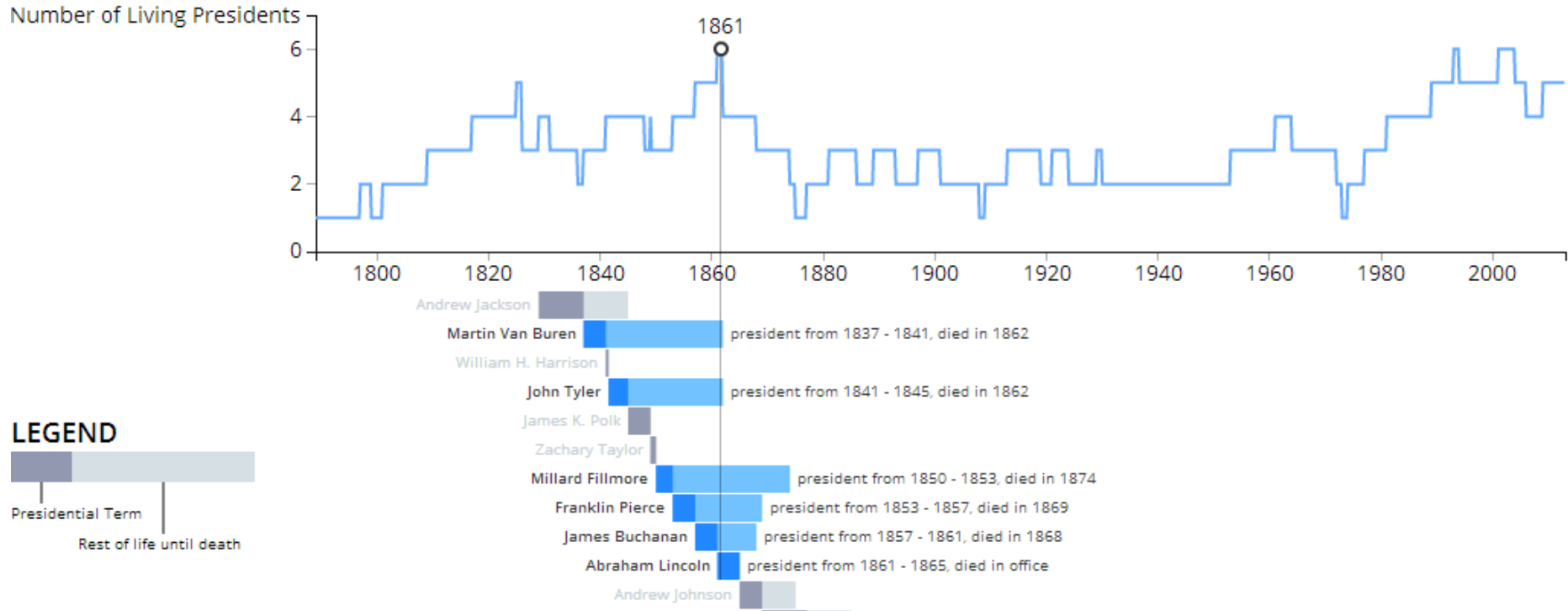


7

Beginning with JavaScript

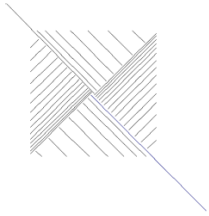
A friendly crash course in a friendly language



<http://www.ravi.io/living-us-presidents>
<http://www.ravi.io/living-us-presidents-2>

The course so far...

The course so far...



