwise be a special character
  - *, +, ? – 0 or more, 1 or more, 0 or 1 occurrences
  - {n}, {n,}, {n, m} – exactly n, n or more occurrences, from n to m occurrences
  - [ ] – grouping
  - [^] – not in the group
  - \s, \S – whitespace, non-whitespace
  - \w, \W – word char (letter or number), non-word char
  - \d, \D – number, a non-digit character
  - (x | y) – any of the alternatives specified
  - ?=n, ?!n – any string followed by n, any string not followed by n

- Modifiers
  - /Pattern/**g** – do global matching (find all matches, not just first one)
  - /Pattern/**i** – do case-insensitive matching
  - /Pattern/**m** – do multiline matching

## String methods that use regular expressions

- Match
  - Returns array of parts of the string that match the regular expression
  - "A**x**b**xx**c**xxx**d".match(/x+/g) → ["x", "xx", "xxx"]
- Replace
  - Replaces all places that match the regular expression with a replacement string
  - "A**x**b**xx**c**xxx**d".replace(/x+/g, "q") → "A**q**b**q**c**q**d"
- Split
  - Returns array of all parts of the string that are in between the regular expressions
  - "**A**x**b**xx**c**xxx**d**".split(/x+/) → ["A", "b", "c", "d"]
- Search
  - Returns the position of the first place that matches the regular expression
  - "A**x**bxxcxxxd".search(/x+/) → 1


Firebug - Regular Expression Testing
File View Help
Inspect Clear Profile
Console HTML CSS Script DOM Net
Options

```
>>> var firstString = "aaxbbxxxcccxddd";
>>> firstString.split("x");
["aa", "bb", "", "", "ccc", "ddd"]
>>> firstString.split(/x*/);
["a", "a", "b", "b", "c", "c", "c", "d", "d", "d"]
>>> firstString.split(/x+/);
["aa", "bb", "ccc", "ddd"]
>>> var secondString = "foo123bar321baz222boo";
>>> secondString.split("123");
["foo", "bar321baz222boo"]
>>> secondString.split(/[123]+/);
["foo", "bar", "baz", "boo"]
>>> var thirdString = "foo <blink>bar</BLINK> baz";
>>> thirdString.replace(/<\/?blink>/gi, "");
"foo bar baz"
>>> thirdString.replace(/b./g, "QQ");
"foo <QQink>QQr</BLINK> QQz"
>>>
```

# Example: regular expressions

## Practical approach

- JavaScript code for validating user input

```
function validateInput(obj)
{
    var pattern = /.../;
    if(obj.value.match(pattern))  {
        //Do something: set class
        return true;
    }  else  {
        //Do something: set error class, focus, show error message, show suggestion,
        //show alert
        return false;
    }
}
```

Examples: validations.zip

## Example 1: username

- JavaScript code for validating user name

```
function allLetter(uname)
{
        var letters = /^[A-Za-z]+$/;
        if(uname.value.match(letters))
        {
                return true;
        }
        else
        {
                alert("Username must have alphabet characters only");
                uname.focus();
                return false;
        }
}
```

## Example 2: useraddress

```
Function alphanumeric(uadd)
{
        var letters = /^[0-9a-zA-Z]+$/;
        if(uadd.value.match(letters))
        {
                return true;
        }
        else
        {
                alert("User address must have alphanumeric characters only");
                uadd.focus();
                return false;
        }
}
```

# Example 3: email

```
function validateEmail(uemail)
{
    var mailformat = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
     if(uemail.value.match(mailformat))
    {
              return true;
     }
     else
   {
              alert("You have entered an invalid email address!");
              uemail.focus();
              return false;
     }
}
```

## Example 4: phone number (1)

- At first we validate a phone number of 10 digits with no comma, no spaces, no punctuation and there will be no + sign in front the number. Simply the validation will remove all non-digits and permit only phone numbers with 10 digits

```
function phonenumber(inputtxt)
{
  var phoneno = /^\d{10}$/;
  if((inputtxt.value.match(phoneno))
  {
      return true;
  }
  else
   {
        alert("The numeric input is not valid");
        return false;
   }
}
```

# Example 4: phone number (2)

To valid a phone number like

- (XXX)-XXX-XXXX
- (XXX).XXX.XXXX
- (XXX) XXX XXXX

use the following code

```
function phonenumber(inputtxt)
{
  var phoneno = /^\(([0-9]{3})\)[-\.\s]([0-9]{3})[-\.\s]([0-9]{4})$/;
  if((inputtxt.value.match(phoneno))
  {
      return true;
  }
  else
  {
        alert("The phone number is not valid");
        return false;
   }
}
```

