Jump to a Section

Intro to Location Services:

Two frameworks are used for dealing with locations:

- CoreLocation → used to manage operations related to coordinates and addresses
- MapKit → used to manage any operations related to *drawing a map* on the screen

Common operations:

- Forward geocoding
- Reverse geocoding
- Including a map
- Adding feature to the map:
    - Pins
    - Driving Directions
    - etc

Forward Geocoding

Translating a human-readable address into a set of (latitude,longitude) coordinates is known as **forward geocoding**

To perform forward geocoding:

- Submit the address to Apple's geocoding server as a *background* task
- At some point in the future, the geocoding server *may* respond with the coordinates of the provided address
- It could be possible that the server *does not respond* or *cannot find matching coordinates.* In those cases, write error handling code.

ForwardGeoVC.swift:

```
private func getLocation(address : String){
    // 1. Take the provided address and send it to apple's geocoding server
    // 2. When we get a response:
    //  2a. Apple server is down (internet is broken) --> error
    //  2b. Network is okay, but server could not find a matching result for your address
    //  2c. Network is okay, AND the server WAS able to find a matching result

    // 3d. output to the screen (error message, results)

}
```

Code Along - Forward Geocoding

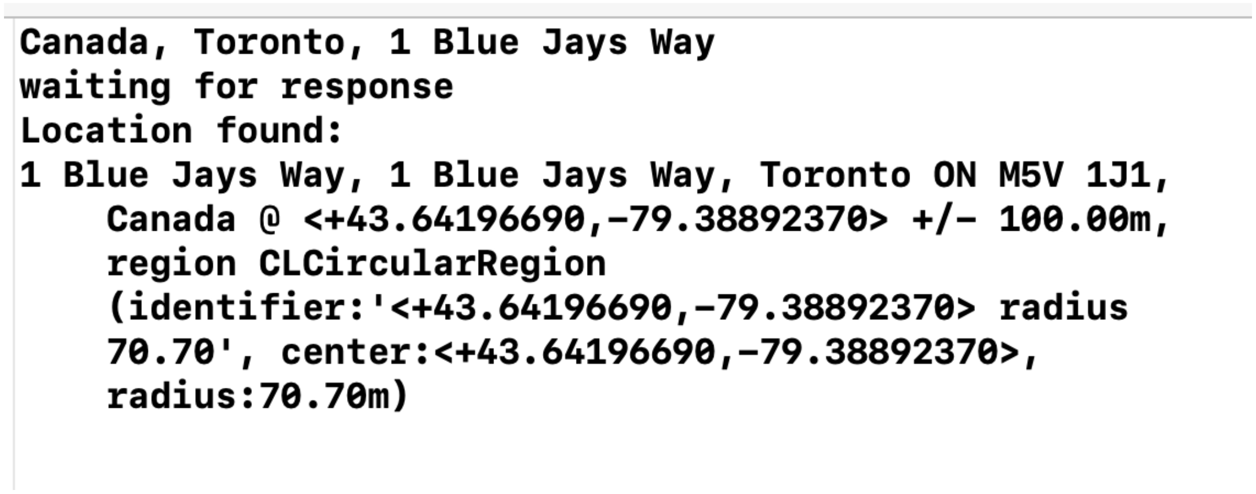
1/ You want to import Core Location
2/ Create a geocoding object (built in object that comes with core loaction)
3/ Use that geocoding object to perform the operations in the getAddress() helper function

```
import UIKit

// 1. Importing the CoreLocation library
import CoreLocation

class ForwardGeoVC: UIViewController {

    @IBOutlet weak var tfCountry: UITextField!
    @IBOutlet weak var tfCity: UITextField!
    @IBOutlet weak var tfStreet: UITextField!
    @IBOutlet weak var lblLocation: UILabel!

    // 2. Create the geocoder variable
    let geocoder = CLGeocoder()

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

    @IBAction func onFetchCoordinateClicked(_ sender: Any) {
        guard let country = tfCountry.text else {return}
        guard let city = tfCity.text else {return}
        guard let street = tfStreet.text else {return}
        print("\(country), \(city), \(street)")
        //self.getLocation(address: "\(country), \(city), \(street)")
    }

    private func getLocation(address : String){
        // 1. Take the provided address and send it to apple's geocoding server
        // 2. When we get a response:
        //  2a. Apple server is down (internet is broken) --> error
        //  2b. Network is okay, but server could not find a matching result for your address
        //  2c. Network is okay, AND the server WAS able to find a matching result

        // 3d. output to the screen (error message, results)
        self.geocoder.geocodeAddressString(address) {
            (resultsList, error) in
            print("waiting for response")
        }


    }
}
```

