

JavaScript Validated Form



Paul Cheney

SPARTAN DESIGN UNIVERSITY

spartandesignuniversity.com



Agenda



Compare

Start file

HTML structure

Logic

CSS

JavaScript



Form Validation

Browser Validation

Easy

Fast

Newer Browsers

JavaScript Validation


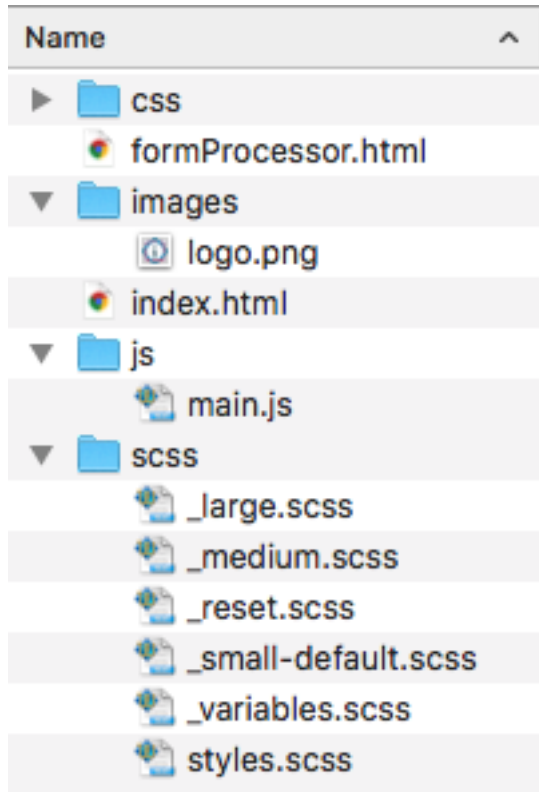
Control Feedback

Use Color

Older Browsers



Start File



search...

Register Agenda Contact Locations

Register

Full Name*

Cell Number*

Payment Method*

Credit Card Number*

Vehicle Preference*

Process My Rental

©2018 iConference • 987-555-1234

The form contains the following fields and options:

- Full Name*: Text input field.
- Cell Number*: Text input field containing "4325556789".
- Payment Method*: Radio button options for VISA, Master Card, and Discover.
- Credit Card Number*: Text input field containing "1478 5236 9874 1584".
- Vehicle Preference*: Dropdown menu with "Please Select" as the current selection.

HTML Structure

```
<section id="xxxxx" class="" >
```

```
<label>
```

```
<div>User Instructions*</div>
```


```
<input name="x" type="x" class="x" autocomplete="xxxx" >
```

```
<span> corrective feedback</span>
```

```
</label>
```

```
</section>
```



 Conference

search...

RegisterAgendaContactLocations

Register

Full Name*

Cell Number*

4325556789

Payment Method*

☐ VISA

☐ Master Card

☐ Discover

Credit Card Number*

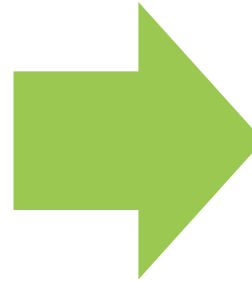
1478 5236 9874 1584


Vehicle Preference*

Please Select

Process My Rental

©2018 iConference • 987-555-1234



 Conference

search...

RegisterAgendaContactLocations

Register

There was a problem with your submission.
Errors have been highlighted below.

