What will this program do?

(A) Nothing (or error)
(B) Print 100.0 forever
(C) Print random numbers between 0 and 100 forever
(D) Print copies of a single random number between 0 and 100 forever
(E) Print 100, then random numbers that get ever closer to 0

```javascript
let x = 100.0;
function draw() {
    print(x);
    x = random(x);
}
```
What will this program do?

(A) Nothing (or error)
(B) Print 100.0 forever
(C) Print random numbers between 0 and 100 forever
(D) Print copies of a single random number between 0 and 100 forever
(E) Print random numbers between 0 and 1 forever

```javascript
let x;
function draw() {
    print(noise(x));
    x = random(100);
}
```
Data challenges

- Creating it
- Storing it
- Moving it around
- Keeping it private
Data challenges

• Creating it
• Storing it
• Moving it around
• Keeping it private

• Making sense of it
The shape of data

How is your information organized? How do the parts relate to each other?

These questions profoundly affect the tools you use and the code you write.
Raw text
Sequence
Dictionary
Table
Tree
Graph
Call me Ishmael. Some years ago—never mind how long precisely—having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off—then, I account it high time to get to sea as soon as I can. This is my substitute for pistol and ball. With a philosophical flourish Cato throws himself upon his sword; I quietly take to the
McCarthy
medium.com/@neuroecology/punctuation-in-novels-8f316d542ec4

Faulkner
<table>
<thead>
<tr>
<th>Book Title</th>
<th>Book Title</th>
<th>Book Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absalom, Absalom!</td>
<td>A Farewell To Arms</td>
<td>Alice in Wonderland</td>
</tr>
<tr>
<td>Blood Meridian</td>
<td>Frankenstein</td>
<td>Great Expectations</td>
</tr>
<tr>
<td>Huckleberry Finn</td>
<td>Pride and Prejudice</td>
<td>Ulysses</td>
</tr>
</tbody>
</table>
46.12 47.88 46.32 45.27 44.32 43.87 44.23 42.95 41.74 40.69 41.68 40.73 40.75 40.55 39.39 39.27 40.89 41.22 40.57 40.43 40.58 39.93 41.08 40.00 37.64 37.46 37.16 36.76 35.65 36.31 37.32 35.55 34.98 34.72 34.55 36.12 36.76 37.62 36.36 37.88 36.59 37.13

The Right Honourable Justin Trudeau
The Right Honourable Stephen Harper
The Right Honourable Paul Edgar Philippe Martin
The Right Honourable Joseph Jacques Jean Chrétien
The Right Honourable A. Kim Campbell
The Right Honourable Martin Brian Mulroney
The Right Honourable John Napier Turner
The Right Honourable Pierre Elliott Trudeau
The Right Honourable Charles Joseph Clark
The Right Honourable Pierre Elliott Trudeau
The Right Honourable Lester Bowles Pearson
The Right Honourable John George Diefenbaker
The Right Honourable Louis Stephen St-Laurent
The Right Honourable William Lyon Mackenzie King
The Right Honourable Richard Bedford Bennett
The Right Honourable William Lyon Mackenzie King
The Right Honourable Arthur Meighen
Dictionary

Associate a set of keys with a set of values. Ask for the value associated with any key without examining every other key/value pair.