## Examples (with test)

```
var dateTime = /\d{1,2}-\d{1,2}-\d{4} \d{1,2}:\d{2}/;
console.log(dateTime.test("30-5-2017 11:25"));  // → true
console.log(dateTime.test("30-5-2017 11:5"));   // → false

var currency = /^\$[0-9][0-9\,]*(\.\d{1,2})?$|^\$[\.]([\d][\d]?)$/;
console.log(currency.test("$100"));      // → true
console.log(currency.test("$95.33"));    // → true
console.log(currency.test("$.33"));      // → true
console.log(currency.test("90"));        // → false
```

## match vs. test

*regexObject*.**test**( *String* )

> Executes the search for a match between a regular expression and a specified string. Returns *true* or *false*.

*string*.**match**( *RegExp* )

> Used to retrieve the matches when matching a string against a regular expression. Returns an array with the matches or `null` if there are none.

Since `null` evaluates to `false`

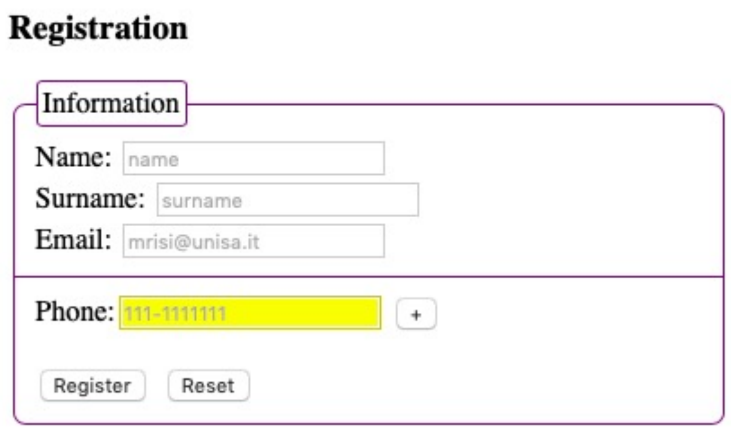
# More information on regular expressions

