HTML+CSS+JS SL LISA XU

ES6 Promises: (31)

time(long) delay task

```
function asyncFunc(work){ work can be func
  return new Promise(function(resolve,reject)
{
    if (work === "") setTimeout(work,time)
      reject(Error("Nothing")); remember
    setTimeout(function(){ // resolve
      resolve(work); is the method for
    }, 1000); success and reject is the
  }); method for failure. it a method
} returns a promise, its calls should
asyncFunc("work 1") use the then method
.then(function(result){ which takes two
  console.log(result); methods as input,
  return asyncFunc("work 2"); one for success
}, function(error){ and the other for failure
  console.log(error);
})
.then(function(result){
  console.log(result);
}, function(error){
  console.log(error);
});
console.log("End");
```

Iterators & Generators

Symbol.iterator — default iterator for an object

```
function* idMaker(){
  let index = 0;
  while (index < 5)
    yield index++;
}
var gen = idMaker();
console.log(gen.next().value);
```

Modules

```
// lib/example.js
export let sum = (x, y) => { return x + y; }
export let pi = 3.14;

// app.js
import * as exa from "lib/example"
console.log('2p = ' + ${exa.sum(exa.pi, exa.pi)});
```

Built-in methods: finding, repeating, searching — common ones

```
old: [4,5,7,1,2].filter(function(x){ return x > 3; })[0]
```

return the first

ES6: `[4,5,7,1,2].find(x => x > 3);` third index

~~`[4,5,7,1,2].findIndex(x => x > 3);`~~

Before: `console.log(Array(3+1).join("foo"));`

// foofoofoo

ES6: `console.log("foo".repeat(3));`

searching: ES6:

```
.startsWith(" ", num);
.endsWith(" ", num);
.includes(" ");
.includes(" ", num);
```