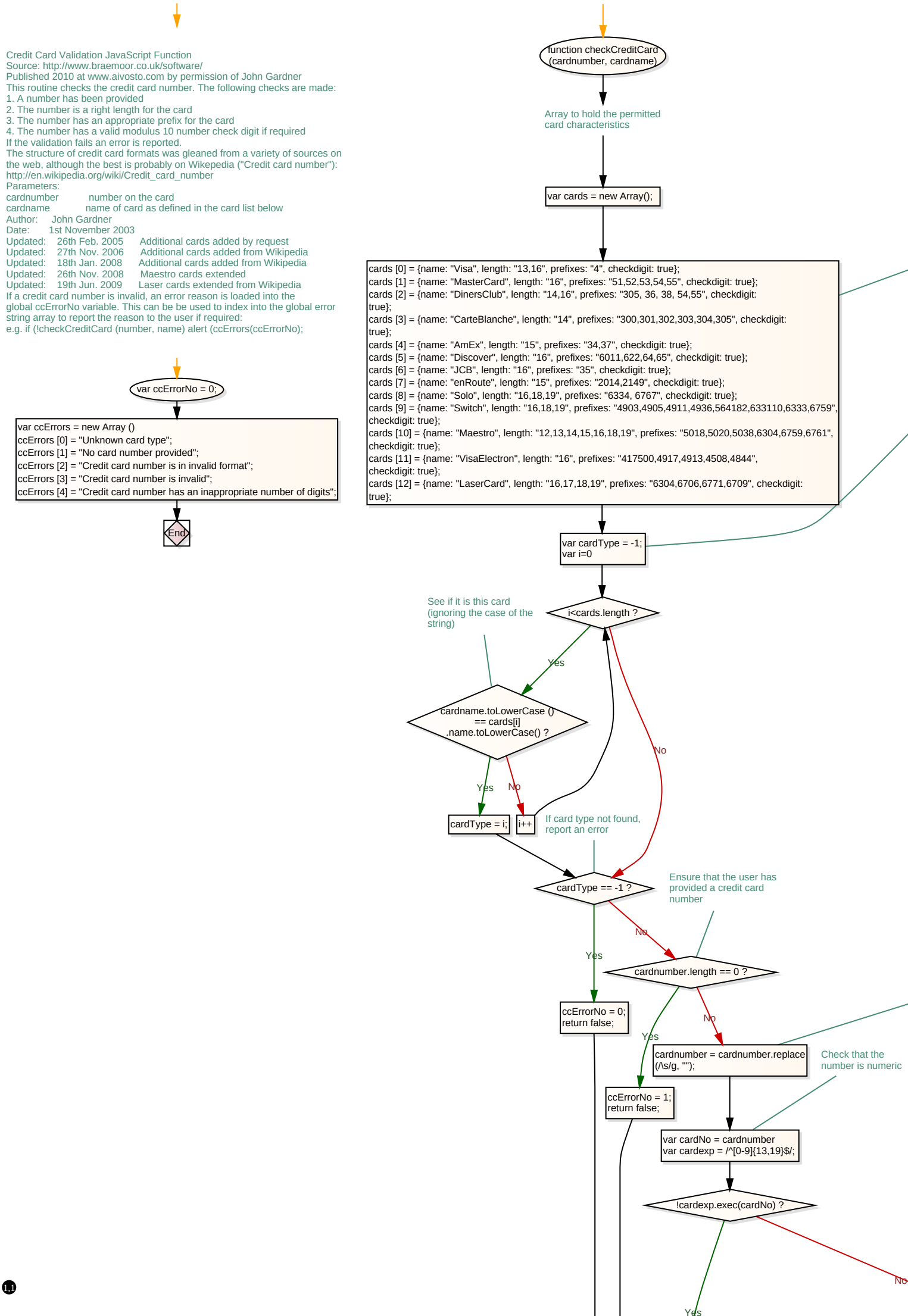
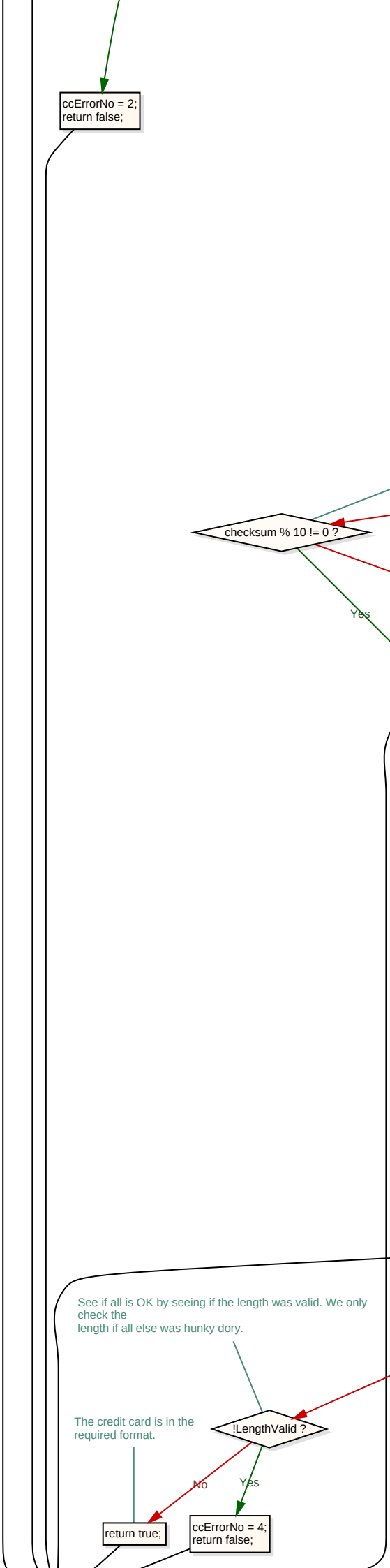

