Arrow Functions

```
() => { }
```

Full Stack Web Development

“Regular” Function Variables

Regular function can be assigned to a variable

```
let getData = function getData() {
  //...
}
```

The variable can be used to invoke the function

```
getData()
```

The function may be anonymous

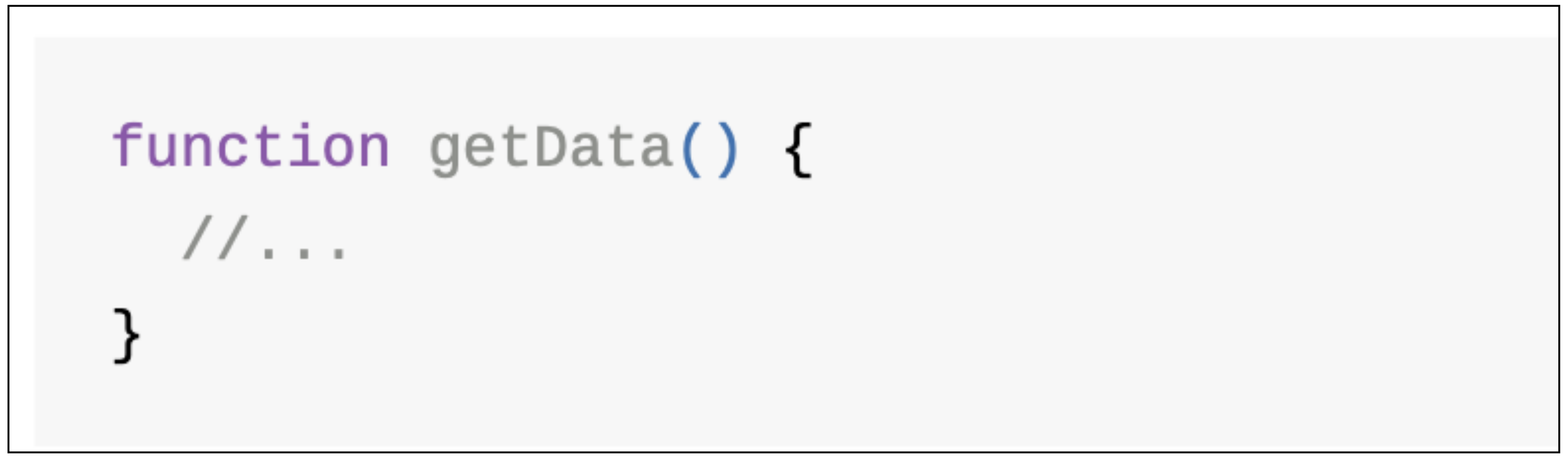
```
let getData = function() {
  //...
}
```

Arrow Functions

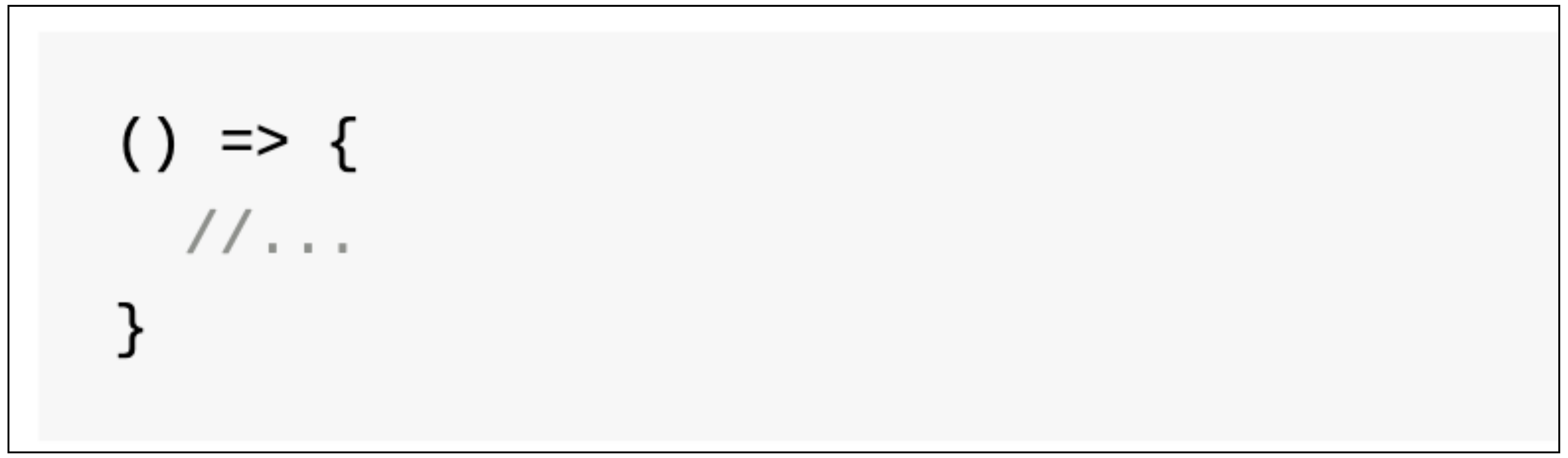
Arrow functions are a recent introduction to JavaScript.