<table>
<thead>
<tr>
<th>Year</th>
<th>City, Country</th>
<th>Year</th>
<th>City, Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1896</td>
<td>Athens, Greece</td>
<td>1968</td>
<td>Mexico City, Mexico</td>
</tr>
<tr>
<td>1900</td>
<td>Paris, France</td>
<td>1972</td>
<td>Munich, West Germany</td>
</tr>
<tr>
<td>1904</td>
<td>St. Louis, United States</td>
<td>1976</td>
<td>Montreal, Canada</td>
</tr>
<tr>
<td>1908</td>
<td>London, United Kingdom</td>
<td>1980</td>
<td>Moscow, Soviet Union</td>
</tr>
<tr>
<td>1912</td>
<td>Stockholm, Sweden</td>
<td>1984</td>
<td>Los Angeles, United States</td>
</tr>
<tr>
<td>1920</td>
<td>Antwerp, Belgium</td>
<td>1988</td>
<td>Seoul, South Korea</td>
</tr>
<tr>
<td>1924</td>
<td>Paris, France</td>
<td>1992</td>
<td>Barcelona, Spain</td>
</tr>
<tr>
<td>1928</td>
<td>Amsterdam, Netherlands</td>
<td>1996</td>
<td>Atlanta, United States</td>
</tr>
<tr>
<td>1932</td>
<td>Los Angeles, United States</td>
<td>2000</td>
<td>Sydney, Australia</td>
</tr>
<tr>
<td>1936</td>
<td>Berlin, Germany</td>
<td>2004</td>
<td>Athens, Greece</td>
</tr>
<tr>
<td>1948</td>
<td>London, United Kingdom</td>
<td>2008</td>
<td>Beijing, China</td>
</tr>
<tr>
<td>1952</td>
<td>Helsinki, Finland</td>
<td>2012</td>
<td>London, United Kingdom</td>
</tr>
<tr>
<td>1956</td>
<td>Melbourne, Australia</td>
<td>2016</td>
<td>Rio de Janeiro, Brazil</td>
</tr>
<tr>
<td>1960</td>
<td>Rome, Italy</td>
<td>2020</td>
<td>Tokyo, Japan</td>
</tr>
<tr>
<td>SONG</td>
<td>ARTIST</td>
<td>ALBUM</td>
<td>Time Ago</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Ways To Go - Margot Mix</td>
<td>Weval, Margot</td>
<td>Weval Remix</td>
<td>11 hours</td>
</tr>
<tr>
<td>Death Is A Girl</td>
<td>Mini Mansions</td>
<td>The Great Pretend</td>
<td>11 hours</td>
</tr>
<tr>
<td>Jumbo</td>
<td>Underworld</td>
<td>Beaucoup Fish</td>
<td>11 hours</td>
</tr>
<tr>
<td>Bug Powder Dust</td>
<td>The Mysterons</td>
<td>Meandering</td>
<td>11 hours</td>
</tr>
<tr>
<td>...To Have No Answer</td>
<td>Flock of Dimes</td>
<td>If You See Me, Sa...</td>
<td>11 hours</td>
</tr>
<tr>
<td>I'll Cut You Down</td>
<td>Uncle Acid &amp; The...</td>
<td>Blood Lust</td>
<td>11 hours</td>
</tr>
<tr>
<td>L'enfer ce n'est pas les autres c'est moi</td>
<td>The Eye Of Time</td>
<td>Myth I: A Last Da...</td>
<td>11 hours</td>
</tr>
<tr>
<td>Terrain</td>
<td>pg.lost</td>
<td>Key</td>
<td>11 hours</td>
</tr>
</tbody>
</table>
Tree
Raw Text

- Web pages are raw text
  - index.html is a web page
  - index.html is raw text
- Let’s create an eReader on a web page
  - Read in text files
  - Display on index.html as raw text
The `<div>` tag defines a division or a section in an HTML document.

The `<div>` element is often used as a container for other HTML elements to style them with CSS or to perform certain tasks with JavaScript.

`createDiv()` in JavaScript P5 creates a `<div></div>` element in the DOM with given inner HTML.
let myDiv;
function setup() {
    createCanvas(100, 100);
    background(220);
    let myText = "A Christmas Carol";
    myDiv = createDiv(myText);
    myDiv.style("font-size", "48");
}
let myDiv;
function setup() {
  noCanvas();
  let myText = "A Christmas Carol";
  myDiv = createDiv(myText);
  myDiv.style("font-size", "48");
}
let myDiv;
let myText;
function setup() {
    noCanvas();
    myText = getText();
    myDiv = createDiv(myText);
    myDiv.style("font-size", "48");
}
function getText() {
    let myText = "Books by Charles Dickens:" +
        "A Christmas Carol" +
        "Great Expectations" +
        "Oliver Twist" +
        "Hard Times";
    return myText;
}
function getText() {
    let myText = "Books by Charles Dickens:" +
    "<br>" + "A Christmas Carol" +
    "<br>" + "Great Expectations" +
    "<br>" + "Oliver Twist" +
    "<br>" + "Hard Times";
    return myText;
}