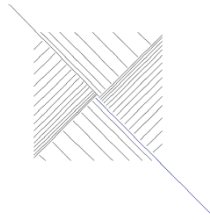
1

Introduction to Visualization

The midpoint between art and engineering



Kevin McVey
CS1501 – Fall 2013



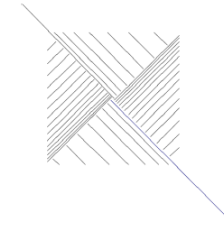
2

Graphical Excellence

The whole and the sum of its parts



Kevin McVey
CS1501 – Fall 2013



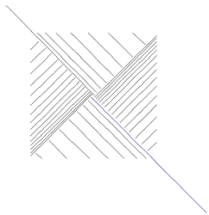
3

Chartjunk

Where design and function part ways



Kevin McVey
CS1501 – Fall 2013



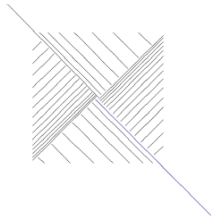
4

Communication Design

Images that speak for themselves



Kevin McVey
CS1501 – Fall 2013



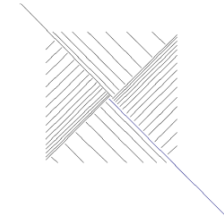
5

Data Ink

Brevity and the soul of visualization



Kevin McVey
CS1501 – Fall 2013



6

Evaluation and Integrity

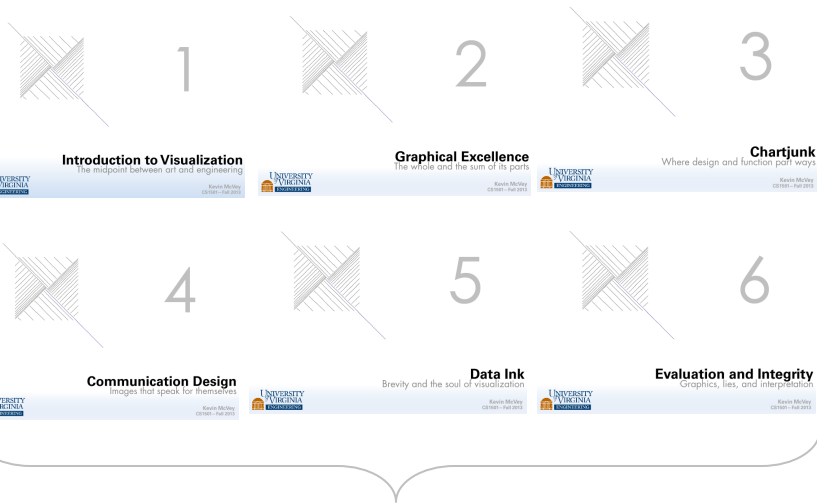
Graphics, lies, and interpretation



Kevin McVey
CS1501 – Fall 2013

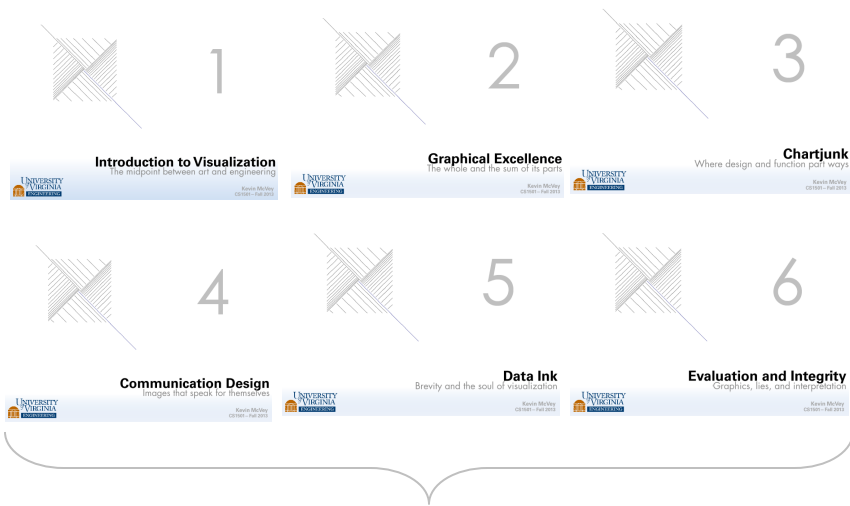
The course so far...

Building the right thing



The course so far...

Building the right thing



1 **Introduction to Visualization**
The midpoint between art and engineering

2 **Graphical Excellence**
The whole and the sum of its parts

3 **Chartjunk**
Where design and function part ways

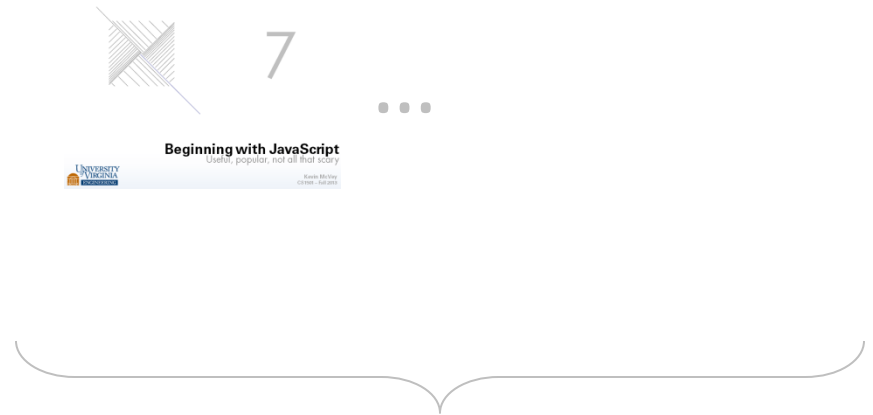
4 **Communication Design**
Images that speak for themselves

5 **Data Ink**
Brevity and the soul of visualization

6 **Evaluation and Integrity**
Graphics, lies, and interpretation

A large curly bracket is positioned below these six thumbnails, grouping them together.

Building the thing right

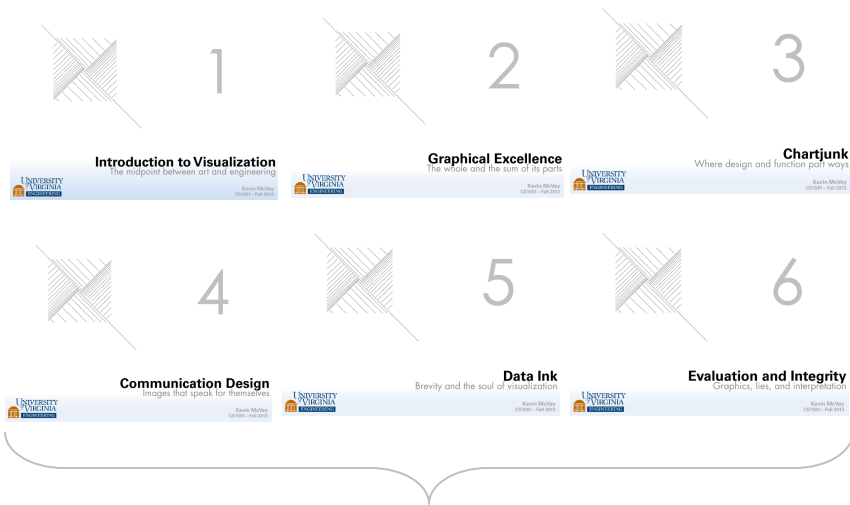


7 **Beginning with JavaScript**
Useful, popular, not all that scary

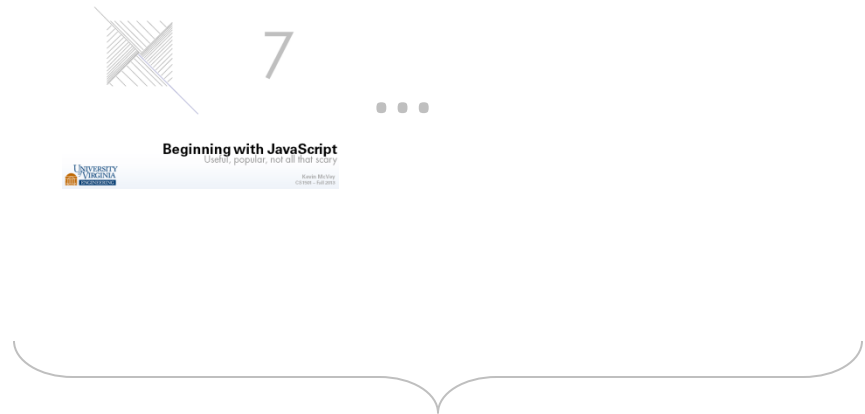
A large curly bracket is positioned below this single thumbnail, grouping it.

The course so far...

Building the right thing



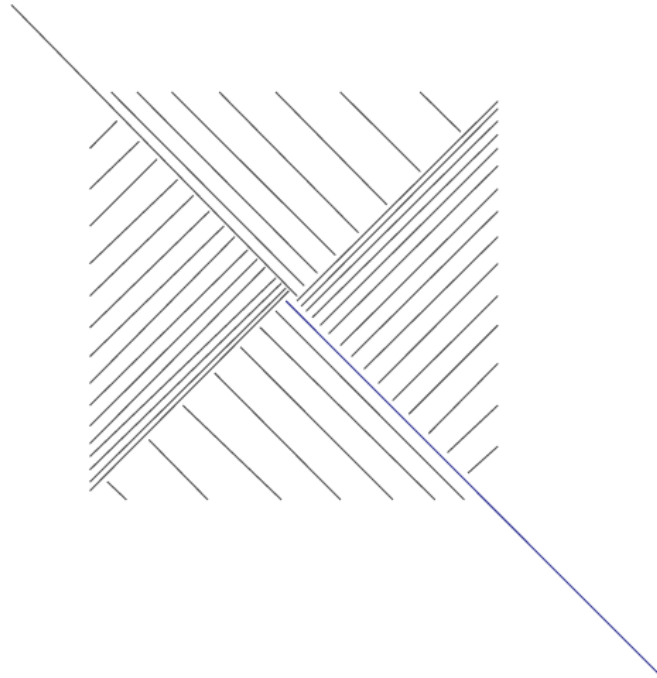
Building the thing right



Success in design

But in case that's not motivation enough...

