

КАК Я ПЕРЕСТАЛ БОЯТЬСЯ И ПОЛЮБИЛ DOM API

Ярослав Сергеевский
@everdimension



github.com/everdimension
telegram: @everdimension
twitter: @everdimension

FORM VALIDATION

ACCESSIBILITY

ACCESSIBILITY

(A11Y)

SEMANTIC HTML

ЗАЧЕМ?

РАСПРОСТРАНЁННЫЕ ОШИБКИ

РАСПРОСТРАНЁННЫЕ ОШИБКИ

<div onClick={} />

**ЕСЛИ ВАМ ВСЁ РАВНО НА А11У,
ИСПОЛЬЗУЙТЕ CANVAS ДЛЯ ВЁРСТКИ**

**ЕСЛИ ВАМ ВСЁ РАВНО НА А11У,
ИСПОЛЬЗУЙТЕ CANVAS ДЛЯ ВЁРСТКИ**

(Seriously)



[Алексей Охрименко](#)

IPONWEB

[MoscowJS 31](#)

Верстка Canvas'ом

Зачем? Как и когда нужно создавать пользовательский интерфейс с помощью Canvas и как обогнать React Native — обо всем этом вы узнаете из этого доклада.

СЛАЙДЫ

ВИДЕО

AOM

(ACCESSIBILITY OBJECT MODEL)

<http://wicg.github.io/aom/>

РАСПРОСТРАНЁННЫЕ ОШИБКИ

РАСПРОСТРАНЁННЫЕ ОШИБКИ

```
<input  
  type="text"  
  onChange={this.handleUsernameChange}  
>
```

РАСПРОСТРАНЁННЫЕ ОШИБКИ