Books by Charles Dickens:
A Christmas Carol
Great Expectations
Oliver Twist
Hard Times
A primitive eReader
Displays all of “A Christmas Carol” on one page

let aChristmasCarol = [];
let myDiv;
let myText;

function preload() {
    aChristmasCarol = loadStrings("/data/AChristmasCarol.txt");
}

function setup() {
    noCanvas();
    myText = join(aChristmasCarol, "<br>");
    myDiv = createDiv(myText);
    myDiv.style("font-size", "48");
}
Add Functionality to the eReader

- 30 lines per page
  - Will need page forward/back buttons
- Allow user to choose font size
- Allow user to select from multiple books by Dickens
let aChristmasCarol = [];
let myDiv;
let allLines;
let linesPerPage = 30;
let currentLine = 0;

function preload() {
  aChristmasCarol = loadStrings("/data/AChristmasCarol.txt");
}
function setup() {
    noCanvas();

    pageForward = createButton("Forward");
    pageForward.mouseClicked(pageForwardFunc);
    pageForward.style('width', '100px');
    pageForward.style('height', '50px');

    myDiv = createDiv();
    myDiv.style("font-size", "48");
}
function draw() {
    let myText = "";
    for (let i = currentLine; i < currentLine + linesPerPage; i++) {
        myText = myText + aChristmasCarol[i] + "<br>";
    }
    myDiv.html(myText);
}

function pageForwardFunc() {
    currentLine = currentLine + linesPerPage;
}
Change Font Size Slider

• Add a global variable
  ```javascript
  let myFontSizeSlide;
  ```

• Add to setup()
  ```javascript
  createP("Font Size: ");
  myFontSizeSlider = createSlider(20, 96, 48);
  ```

• Add to draw()
  ```javascript
  myDiv.style("font-size",
    myFontSizeSlider.value());
  ```
Add radio for Books (1 of 5)

- Add a global variable
  ```javascript
  let bookSelectorRadio;
  ```
function preload() {
    aChristmasCarol = loadStrings("/data/AChristmasCarol.txt");
    greatExpectations = loadStrings("/data/GreatExpectations.txt");
    oliverTwist = loadStrings("/data/OliverTwist.txt");
    hardTimes = loadStrings("/data/HardTimes.txt");
}
bookSelectorRadio = createRadio();
bookSelectorRadio.option("Carol");
bookSelectorRadio.option("Expectations");
bookSelectorRadio.option("Twist");
bookSelectorRadio.option("Times");
bookSelectorRadio.value("Carol");
bookSelectorRadio.style("font-size", "48");
let currentBook = [];
    if (bookSelectorRadio.value() === "Carol") {
        currentBook = aChristmasCarol;
    } else if (bookSelectorRadio.value() === "Expectations") {
        currentBook = greatExpectations;
    } else if (bookSelectorRadio.value() === "Twist") {
        currentBook = oliverTwist;
    } else if (bookSelectorRadio.value() === "Times") {
        currentBook = hardTimes;
    }
if (previousBook !== currentBook) {
  currentLine = 0;
  previousBook = currentBook;
}

for (let i = currentLine; i < currentLine + rowsPerPage; i++) {
  tempText = tempText + currentBook[i] + "<br>";
}
myDiv.html(tempText);
Fully implemented eReader is in demo code

“eReaderOnTheDOM”
What will this program display?
```javascript
let myDiv;
let myText = " 
function setup() {
  noCanvas();
  for (let i = 0; i < 3; i++) {
    myText = myText + i;
  }
  myDiv = createDiv(myText);
  myDiv.style("font-size", "48");
}
```

What will this program display?
Review from Week 3 (I/O)

- Two examples for quick review
  - Shopping List
  - Speed Reader
let lines = [];
let words = [];
let nextI;

function preload() {
    lines = loadStrings("data/shoppinglist.txt");
}

function setup() {
    createCanvas(600, 600);
    textSize(24);
}

function draw() {
    background(220);
    let totalKG = 0;
    for (let i = 0; i < lines.length; i++) {
        words = splitTokens(lines[i], " ");
        text(lines[i], 10, 30 + (i * 30));
        totalKG = totalKG + int(words[1]);
        nextI = i + 1;
    }
    text("Total KG: " + totalKG, 10, 30 + (nextI * 30));
}
SpeedReader Example

• Read in a text file.