Announcing a new course partnership



+



VividCortex

Announcing a new course partnership

VividCortex is a SaaS-based MySQL monitoring and analysis company founded by Baron Schwartz and Kyle Redinger. We create amazing tools for systems management, so you can run your servers better with less cost and effort. We're delivering *Technical Intelligence for MySQL*.



VividCortex

Java

Class-based, object-oriented, *compiles* to the Java Virtual Machine

JavaScript

Prototype-based, dynamically typed, *interpreted* scripting language

Java

Class-based, object-oriented, compiles to the Java Virtual Machine

IS NOT

JavaScript

Prototype-based, dynamically typed, *interpreted* scripting language

Let's jump in

If you have a laptop, open Chrome. Hit F12 and go to Console.

Let's jump in

If you have a laptop, open Chrome. Hit F12 and go to Console.

"Kevin"

3

1+1

1/4

1e2

false

TheMeaningOfLife

Let's jump in

If you have a laptop, open Chrome. Hit F12 and go to Console.

"Kevin"	→	"Kevin"
3	→	3
1+1	→	2
1/4	→	0.25
1e2	→	100
false	→	false
TheMeaningOfLife	→	ReferenceError

Let's jump in

If you have a laptop, open Chrome. Hit F12 and go to Console.

"Kevin"	→	"Kevin"
3	→	3
1+1	→	2
1/4	→	0.25
1e2	→	100
false	→	false
TheMeaningOfLife	→	ReferenceError

“Oh man oh man, what’s going on here?”
-you

JSON

JavaScript Object Notation
or: What is data and what isn't

JSON

JavaScript Object Notation
or: What is data and what isn't

Number 3, 2.335, 2.3e5

JSON

JavaScript Object Notation
or: What is data and what isn't

Number 3, 2.335, 2.3e5

String “amanaplanacanalpanama”, “Hello World!”

JSON

JavaScript Object Notation
or: What is data and what isn't

Number 3, 2.335, 2.3e5

String “amanaplanacanalpanama”, “Hello World!”

Boolean false, true

JSON

JavaScript Object Notation
or: What is data and what isn't

Number 3, 2.335, 2.3e5

String “amanaplanacanalpanama”, “Hello World!”

Boolean false, true

Array [0, 1, 2, 3], [“hello”, 2.3e5, “world!”, false]

JSON

JavaScript Object Notation
or: What is data and what isn't

Number 3, 2.335, 2.3e5

String “amanaplanacanalpanama”, “Hello World!”

Boolean false, true

Array [0, 1, 2, 3], [“hello”, 2.3e5, “world!”, false]

Object {“Name”: “CS1501”, “Credits”: 1, “Your Favorite”: true}

JSON

JavaScript Object Notation
or: What is data and what isn't

Number 3, 2.335, 2.3e5

String “amanaplanacanalpanama”, “Hello World!”

Boolean false, true

Array [0, 1, 2, 3], [“hello”, 2.3e5, “world!”, false]

Object {“Name”: “CS1501”, “Credits”: 1, “Your Favorite”: true}

null null

Variables

“Objects you can store and use later!”

Variables

“Objects you can store and use later!”

Number, String, Boolean, Array, Object, Null? Use var.

Variables

“Objects you can store and use later!”

Number, String, Boolean, Array, Object, Null? Use var.

```
var name = "Kevin";  
name + " is the best teacher.";
```

→ “Kevin is the best teacher.”

Variables

“Objects you can store and use later!”

Number, String, Boolean, Array, Object, Null? Use var.

```
var name = "Kevin";  
name + " is the best teacher.";
```

→ “Kevin is the best teacher.”

```
var degrees_f = 98.6;  
var degrees_c = (degrees_f-32)*(5/9);  
degrees_f + "F is equal to " + degrees_c + "C";
```

→ 98.6F is equal to 37C

Variables

“Objects you can store and use later!”

Variables in JavaScript are “dynamically typed.”

Variables

“Objects you can store and use later!”

Variables in JavaScript are “dynamically typed.”

5>true;  6

Variables

“Objects you can store and use later!”

Variables in JavaScript are “dynamically typed.”

5>true; → 6

“hello”+true; → hellotrue

Variables

“Objects you can store and use later!”

Variables in JavaScript are “dynamically typed.”

5>true; → 6

“hello”+true; → hellotrue

```
var my_object = {"hello": "world"};
var my_name = "kevin";
var confusing = my_name + my_object;
confusing
```

→ “kevin[object Object]”

Variables

“Objects you can store and use later!”

Variables in JavaScript are “dynamically typed.”

`5>true;` → 6

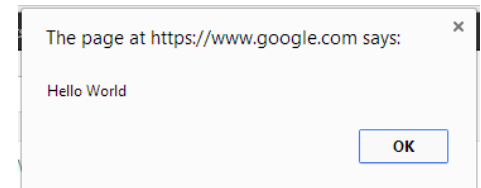
`“hello”+true;` → `hellotrue`

```
var my_object = {“hello”: “world”};  
var my_name = “kevin”;  
var confusing = my_name + my_object;  
confusing
```

→ “kevin[object Object]”

```
var my_var = function() {  
  alert(“Hello World!”);  
};
```

→ `my_var();` →



“Golly gee! Variables sure are cool. What can I do with them?”
- you

“Golly gee! Variables sure are cool. What can I do with them?”

- you

MATH

“Golly gee! Variables sure are cool. What can I do with them?”

- you

MATH

COMPARISON

“Golly gee! Variables sure are cool. What can I do with them?”

- you

MATH

COMPARISON

BRANCHING

“Golly gee! Variables sure are cool. What can I do with them?”
- you

MATH

COMPARISON

LOOPING

BRANCHING

“Golly gee! Variables sure are cool. What can I do with them?”

- you

MATH

HEART!

(first-class functions)

COMPARISON

LOOPING

BRANCHING

“Golly gee! Variables sure are cool. What can I do with them?”

- you

MATH

HEART!

(first-class functions)



COMPARISON

LOOPING

BRANCHING

“Golly gee! Variables sure are cool. What can I do with them?”

MATH

- you

```
var a = 100;  
var b = 5;
```

“Golly gee! Variables sure are cool. What can I do with them?”