```
<input  
  type="text"  
  name="username"  
  onChange={this.handleUsernameChange}  
>
```

```
<input  
  type="text"  
  name="username"  
  onChange={this.handleChange}  
>
```


ИСПОЛЬЗУЙТЕ ПРАВИЛЬНЫЕ ПОЛЯ ВВОДА

```
<input type="email" />
```

```
<input type="tel" />
```

```
<input type="text" pattern="\d*" />
```

```
<input type="url" />
```

USE APPROPRIATE INPUTS

USE APPROPRIATE INPUTS

```
<input type="text" pattern="\d*" />
```

USE APPROPRIATE INPUTS

```
<input type="text" pattern="\d*" />
```



USE APPROPRIATE INPUTS

USE APPROPRIATE INPUTS

```
<input type="tel" />
```

USE APPROPRIATE INPUTS

```
<input type="tel" />
```



USE APPROPRIATE INPUTS

`<input type="tel" />`



USE APPROPRIATE INPUTS

USE APPROPRIATE INPUTS

```
<input type="url" />
```

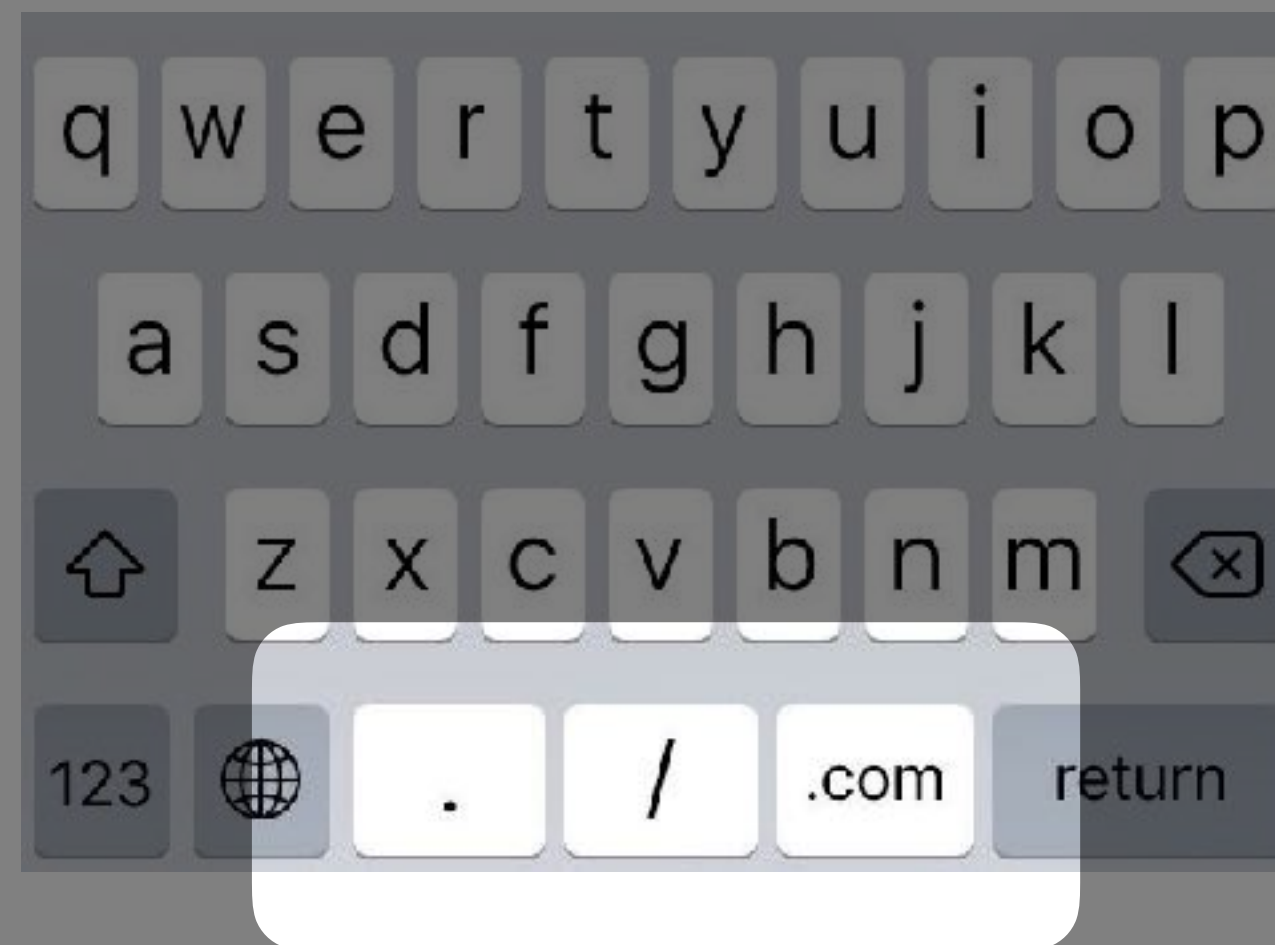
USE APPROPRIATE INPUTS

```
<input type="url" />
```