• Make one big long list (array) of “words”
  • Words may contain punctuation in this example

• Display one word at a time
let lines = [];
let words = [];
let index = 0;
function preload() {
    lines = loadStrings("data/marley.txt");
}
function setup() {
    createCanvas(400, 200);
    textSize(50);
    textAlign(CENTER);
    fill(255);
    let allLines = join(lines, " ");
    words = splitTokens(allLines);
    frameRate(1);
}
function draw() {
    background(80);
    text(words[index], width/2, height/2);
    index = (index + 1) % words.length;
}
Messier text

function splitTokens(text, delims) { ... }

Break the long string text into “words”, where the characters in delims (and not whitespace) are treated as breakpoints.

function trim(text) { ... }

Return a copy of text with any excess whitespace removed from the start and end.
Example: the Region of Waterloo’s list of reserved street names

<table>
<thead>
<tr>
<th>FullStreetName</th>
<th>Municipality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abbey Glen</td>
<td>Kitchener</td>
</tr>
<tr>
<td>Aberle</td>
<td>Woolwich</td>
</tr>
<tr>
<td>Abeth</td>
<td>Kitchener</td>
</tr>
<tr>
<td>Abitibi</td>
<td>Cambridge</td>
</tr>
<tr>
<td>Able</td>
<td>Cambridge</td>
</tr>
<tr>
<td>Abram Clemens St</td>
<td>Kitchener</td>
</tr>
<tr>
<td>Accobee</td>
<td>Cambridge</td>
</tr>
<tr>
<td>Adair</td>
<td>Cambridge</td>
</tr>
</tbody>
</table>

let lines;
let myDiv;

function preload() {
    lines = loadStrings( "/data/ReservedStreetnames.csv" );
}
function setup() {
  noCanvas();
  myDiv = createDiv();
  myDiv.style("font-size", 36);
  let list = "";
  for ( let idx = 0; idx < lines.length; ++idx ) {
    let line = lines[idx];
    if ( line.charAt(0) != '-' ) {
      let words = splitTokens( line, "|" );
      let street = trim( words[0] );
      let municipality = trim( words[1] );
      list = list + street + "---" + municipality + "<br>";
    }
  }
  myDiv.html(list);
}
Fully implemented reserved street names in demo code

“ReservedStreetNamesP5”
Time permitting:

There is one more example of using JavaScript p5 and HTML in the demo code. It builds an unordered list of Canadian Prime Ministers.

“UnorderedList”
Reading the dictionary to solve Dictionary “problems”

Find the longest word
Find all words with three or more Ys
Find all words ending with MT
Find all words starting with TM
Find all words ending with DOUS
Find all words containing UFA
Find all words ending in GRY
Find all palindromes
Find words with three consecutive double letters
Find the longest word whose letters are in
let words = []; 
let palindromes = []; 
let myDiv; 

function preload() {
    words = loadStrings("/data/H_words.txt"); 
}
Find all Palindromes (2 of 2)

```javascript
function setup() {

    myDiv = createDiv();
    myDiv.style("font-size", "48");

    for (let i = 0; i < words.length; i++) {

        let pal = true;
        let wordLength = words[i].length;
        let halfWordLength = int(words[i].length / 2);

        for (let j = 0; j < halfWordLength; j++) {
            if (words[i][j] != words[i][wordLength - j - 1]) {
                pal = false;
            }
        }
        if (pal === true) {
            palindromes.push(words[i]);
        }
    }

    myDiv.html(join(palindromes, "<br>"));}
```
Fully implemented palindrome is in the demo code

“Palindromes”
Dictionaries

In programming, a dictionary is a mapping from a set of keys to a set of values. Any given key may have at most one associated value. Here are 5 examples.