MATH

- you

```
var a = 100;
```

```
var b = 5;
```

```
a + b;
```

```
a - b;
```

```
a * b;
```

```
a / b;
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

MATH

```
var a = 100;
```

```
var b = 5;
```

```
a + b; 105
```

```
a - b; 95
```

```
a * b; 500
```

```
a / b; 20
```

(easy!)

“Golly gee! Variables sure are cool. What can I do with them?”

- you

MATH

```
var a = 100;
```

```
var b = 5;
```

```
Math.log(a);
```

```
Math.min(a, b);
```

```
Math.pow(a, b);
```

```
Math.sin(a) / Math.cos(b);
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

MATH

```
var a = 100;
```

```
var b = 5;
```

```
Math.log(a); 4.605170185988092
```

```
Math.min(a, b); 5
```

```
Math.pow(a, b); 10000000000
```

```
Math.sin(a) / Math.cos(b); -1.7851009653713736
```

(cool!)

“Golly gee! Variables sure are cool. What can I do with them?”

- you

COMPARISON

```
var a = 100;
```

```
var b = 5;
```

```
var c = 5;
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

COMPARISON

```
var a = 100;
```

```
var b = 5;
```

```
var c = 5;
```

```
a > b
```

```
a < b
```

```
b < c
```

```
b <= c
```

```
a == b
```

```
b == c
```

```
a != b
```

```
c != b
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

COMPARISON

```
var a = 100;
```

```
var b = 5;
```

```
var c = 5;
```

```
a > b true
```

```
a < b false
```

```
b < c false
```

```
b <= c true
```

```
a == b false
```

```
b == c true
```

```
a != b true
```

```
c != b false
```


“Golly gee! Variables sure are cool. What can I do with them?”

- you

COMPARISON

```
var a = 100;  
var b = 5;  
var c = 5;
```

a > b true

a < b false

b < c false

b <= c true

a == b false

b == c true

a != b true

c != b false

PAY ATTENTION.

a = b : assign value of b to a

a == b : check if a equals b

“Golly gee! Variables sure are cool. What can I do with them?”

- you

COMPARISON

```
var a = 100;  
var b = 5;  
var c = 5;
```

REMEMBER.
JavaScript's dynamic
typing can allow...

PAY ATTENTION.

a = b : assign value of b to a
a == b : check if a equals b

```
a > b true  
a < b false  
b < c false  
b <= c true  
a == b false  
b == c true  
a != b true  
c != b false
```

```
a > true true  
"hello" < false false  
[1, 2, 3] < true false  
"test" == true false  
"test" == false false  
"test" != true true  
"test" != false true
```

(learning!)

“Golly gee! Variables sure are cool. What can I do with them?”

- you

BRANCHING

```
var a = 10;
```

```
var b = 5;
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

BRANCHING

```
var a = 10;
```

```
var b = 5;
```



a > b?



Yes!



No!

“Golly gee! Variables sure are cool. What can I do with them?”

- you

BRANCHING

```
var a = 10;  
var b = 5;
```



a > b?



Yes!

No!



```
launch_missiles(); do_nothing();
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

BRANCHING

```
var a = 10;  
var b = 5;
```



a > b?



Yes!

No!



```
launch_missiles(); do_nothing();
```

```
var a = 10;  
var b = 5;
```

```
if(a > b) {  
    launch_missiles();  
} else {  
    do_nothing();  
}
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

BRANCHING

```
var a = 10;  
var b = 5;
```



a > b?



Yes!

No!



```
launch_missiles(); do_nothing();
```

```
var a = 10;  
var b = 5;
```

```
if(a > b) {  
→ launch_missiles();  
} else {  
    do_nothing();  
}
```



“Golly gee! Variables sure are cool. What can I do with them?”

- you

BRANCHING

```
var a = Math.floor(Math.random() * 5);
```


“Golly gee! Variables sure are cool. What can I do with them?”

- you

BRANCHING

```
var a = Math.floor(Math.random() * 5);
```

```
if(a == 0) {  
    // do thing 0  
} else if(a == 1) {  
    // do thing 1  
} else if(a == 2) {  
    // do thing 2  
} else if(a == 3) {  
    // do thing 3  
} else if(a == 4) {  
    // do thing 4  
}
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

BRANCHING

```
var a = Math.floor(Math.random() * 5);
```

```
if(a == 0) {  
    // do thing 0  
} else if(a == 1) {  
    // do thing 1  
} else if(a == 2) {  
    // do thing 2  
} else if(a == 3) {  
    // do thing 3  
} else if(a == 4) {  
    // do thing 4  
}
```

You can carry this out as much as you like!

Lines starting with // are **comments**

They do not get run and are for your eyes only.

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var a = 0;
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var a = 0;
```



