

# Проверка на коректността на въведени данни — 11а, Икономическа информатика

## Увод

Коректността на данните е критична за надеждни системи. Валидацията трябва да е многослойна и последователна.

## Основни понятия

- **Валидация срещу верификация:**
  - **Валидация:** "Правим ли правилния продукт?" (потребителски изисквания)
  - **Верификация:** "Правим ли продукта правилно?" (технически изисквания)
- **Синтактична и семантична валидация:**
  - **Синтактична:** правилен формат (например валиден email)
  - **Семантична:** логичен смисъл (например възраст между 0 и 150)
- **Sanitization, маски за въвеждане** - пречистване и форматиране на данни

## Цели

- **Прилагане на клиентска и сървърна валидация** - двойна защита
- **Използване на вградени средства и регулярни изрази** - ефективни инструменти

- **Обратна връзка към потребителя и тестови случаи** - добро потребителско изживяване

## Съдържание

### Клиентска валидация (Web)

- **HTML атрибути:**
  - `required` - задължително поле
  - `min` , `max` - минимални/максимални стойности
  - `pattern` - регулярен израз за валидация
  - `minlength` , `maxlength` - дължина на текста
- **Типове input:**
  - `email` - автоматична валидация на email формат
  - `number` - само числа
  - `tel` - телефонен номер
  - `date` - календар за дата
  - `url` - валиден URL адрес
- **Constraint Validation API** и съобщения за грешки

### Практически пример:

## Сървърна валидация

- **Недоверие към клиента** - повтаряне на всички проверки на сървър
- **Съобщения за грешки** - ясни и полезни за потребителя
- **Локализация** - съобщения на подходящ език
- **Логиране** - записване на грешки за анализ
- **Бизнес правила:**
  - Уникалност (например уникален email)
  - Зависимости между полета (например парола и потвърждение)

## Регулярни изрази

- **E-mail:** `^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$`
- **Телефон (България):** `^(\\+359|0)[0-9]{9}$`
- **ЕГН:** `^[0-9]{10}$`
- **Гранични случаи и негативни тестове:**
  - Празни стойности
  - Прекалено дълги стойности
  - Специални символи
  - SQL injection опити

## Примери за регулярни изрази:

```
// ЕГН валидация
const egnRegex = /^[0-9]{10}$/;
if (!egnRegex.test(egn)) {
    throw new Error('ЕГН трябва да съдържа точно 10 цифри');
}
```

```
// Email валидация
const emailRegex = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
if (!emailRegex.test(email)) {
    throw new Error('Невалиден email формат');
}
```

## UX насоки

- **Инлайн грешки** - показване на грешки веднага до полето
- **Ясни подсказки** - обяснение на очаквания формат
- **Предотвратяване на дублиране на заявки** - деактивиране на бутона след натискане
- **Прогресивна валидация** - проверка докато потребителят пише
- **Положителна обратна връзка** - показване когато данните са правилни

## Пример (HTML)







```
<form>
  <input type="email" required placeholder="Въведете email" />
  <input type="tel" pattern="^[0-9\-\s]{6,15}$"
    title="Телефон с 6-15 цифри" />
  <input type="password" minlength="8" required
    placeholder="Минимум 8 символа" />
  <button type="submit">Изпрати</button>
</form>
```

## Примери и задачи

## Практически задачи:

- **Проектиране на форма за регистрация** с двустранна валидация
- **Списък от тестови случаи** за невалидни входи и очакван резултат
- **Валидация на файлове** - размер, тип, съдържание
- **Защита срещу XSS** - sanitization на потребителски вход

## Тестови случаи за валидация:

- **Email:**
  -  Валиден: user@example.com
  -  Невалиден: user@, @example.com, user.example.com
- **ЕГН:**
  -  Валиден: 1234567890 (10 цифри)
  -  Невалиден: 123456789 (9 цифри), 12345678901 (11 цифри)
- **Телефон:**
  -  Валиден: +359888123456, 0888123456
  -  Невалиден: 888123456, +3598881234567

## Заклучение

Комбинацията от клиентска и сървърна валидация намалява грешките и подобрява надеждността на системата.

## Ключови принципи:

- **Двойна защита** - валидация на клиент и сървър
- **Добро UX** - ясни съобщения и моментална обратна връзка
- **Безопасност** - защита срещу злонамерени входи
- **Тестване** - проверка на всички гранични случаи