Higher-order Functions II

Overview

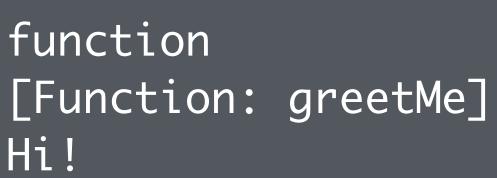
```
- Returning functions from functions
  - Closure
*/
```

Returning values from functions

```
/* we know that functions can return values */
   function returnsAString() {
     return 'I am a string';
   let returnedValue = returnsAString();
   console.log(returnedValue);
10
```

Returning values from functions

```
/* in the previous unit, we learned that functions are like any other
   value in JS */
/* so if functions can return values, and functions are values... */
/* ...functions can return other functions! */
```



Returning functions from functio

```
function greetMe() {
     console.log('Hi!');
   function getGreeter() {
     return greetMe; // note: we're returning the function without calling it
   let greeter = getGreeter();
10
   console.log(typeof greeter);
   console.log(greeter);
13 greeter();
```



Returning functions from functio

Hi!
undefined
undefined
TypeError: greeter
is not a function

```
function greetMe() {
     console.log('Hi!');
   function getGreeter() {
     return greetMe(); // what if we did invoke it?
   let greeter = getGreeter();
10
   console.log(typeof greeter);
   console.log(greeter);
13 greeter();
```

Returning functions from functions

```
function getGreeter() {
  // we can declare a new inner function and then return it
  function greetMe() {
    console.log('Hi!');
  return greetMe;
let greeter = getGreeter();
greeter();
```

Returning functions from functions

```
function getGreeter() {
 // we can return anonymous function, too
  return function() {
    console.log('Hi!');
 };
let greeter = getGreeter();
greeter();
```

Returning functions from functions

```
function getGreeter() {
  // what if our returned function takes a parameter?
  return function(name) {
    console.log('Hi', name);
  };
let greeter = getGreeter();
greeter('Marie');
greeter('Rosalind');
greeter();
```

Closure: scope refresher

```
/* before we talk about closure, let's review how scope works with nested
      functions */
   /* inner functions can access variables defined in outer functions */
   function greeter() {
     let name = 'Marie';
     let saysHi = function() {
       console.log('Hi', name);
     };
     saysHi();
14 greeter();
```



```
/* closure is the fact that an inner function can STILL to access values
   defined in the outer function even after the outer function is finished
   running! */
```

```
function getGreeter() {
     console.log('getGreeter is running');
     let name = 'Marie';
     console.log('getGreeter is finishing');
     return function() {
       console.log('Hi', name);
     };
13 let greeter = getGreeter();
14 greeter();
```

```
// what if getGreeter take a parameter?
function getGreeter(name) {
  return function() {
    console.log('Hi', name);
  };
let greeter = getGreeter('Rosalind');
greeter();
```

```
// what if both functions take parameters?
   function getGreeter(name1) {
     return function(name2) {
       console.log('Hi ' + name1 + ', meet ' + name2);
    };
   let greeter = getGreeter('Marie');
   greeter('Rosalind');
10
```

```
function getGreeter(name1) {
     return function(name2) {
       console.log('Hi' + name1 + ', meet ' + name2);
     };
   let introduceMarieTo = getGreeter('Marie');
   let introduceRosalindTo = getGreeter('Rosalind');
   introduceMarieTo('Dorothy');
   introduceMarieTo('Albert');
13 introduceRosalindTo('Barbara');
14 introduceRosalindTo('Isaac');
```

Hi Marie, meet Dorothy Hi Marie, meet Albert Hi Rosalind, meet Barbara Hi Rosalind, meet Isaac

Recap

```
- Returning functions from functions
  - Closure
*/
```