```
a % 3 == 0?
```



Yes!

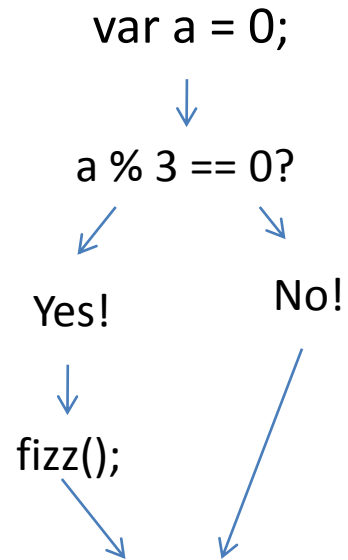


No!

“Golly gee! Variables sure are cool. What can I do with them?”

- you

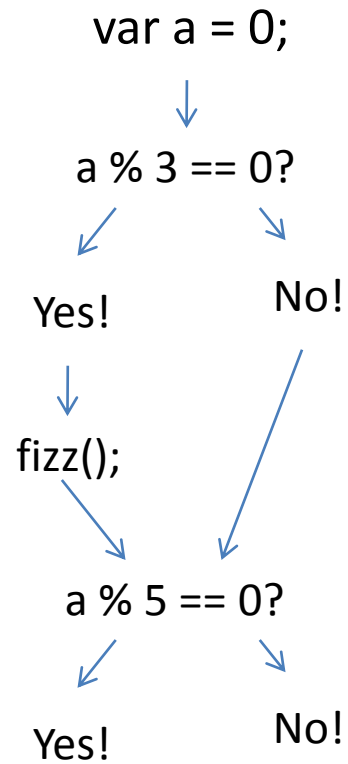
LOOPING



“Golly gee! Variables sure are cool. What can I do with them?”

- you

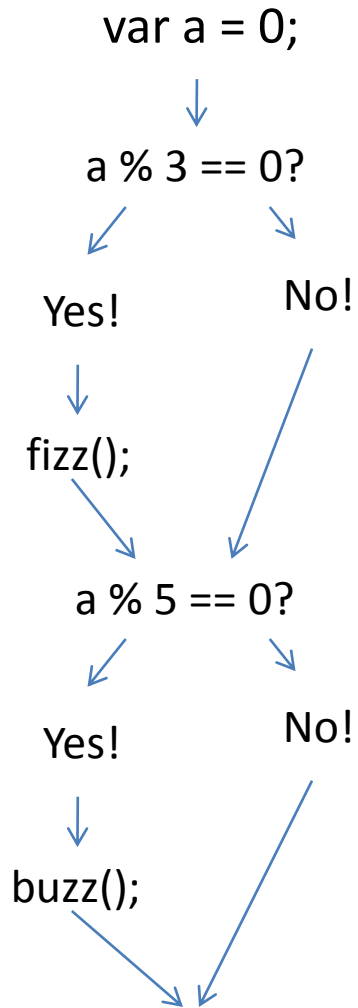
LOOPING



“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

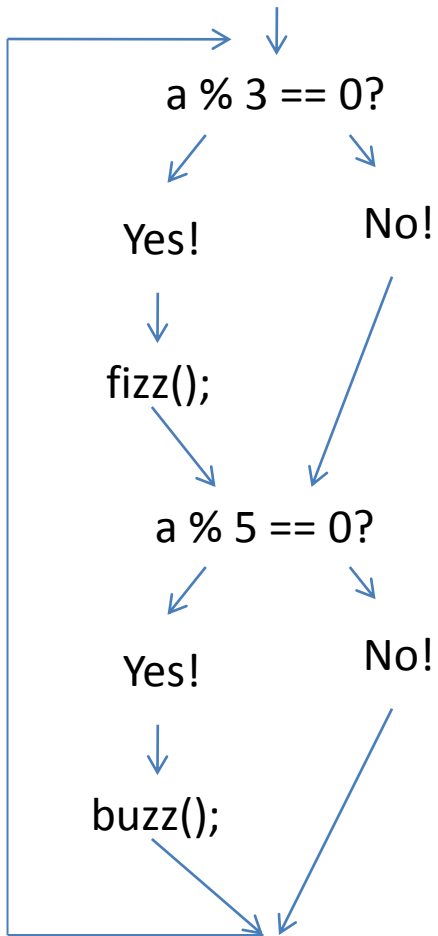


“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var a = 0;
```

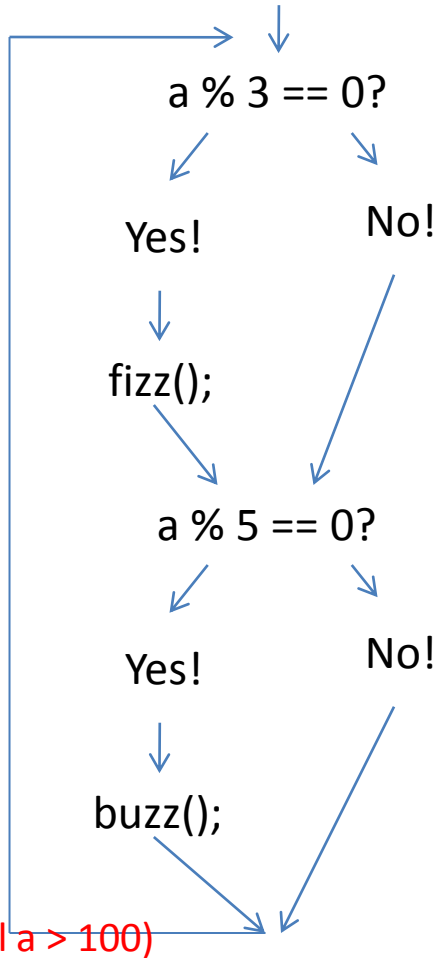


“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var a = 0;
```



“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var a = 0;
```

$a \% 3 == 0?$

Yes!

```
fizz();
```

$a \% 5 == 0?$

Yes!

```
buzz();
```

No!

No!

$\%$: “modulus” or “mod”
aka: remainder from division

(until $a > 100$)

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var a = 0;
```

```
a % 3 == 0?
```

Yes!

No!

```
fizz();
```

```
a % 5 == 0?
```

Yes!

No!

```
buzz();
```

% : “modulus” or “mod”
aka: remainder from division

2 % 1

5 % 2

213 % 10

(until a > 100)

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var a = 0;
```

```
a % 3 == 0?
```

Yes!

No!

```
fizz();
```

```
a % 5 == 0?
```

Yes!

No!

```
buzz();
```

% : “modulus” or “mod”
aka: remainder from division

2 % 1 **0**

5 % 2 **1**

213 % 10 **3**

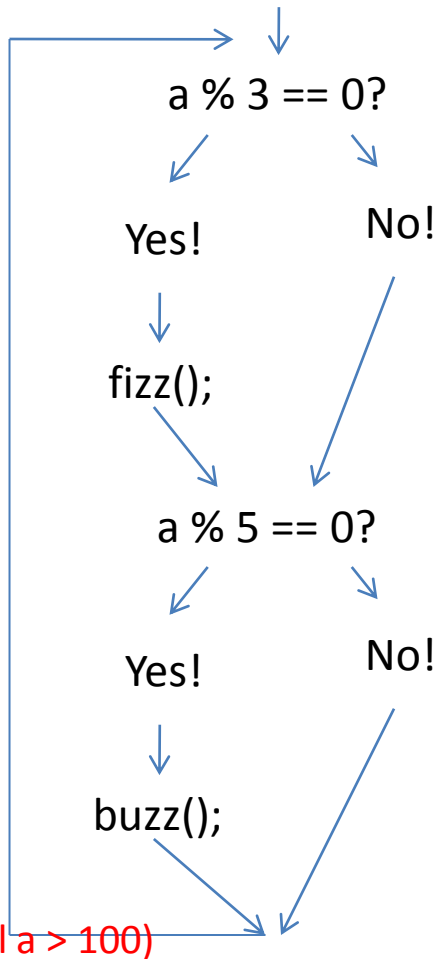
(until a > 100)

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var a = 0;
```



“While” loop