4/ Handle the results of the geocoding request
4a/ If matching coordinates found, they will be located in the *resultsList* variable and the *error* variable will be NIL
4b/ If something went wrong, the *error* variable will be NOT nil (and thus contain information about what happened)

```
import UIKit
// 1. Importing the CoreLocation library
import CoreLocation

class ForwardGeoVC: UIViewController {

    @IBOutlet weak var tfCountry: UITextField!
    @IBOutlet weak var tfCity: UITextField!
    @IBOutlet weak var tfStreet: UITextField!
    @IBOutlet weak var lblLocation: UILabel!

    // 2. Create the geocoder variable
    let geocoder = CLGeocoder()

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }

    @IBAction func onFetchCoordinateClicked(_ sender: Any) {
        guard let country = tfCountry.text else {return}
        guard let city = tfCity.text else {return}
        guard let street = tfStreet.text else {return}


        print("\(country), \(city), \(street)")

        self.getLocation(address: "\(country), \(city), \(street)")
    }

    private func getLocation(address : String){
        // 1. Take the provided address and send it to apple's geocoding server
        // 2. When we get a response:

        // 3d. output to the screen (error message, results)
        self.geocoder.geocodeAddressString(address) {
            (resultsList, error) in
            print("waiting for response")

            //  2a. Apple server is down (internet is broken) --> error
            if let err = error {
                print("Error while trying to geocode the address")
                print(err)
                return
            }
            //  2b. Network was okay
            if (resultsList != nil) {
                // 2c. Network is okay, but server could not find a matching result
for your address
                if (resultsList!.count == 0) {
                    print("No results found.")
                }
                // 2d. Network is okay, AND the server WAS able to find a matching
result
                else {
                    let placemark:CLPlacemark = resultsList!.first!
                    print("Location found: ")
                    print(placemark)
                }
            }
        }
    }
}
```

Expected result:

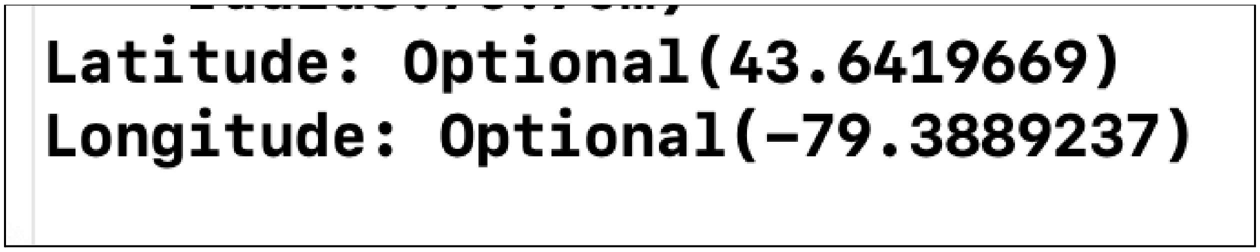


```
Canada, Toronto, 1 Blue Jays Way
waiting for response
Location found:
1 Blue Jays Way, 1 Blue Jays Way, Toronto ON M5V 1J1,
    Canada @ <+43.64196690,-79.38892370> +/- 100.00m,
    region CLCircularRegion
    (identifier:'<+43.64196690,-79.38892370> radius
    70.70', center:<+43.64196690,-79.38892370>,
    radius:70.70m)
```