Full Name*

Please provide your full name

Cell Number*

4325556789

Phone number needs to be 10-15 digits

Payment Method*

☐ VISA

☐ Master Card

☐ Discover

Please select a payment method for your rental

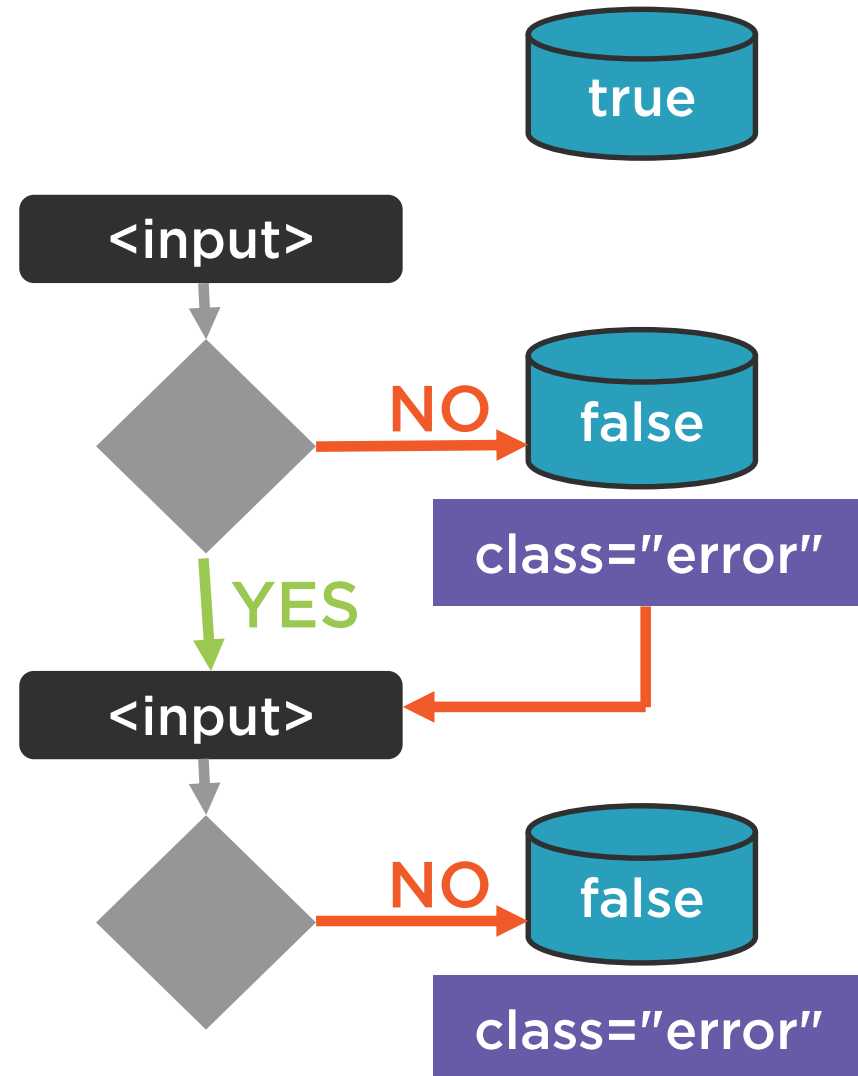
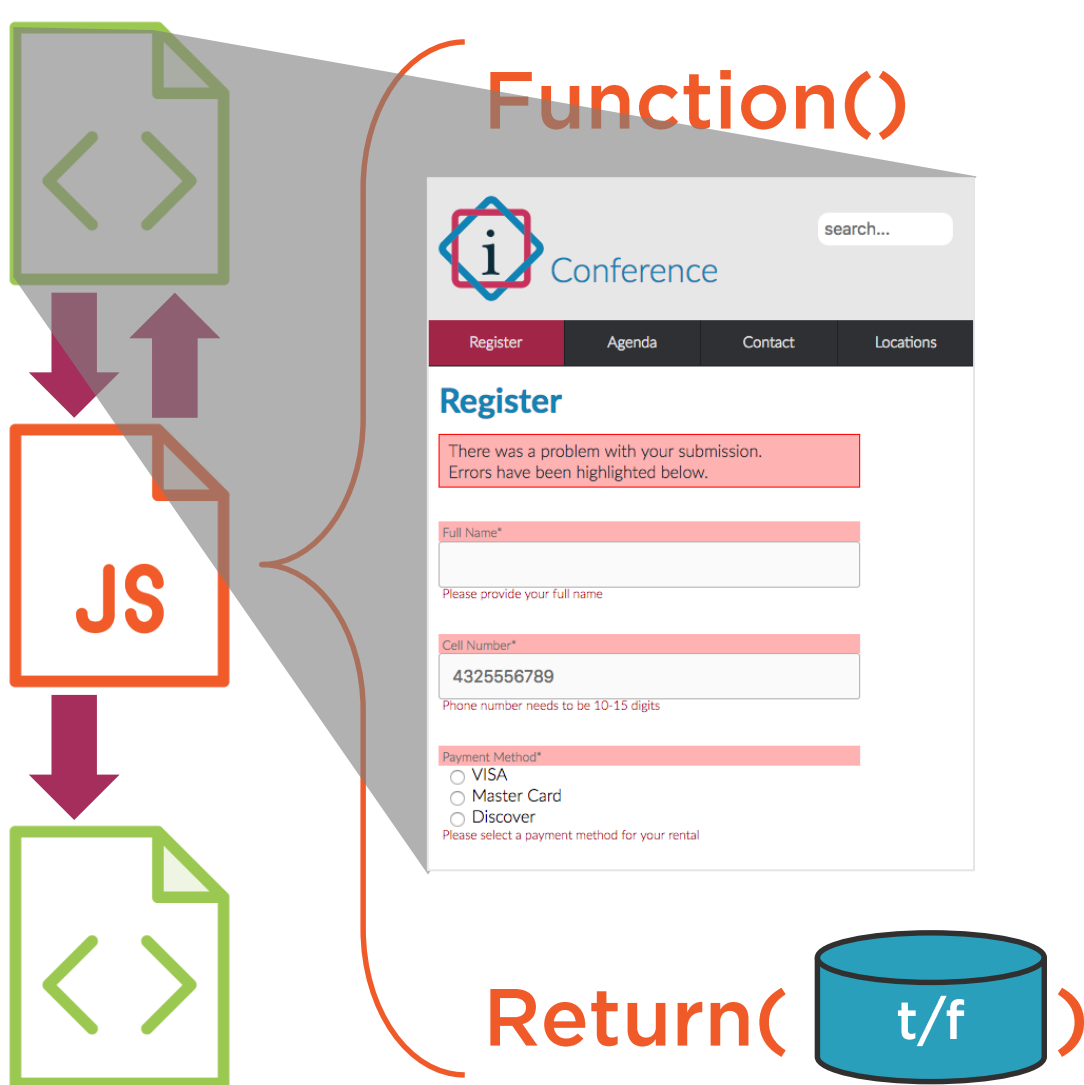
Form Processor (PHP or some X)Paul