Credit Card Validation JavaScript Function
Source: http://www.braemoor.co.uk/software/
Published 2010 at www.aivosto.com by permission of John Gardner
This routine checks the credit card number. The following checks are made:
1. A number has been provided
2. The number is a right length for the card
3. The number has an appropriate prefix for the card
4. The number has a valid modulus 10 number check digit if required
If the validation fails an error is reported.
The structure of credit card formats was gleaned from a variety of sources on the web, although the best is probably on Wikepedia ("Credit card number"):
http://en.wikipedia.org/wiki/Credit_card_number
Parameters:
cardnumber number on the card
cardname name of card as defined in the card list below
Author: John Gardner
Date: 1st November 2003
Updated: 26th Feb. 2005 Additional cards added by request
Updated: 27th Nov. 2006 Additional cards added from Wikipedia
Updated: 18th Jan. 2008 Additional cards added from Wikipedia
Updated: 26th Nov. 2008 Maestro cards extended
Updated: 19th Jun. 2009 Laser cards extended from Wikipedia
If a credit card number is invalid, an error reason is loaded into the global ccErrorNo variable. This can be be used to index into the global error string array to report the reason to the user if required:
e.g. if (!checkCreditCard (number, name) alert (ccErrors(ccErrorNo);
var ccErrorNo = 0;
var ccErrors = new Array ()
ccErrors [0] = "Unknown card type";
ccErrors [1] = "No card number provided";
ccErrors [2] = "Credit card number is in invalid format";
ccErrors [3] = "Credit card number is invalid";
ccErrors [4] = "Credit card number has an inappropriate number of digits";
End
function checkCreditCard (cardnumber, cardname)
Array to hold the permitted card characteristics
var cards = new Array();
cards [0] = {name: "Visa", length: "13,16", prefixes: "4", checkdigit: true};
cards [1] = {name: "MasterCard", length: "16", prefixes: "51,52,53,54,55", checkdigit: true};
cards [2] = {name: "DinersClub", length: "14,16", prefixes: "305, 36, 38, 54,55", checkdigit: true};
cards [3] = {name: "CarteBlanche", length: "14", prefixes: "300,301,302,303,304,305", checkdigit: true};
cards [4] = {name: "AmEx", length: "15", prefixes: "34,37", checkdigit: true};
cards [5] = {name: "Discover", length: "16", prefixes: "6011,622,64,65", checkdigit: true};
cards [6] = {name: "JCB", length: "16", prefixes: "35", checkdigit: true};
cards [7] = {name: "enRoute", length: "15", prefixes: "2014,2149", checkdigit: true};
cards [8] = {name: "Solo", length: "16,18,19", prefixes: "6334, 6767", checkdigit: true};
cards [9] = {name: "Switch", length: "16,18,19", prefixes: "4903,4905,4911,4936,564182,633110,6333,6759", checkdigit: true};
cards [10] = {name: "Maestro", length: "12,13,14,15,16,18,19", prefixes: "5018,5020,5038,6304,6759,6761", checkdigit: true};
cards [11] = {name: "VisaElectron", length: "16", prefixes: "417500,4917,4913,4508,4844", checkdigit: true};
cards [12] = {name: "LaserCard", length: "16,17,18,19", prefixes: "6304,6706,6771,6709", checkdigit: true};
var cardType = -1;
var i=0
See if it is this card (ignoring the case of the string)
i<cards.length ?
Yes
No
cardname.toLowerCase () == cards[i] .name.toLowerCase() ?
Yes
No
cardType = i;
i++
If card type not found, report an error
cardType == -1 ?
Ensure that the user has provided a credit card number
No
Yes
cardnumber.length == 0 ?
ccErrorNo = 0;
return false;
No
Yes
cardnumber = cardnumber.replace (/\s/g, "");
Check that the number is numeric
ccErrorNo = 1;
return false;
var cardNo = cardnumber
var cardexp = /^[0-9]{13,19}$/;
!cardexp.exec(cardNo) ?
No
Yes