USE APPROPRIATE INPUTS

```
<input type="url" />
```



ЧТОБЫ ВНЕДРЯТЬ A11Y БЫЛО ПРОЩЕ,
РАЗРАБОТЧИК ДОЛЖЕН ХОТЕТЬ ПИСАТЬ
СЕМАНТИЧНЫЙ КОД

**ЛЮБАЯ ФОРМА
ЭТО KEY-VALUE DATA STRUCTURE**

ЛЮБАЯ ФОРМА — ЭТО KEY-VALUE DATA STRUCTURE

<form>

<input type="text" name="name" **>**

<input type="tel" name="phone" **>**

<input type="text" name="age" **>**

</form>

ЛЮБАЯ ФОРМА – ЭТО KEY-VALUE DATA STRUCTURE

```
<form>
```

```
  <input type="text"  name="name"  />
```

```
  <input type="tel"  name="phone"  />
```

```
  <input type="text"  name="age"  />
```

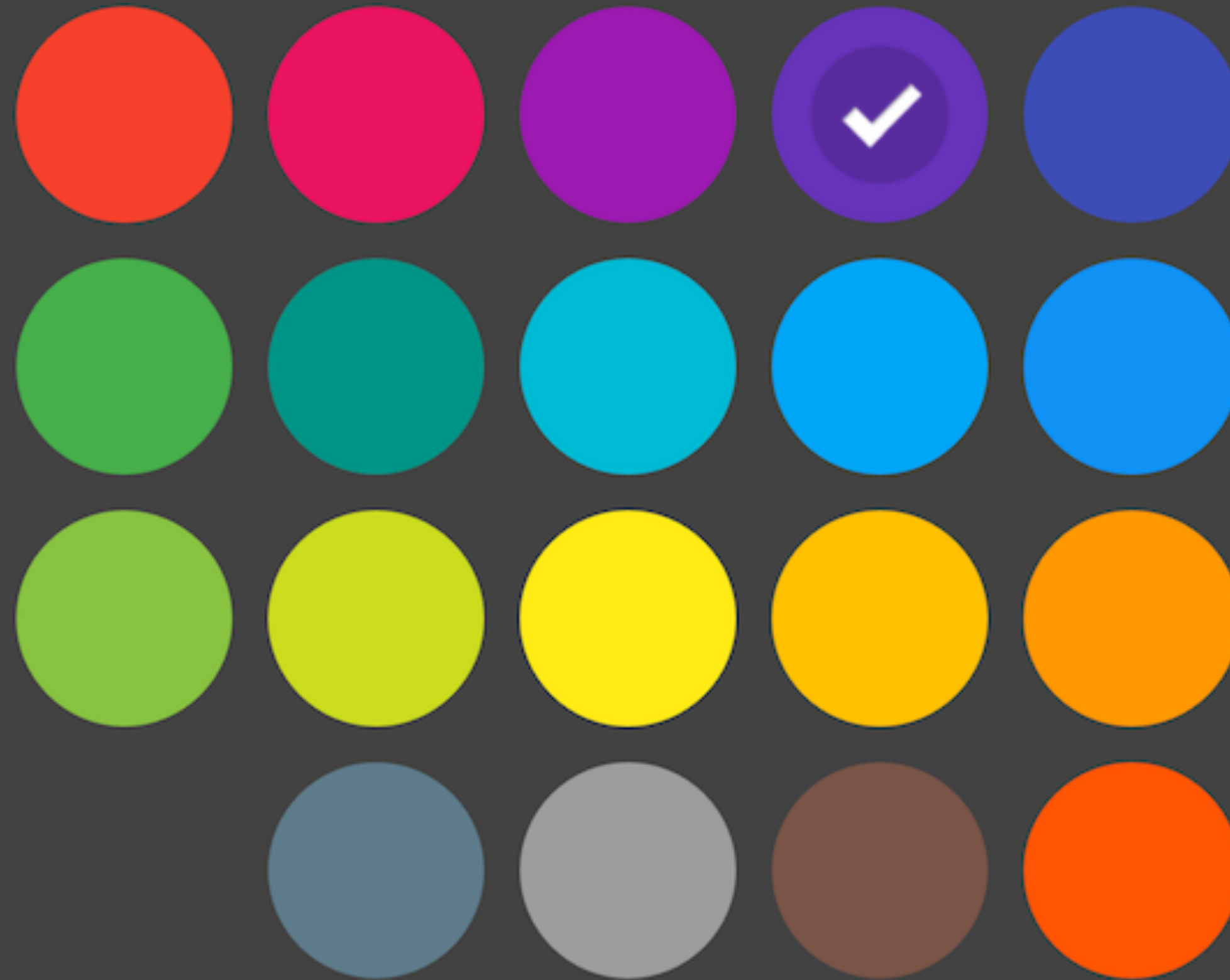
```
</form>
```


ЛЮБАЯ ФОРМА — ЭТО KEY-VALUE DATA STRUCTURE

```
<form>
  <input type="text" name="name" />
  <input type="tel" name="phone" />
  <input type="text" name="age" />
</form>
```

```
{
  name: 'john',
  phone: '+79031234567',
  age: 25,
}
```

Select your color:



CANCEL

OK

Select Date

< April 2016 >

SUN	MON	TUE	WED	THU	FRI	SAT
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

А ЗАЧЕМ ВАЛИДИРОВАТЬ?

А ЗАЧЕМ ВАЛИДИРОВАТЬ?

А ЗАЧЕМ ВАЛИДИРОВАТЬ?

- *Запретить отправку неполных данных на сервер*

А ЗАЧЕМ ВАЛИДИРОВАТЬ?

- ~~Запретить отправку неполных данных на сервер~~

А ЗАЧЕМ ВАЛИДИРОВАТЬ?

- ~~Запретить отправку неполных данных на сервер~~
- Помочь пользователю заполнить форму правильно

А ЗАЧЕМ ВАЛИДИРОВАТЬ?

- ~~Запретить отправку неполных данных на сервер~~
- Помочь пользователю заполнить форму правильно
 - Сразу показать ошибки и дать подсказки

А ЗАЧЕМ ВАЛИДИРОВАТЬ?