```
var a = 0;
```

```
while(a <= 100) {  
  if(a % 3 == 0) {  
    fizz();  
  }
```

```
  if(a % 5 == 0) {  
    buzz();  
  }
```

```
  a = a + 1;
```

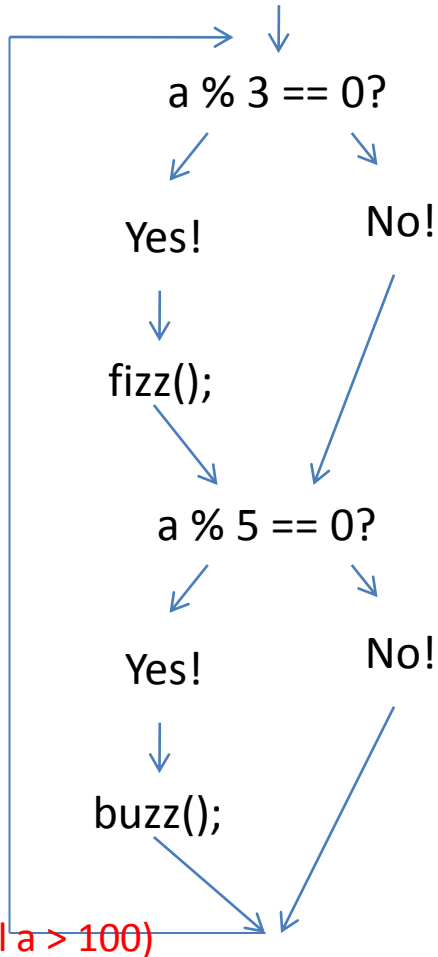
```
}
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var a = 0;
```



“While” loop

```
var a = 0;

while(a <= 100) {
  if(a % 3 == 0) {
    fizz();
  }

  if(a % 5 == 0) {
    buzz();
  }

  a = a + 1;
}
```

“For” loop

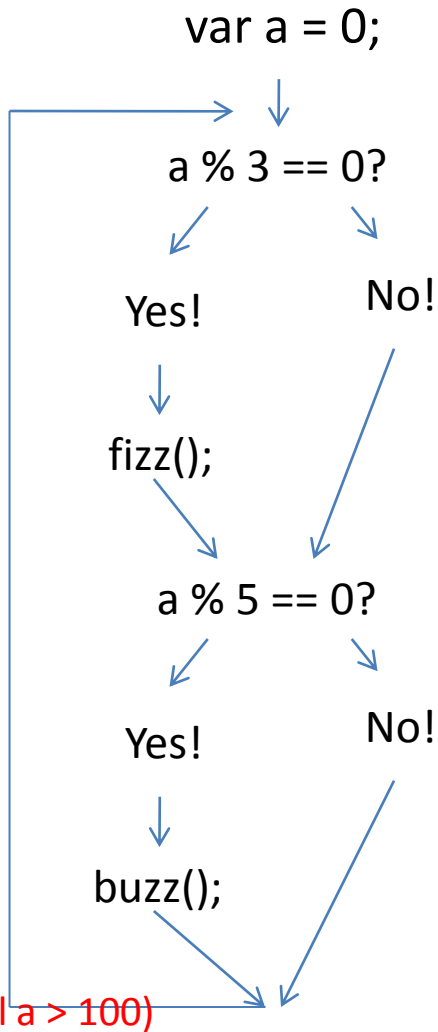
```
for(var a = 0; a <= 100; a = a + 1) {
  if(a % 3 == 0) {
    fizz();
  }

  if(a % 5 == 0) {
    buzz();
  }
}
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING



“While” loop

```
var a = 0; Initialization
while(a <= 100) Condition
    if(a % 3 == 0) {
        fizz();
    }
    if(a % 5 == 0) {
        buzz();
    }
    a = a + 1; Afterthought
}
```

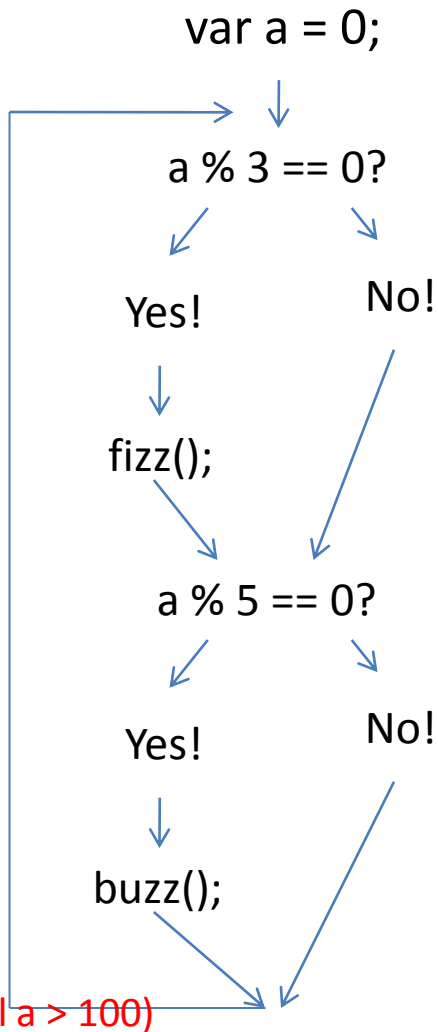
“For” loop

```
for(var a = 0; a <= 100; a = a + 1) {
    if(a % 3 == 0) {
        fizz();
    }
    if(a % 5 == 0) {
        buzz();
    }
}
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING



“While” loop

```
var a = 0; Initialization  
while(a <= 100) Condition  
  if(a % 3 == 0) {  
    fizz();  
  }  
  if(a % 5 == 0) {  
    buzz();  
  }  
  a = a + 1; Afterthought  
}
```

“For” loop

Initialization	Condition	Afterthought
for(var a = 0; a <= 100; a = a + 1) {	if(a % 3 == 0) {	fizz();
	if(a % 5 == 0) {	buzz();
	a = a + 1;	
}		

“I bet this is really useful for data-rich projects, right?”
- still you

“I bet this is really useful for data-rich projects, right?”
- still you

BINGO.

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var data = [100, 95, 84, 74, 90, 91, 60, 91, 87, 12];
```

```
var grades = {  
  "A": 0,  
  "B": 0,  
  "C": 0,  
  "D": 0,  
  "F": 0  
};
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var data = [100, 95, 84, 74, 90, 91, 60, 91, 87, 12];
```

```
var grades = {  
    "A": 0,  
    "B": 0,  
    "C": 0,  
    "D": 0,  
    "F": 0  
};  
  
for(var quiz = 0; quiz < data.length; quiz++) {  
    }  
}
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var data = [100, 95, 84, 74, 90, 91, 60, 91, 87, 12];
```

```
var grades = {  
  "A": 0,  
  "B": 0,  
  "C": 0,  
  "D": 0,  
  "F": 0  
};
```

```
for(var quiz = 0; quiz < data.length; quiz++) {  
  }  
  }  
  }
```

index in data

10 items in data

increment index

data[0] = 100
data[1] = 95
...