- JavaScript RegExp Reference

  https://www.w3schools.com/jsref/jsref_obj_regexp.Asp

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.
PERL!
TAP TAP

# Dynamic form and validation (DynamicForm.zip)



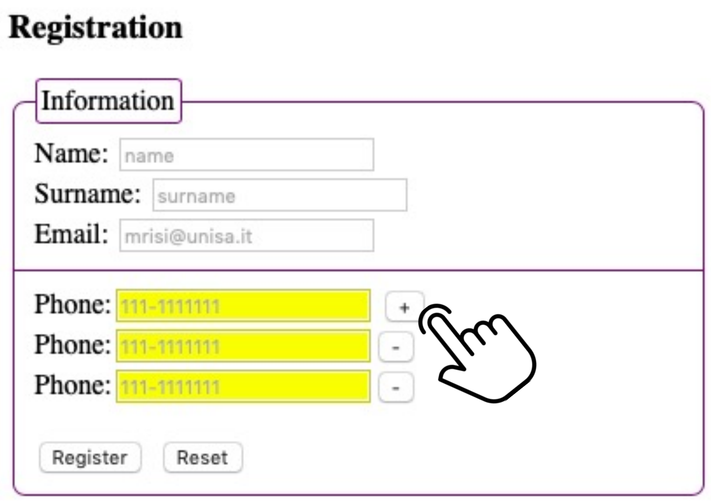
**Registration**

Information

Name: name
Surname: surname
Email: mrisi@unisa.it

Phone: 111-1111111 +

Register Reset



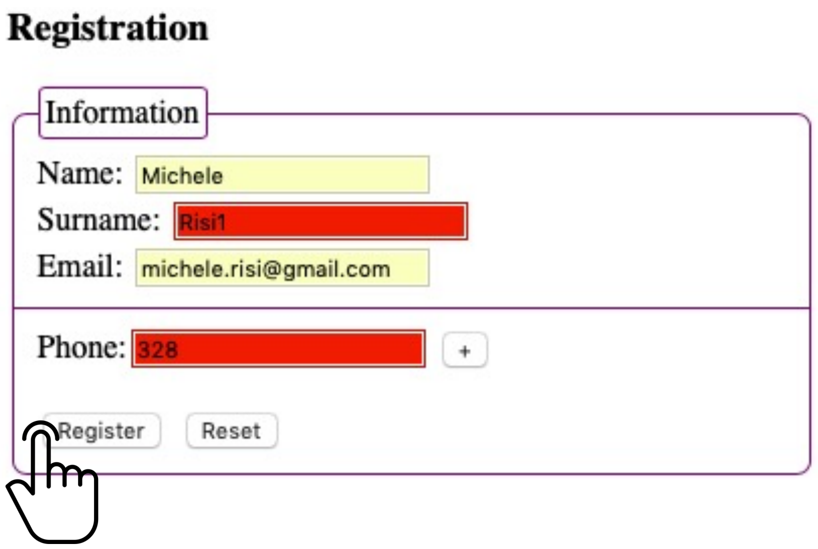
**Registration**

Information

Name: name
Surname: surname
Email: mrisi@unisa.it

Phone: 111-1111111 +
Phone: 111-1111111 -
Phone: 111-1111111 -

Register Reset



**Registration**

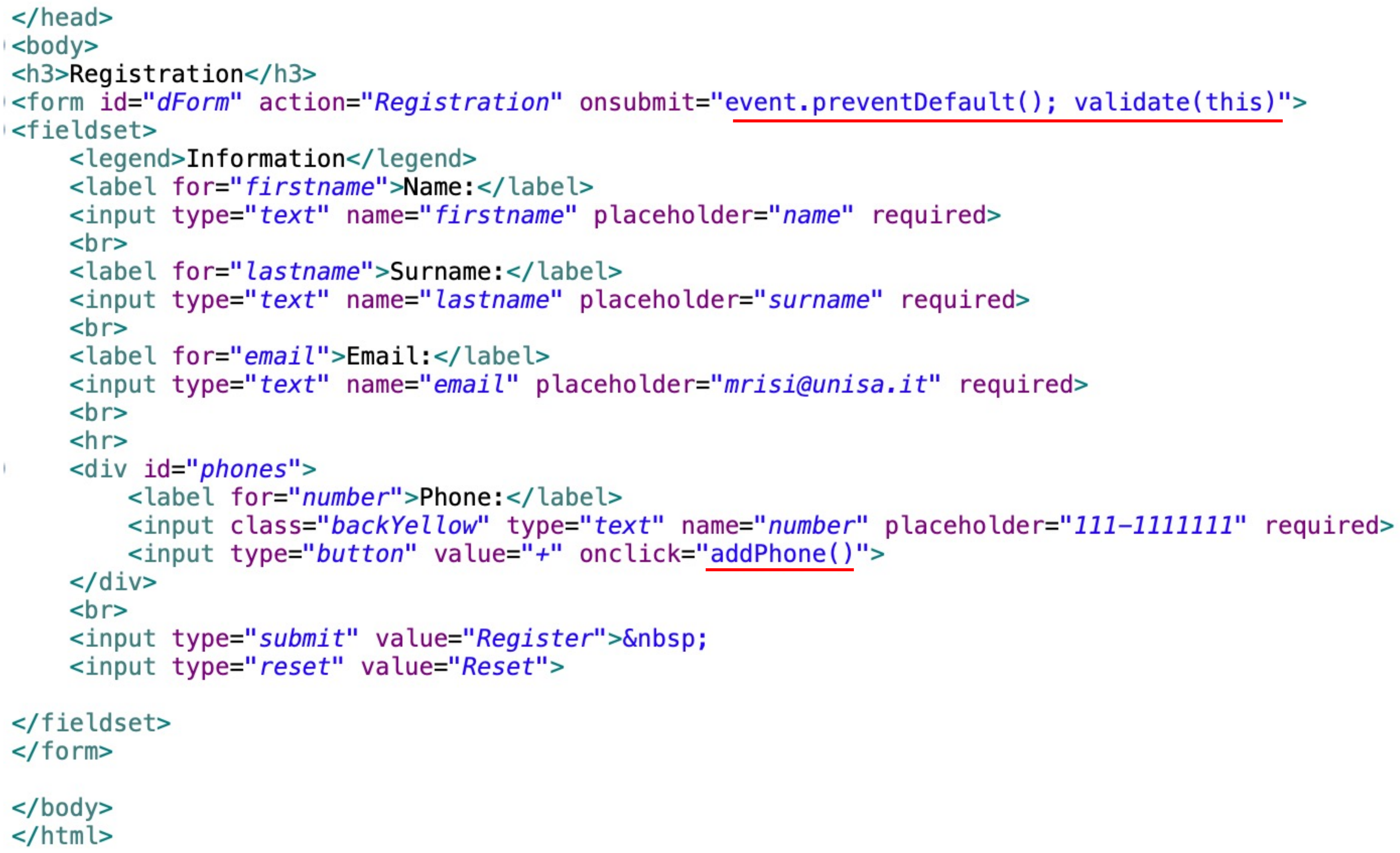
Information

Name: Michele
Surname: Risi1
Email: michele.risi@gmail.com

Phone: 328 +

Register Reset

# The form



```html
</head>
<body>
<h3>Registration</h3>
<form id="dForm" action="Registration" onsubmit="event.preventDefault(); validate(this)">
<fieldset>
    <legend>Information</legend>
    <label for="firstname">Name:</label>
    <input type="text" name="firstname" placeholder="name" required>
    <br>
    <label for="lastname">Surname:</label>
    <input type="text" name="lastname" placeholder="surname" required>
    <br>
    <label for="email">Email:</label>
    <input type="text" name="email" placeholder="mrisi@unisa.it" required>
    <br>
    <hr>
    <div id="phones">
        <label for="number">Phone:</label>
        <input class="backYellow" type="text" name="number" placeholder="111-1111111" required>
        <input type="button" value="+" onclick="addPhone()">
    </div>
    <br>
    <input type="submit" value="Register">&nbsp;
    <input type="reset" value="Reset">

</fieldset>
</form>

</body>
</html>
```

## CSS

```
<style>
    body {
        width: 400px;
        margin: 0 auto;
    }
    legend {
        padding: 3px;
        border: 1px solid purple;
        border-radius: 3px;
    }
    fieldset {
        border: 1px solid purple;
        border-radius: 7px;
    }
    .backYellow {
        background-color: yellow;
    }
    .error {
        background-color: red;
    }
    hr {
        border: 0;
        border-top: 1px solid purple;
        -webkit-margin-start: -0.8em;
        -webkit-margin-end: -0.8em;
    }
</style>
```

## Regular expressions

```
function checkNamesurname(inputtxt) {
    var name = /^[A-Za-z]+$/;
    if(inputtxt.value.match(name))
        return true

    return false;
}

function checkEmail(inputtxt) {
    var email = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
    if(inputtxt.value.match(email))
        return true;

    return false;
}

function checkPhonenumber(inputtxt) {
    var phoneno = /^([0-9]{3}-[0-9]{7})$/;