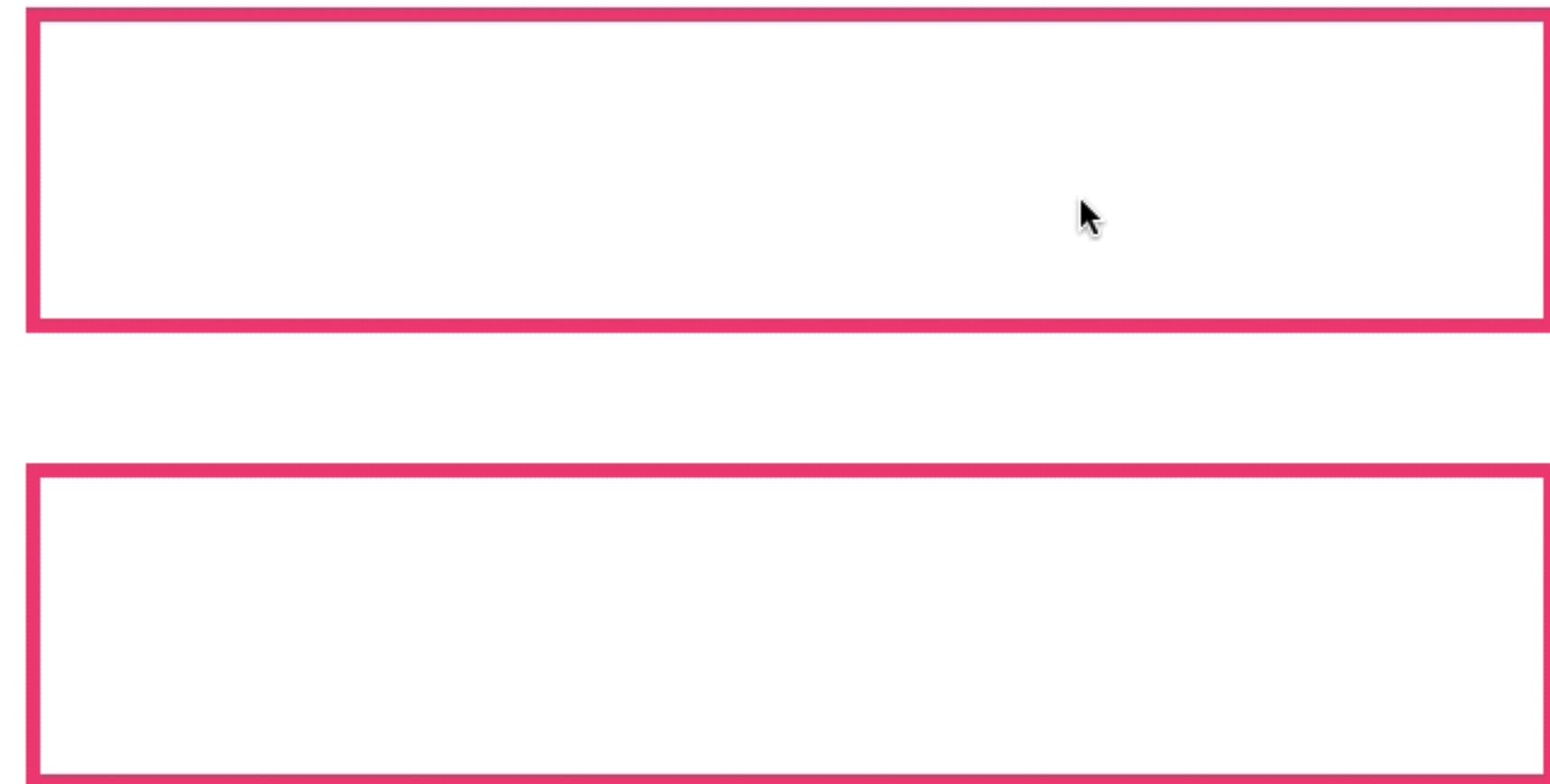
- ~~Запретить отправку неполных данных на сервер~~
- Помочь пользователю заполнить форму правильно
 - Сразу показать ошибки и дать подсказки
 - Не терять состояние, если пользователь попытался отправить неверные данные

ValidityState

```
{  
  browserslist: [  
    'someOldBrowser'  
  ]  
}
```

```
<form>
  <input
    name="name"
    type="text"
    required
  />
  <input
    name="phone"
    type="tel"
    required
  />
</form>
```