They are very often used instead of "regular" functions

Visually, they allows you to write functions with a shorter syntax



```
function getData() {
  //...
}
```



```
() => {
  //...
}
```

Arrow Function Variables

- Arrow functions are always anonymous.

```
let getData = function getData() {
  //...
}
```

Is equivalent to

```
let getData = () => {
  //...
}
getData()
```

- They can be assigned to a variable and invoked

Single Statement Functions

If the function body contains just a single statement, you can omit the parentheses and write all on a single line

```
const getData = () => console.log('hi!')
```

Parameters are passed in the parentheses:

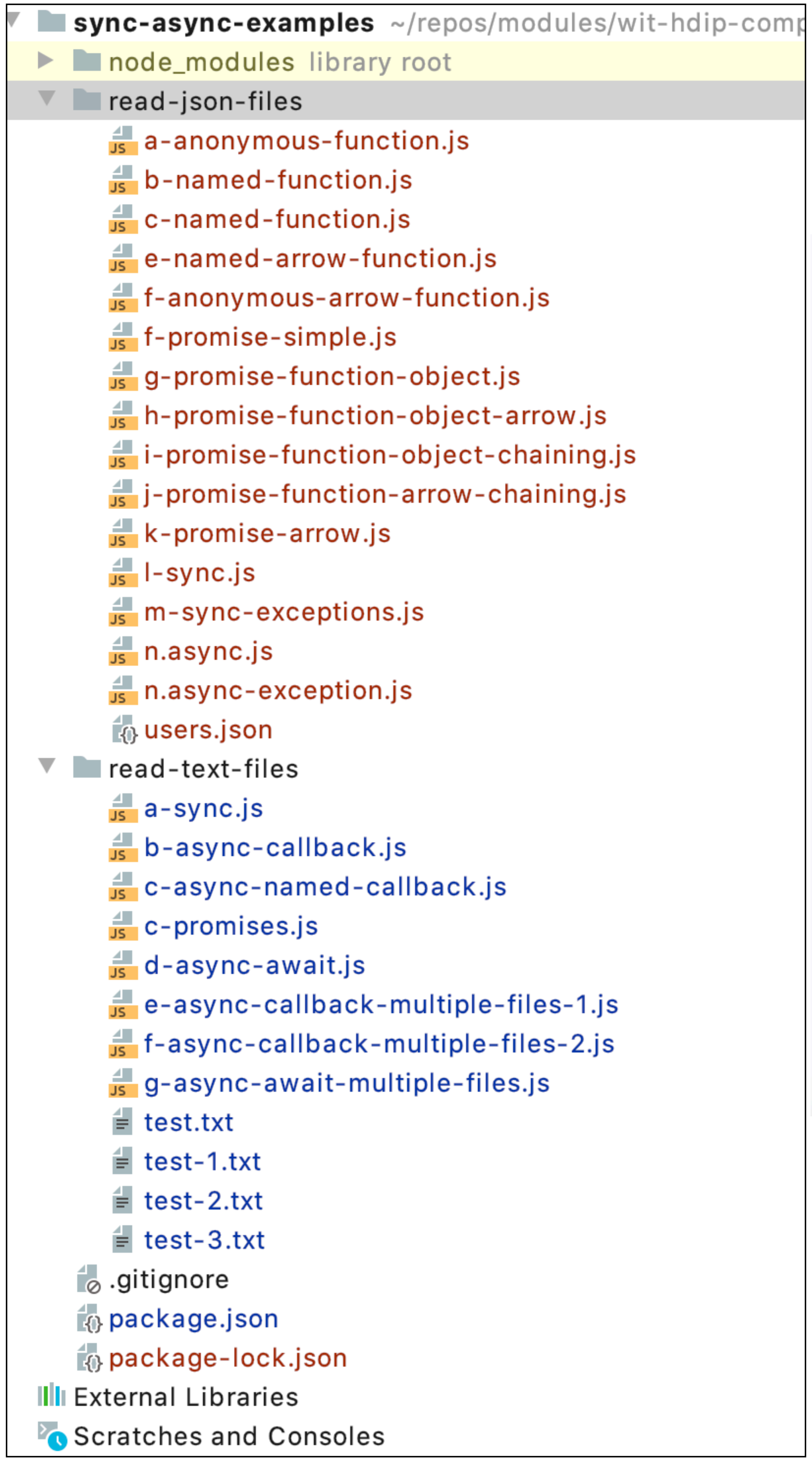
```
const getData = (param1, param2) =>
  console.log(param1, param2)
```