ccErrorNo = 2;
return false;
checksum % 10 != 0 ?
Yes
See if all is OK by seeing if the length was valid. We only check the
length if all else was hunky dory.
The credit card is in the required format.
!LengthValid ?
No
Yes
return true;
ccErrorNo = 4;
return false;


End

Define the cards we support. You may add addtional card types.
Name: As in the selection box of the form - must be same as user's
Length: List of possible valid lengths of the card number for the card
prefixes: List of possible prefixes for the card
checkdigit Boolean to say whether there is a check digit

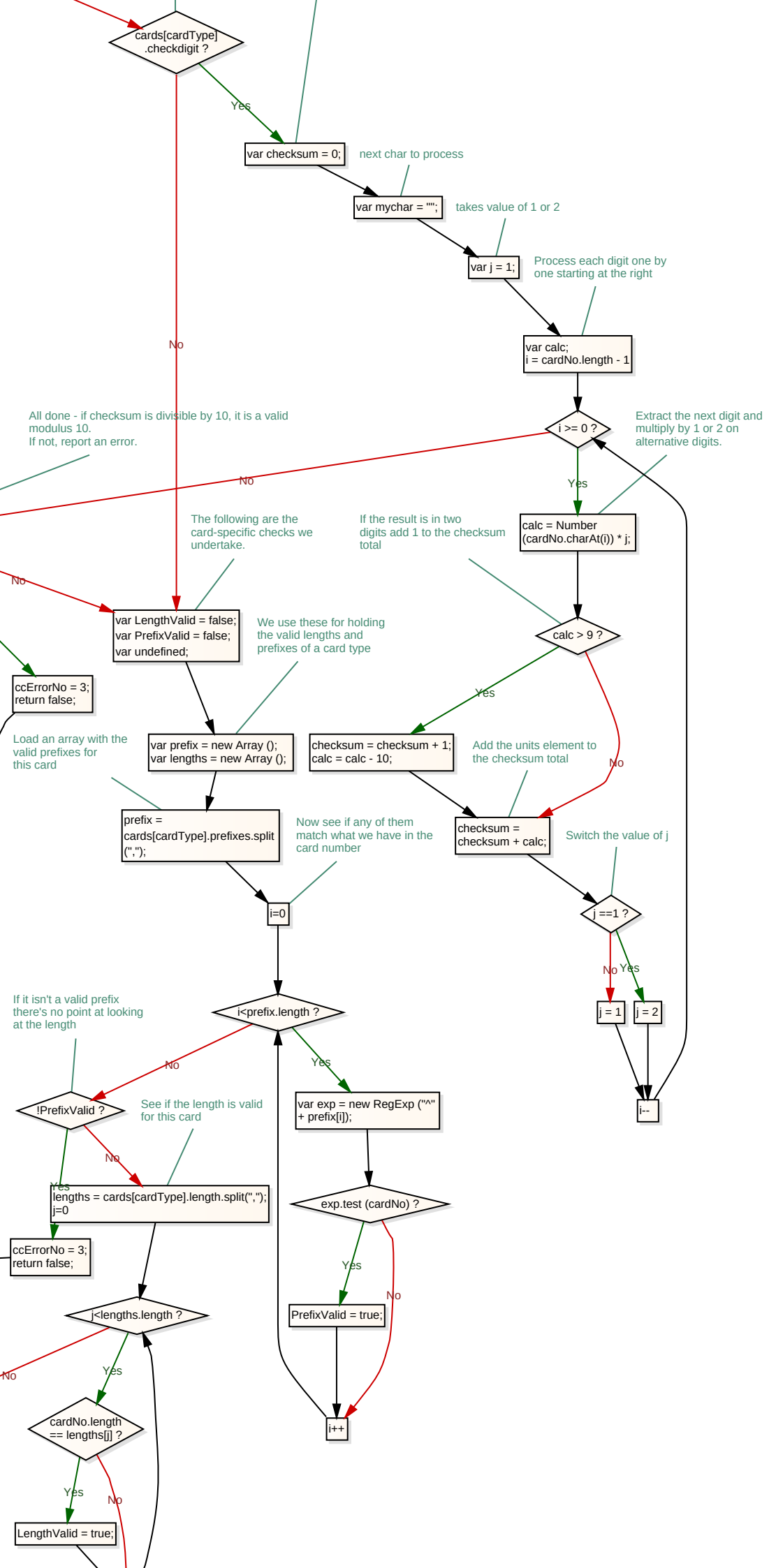
Establish card type

Now remove any spaces from
the credit card number

Now check the modulus 10
check digit - if required

2,1

running checksum total


cards[cardType]
.checkdigit ?
Yes
No
var checksum = 0;
next char to process
var mychar = "";
takes value of 1 or 2
var j = 1;
Process each digit one by
one starting at the right
var calc;
i = cardNo.length - 1
All done - if checksum is divisible by 10, it is a valid
modulus 10.
If not, report an error.
i >= 0 ?
Extract the next digit and
multiply by 1 or 2 on
alternative digits.