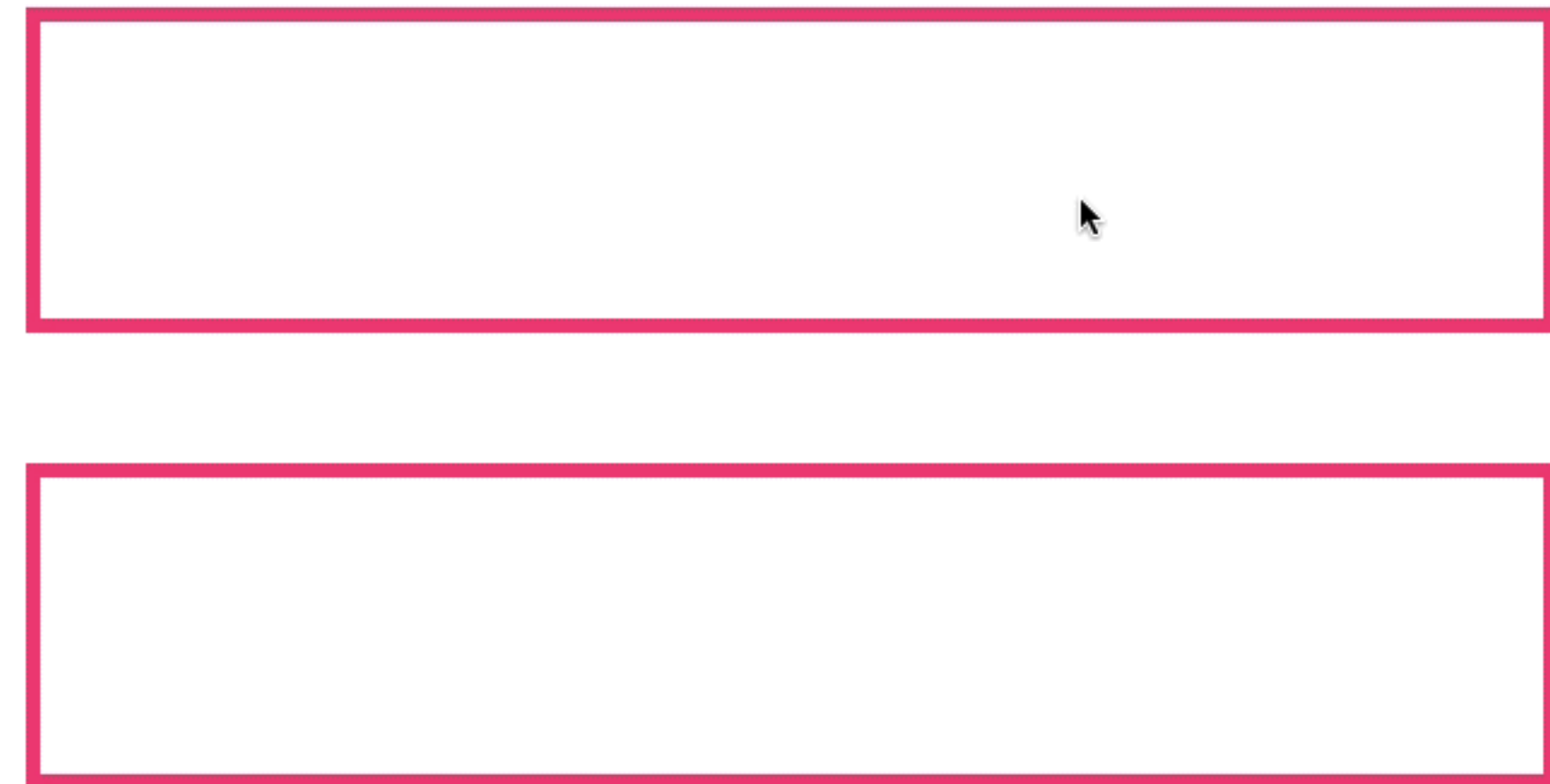
```
input:invalid {
  border: 2px solid red;
}
```



The image shows two empty text input fields stacked vertically. Both fields have a thick red border, which is the visual result of the CSS rule `input:invalid { border: 2px solid red; }` applied to the form elements. The top field is a standard text input, and the bottom field is a telephone input, both of which are currently empty and thus marked as invalid.

```
<form>
  <input
    name="name"
    type="text"
    required
  />
  <input
    name="phone"
    type="tel"
    required
  />
</form>
```

```
input:invalid {
  border: 2px solid red;
}
```



The image shows two empty text input fields stacked vertically. Both fields have a thick red border, which is the visual result of the CSS rule `input:invalid { border: 2px solid red; }` applied to the form elements. The top field is a standard text input, and the bottom field is a telephone input, both of which are currently empty and thus trigger the invalid state styling.

КАК ЭТО РАБОТАЕТ?

FORM

FORM

INPUT

FORM

INPUT

INPUT

FORM

INPUT

INPUT

```
interface Form {  
    checkValidity(): true | false  
}
```

FORM

INPUT

INPUT

```
interface Form {  
    checkValidity(): true | false  
}
```

```
interface Input {  
    validity: {  
        /* ... */  
        valid: true | false  
    }  
}
```

<form>

<input

name="name"

type="text"

required

>

<input

name="phone"

type="tel"

required

>

</form>

`form.checkValidity():` **false**

`input.validity.valid:` **false**

`input.validity.valid:` **false**

`form.checkValidity():` **false**

`input.validity.valid:` **false**

`input.validity.valid:` **false**

<form>

<input

name="name"

type="text"

required

>

<input

name="phone"

type="tel"

required

>

</form>

```
<form>
  <input
    name="name"
    type="text"
    required
  />
  <input
    name="phone"
    type="tel"
    required
  />
</form>
```

USE APPROPRIATE INPUTS

USE APPROPRIATE INPUTS

```
<input type="text" />
```

USE APPROPRIATE INPUTS

```
<input type="text" />
```

```
<input type="email" />
```

USE APPROPRIATE INPUTS

```
<input type="text" />
```

```
<input type="email" />
```

```
<input type="url" />
```

USE APPROPRIATE INPUTS

```
<input type="text" />
```

```
<input type="email" />
```

```
<input type="url" />
```

```
<input type="text" minlength="3" />
```

