# OBJECTS, THE DOM, AND APIS

## AGENDA

- Review
- Higher-Order Functions
- Higher-Order Methods
- Prototypes
- Constructors
- The DOM
- APIs

## DIFFERENCE BTWN/ AN ARGUMENT AND A PARAMETER?

Parameters used inside of a function:

```
function foo(x,y){
    return x+y;
}
```

vs

Arguments are the values passed when the function is called:

```
foo(2,3);
```

## ARRAY CREATION

Array Constructor:

```
var myZebra = new Array();
```

vs

Literal Notation:

```
var myZebra = [];
```

## COMPUTERS ONLY UNDERSTAND DATA

- Data is just a sequence of bits (0,1)
- JavaScript data is seperated into values
- Every value has a type which determines the role it can play (numbers, strings, booleans, objects, functions, and undefined)

## FUNCTIONS ARE REGULAR VALUES

## HIGHER-ORDER FUNCTIONS

- Functions that operate on other functions, either by taking them as arguments or by returning them, are called higher-order functions.
- Higher-Order functions allow us to abstract over actions!!!, not just values.

## FUNCTION THAT CREATES NEW FUNCTIONS

```
function greaterThan(n) {
return function(m) { return m > n; };
}
var greaterThan10 = greaterThan(10);
console.log(greaterThan10(11));
// → true
```

## WHY DOES THIS MATTER?

- Complexity is the devil
- We need to abstract out the Complexity

```
Put 1 cup of dried peas per person into a container. Add water
until the peas are well covered. Leave the peas in water for at
least 12 hours. Take the peas out of the water and put them in
a cooking pan. Add 4 cups of water per person. Cover the pan
and keep the peas simmering for two hours. Take half an
onion per person. Cut it into pieces with a knife. Add it to
the peas. Take a stalk of celery per person. Cut it into
pieces with a knife. Add it to the peas. Take a carrot per
person. Cut it into pieces. With a knife! Add it to the
peas. Cook for 10 more minutes.
```

vs

```
Per person: 1 cup dried split peas, half a chopped onion, a stalk of celery,
and a carrot.

Soak peas for 12 hours. Simmer for 2 hours in 4 cups of water
(per person). Chop and add vegetables. Cook for 10 more minutes.
```

## ABSTRACTION REQUIRES EFFORT OR A DICTIONARY

- Methods - object functions
- The JavaScript Array global object is a constructor for arrays , which are high-level, list-like objects. (We will get to this later. Promise!)
- Arrays have standard methods

## FOREACH

- Moves through each item in the array

```
var nums = [1,2,3,4];
nums.forEach(function(element,index){
    nums[index]=++element;
});
console.log(nums);
```

## MAP

- map calls a provided callback function once for each element

```
var numbers = [1, 4, 9];
var roots = numbers.map(Math.sqrt);
// roots is now [1, 2, 3], numbers is still [1, 4, 9]
```

## OBJECTS

## WHAT THE F^*% ARE OBJECTS?

- An aggregation of variables and functions
- Ideally, the components of an object are put together in such a way that the object represents both the attributes and behavior of some "thing" being modeled in the program.
- In object oriented languages, object variables are called "properties" and object functions are called "methods."

## OBJECT EXAMPLE

- Stopwatch
- Might contain a variable to represent hours, another to represent minutes, and another to represent seconds. (Properties).
- Might also contain functions that manipulate those values, such as "start ()" and "stop ()". (Methods)

## STOPWATCH OBJECT

```
var stopwatch = {};

stopwatch.currentTime = 12;

stopwatch.tellTime = function(time){
    console.log("The current time is "+time+".");
}

stopwatch.tellTime(stopwatch.currentTime);
```

## THE MAGIC OF PROTOTYPES

```
stopwatch.toString()

// "[object Object]"
```

- We never created the toString method
- Enter Prototypes

## PROTOTYPES

- A prototype is another object that is used as a fallback source of properties.
- When an object gets a request for a property that it does not have, its prototype will be searched for the property, then the prototype's prototype, and so on.
- Many objects don't directly have Object.prototype as their prototype, but instead have another object, which provides its own default properties. Functions derive from Function.prototype, and arrays derive from Array.prototype.