5/ Extract just the coordinates from the results

```swift
import UIKit
// 1. Importing the CoreLocation library
import CoreLocation

class ForwardGeoVC: UIViewController {

    @IBOutlet weak var tfCountry: UITextField!
    @IBOutlet weak var tfCity: UITextField!
    @IBOutlet weak var tfStreet: UITextField!
    @IBOutlet weak var lblLocation: UILabel!

    // 2. Create the geocoder variable
    let geocoder = CLGeocoder()

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }

    @IBAction func onFetchCoordinateClicked(_ sender: Any) {
        guard let country = tfCountry.text else {return}
        guard let city = tfCity.text else {return}
        guard let street = tfStreet.text else {return}


        print("\(country), \(city), \(street)")

        self.getLocation(address: "\(country), \(city), \(street)")
    }

    private func getLocation(address : String){
        // 1. Take the provided address and send it to apple's geocoding server
        // 2. When we get a response:

        // 3d. output to the screen (error message, results)
        self.geocoder.geocodeAddressString(address) {
            (resultsList, error) in
            print("waiting for response")

            //  2a. Apple server is down (internet is broken) --> error
            if let err = error {
                print("Error while trying to geocode the address")
                print(err)
                return
            }


            //  2b. Network was okay
            if (resultsList != nil) {
                // 2c. Network is okay, but server could not find a matching result
for your address
                if (resultsList!.count == 0) {
                    print("No results found.")
                }
                // 2d. Network is okay, AND the server WAS able to find a matching
result
                else {
                    let placemark:CLPlacemark = resultsList!.first!
                    print("Location found: ")
                    print(placemark)

                    // 2e. Extract only the coordinates
                    print("Latitude: \(placemark.location?.coordinate.latitude)")
                    print("Longitude: \(placemark.location?.coordinate.longitude)")
                }
            }


        }


    }
}
```

Expected result:



```
Latitude: Optional(43.6419669)
Longitude: Optional(-79.3889237)
```

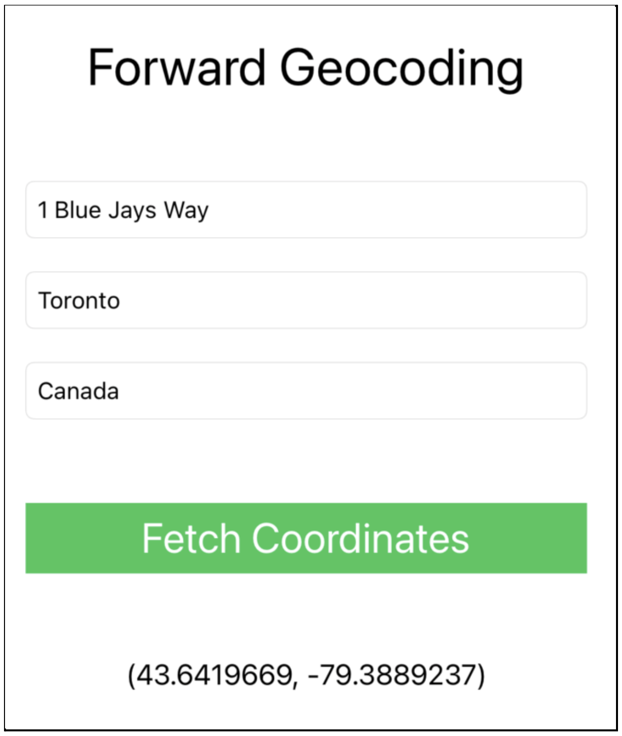
6/ Output results to ui

When writing code in a closure, you must prefix your outlets and other global variables with *self*
The closure cannot directly access any of the global variables or outlet, so you must prefix them with *self* for the compiler to build your code.

```
private func getLocation(address : String){
        // 1. Take the provided address and send it to apple's geocoding server
        // 2. When we get a response:

        // 3d. output to the screen (error message, results)
        self.geocoder.geocodeAddressString(address) {
            (resultsList, error) in
            print("waiting for response")

            //  2a. Apple server is down (internet is broken) --> error
            if let err = error {
                print("Error while trying to geocode the address")
                print(err)
                self.lblLocation.text = "\(err)"
                return
            }


            //  2b. Network was okay
            if (resultsList != nil) {
                // 2c. Network is okay, but server could not find a matching result
for your address
                if (resultsList!.count == 0) {
                    print("No results found.")
                    self.lblLocation.text = "No results found"
                }
                // 2d. Network is okay, AND the server WAS able to find a matching
result
                else {
                    let placemark:CLPlacemark = resultsList!.first!
                    print("Location found: ")
                    print(placemark)

                    // 2e. Extract only the coordinates
                    print("Latitude: \(placemark.location?.coordinate.latitude)")
                    print("Longitude: \(placemark.location?.coordinate.longitude)")
                    self.lblLocation.text =
"(\(placemark.location!.coordinate.latitude),
\(placemark.location!.coordinate.longitude))"
                }
            }

        }

    }
```