file:///Users/paulcheney/Desktop/st... ☆

Now Procesing your Valid Form



JavaScript Logic





First we built the HTML with the errors messages in place and you can see them in the span tags below each input.

Next we styled the form to follow all our best practices. Notice that all the error messages are not showing. We hid the span tags with the css here.

Now we will manually add an error class to each section and style the error messages.

Now lets show the span tag when the class is set to error by adding section dot error span and display block.

Now you can see the instruction appearing below each input.

Lets style them by adding a font size of .75rem.

A color of red,

and finally indenting them a little with padding on the left side.

Now that looks pretty good but take it a step further.

Add section.error div which is the user instructions above each input. lets set that to a background color of red at 30%.

That may be enough but were not done yet. Lets add a message at the top of the page.

In the html on line 49 after the opening form tag, add the following code. YOu can copy and paste this from the snippets file in this module. Notice that the section has an id so we can control it from the javascript.



Now lets style this message.

We want it hidden by default so lets type section p and then display none. It should now be gone.

However, right now we are styling the page when errors are present so lets manually add a class of error in the html.

Now we can jump back to the small page and add section dot error p.

First we display as a block so it appears.

Then we add a red border

Fill the background with the same red at 30%

To push the text away from the border we will add padding .3rem and the variable gutter.

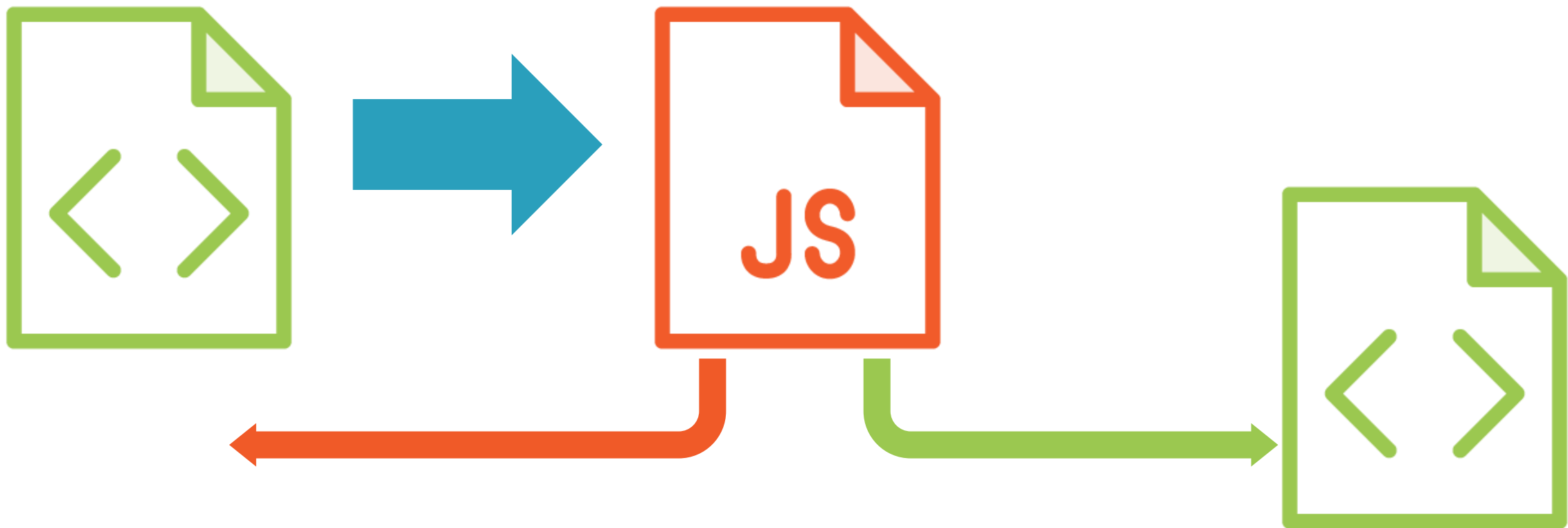
Now if you look at the results there is no way you can miss the fact that there are inputs that still need to be completed. There is also no question as to exactly what should be in each one.

Please remove all the error classes from the html and we will get started on the JavaScript.





Making a Connection



Lets open the form. The action if the form is completed is to show the formprocessor.html page which looks like this.

Lets add an onsubmit action which expects a return value and calls a yet to be created function called validateForm.

Now open the Javascript file

On line 4 create a function called validateForm. Remember to add a comment after the closing form curly so we can remember what it is for.

Add an alert with the message It Worked.

Refresh the web page and click the process button. You should see the alert pop up... when you dismiss it, you yee the processing page.

Please remove the alert.

Create a variable called status and set it to true.

Then below that, lets return the value of status back to the form.

With the value set to true, when we click the button, we see the second page

Now change the value to false.

Now no matter how much you click the process button, the form will not advance to the next page.



Lets code our first test of the full name field.

We will begin by creating a variable X we can use for all the form field tests.

Into that we will drill down to see what the user actually typed. We start with the document then look for the forms in the document. Then look for the form with a name of myForm. If we look at the html we can see that we have used name="myForm" inside the opening form tag.

Next we look for the input with a name of "name" and then we want the value of that input which is the information entered by the user. This is called dot notation.

Now we will make sure we did it correctly by sending the value of x to the console log.

Refresh the page and click the button.

We see a line number with no value. Why, because there is nothing typed in the full name input. Enter a name and press the button again. Now you see text appearing in the console. So that means it's working!

Return to the JS file and remove the console statement.

Lets add a conditional.

If (x === null || x === "") then lets put a false in the status variable.

Lets also change the class so the user knows there is a problem.

If we look at the html, we can see that the full name input is inside a section with an id of fullName.



In our Javascript file, type `document.getElementById(fullName)` dot. Now we will use `className equals...` to assign a class of error to the section element.

In order for this to work we need to change the default value of status to true up here. Then if the test fails, the value will be changed to false.

Save your work.

Now when we click the button, the form page does not advance and we can see the error styling appear for the full name.

If we inspect the section we can see that JavaScript has added the error class for us.

Try entering a name.

Now when you process the form it shows the second web page.



Phone Validation

Cell Number*

123-456-7890

Cell Number*

555-1234

Cell Number*

1234567890

Cell Number*

ext 1234

Cell Number*

123-456-7890 ext 23

Cell Number*

9859849399378973987398735

Cell Number*

1-123-456-7890



We will start with putting the entered phone number in the variable x just like before.