- Year ➔ Olympic host city
- Name ➔ Phone number
- Student ID number ➔ Exam seating code
- Clicker ID ➔ Student ID number
- Server name ➔ IP address
Dictionaries

Dictionary operations we might care about:

• Look up the value associated with a given key
• Ask if the dictionary has a given key
• Add a new key to the dictionary, with its associated value
• Remove a key and its value from the dictionary
let words;
let myDiv;
let carolLines;

function preload() {
  words = loadStrings( "../data/H_words.txt" );
  carolLines = loadStrings( "../data/AChristmascarol.txt" );
}
function setup() {
    myDiv = createDiv();
    myDiv.style("font-size", 48);

    let text = join( carolLines, ' ' );
    let carolWords = splitTokens( text, " ,!0123456789[]*:" );

    let badWords = [];
    for ( let idx = 0; idx < carolWords.length; ++idx ) {
        let word = carolWords[idx].toLowerCase();
        let wordFound = isWord(word);
        if (!wordFound) {
            badWords.push(word);
        }
    }
    myDiv.html(join(badWords, "<br>"));}

function isWord(wd) {
    for (let j = 0; j < words.length; j++) {
        if (wd === words[j]) {
            return true;
        }
    }
    return false;
}
Fully implemented inefficient spell checker is in the demo code

“ArraySpellCheckerP5”
let dictionary = {};  
let words;  
let myDiv;  
let carolLines;  

function preload() {  
    words = loadStrings( "./data/H_words.txt" );  
    carolLines = loadStrings(  
        "./data/AChristmasmascarol.txt" );  
}
function setup()
{
    myDiv = createDiv();
    myDiv.style("font-size", 48);
    for ( let idx = 0; idx < words.length; ++idx ) {
        dictionary[words[idx]] = 1;
    }

    let text = join( carolLines, ' ' );
    let carolWords = splitTokens(text, " ,.!0123456789[]*:!");

    let badWords = [];
    for ( let idx = 0; idx < carolWords.length; ++idx ) {
        let word = carolWords[idx].toLowerCase();
        if ( !dictionary[word] ) {
            badWords.push(word);
        }
    }

    myDiv.html(join(badWords, "<br>")));
Fully implemented efficient spell checker is in the demo code

“DictSpellCheckerP5”
Regular Expressions: match()

To help us search through data, p5 provides us with the “match” function.

function match( text, pattern ) { ... }
Regular Expressions are a general tool for finding patterns in strings.

**Example**

```
var string = 'Hello p5js*!';
var regexp = 'p5js\\*';
var m = match(string, regexp);
text(m, 5, 50);
```

**Description**

This function is used to apply a regular expression to a piece of text, and return matching groups (elements found inside parentheses) as a String array. If there are no matches, a null value will be returned. If no groups are specified in the regular expression, but the sequence matches, an array
Finding patterns
Regular Expressions are a general tool for finding patterns in strings.

Reference

**match()**

**Example**

```javascript
var string = 'Hello p5js*!';
var regexp = 'p5js\\*';
var m = match(string, regexp);
text(m, 5, 50);
```

**Description**

This function is used to apply a regular expression to a piece of text, and return matching groups (elements found inside parentheses) as a String array. If there are no matches, a null value will be returned. If no groups are specified in the regular expression, but the sequence matches, an array
function match( text, pattern ) { ... }

Look for an instance of the regular expression pattern inside of the string text. If the answer is not null, the pattern was found.
Finding patterns

It’s easy to search a string for a given phone number:

```java
if (match(myString, "(519) 888-4567") != null) { … }
```

But what if we wanted to find all the phone numbers in a string?
Finding patterns

*Regular Expressions* are a general tool for finding patterns in strings.
Finding patterns

Regular Expressions are a **programming language** for finding patterns in strings.
Finding patterns

Regular Expressions are a cryptic programming language for finding patterns in strings.
**Regular Expressions - Quick Reference Guide**

### Anchors
- `^` start of line
- `$` end of line
- `\b` word boundary
- `\B` not at word boundary
- `\A` start of subject
- `\G` first match in subject
- `\Z` end of subject
- `\z` end of subject
- `\Z` or before newline at end

### Non-printing characters
- `\a` alarm (BEL, hex 07)
- `\c` "control-x"
- `\e` escape (hex 1B)
- `\f` formfeed (hex 0C)
- `\n` newline (hex 0A)
- `\r` carriage return (hex OD)
- `\t` tab (hex 09)
- `\ddd` octal code ddd
- `\hh` hex code hh
- `\{hh..}` hex code hhh...