Expected result

- Coordinates will be output to the screen



(43.6419669, -79.3889237)

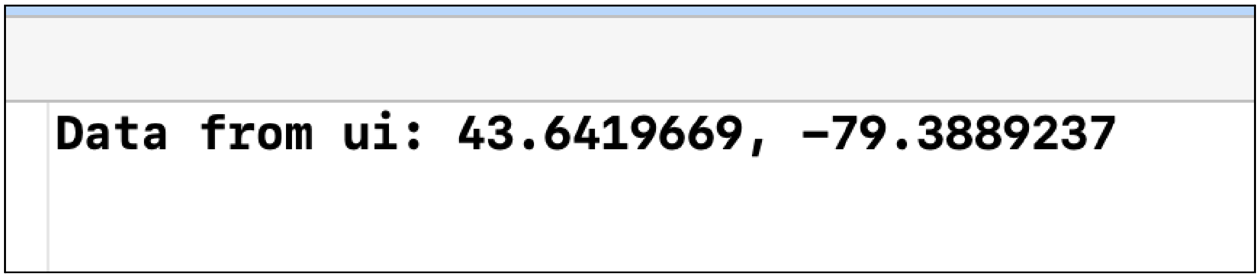
Reverse Geocoding

Reverse geocoding is the opposite of forward geocoding. Reverse geocoding is the process of translating a set of (latitude,longitude) coordinates into a human readable address.

1/ Get data from text box

```
import UIKit

class ReverseGeoVC: UIViewController {

    @IBOutlet weak var tfLatitude: UITextField!
    @IBOutlet weak var tfLongitude: UITextField!
    @IBOutlet weak var lblAddress: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }

    @IBAction func onFetchAddressClicked(_ sender: Any) {
        // 1. get the lat/lng from the ui

        // 1a. Get the latitude from the text box (textbox will return a string or 
nil)
        // 1b. If you got a string (and not a nil), then try to convert that string 
to a Double
        // - If the conversion succeeeds, then store the converted value in the 
variable called "lat"
        // - If the conversion fails, then return / exit this function
        guard let latAsString = tfLatitude.text, let latAsDouble = 
Double(latAsString) else {
            return
        }

        guard let lngAsString = tfLongitude.text, let lngAsDouble = 
Double(lngAsString) else {
            return
        }
        print("Data from ui: \(latAsDouble), \(lngAsDouble)")


        // 2. call the helper function that will translate that lat/lng into a human 
readable address
        // TODO: Call the getAddress() function
    }

    func getAddress(){

    }
}
```

Expected result:

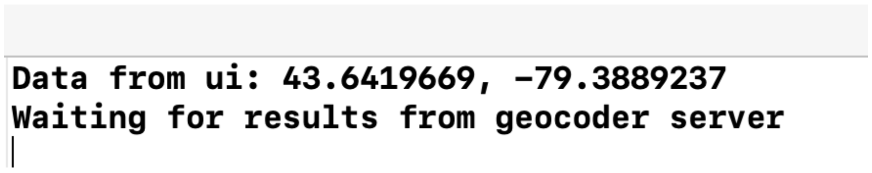


```
Data from ui: 43.6419669, -79.3889237
```

