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

- **Function expression**
  ```javascript
  const double = function (x) {
    return x * 2;
  };
  ```

- **Named function expression**
  ```javascript
  const double = function f(x) {
    return x * 2;
  };
  ```

- **Function expression (fat arrow syntax)**
  ```javascript
  const double = (x) => {
    return x * 2;
  };
  ```

- **Function expression (fat arrow, implicit return)**
  ```javascript
  const double = (x) => x * 2;
  ```
Higher-order functions

Abstract over “actions” not just values by passing functions as arguments.

```javascript
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

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

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

### SPA Lifecycle

- **Client**
  - HTML
  - AJAX
  - JSON

- **Server**

What is happening during this time?

---

## Making callbacks work in JS

```javascript
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?
console.log(wrap2()); // What will print here?
```

Function "closes" over `local`

- `() => 1`
- `() => 2`

---

## What does the following code print?

```javascript
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?")
```

---

## What does the following code print?

```javascript
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?")
```

---

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>First?</td>
<td>Time elapsed (ms): 100</td>
<td>First? Time elapsed (ms): 100</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>First?</td>
<td>First? Time elapsed (ms): 100</td>
<td>First?</td>
</tr>
</tbody>
</table>

---
Closures

```javascript
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?");
}
> version1()
First?
undefined
> Time elapsed (ms): 101

Output
```

Function "closes" over current variable

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

Output
```

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

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

Output
```