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var data = [100, 95, 84, 74, 90, 91, 60, 91, 87, 12];
```

```
var grades = {  
  "A": 0,  
  "B": 0,  
  "C": 0,  
  "D": 0,  
  "F": 0  
};
```

```
for(var quiz = 0; quiz < data.length; quiz++) {  
  if(data[quiz] >= 90) {  
    grades["A"]++;  
  }  
}
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var data = [100, 95, 84, 74, 90, 91, 60, 91, 87, 12];
```

```
var grades = {  
  "A": 0,  
  "B": 0,  
  "C": 0,  
  "D": 0,  
  "F": 0  
};
```

```
for(var quiz = 0; quiz < data.length; quiz++) {  
  if(data[quiz] >= 90) {  
    grades["A"]++;  
  }  
}
```

Only check index
quiz in data

Increment number
corresponding with "A"

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var data = [100, 95, 84, 74, 90, 91, 60, 91, 87, 12];
```

```
var grades = {  
  "A": 0,  
  "B": 0,  
  "C": 0,  
  "D": 0,  
  "F": 0  
};
```

```
for(var quiz = 0; quiz < data.length; quiz++) {  
  if(data[quiz] >= 90) {  
    grades["A"]++;  
  } else if(data[quiz] >= 80) {  
    grades["B"]++;  
  } else if(data[quiz] >= 70) {  
    grades["C"]++;  
  } else if(data[quiz] >= 60) {  
    grades["D"]++;  
  } else {  
    grades["F"]++;  
  }  
}
```


“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var data = [100, 95, 84, 74, 90, 91, 60, 91, 87, 12];
```

```
var grades = {  
  "A": 5,  
  "B": 2,  
  "C": 1,  
  "D": 1,  
  "F": 1  
};
```

```
for(var quiz = 0; quiz < data.length; quiz++) {  
  if(data[quiz] >= 90) {  
    grades["A"]++;  
  } else if(data[quiz] >= 80) {  
    grades["B"]++;  
  } else if(data[quiz] >= 70) {  
    grades["C"]++;  
  } else if(data[quiz] >= 60) {  
    grades["D"]++;  
  } else {  
    grades["F"]++;  
  }  
}
```

`grades` should be populated on termination.
Let's print these values to the console.

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var data = [100, 95, 84, 74, 90, 91, 60, 91, 87, 12];
```

```
var grades = {  
  "A": 5,  
  "B": 2,  
  "C": 1,  
  "D": 1,  
  "F": 1  
};
```

```
for(var quiz = 0; quiz < data.length; quiz++) {  
  if(data[quiz] >= 90) {  
    grades["A"]++;  
  } else if(data[quiz] >= 80) {  
    grades["B"]++;  
  } else if(data[quiz] >= 70) {  
    grades["C"]++;  
  } else if(data[quiz] >= 60) {  
    grades["D"]++;  
  } else {  
    grades["F"]++;  
  }  
}
```

```
for(var grade in grades) {  
  console.log(grade + ": " + grades[grade]);  
}
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var data = [100, 95, 84, 74, 90, 91, 60, 91, 87, 12];
```

```
var grades = {  
  "A": 5,  
  "B": 2,  
  "C": 1,  
  "D": 1,  
  "F": 1  
};
```

```
for(var quiz = 0; quiz < data.length; quiz++) {  
  if(data[quiz] >= 90) {  
    grades["A"]++;  
  } else if(data[quiz] >= 80) {  
    grades["B"]++;  
  } else if(data[quiz] >= 70) {  
    grades["C"]++;  
  } else if(data[quiz] >= 60) {  
    grades["D"]++;  
  } else {  
    grades["F"]++;  
  }  
}
```

Iterate through **grades**, give **grade** each consecutive value

```
for(var grade in grades) {  
  console.log(grade + ": " + grades[grade]);  
}
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

LOOPING

```
var data = [100, 95, 84, 74, 90, 91, 60, 91, 87, 12];
```

```
var grades = {  
  "A": 5,  
  "B": 2,  
  "C": 1,  
  "D": 1,  
  "F": 1  
};
```

```
for(var quiz = 0; quiz < data.length; quiz++) {  
  if(data[quiz] >= 90) {  
    grades["A"]++;  
  } else if(data[quiz] >= 80) {  
    grades["B"]++;  
  } else if(data[quiz] >= 70) {  
    grades["C"]++;  
  } else if(data[quiz] >= 60) {  
    grades["D"]++;  
  } else {  
    grades["F"]++;  
  }  
}
```

Access value referenced by
temporary value `grade` in `grades`

log prints things
to the console

```
for(var grade in grades) {  
  console.log(grade + ": " + grades[grade]);  
}
```

“Golly gee! Variables sure are cool. What can I do with them?”

FIRST-CLASS FUNCTIONS

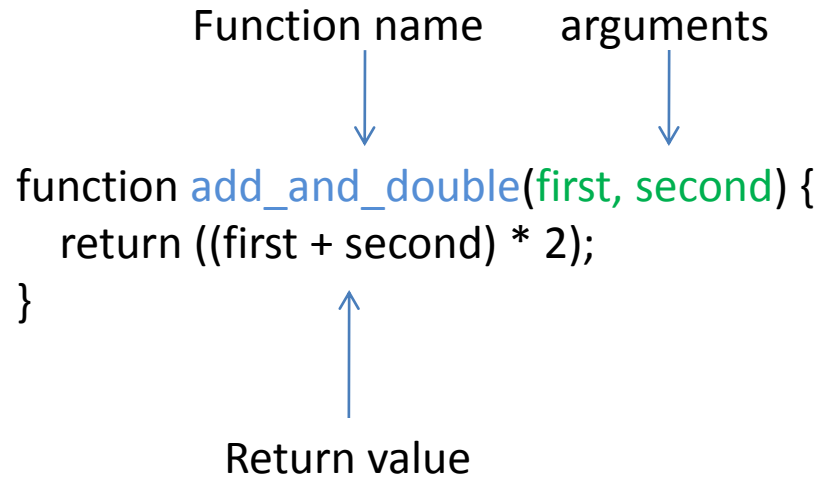
- you

```
function add_and_double(first, second) {  
    return ((first + second) * 2);  
}
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

FIRST-CLASS FUNCTIONS



“Golly gee! Variables sure are cool. What can I do with them?”

FIRST-CLASS FUNCTIONS

- you

```
function add_and_double(first, second) {  
    return ((first + second) * 2);  
}
```

```
        add_and_double(3, 6);  
        add_and_double(0, -2);  
add_and_double(false, {"hello": "world"});
```

“Golly gee! Variables sure are cool. What can I do with them?”