USE APPROPRIATE INPUTS

```
<input type="text" />
```

```
<input type="email" />
```

```
<input type="url" />
```

```
<input type="text" minlength="3" />
```


USE APPROPRIATE INPUTS

```
<input type="text" />
```

```
<input type="email" />
```

```
<input type="url" />
```

```
<input type="text" minlength="3" />
```

```
<input type="text" pattern=".{3,}" />
```

USE APPROPRIATE INPUTS

```
<input type="text" />
```

```
<input type="email" />
```

```
<input type="url" />
```

```
<input type="text" minlength="3" />
```

```
<input type="text" pattern=".{3,}" />
```

```
<input type="text" pattern=".{3,10}" />
```

USE APPROPRIATE INPUTS

```
<input type="text" />
```

```
<input type="email" />
```

```
<input type="url" />
```

```
<input type="text" minlength="3" />
```

```
<input type="text" pattern=".{3,}" />
```

```
<input type="text" pattern=".{3,10}" />
```

```
<input type="text" pattern="\S{3,}" />
```

INSTEAD OF THIS...

```
import Yup from 'yup';

const schema = Yup.object().shape({
  email: Yup.string()
    .email('Invalid email address')
    .required('Email is required!'),
  password: Yup.string()
    .required('Password is required'),
});
```

INSTEAD OF THIS...

```
import Yup from 'yup';
```

```
const schema = Yup.object().shape({  
  email: Yup.string()  
    .email('Invalid email address')  
    .required('Email is required!'),  
  password: Yup.string()  
    .required('Password is required'),  
});
```

```
/** Now write code to apply this schema.... */
```

...USE THIS

```
<form>  
  <input name="email" type="email" required />  
  <input name="password" type="password" required />  
</form>
```

```
render() {  
  return (  
    <form onSubmit={this.handleSubmit} noValidate>  
      { /* ... */ }  
    </form>  
  );  
}
```

```
handleSubmit(event) {
```

```
}
```

```
render() {
```

```
  return (
```

```
    <form onSubmit={this.handleSubmit} noValidate>
```

```
      { /* ... */ }
```

```
    </form>
```

```
  );
```

```
}
```



```
handleSubmit(event) {  
    event.preventDefault();
```

```
}
```

```
render() {  
    return (  
        <form onSubmit={this.handleSubmit} noValidate>  
            { /* ... */ }  
        </form>  
    );  
}
```

```
handleSubmit(event) {  
  event.preventDefault();  
  const form = event.target;
```

```
}
```

```
render() {  
  return (  
    <form onSubmit={this.handleSubmit} noValidate>  
      { /* ... */ }  
    </form>  
  );  
}
```

```
handleSubmit(event) {  
  event.preventDefault();  
  const form = event.target;  
  if (!form.checkValidity()) {  
  
  }  
}
```

```
render() {  
  return (  
    <form onSubmit={this.handleSubmit} noValidate>  
      { /* ... */ }  
    </form>  
  );  
}
```

```

handleSubmit(event) {
  event.preventDefault();
  const form = event.target;
  if (!form.checkValidity()) {

  }

}

render() {
  return (
    <form onSubmit={this.handleSubmit} noValidate>
      { /* ... */ }
    </form>
  );
}

```

```
handleSubmit(event) {  
  event.preventDefault();  
  const form = event.target;  
  if (!form.checkValidity()) {  
    return;  
  }  
}  
  
render() {  
  return (  
    <form onSubmit={this.handleSubmit} noValidate>  
      { /* ... */ }  
    </form>  
  );  
}
```

```

handleSubmit(event) {
  event.preventDefault();
  const form = event.target;
  if (!form.checkValidity()) {
    this.setState({ submitAttempted: true });
    return;
  }
}

render() {
  return (
    <form onSubmit={this.handleSubmit} noValidate>
      { /* ... */ }
    </form>
  );
}

```

```
handleSubmit(event) {  
  event.preventDefault();  
  const form = event.target;  
  if (!form.checkValidity()) {  
    this.setState({ submitAttempted: true });  
    return;  
  }  
  /* handle valid form */  
}
```

```
render() {  
  return (  
    <form onSubmit={this.handleSubmit} noValidate>  
      { /* ... */ }  
    </form>  
  );  
}
```