2/ You want to import Core Location
3/ Create a geocoding object (built in object that comes with core location)
4/ Use that geocoding object to perform the operations in the getAddress() helper function

```
import UIKit
import CoreLocation

class ReverseGeoVC: UIViewController {

    @IBOutlet weak var tfLatitude: UITextField!
    @IBOutlet weak var tfLongitude: UITextField!
    @IBOutlet weak var lblAddress: UILabel!

    let geocoder = CLGeocoder()

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func onFetchAddressClicked(_ sender: Any) {
        guard let latAsString = tfLatitude.text, let latAsDouble = 
Double(latAsString) else {
            return
        }

        guard let lngAsString = tfLongitude.text, let lngAsDouble = 
Double(lngAsString) else {
            return
        }
        print("Data from ui: \(latAsDouble), \(lngAsDouble)")

        // 2. call the helper function that will translate that lat/lng into a human 
readable address
        let loc = CLLocation(latitude: latAsDouble, longitude: lngAsDouble)
        self.getAddress(locationFromUser: loc)

    }

    func getAddress(locationFromUser:CLLocation){

        // reverseGeocodeLocation accepts 1 parameter of type CLLocation
        // and CLLocation should contain your lat/lng that you want to reverse 
geocode
        self.geocoder.reverseGeocodeLocation(locationFromUser) {
            (resultsList, error) in
            print("Waiting for results from geocoder server")
        }
    }
}
```

Expected:

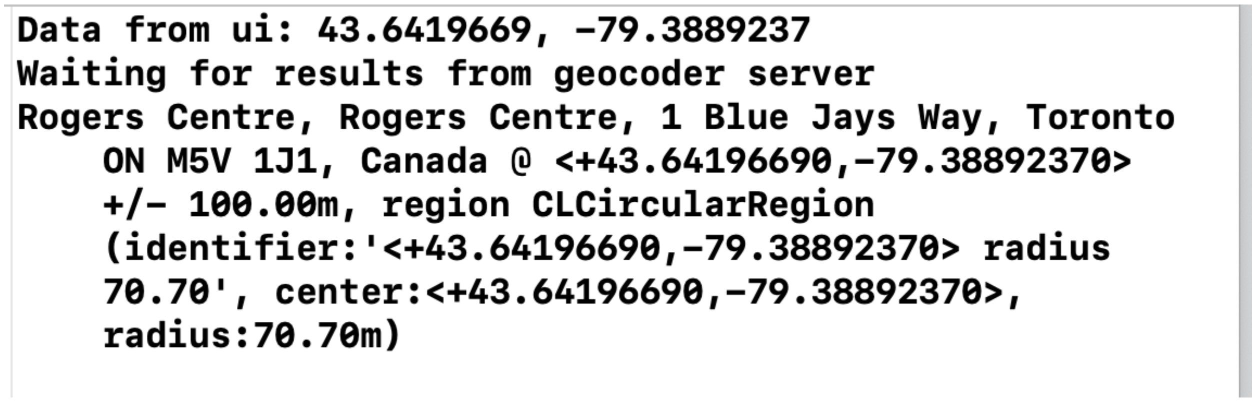


```
Data from ui: 43.6419669, -79.3889237
Waiting for results from geocoder server
```

2/ Process results from apple server
The code is the same as the code for forward geocoding, but write using the if-let syntax instead of the (if x != nil) format:

```
import UIKit

import CoreLocation

class ReverseGeoVC: UIViewController {

    @IBOutlet weak var tfLatitude: UITextField!
    @IBOutlet weak var tfLongitude: UITextField!
    @IBOutlet weak var lblAddress: UILabel!

    let geocoder = CLGeocoder()


    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func onFetchAddressClicked(_ sender: Any) {
        // 1. get the lat/lng from the ui

        // 1a. Get the latitude from the text box (textbox will return a string or
nil)
        // 1b. If you got a string (and not a nil), then try to convert that string
to a Double
        // - If the conversion succeeeds, then store the converted value in the
variable called "lat"
        // - If the conversion fails, then return / exit this function
        guard let latAsString = tfLatitude.text, let latAsDouble =
Double(latAsString) else {
            return
        }

        guard let lngAsString = tfLongitude.text, let lngAsDouble =
Double(lngAsString) else {
            return
        }
        print("Data from ui: \(latAsDouble), \(lngAsDouble)")


        // 2. call the helper function that will translate that lat/lng into a human
readable address
        // TODO: Call the getAddress() function
        let loc = CLLocation(latitude: latAsDouble, longitude: lngAsDouble)
        self.getAddress(locationFromUser: loc)

    }

    func getAddress(locationFromUser:CLLocation){

        // reverseGeocodeLocation accepts 1 parameter of type CLLocation
        // and CLLocation should contain your lat/lng that you want to reverse
geocode
        self.geocoder.reverseGeocodeLocation(locationFromUser) {
            (resultsList, error) in
            print("Waiting for results from geocoder server")

            if let err = error {
                print("Error when performing reverse geocoding")
                print(err)
                self.lblAddress.text = "\(err)"
                return
            }

            // otherwise, network was okay so lets get the results if any
            if let results = resultsList {
                if (results.isEmpty) {
                    print("No matching addresses found")
                    self.lblAddress.text = "No matching addresses found"
                }
                else {
                    // if not empty, then you were able to find results!
                    // of type CLPlacemark
                    if let matchingLocation:CLPlacemark = results.first {
                        print(matchingLocation)
                    }
                }
            }

        }
    }
}
```