Then we will take the value of x and replace a dash with quote quote which means nothing. If we then add /g it means replace all dashes in the string.

Lets display that to the console and make sure it is working.

Enter a phone number with dashes and you can see in the console that the dashes have been removed. You can now remove the console statement.

Were now going to put our cleaned up phone number back into the... document . Forms . myForm . Phone... as the value which will replace the one entered by the user.

Lets test our work by adding a number with dashes and when we click the button, we can see the cleaned up number in the phone input.

The rest is very similar to the full name so copy the if statement from full name and past it below.

Lets change the condition to make sure the length of x is not less than 10 OR that the length of x is not greater than 15.

Check the html to see what the id is for this section.

Now change full name to phone number and lets give it a try.

Pressing the button generates the appropriate errors.

Adding a valid phone number allows you to advance to the next page.

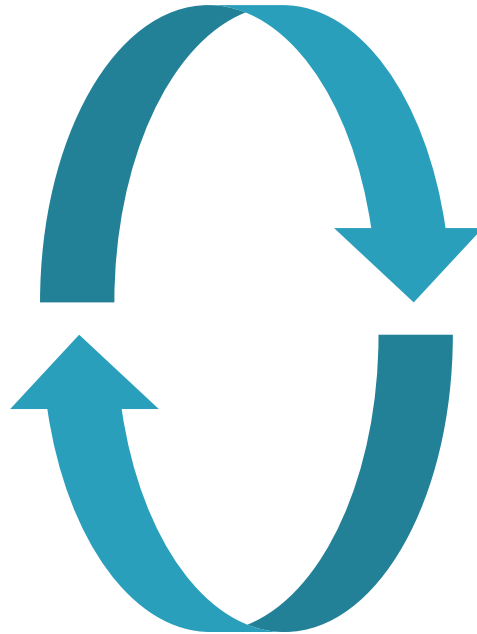




Radio Buttons

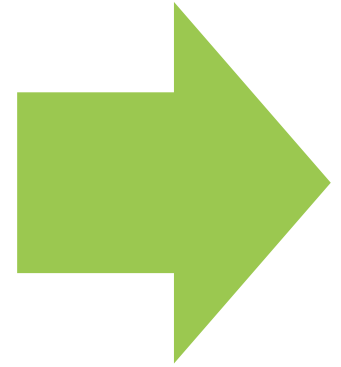
Payment Method*

- ☐ VISA
- ☐ Master Card
- ☐ Discover



Payment Method*

- ☐ VISA
- ☒ Master Card
- ☐ Discover



Payment Method*

- ☐ VISA
- ☐ Master Card
- ☐ Discover



Create a new section for the payment.

Create a new variable called `foundChecked` and set it to `false`. If we find that one of the radio buttons has been checked we will change the value to `true`.

If we look at our html we can see that the name of all three radio buttons is `payMethod`.

Let's set out variable `x` to `document dot get element by NAME (payMethod)`.

If we display this to the console we see a node list of three radio buttons.

We will create a new variable called `i` and use it for the loop.

Let's build a for loop that starts at 0... because it is an array... and ends at less than the length of the `x` array, and counts up by 1.

Remember to add a comment after the closing curly.

We will now check each radio button with an if statement.

Because `x` is now an array, we grab each element in the array using `x` followed by a number in square braces like this. Then we use dot notation to see if that element of the array has been checked.

If it has been checked then we change the value of this `foundChecked` variable from `false` to `true`.

Let's display the value of `foundChecked` to the console AFTER the closing "if" curly.



The first time through the loop, the variable `i` will be zero which means it will check the option for Visa. The second time the variable `i` will be increased by one and become 1. This will check the next item which is MasterCard. The third time through the loop `i` will again be increased by 1 and become a 2. It will then check the Discover button. When `i` is increased to three it is no longer less than three which is the length of the array so it will not loop any more.

Now lets try our code with the console log.

If nothing is checked then we see three false statements reported.

If the last one is checked then we see two false and one true

If the middle one is checked, we see a false and two trues. The reason we see true twice is because once the value has been changed to true, nothing changes it back to false.