If you have one (and just one) parameter, you could omit the parentheses completely:

```
const getData = param => console.log(param)
```

Function Styles

- Anonymous Function
- Named Function
- Named Function (variable)
- Named Arrow Function
- Anonymous Arrow Function



```
sync-async-examples ~/repos/modules/wit-hdip-comp
  node_modules library root
  read-json-files
    a-anonymous-function.js
    b-named-function.js
    c-named-function.js
    e-named-arrow-function.js
    f-anonymous-arrow-function.js
    f-promise-simple.js
    g-promise-function-object.js
    h-promise-function-object-arrow.js
    i-promise-function-object-chaining.js
    j-promise-function-arrow-chaining.js
    k-promise-arrow.js
    l-sync.js
    m-sync-exceptions.js
    n.async.js
    n.async-exception.js
    users.json
  read-text-files
    a-sync.js
    b-async-callback.js
    c-async-named-callback.js
    c-promises.js
    d-async-await.js
    e-async-callback-multiple-files-1.js
    f-async-callback-multiple-files-2.js
    g-async-await-multiple-files.js
    test.txt
    test-1.txt
    test-2.txt
    test-3.txt
  .gitignore
  package.json
  package-lock.json
External Libraries
Scratches and Consoles
```

Node Sync/Async Examples

Source project for the node sync/async examples discussed in the talk.

Anonymous Function

```
fs.readFile("users.json", function(error, text) {
  if (error) {
    console.error(error.message);
  } else {
    try {
      var obj = JSON.parse(text);
      console.log(obj);
    } catch (e) {
      console.error(err.message);
    }
  }
});
```

Named Function

```
function readFileSimple(error, text) {
  if (error) {
    console.error(error.message);
  } else {
    try {
      var obj = JSON.parse(text);
      console.log(obj);
    } catch (e) {
      console.error(err.message);
    }
  }
}

fs.readFile('users.json', readFileSimple)
```

Named Function Variable

```
const readFileFunc = function(error, text) {
  if (error) {
    console.error(error.message);
  } else {
    try {
      var obj = JSON.parse(text);
      console.log(obj);
    } catch (err) {
      console.error(err.message);
    }
  }
};

fs.readFile('users.json', readFileFunc);
```

Named Arrow Function

```
const readFileArrow = (error, text) => {
  if (error) {
    console.error(error);
  } else {
    try {
      var obj = JSON.parse(text);
      console.log(obj);
    } catch (err) {
      console.error(err.message);
    }
  }
};

fs.readFile('users.json', readFileArrow);
```

Anonymous Arrow Function

```
fs.readFile("users.json", (error, text) => {
  if (error) {
    console.error(error);
  } else {
    try {
      var obj = JSON.parse(text);
      console.log(obj);
    } catch (err) {
      console.error(err.message);
    }
  }
});
```

Arrow Functions

```
() => {}
```

Full Stack Web Development