Yes
No
The following are the
card-specific checks we
undertake.
If the result is in two
digits add 1 to the checksum
total
calc = Number
(cardNo.charAt(i)) * j;
No
var LengthValid = false;
var PrefixValid = false;
var undefined;
We use these for holding
the valid lengths and
prefixes of a card type
calc > 9 ?
ccErrorNo = 3;
return false;
Yes
Load an array with the
valid prefixes for
this card
var prefix = new Array ();
var lengths = new Array ();
checksum = checksum + 1;
calc = calc - 10;
Add the units element to
the checksum total
No
prefix =
cards[cardType].prefixes.split
(",");
Now see if any of them
match what we have in the
card number
checksum =
checksum + calc;
Switch the value of j
i=0
j ==1 ?
No
Yes
If it isn't a valid prefix
there's no point at looking
at the length
i<prefix.length ?
j = 1
j = 2
No
Yes
!PrefixValid ?
See if the length is valid
for this card
var exp = new RegExp ("^"
+ prefix[i]);
i--
No
Yes
lengths = cards[cardType].length.split(",");
j=0
exp.test (cardNo) ?
ccErrorNo = 3;
return false;
Yes
No
j<lengths.length ?
PrefixValid = true;
No
Yes
cardNo.length
== lengths[j] ?
i++
Yes
No
LengthValid = true;


j++