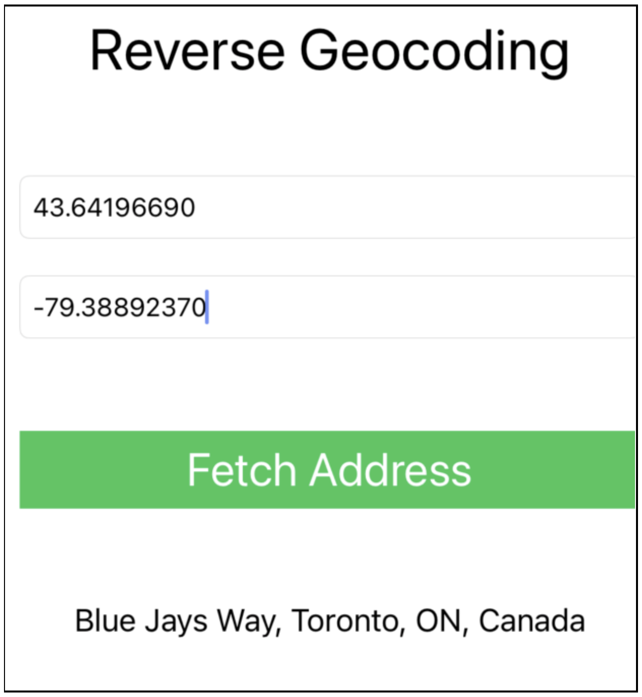
Expected result:



```
Data from ui: 43.6419669, -79.3889237
Waiting for results from geocoder server
Rogers Centre, Rogers Centre, 1 Blue Jays Way, Toronto
    ON M5V 1J1, Canada @ <+43.64196690,-79.38892370>
    +/- 100.00m, region CLCircularRegion
    (identifier:'<+43.64196690,-79.38892370> radius
    70.70', center:<+43.64196690,-79.38892370>,
    radius:70.70m)
```

3/ Extract the relevant data from the results (city name, address, etc)

```
func getAddress(locationFromUser:CLLocation){

        // reverseGeocodeLocation accepts 1 parameter of type CLLocation
        // and CLLocation should contain your lat/lng that you want to reverse
geocode
        self.geocoder.reverseGeocodeLocation(locationFromUser) {
            (resultsList, error) in
            print("Waiting for results from geocoder server")

            if let err = error {
                print("Error when performing reverse geocoding")
                print(err)
                self.lblAddress.text = "\(err)"
                return
            }

            // otherwise, network was okay so lets get the results if any
            if let results = resultsList {
                if (results.isEmpty) {
                    print("No matching addresses found")
                    self.lblAddress.text = "No matching addresses found"
                }
                else {
                    // if not empty, then you were able to find results!
                    // of type CLPlacemark
                    if let matchingLocation:CLPlacemark = results.first {
                        print(matchingLocation)

                        // NIL COALESCING
                        // try to assign city = matchingLocation.locality
                        // If matchingLocation.locatility = nil, then assign city =
"N/A"
                        let city = matchingLocation.locality ?? "N/A"
                        let province = matchingLocation.administrativeArea ?? "N/A"
                        let country = matchingLocation.country ?? "N/A"
                        let street = matchingLocation.thoroughfare ?? "N/A"

                        let address = "\(street), \(city), \(province), \(country)"
                        print("Final address: \(address)")
                        self.lblAddress.text = address
                    }
                }
            }
        }
    }
```



43.64196690

-79.38892370

Blue Jays Way, Toronto, ON, Canada