### Generic character types
- `\d` decimal digit
- `\D` not a decimal digit
- `\s` whitespace character
- `\S` not a whitespace char
- `\w` "word" character
- `\W` "non-word" character

### POSIX character classes
- `alnum` letters and digits
- `alpha letters`
- `ascii` character codes 0-127
- `blank` space or tab only
- `cntrl` control characters
- `digit` decimal digits
- `graph` printing chars -space
- `lower` lower case letters
- `print` printing chars +space
- `punct` printing chars -alnum
- `space` white space
- `upper` upper case letters
- `word` "word" characters
- `xdigit` hexadecimal digits

### Literal Characters
- Letters and digits match exactly: `a \ x \ B \ 7 \ 0`
- Some special characters match exactly: `@ = - %`
- Escape other specials with backslash: `\ \ \ 1 \ . \ \ \ \ $ \ \ [`

### Character Groups
- Almost any character (usually not newline): `.`
- Lists and ranges of characters: `[ ]`
- Any character except those listed: `[^ ]`

### Counts (add ? for non-greedy)
- 0 or more ("perhaps some"): `*`
- 0 or 1 ("perhaps a"): `?`
- 1 or more ("some"): `+`
- Between "n" and "m" of: `{n,m}`
- Exactly "n", "n" or more: `{n}, {{n},`

### Alternation
- Either/or: `|`

### Lookahead and Lookbehind
- Followed by: `(?= )`
- NOT followed by: `(?! )`
- Following: `(?<= )`
- NOT following: `(?<! )`

### Grouping
- For capture and counts: `( )`
- Non-capturing: `(?: )`
- Named captures: `( ?<name> )`

### Back references
- Numbered: `\n \ \ \n \ \ \ \ \g{n}`
- Relative: `\g{-n}`
- Named: `\k<name>`

### Character group contents
- `x` individual chars
- `x-y` character range
- `[:class:]` character class
- `[^:]` negated class

### Examples
- `[a-zA-Z0-9_]`
- `[[alnum:]]`

### Comments
- `(?#comment)`

### Conditional subpatterns
- `(?condition)yes-pattern`
- `(?condition)yes|no-pattern`

### Recursive patterns
- `(?n)` Numbered
- `(?0) (?R)` Entire regex
- `(?&name)` Named

### Replacements
- `$n` reference capture

### Case foldings
- `
` upper case next char
- `
` upper case following
- `
` lower case next char
- `
` lower case following
- `
` end case folding

### Conditional insertions
- `(?:insertion)`
- `(?:insertion:otherwise)`

[http://www.e-texteditor.com](http://www.e-texteditor.com)
Substring “ufa” anywhere in a word:

ufa

Word ending in “mt”:

mt$

Word with three or more “y”s, on a line by itself:

y.*y.*y

An integer:

^(-?[1-9]+d*)$|^0$

An email address:

\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b

A URL:

^(https?:\/\/)?([\da-z\.-]+\.)([a-z\.-]{2,6})([\w\-]*)*$
A regular expression is like a little “machine”:

^(-?[1-9]+\d*)$|^0$
Phone Number Patterns (1 of 2)

let p = "(\(?:\([\d]{3}\)\)?\([\d]{3}\)\([\d]{4}\)\)|\([\d]{3}\)\([\d]{3}\)\([\d]{4}\)\)\([\d]{2}\)\([\d]{2}\)\([\d]{2}\)\)";
let lines = [];
let myDiv;
let phoneNumbers = [];

function preload() {
  print("preload");
  lines = loadStrings( "/data/input.txt" );
}
function setup()
{
    noCanvas();
    myDiv = createDiv();
    myDiv.style("font-size", "48");

    for ( let idx = 0; idx < lines.length; ++idx ) {
        let m = match( lines[idx], p );
        if ( m != null ) {
            if ( m[1] == null ) {
                phoneNumbers = phoneNumbers + "<br>" + "(" + m[2] + ") " + m[3] + "-" + m[4];
            } else {
                phoneNumbers = phoneNumbers + "<br>" + "(" + m[1] + ") " + m[3] + "-" + m[4];
            }
        }
    }

    myDiv.html(phoneNumbers);
}
Fully implement of finding phone number patterns is in the demo code

“FindPhoneNumbersP5”