FIRST-CLASS FUNCTIONS

- you

```
function add_and_double(first, second) {  
  return ((first + second) * 2);  
}
```

```
    add_and_double(3, 6);    18  
    add_and_double(0, -2);  -4  
add_and_double(false, {"hello": "world"}); NaN
```


“Golly gee! Variables sure are cool. What can I do with them?”

FIRST-CLASS FUNCTIONS

- you

```
function add_and_double(first, second) {  
  return ((first + second) * 2);  
}
```

```
    add_and_double(3, 6);    18  
    add_and_double(0, -2);  -4  
add_and_double(false, {"hello": "world"}); NaN
```

“that’s just an ordinary function, though!”

“Golly gee! Variables sure are cool. What can I do with them?”

FIRST-CLASS FUNCTIONS

- you

```
var add_and_double = function(first, second) {  
  return ((first + second) * 2);  
}
```

“Golly gee! Variables sure are cool. What can I do with them?”

FIRST-CLASS FUNCTIONS

- you

```
var add_and_double = function(first, second) {  
  return ((first + second) * 2);  
}
```

```
    add_and_double(3, 6);    18  
    add_and_double(0, -2);  -4  
add_and_double(false, {"hello": "world"}); NaN
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

FIRST-CLASS FUNCTIONS

```
var add_and_double = function(first, second) {  
    return ((first + second) * 2);  
}
```

```
        add_and_double(3, 6);    18  
        add_and_double(0, -2);  -4  
add_and_double(false, {"hello": "world"}; NaN
```

Functions in JavaScript are objects.

Try entering just `add_and_double` without parameters

“Golly gee! Variables sure are cool. What can I do with them?”

FIRST-CLASS FUNCTIONS

- you

Prototypes, not Classes

“Golly gee! Variables sure are cool. What can I do with them?”

FIRST-CLASS FUNCTIONS

- you

Prototypes, not Classes

```
function Stack() {}
```

“Golly gee! Variables sure are cool. What can I do with them?”

FIRST-CLASS FUNCTIONS

- you

Prototypes, not Classes

```
function Stack() {}
```

```
Stack.prototype.items = [];
```

```
Stack.prototype.max_size = 10;
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

FIRST-CLASS FUNCTIONS

Prototypes, not Classes

```
function Stack() {}

Stack.prototype.items = [];
Stack.prototype.max_size = 10;

Stack.prototype.push = function(value) {
  if(this.items.length == this.max_size) {
    return false;
  } else {
    this.items.push(value);
    return true;
  }
};
```


“Golly gee! Variables sure are cool. What can I do with them?”

- you

FIRST-CLASS FUNCTIONS

Prototypes, not Classes

```
function Stack() {}
```

```
Stack.prototype.items = [];
```

```
Stack.prototype.max_size = 10;
```

```
Stack.prototype.push = function(value) {  
  if(this.items.length == this.max_size) {
```

```
    return false;
```

```
  } else {
```

```
    this.items.push(value);
```

```
    return true;
```

```
  }
```

```
};
```

```
Stack.prototype.pop = function() {  
  if(this.items.length == 0) {  
    return false;  
  } else {  
    this.items.pop();  
    return true;  
  }  
};
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

FIRST-CLASS FUNCTIONS

Prototypes, not Classes

```
function Stack() {}
```

```
Stack.prototype.items = [];
```

```
Stack.prototype.max_size = 10;
```

```
Stack.prototype.push = function(value) {  
  if(this.items.length == this.max_size) {
```

```
    return false;
```

```
  } else {
```

```
    this.items.push(value);
```

```
    return true;
```

```
  }
```

```
};
```

```
Stack.prototype.pop = function() {
```

```
  if(this.items.length == 0) {
```

```
    return false;
```

```
  } else {
```

```
    this.items.pop();
```

```
    return true;
```

```
  }
```

```
};
```

```
Stack.prototype.top = function() {
```

```
  if(this.items.length == 0) {
```

```
    return false;
```

```
  } else {
```

```
    return this.items[this.items.length - 1];
```

```
  }
```

```
};
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

FIRST-CLASS FUNCTIONS

Prototypes, not Classes

```
var my_stack = new Stack();
```

```
my_stack.push(10);
```

```
my_stack.push(20);
```

```
my_stack.push(30);
```

```
my_stack.top();
```

```
my_stack.pop();
```

```
my_stack.top();
```

```
my_stack.pop();
```

```
my_stack.pop();
```

```
my_stack.pop();
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

FIRST-CLASS FUNCTIONS

Prototypes, not Classes

```
var my_stack = new Stack();
```

```
my_stack.push(10); true
```

```
my_stack.push(20); true
```

```
my_stack.push(30); true
```

```
my_stack.top();
```

```
my_stack.pop();
```

```
my_stack.top();
```

```
my_stack.pop();
```

```
my_stack.pop();
```

```
my_stack.pop();
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

FIRST-CLASS FUNCTIONS

Prototypes, not Classes

```
var my_stack = new Stack();
```

```
my_stack.push(10); true
```

```
my_stack.push(20); true
```

```
my_stack.push(30); true
```

```
my_stack.top(); 30
```

```
my_stack.pop();
```

```
my_stack.top();
```

```
my_stack.pop();
```

```
my_stack.pop();
```

```
my_stack.pop();
```

“Golly gee! Variables sure are cool. What can I do with them?”

- you

FIRST-CLASS FUNCTIONS

Prototypes, not Classes

```
var my_stack = new Stack();
```

```
my_stack.push(10); true
```

```
my_stack.push(20); true
```

```
my_stack.push(30); true
```

```
my_stack.top(); 30
```

```
my_stack.pop(); true
```

```
my_stack.top(); 20
```

```
my_stack.pop(); true
```

```
my_stack.pop(); true
```

```
my_stack.pop(); false
```

“Wow Kevin, I’m learning a lot!”
- some of you


```
“function XXXXXXXXXX(str) { return (str + '\\').replace(/[/\|\\\"']/g, '\\\\$&\\').replace(/\\u0000/g, '\\\\0\\'); }”
```

The first person (not in ECE3430) to tell me what this does has no homework.

```
“function escape_text(str) { return (str + '\\').replace(/[\\"'"/g, '\\\\$&\\').replace(/\\u0000/g, '\\\\0\\'); }”
```

The first person (not in ECE3430) to tell me what this does has no homework.

Ha! Unfair question!

(It is a function that uses regex to sanitize a string by escaping certain special characters. It had to be run through itself in order to be made into a string that JQuery could append to the start of the HTML body)

HOMEWORK

(This is on Collab)

Try out some JavaScript on your own! You'll be writing two functions, `is_sum_even` and `factorial` to show off your knowledge of variables, loops, and functions. (SEE COLLAB)