

CS 312 Software Development

Introduction to JavaScript

Learning JavaScript (in CS312)

JavaScript is an object-oriented, prototype-based, dynamic, “brackets” language

A pragmatic language that “evolved” (instead of being “designed”)

Gotchas abound

Recent versions have smoothed some rough edges (e.g. introduced “classes”)

Declaring variables

~~no declaration~~

implicitly create a new global variable

~~var~~

create new variable with function (or global) scope

variables are *hoisted* to the top of their context

let

create new variable with block-level scope

const

create a new constant with block-level scope

Declaring functions

Function declaration

```
function double(x) {  
  return x * 2;  
}
```

Function expression (fat arrow syntax)

```
const double = (x) => {  
  return x * 2;  
};
```

Function expression

```
const double = function (x) {  
  return x * 2;  
};
```

Function expression (fat arrow, implicit return)

```
const double = (x) => x * 2;
```

Named function expression

```
const double = function f(x) {  
  return x * 2;  
};
```

CS 312 Software Development

Introduction to JavaScript: Higher-order functions

Higher-order functions

Abstract over "actions" not just values by passing functions as arguments

```
const m = [4,6,2,7];
for (let i=0; i<m.length; i++) {
  console.log(m[i]);
}
```

```
const m = [4,6,2,7];
m.forEach((i) => { console.log(i); });
```

Common operations of this kind are `forEach`, `map`, `filter`, `reduce` and `sort`

map vs. forEach

```
const m = [4,6,2,7];
m.forEach((i) => { console.log(i); });
```

```
const m = [4,6,2,7];
m.map((i) => { console.log(i); });
```

4
6
2
7
undefined

4
6
2
7
[undefined, undefined, undefined, undefined]

```
const m = [4,6,2,7];
m.forEach((i) => 2 * i);
```

undefined

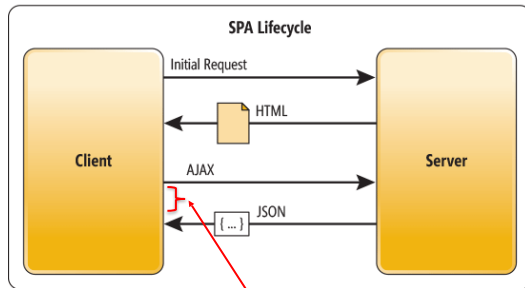
```
const m = [4,6,2,7];
m.map((i) => 2 * i);
```

[8, 12, 4, 14]

CS 312 Software Development

Introduction to JavaScript: Callbacks and closures

Callbacks



What is happening during this time?

Wasson_Microsoft

Making callbacks work in JS

```
const wrapValue = (n) => { // function(n) {
  const local = n;
  return () => local; // function () { return local; }
}

const wrap1 = wrapValue(1);
const wrap2 = wrapValue(2);
console.log(wrap1()); // What will print here? // () => 1
console.log(wrap2()); // What will print here? // () => 2
```

Function "closes" over local

What does the following code print?

```
let current = Date.now(); // Time in ms since epoch
const action = () => { console.log("Time elapsed (ms): " + (Date.now() - current));

// setTimeout(callback, delay[,param1[,param2...]]) delay in ms
setTimeout(action, 100);

console.log("First?")
```

A	B	C
First?	First? Time elapsed (ms): 100	Time elapsed (ms): 100 First?

What does the following code print?

```
let current = Date.now(); // Time in ms since epoch

// setTimeout(callback, delay[,param1[,param2...]]) delay in ms
setTimeout(() => {
  console.log("Time elapsed (ms): " + (Date.now() - current))
}, 100);

console.log("First?")
```

A	B	C
First?	First? Time elapsed (ms): 100	Time elapsed (ms): 100 First?

Closures

```
const version1 = ()=>{
  let current = Date.now(); // Time in ms since epoch

  // setTimeout(callback, delay[,param1[,param2...]]) delay in ms
  setTimeout(() => {
    console.log("Time elapsed (ms): " + (Date.now() - current))
  }, 100);

  console.log("First?");
}
```

Function "closes" over current variable

Output

```
> version1()
First?
undefined
> Time elapsed (ms): 101
```

version1 doesn't return a value, so this marks the completion of the function

Closures

```
const version2 = ()=>{
  let current = Date.now(); // Time in ms since epoch

  // setTimeout(callback, delay[,param1[,param2...]]) delay in ms
  setTimeout(() => {
    console.log("Time elapsed (ms): " + (Date.now() - current))
  }, 100);

  console.log("First?");
  current = new Date('1 Jan 2000');
}
```

what will happen now?

Output

```
> version2()
First?
undefined
> Time elapsed (ms): 666181055705
```

Closures

```
const version3 = ()=>{
  let current = Date.now(); // Time in ms since epoch

  // setTimeout(callback, delay[,param1[,param2...]]) delay in ms
  setTimeout((past) => {
    console.log("Time elapsed (ms): " + (Date.now() - past))
  }, 100, current);

  console.log("First?");
  current = new Date('1 Jan 2000');
}
```

Output

```
> version3()
First?
undefined
> Time elapsed (ms): 100
```