Remember that our goal is to see if any one of the radio buttons are true so once we have found a true, we really don't need to check any more of them.

Lets return to the JS and add a break after we find one that is checked. Break means exit the loop.

Lets move the console log after the closing curly of the for loop.

If none are checked we see a false in the console log.

If any of the radio buttons are checked we see a true in the console.



Now that we have the ability to tell if any radio buttons are selected we can alter the value of the "status" variable if needed.

Once again this is almost identical to the if statement from phone so copy and paste below the console log statement. You can now remove the console log statement.

Our condition this time is foundChecked strictly equal to false

Our section in the html has an id of payment

Lets change this to payment

Check your work by pressing the button. You should see the red error message and the help text below.

If we fill in all three inputs correctly it advanced to the second page.

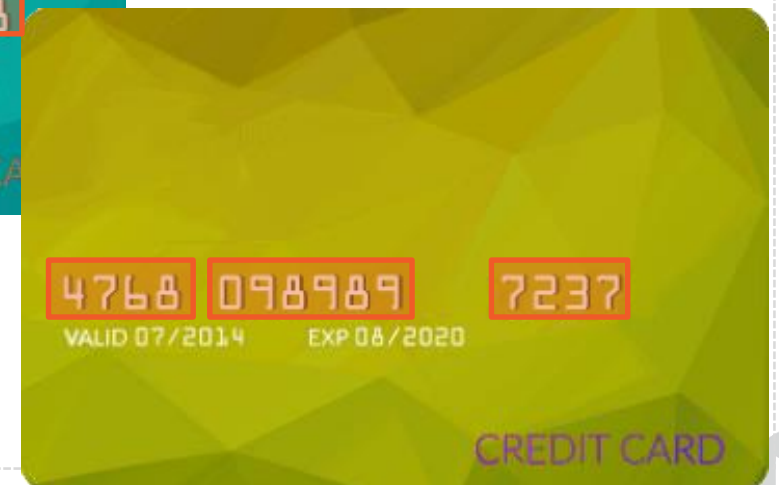
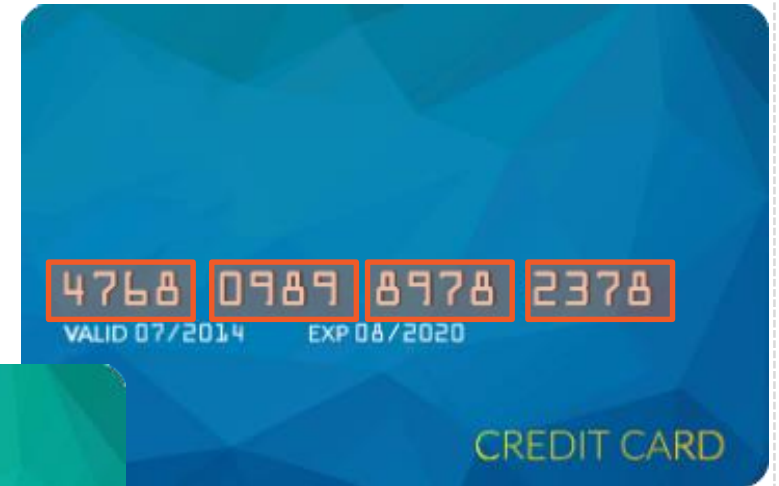
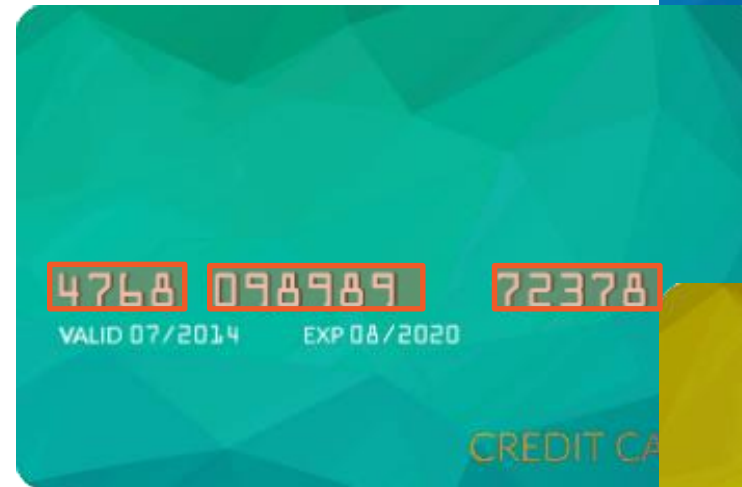