    if(inputtxt.value.match(phoneno))
        return true;

    return false;
}
```

## Validation

```
function validate(obj) {
    var valid = true;

    var name = document.getElementsByName("firstname")[0];
    if(!checkNamesurname(name)) {
        valid = false;
        name.classList.add("error");
    } else {
        name.classList.remove("error");
    }

    var surname = document.getElementsByName("lastname")[0];
    if(!checkNamesurname(surname)) {
        valid = false;
        surname.classList.add("error");
    } else {
        surname.classList.remove("error");
    }

    var email = document.getElementsByName("email")[0];
    if(!checkEmail(email)) {
        valid = false;
        email.classList.add("error");
    } else {
        email.classList.remove("error");
    }

    var numbers = document.getElementsByName("number");
    for(var i=0; i < numbers.length; i++) {
        if(!checkPhonenumber(numbers[i])) {
            valid = false;
            numbers[i].classList.add("error");
        } else  {
            numbers[i].classList.remove("error");
        }
    }

    if(valid) obj.submit();
}
```

# Add/Remove phone input fields

```
var count = 2;

function addPhone() {
    var container = document.getElementById("phones");

    var divv = document.createElement("div");
    divv.id = "id"+count;
    count++;

    var label = document.createElement("label");
    label.htmlFor = "number";
    label.appendChild(document.createTextNode("Phone:"));
    divv.appendChild(label);

    var element = document.createElement("input");
    element.type = "text";
    element.name = "number";
    element.placeholder = "111-1111111";
    element.required = "required";
    element.className = "backYellow";
    divv.appendChild(element);

    var input = document.createElement("input");
    input.type = "button";
    input.value = "-";
    input.addEventListener("click", function() {removePhone(divv.id)});
    divv.appendChild(input);

    container.appendChild(divv);
}

function removePhone(idd) {
    var element = document.getElementById(idd);
    element.parentNode.removeChild(element);
}
```

# Other resources

- Form with Multiple Steps
  - https://www.w3schools.com/howto/howto_js_form_steps.asp
- Autocomplete
  - https://www.w3schools.com/howto/howto_js_autocomplete.asp
- Modal Login Form
  - https://www.w3schools.com/howto/howto_css_login_form.asp
- Checkout Form
  - https://www.w3schools.com/howto/howto_css_checkout_form.asp
- Form with Icons
  - https://www.w3schools.com/howto/howto_css_form_icon.asp
- Password Validation
  - https://www.w3schools.com/howto/howto_js_password_validation.asp
- …