**ЧТО ЕСЛИ
ПОЛЯ ФОРМЫ ЗАВИСЯТ
ДРУГ ОТ ДРУГА?**

Password

enter password

Repeat Password

repeat password

submit

Password

enter password

Repeat Password

repeat password

submit

```
<form onSubmit={this.handleSubmit}>  
  <input  
    name="password"  
    type="password"  
    placeholder="enter password"  
    required  
  />  
  <input  
    name="repeatPassword"  
    type="password"  
    placeholder="repeat password"  
    required  
  />  
</form>
```

```
<form onSubmit={this.handleSubmit}>  
  <input  
    name="password"  
    type="password"  
    placeholder="enter password"  
    onChange={this.handleChange}  
    required  
  />  
  <input  
    name="repeatPassword"  
    type="password"  
    placeholder="repeat password"  
    required  
  />  
</form>
```

```
<form onSubmit={this.handleSubmit}>
  <input
    name="password"
    type="password"
    placeholder="enter password"
    onChange={this.handleChange}
    required
  />
  <input
    name="repeatPassword"
    type="password"
    placeholder="repeat password"
    pattern={this.state.password}
    required
  />
</form>
```

```
<form onSubmit={this.handleSubmit}>
  <input
    name="password"
    type="password"
    placeholder="enter password"
    onChange={this.handleChange}
    required
  />
  <input
    name="repeatPassword"
    type="password"
    placeholder="repeat password"
    pattern={escapeRegex(this.state.password)}
    required
  />
</form>
```

Password

enter password

Repeat Password

repeat password

submit

Password

enter password

Repeat Password

repeat password

submit

АСИНХРОННАЯ ВАЛИДАЦИЯ

Name

enter username

Email

enter email

submit

Name

enter username

Email

enter email

submit

FORM

INPUT

INPUT

```
interface Form {  
    checkValidity(): true | false  
}
```

```
interface Input {  
    validity: {  
        /* ... */  
        valid: true | false  
    }  
}
```

```
interface Input {  
  validity: {  
    /* ... */  
    valid: true | false  
  }  
}
```

```
interface Input {  
    setCustomValidity(message: string): void;  
    validity: {  
        /* ... */  
        valid: true | false  
    }  
}
```



```
input.setCustomValidity( 'checking' );
```

```
input.setCustomValidity( 'checking' );
```

```
input.setCustomValidity( 'checking' );
```

```
checkEmail(value)
```

```
input.setCustomValidity('checking');
```

```
checkEmail(value)  
  .then(() => input.setCustomValidity(''))
```

```
input.setCustomValidity('checking');
```

```
checkEmail(value)  
  .then(() => input.setCustomValidity(''))  
  .catch(() =>
```

```
input.setCustomValidity('checking');  
  
checkEmail(value)  
  .then(() => input.setCustomValidity(''))  
  .catch(() =>  
    input.setCustomValidity('alreadyTaken'));
```

```
<form>
  <label htmlFor="username">Name</label>
  <input name="username" type="text" required />

  <label htmlFor="email">Email</label>
  <input name="email" type="email" required />
</form>
```



```
import { EmailField } from './EmailField'
```

```
<form>
```

```
  <label htmlFor="username">Name</label>
```

```
  <input name="username" type="text" required />
```

```
  <label htmlFor="email">Email</label>
```

```
  <EmailField name="email" type="email" required />
```

```
</form>
```

```
import { EmailField } from './EmailField'
```

```
<form>
```

```
  <label htmlFor="username">Name</label>
```

```
  <input name="username" type="text" required />
```

```
  <label htmlFor="email">Email</label>
```

```
  <EmailField name="email" type="email" required />
```

```
</form>
```