Credit Card Validation

Credit Card Issuer ▲	Length (Number of digits)
American Express	15
Diners Club - Carte Blanche	14
Diners Club - International	14
Diners Club - USA & Canada	16
Discover	16-19
InstaPayment	16
JCB	16-19
Maestro	16-19
MasterCard	16
Visa	13-16-19
Visa Electron	16

type="text"



The validation JavaScript for the credit card will be very similar to the phone number. Please copy that script and paste it down here.

Change the comment

Change the "phone" value here to "ccNumber".

This needs to match the name on the input.

Instead of replacing dashes, we will replace spaces so this should be a space here.

We are NOT going to replace the credit card value in the form because it makes it harder to verify if we remove the distinct blocks. We will just check the length of the variable x without the spaces to make sure that it is the correct length.

Remember that short credit cards are 14 and long credit cards are 19.

The id used in the html for the section tag is called ccNumber so put that here.

When we submit the form empty, the error messages are working.

If we submit a number that is too short we get an error.

If we submit a number that is too long we get an error.

A correct number does not show an error.



Now for the vehicle selection. This one is really easy compared to what we have done so far. Let me introduce you to "selected index".

Please set the value of X to `document.forms.myForm.vehicle.selectedIndex`;

Lets console the value of x

Notice that the option "Please Select" is a value of 0 and would not be considered a valid selection.

Choosing "convertible" displays a 1 to the console. So in reality any selection that is NOT 0 is a valid selection.

Please grab the if statement from the credit card and paste it down here.

Change the if to... x is strictly equalivant to zero

I so, the status changes to false.

Also change the ccNumber to vehicle... which is the id of that last section.

If we process without a selection we see the red errors. If make any selection and process the form, we do not get an error.





Now it's time to run this form through its paces and see what still needs to be fixed. If nothing is filled out, everything turns red and the form does not process. If everything is filled out, we see the second page.

So what if we miss one or two items. Only these items are highlighted in red,, which is good.

What happens if we then correct one of the items. Even though the entry is now valid we are still seeing red. What we need to do is use JavaScript to remove all the error classes before we test the form each time.

Please open the JS file.

Right after we create the variables, lets add a section for removing the error classes. We will create an array of all the id's of the required sections. If your form has non-required inputs, they would NOT be listed here.

Next we will loop through the list and remove the error class. We did the same thing when we checked the radio buttons.

Create a for loop starting at 0 and ending at less than the number of items and incrementing by one each time.

Next set the document dot get element by id of each item from this list to a class name of "" or empty.



Remember at the beginning of this module, we styled an error message at the top of the form page. We will show that using the section id of formProblems.

Just above the closing return... create an if with status strictly equal to false. Then set the document dot get element by id of formProblems and change the className to error.

Now each time we correctly complete an input and then submit... the error messages are removed for that item.



