Visit the Wondrous

FOREST OF FUNCTION EXPRESSIONS

LEVEL 1

FOREST OF FUNCTION EXPRESSIONS

FUNCTIONS ON THE FLY

Building functions within code execution rather than at program load time

```
function diffOfSquares (a, b) {
    return a*a - b*b;
}
```

Builds in memory immediately when the program loads

FUNCTIONS ON THE FLY

Building functions within code execution rather than at program load time

```
function diffOfSquares (a, b) {
    return a*a - b*b;
}
```

Builds in memory immediately when the program loads

```
var diff =
```

FUNCTIONS ON THE FLY

Building functions within code execution rather than at program load time

```
function diffOfSquares (a, b) {
    return a*a - b*b;
}
```

Builds in memory immediately when the program loads

```
var diff =  function diffOfSquares (a, b) {
    return a*a - b*b;
};
```

The function keyword will now assign the following function to the variable.

FUNCTIONS ON THE FLY

Building functions within code execution rather than at program load time

```
function diffOfSquares (a, b) {
    return a*a - b*b;
}
```

Builds in memory immediately when the program loads

```
var diff =  function diffOfSquares (a, b) {
    return a*a - b*b;
};
```

Needs a semicolon to complete the assignment statement in a file.

FUNCTIONS ON THE FLY

Building functions within code execution rather than at program load time

```
function diffOfSquares (a, b) {
    return a*a - b*b;
}
```

Builds in memory immediately when the program loads

```
var diff =  function diffOfSquares (a, b) {
    return a*a - b*b;
};
```

Now the function builds ONLY when this line of code is reached.

```
diff( 9, 5 );
```

→ 56

FUNCTIONS ON THE FLY

Building functions within code execution rather than at program load time

```
function diffOfSquares (a, b) {
    return a*a - b*b;
}
```

This name is optional in JavaScript since we now use the variable name.

```
var diff =  function diffOfSquares (a, b) {
    return a*a - b*b;
};
```

```
diff( 9, 5 );
```

→ 56

Notice the variable name needs parentheses, parameters and a semicolon to execute the function it contains.

ANONYMOUS FUNCTIONS

No need for naming the function a second time

```
var diff =  function diffOfSquares (a, b) {
    return a*a - b*b;
};
```

ANONYMOUS FUNCTIONS

No need for naming the function a second time

```
var diff =  function                    (a, b) {
    return a*a - b*b;
};
```

ANONYMOUS FUNCTIONS

No need for naming the function a second time

```
var diff =  function (a, b) {
    return a*a - b*b;
};
```

Look Ma, no name!

```
diff( 4, 2 );
```

→ 12

Parameters are still passed to the variable name, which JavaScript believes is a function.

ANONYMOUS FUNCTIONS

No need for naming the function a second time

```
var diff =  function  (a, b) {
    return a*a - b*b;
};
```

```
console.log(diff);
```

```
→ function (a, b) {
      return a*a - b*b;
  }
```

Logging out using the variable name will display the entire function it contains.

STORED FUNCTIONS IN A NATIONAL PARK TERMINAL

A variable that holds a function can be passed into other functions

terminal.js

```
var greeting = function () {
    alert("Thanks for visiting the Badlands!\n" +
          "We hope your stay is...better than most.");
};


...


closeTerminal( greeting );


function closeTerminal( message ){
    ...
    message();
    ...
}
```

The greeting variable is passed in as a parameter to an existing declared function.

STORED FUNCTIONS IN A NATIONAL PARK TERMINAL

A variable that holds a function can be passed into other functions

terminal.js

```
var greeting = function () {
    alert("Thanks for visiting the Badlands!\n" +
          "We hope your stay is...better than most.");
};


...


closeTerminal( greeting );


function closeTerminal( message ){
    ...
    message();
    ...
}
```

```
function closeTerminal( message ){
    ...
    message();
    ...
}
```

STORED FUNCTIONS IN A NATIONAL PARK TERMINAL

A variable that holds a function can be passed into other functions

terminal.js

```
var greeting = function () {
    alert("Thanks for visiting the Badlands!\n" +
          "We hope your stay is...better than most.");
};


...


closeTerminal( greeting );


function closeTerminal( message ){
    ...
    message();
    ...
}
```

```
function closeTerminal(          ){
    ...
    message();
    ...
}
```

STORED FUNCTIONS IN A NATIONAL PARK TERMINAL

A variable that holds a function can be passed into other functions

terminal.js

```
var greeting = function () {
    alert("Thanks for visiting the Badlands!\n" +
          "We hope your stay is...better than most.");
};


...


closeTerminal( greeting );


function closeTerminal( message ){
    ...
    message();
    ...
}
```

```
function closeTerminal( greeting ){
    ...
    message();
    ...
}
```

STORED FUNCTIONS IN A NATIONAL PARK TERMINAL

A variable that holds a function can be passed into other functions

terminal.js

```
var greeting = function () {
    alert("Thanks for visiting the Badlands!\n" +
          "We hope your stay is...better than most.");
};

...

closeTerminal( greeting );


function closeTerminal( message ){
    ...
    message();
    ...
}
```

```
function closeTerminal( greeting ){
    ...
    greeting();
    ...
}
```

NOW, WHAT IF WE HAD MULTIPLE GREETINGS?

Function expressions can give flexibility in choosing which functionality to build

terminal.js

```
var greeting = function () {
    alert("Thanks for visiting the Badlands!\n" +
          "We hope your stay is...better than most.");
};

...

closeTerminal( greeting );


function closeTerminal( message ){
    ...
    message();
    ...
}
```

```
function closeTerminal( greeting ){
    ...
    greeting();
    ...
}
```

NOW, WHAT IF WE HAD MULTIPLE GREETINGS?

Function expressions can give flexibility in choosing which functionality to build

terminal.js

```
var greeting;
...some code sets newCustomer to true or false...
if( newCustomer ){
    greeting = function () {
        alert("Thanks for visiting the Badlands!\n" +
              "We hope your stay is...better than most.");
    };
} else {
    greeting = function () {
        alert("Welcome back to the Badlands!\n" +
              "Guess they aren't so bad huh?");
    };
}
closeTerminal( greeting );
function closeTerminal( message ){
    ...
    message();
    ...
}
```

```
var newCustomer = false;
```

NOW, WHAT IF WE HAD MULTIPLE GREETINGS?

Function expressions can give flexibility in choosing which functionality to build

terminal.js

```
var greeting;
...some code sets newCustomer to true or false...
if( newCustomer ){
    greeting = function () {
        alert("Thanks for visiting the Badlands!\n" +
              "We hope your stay is...better than most.");
    };
} else {
    greeting = function () {
        alert("Welcome back to the Badlands!\n" +
              "Guess they aren't so bad huh?");
    };
}
closeTerminal( greeting );
function closeTerminal( message ){
    ...
    message();
    ...
}
```

```
var newCustomer = true;
```

Visit the Wondrous

FOREST OF FUNCTION EXPRESSIONS