Name

enter username

Email

enter email

submit

Name

enter username

Email

enter email

submit

**COMMON ADVICE: DON'T
VALIDATE EMAILS WITH
REGEX**

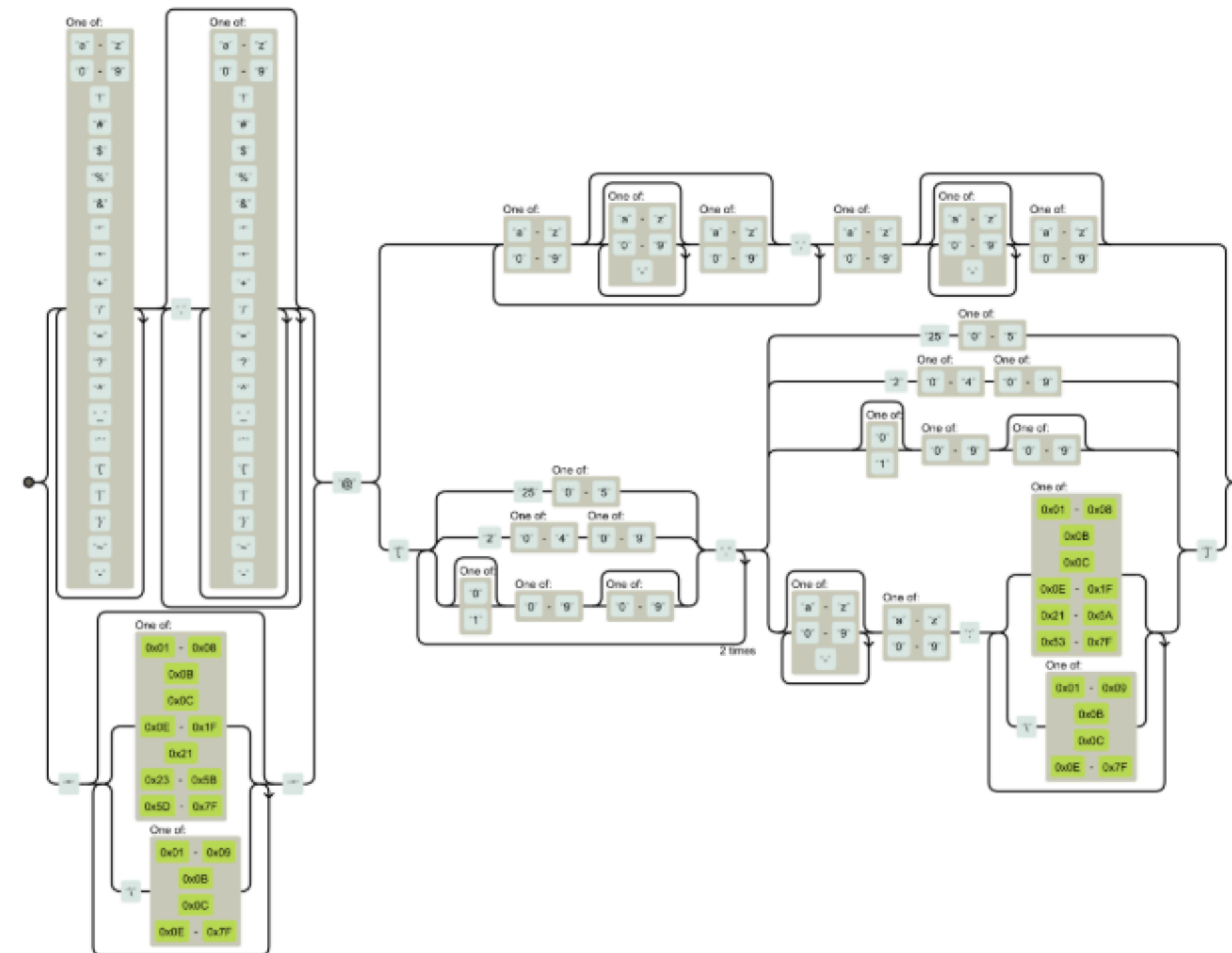
General Email Regex (RFC 5322 Official Standard)

```
(?:[a-z0-9!#$%&'*/+=?^_`{|}~-]+(?:\.[a-z0-9!#$%&'*/+=?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:(25[0-5]|2[0-4][0-9]|01[0-9]?[0-9]
[0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|01[0-9]?[0-9][0-9]?)|
[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\\])
```

General Email Regex (RFC 5322 Official Standard)

```
(?:[a-z0-9!#$%&'*/=?^_`{|}~]+(?:\.[a-z0-9!#$%&'*/=?^_`{|}~]+)*)@"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\[\x01-\x09\x0b\x0c\x0e-\x7f\])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\[(?:(?:(25[0-5]|2[0-4][0-9]|01[0-9]?[0-9]?[0-9])?|\.){3}(?:25[0-5]|2[0-4][0-9]|01[0-9]?[0-9]?[0-9])?|[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\[\x01-\x09\x0b\x0c\x0e-\x7f\])+\)])
```

Railroad Diagram of Above Regex



```
function isValidEmail(str) {  
    // ? ...  
}
```



```
function isValidEmail(str) {
```

```
}
```

```
function isValidEmail(str) {  
    const input = document.createElement('input');  
  
}
```

```
function isValidEmail(str) {  
    const input = document.createElement('input');  
    input.type = 'email';  
  
}
```

```
function isValidEmail(str) {  
    const input = document.createElement('input');  
    input.type = 'email';  
    input.value = str;  
  
}
```

```
function isValidEmail(str) {  
    const input = document.createElement('input');  
    input.type = 'email';  
    input.value = str;  
    return ... ?  
}
```

```
function isValidEmail(str) {  
    const input = document.createElement('input');  
    input.type = 'email';  
    input.value = str;  
    return input.validity.valid;  
}
```

ValidityState

**ИМПЕРАТИВНЫЙ КОД ВСЕГДА
МОЖНО АБСТРАГИРОВАТЬ**

**ДАВАЙТЕ ПИСАТЬ
СЛОЖНЫЕ ВЕЩИ ПРОСТО**

THE END

DEMO AND LINKS:

bit.ly/2LcSWRB

.....

Ярослав Сергеевский
@everdimension



github.com/everdimension
telegram: @everdimension
twitter: @everdimension