Register

Full Name*

Cell Number*

4325556789

Payment Method*

☐ VISA
☐ Master Card
☐ Discover

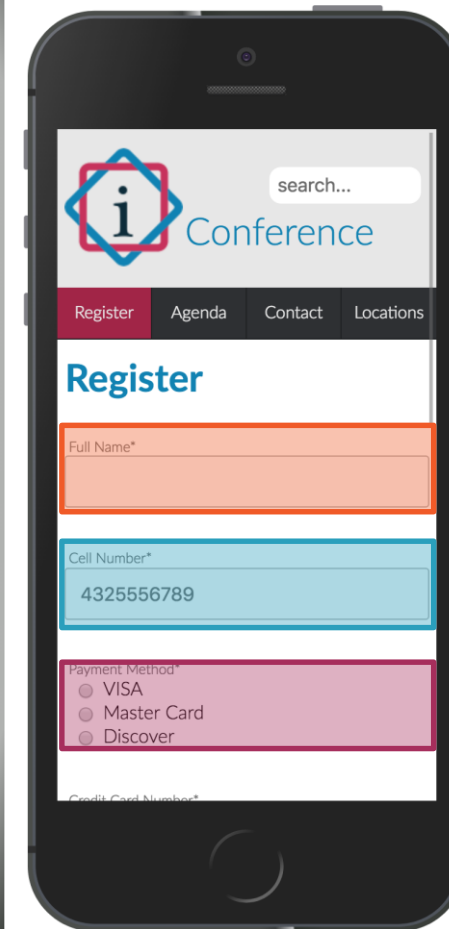
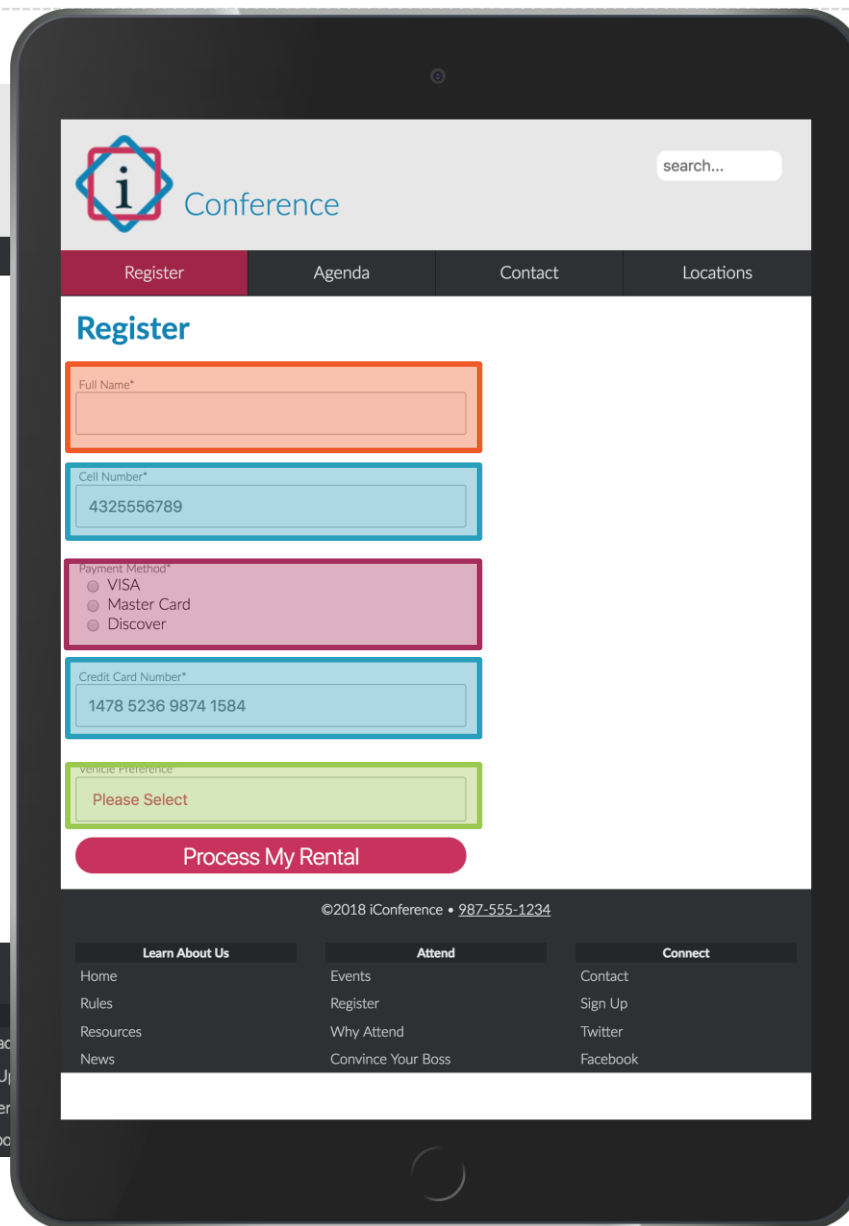
Credit Card Number*

1478 5236 9874 1584

Vehicle Preference*

Please Select

Process My Rental



Summary



Compare

Start file

HTML structure

Logic

CSS

JavaScript





Introduction

Web Form Best Practices

Simple Login Form

Long Web Form

JavaScript Validated Form