## STOPWATCH PROTOTYPE

```
var protoStopwatch = {
  tellTime: function(time) {
    console.log("The current time is "+time+".");
  },
  sayColor: function(color) {
    console.log("I am "+color+".");
  }
};

var redStopwatch = Object.create(protoStopwatch);
redStopwatch.color = "red";
redStopwatch.currentTime = "7pm";
redStopwatch.tellTime(redStopwatch.currentTime);
redStopwatch.sayColor(redStopwatch.color);
```

## CONSTRUCTORS

- A more convenient way to create objects that derive from some shared prototype is to use a constructor.
- In JavaScript, calling a function with the new keyword in front of it causes it to be treated as a constructor.
- The constructor will have its this variable bound to a fresh object, and unless it explicitly returns another object value, this new object will be returned from the call.

## STOPWATCH CONSTRUCTOR

```
function Stopwatch(color) {
  this.color = color;
}

var redStopwatch = new Stopwatch("red");
var blueStopwatch = new Stopwatch("blue");

console.log(blueStopwatch.color);
```

## STOPWATCH EXTENSION

```javascript
Stopwatch.prototype.tellColor = function() {
console.log("I am "+this.color+".");
};

blueStopwatch.tellColor();
```

## THE DOM

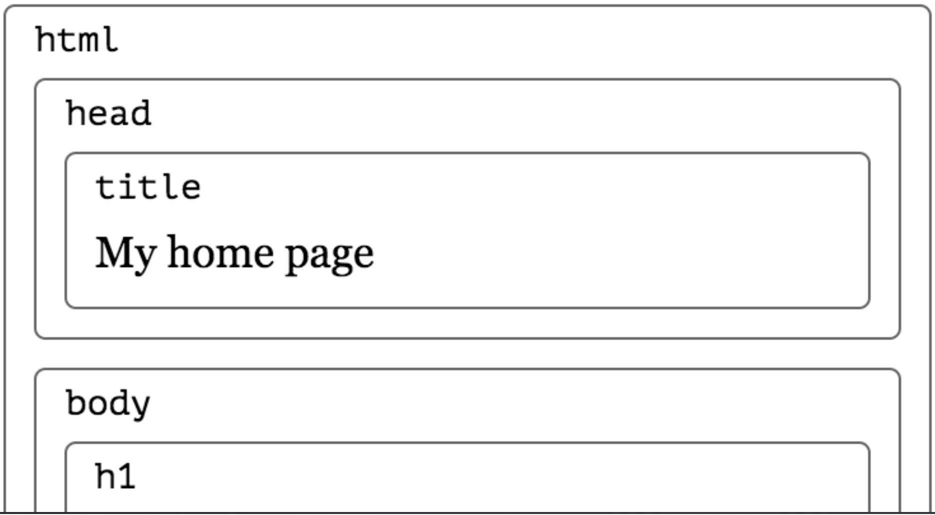
## DOM

### DOCUMENT OBJECT MODEL

- When you open a web page in your browser, the browser retrieves the page's HTML text and parses it. The browser builds up a model of the document's structure and then uses this model to draw the page on the screen.

```html
<!doctype html>
<html>
  <head>
    <title>My home page</title>
  </head>
  <body>
    <h1>My home page</h1>
    <p>Hello, I am Marijn and this is my home page.</p>
    <p>I also wrote a book! Read it
      <a href="http://eloquentjavascript.net">here</a>.</p>
  </body>
</html>
```

This page has the following structure:



## DOM NODES

- document.body.childNodes
- document.body.firstChild
- document.body.firstElementChild

Attributes:

```
var link = document.body.getElementsByTagName("a")[0];
console.log(link.href);
```

## QUERY SELECTORS

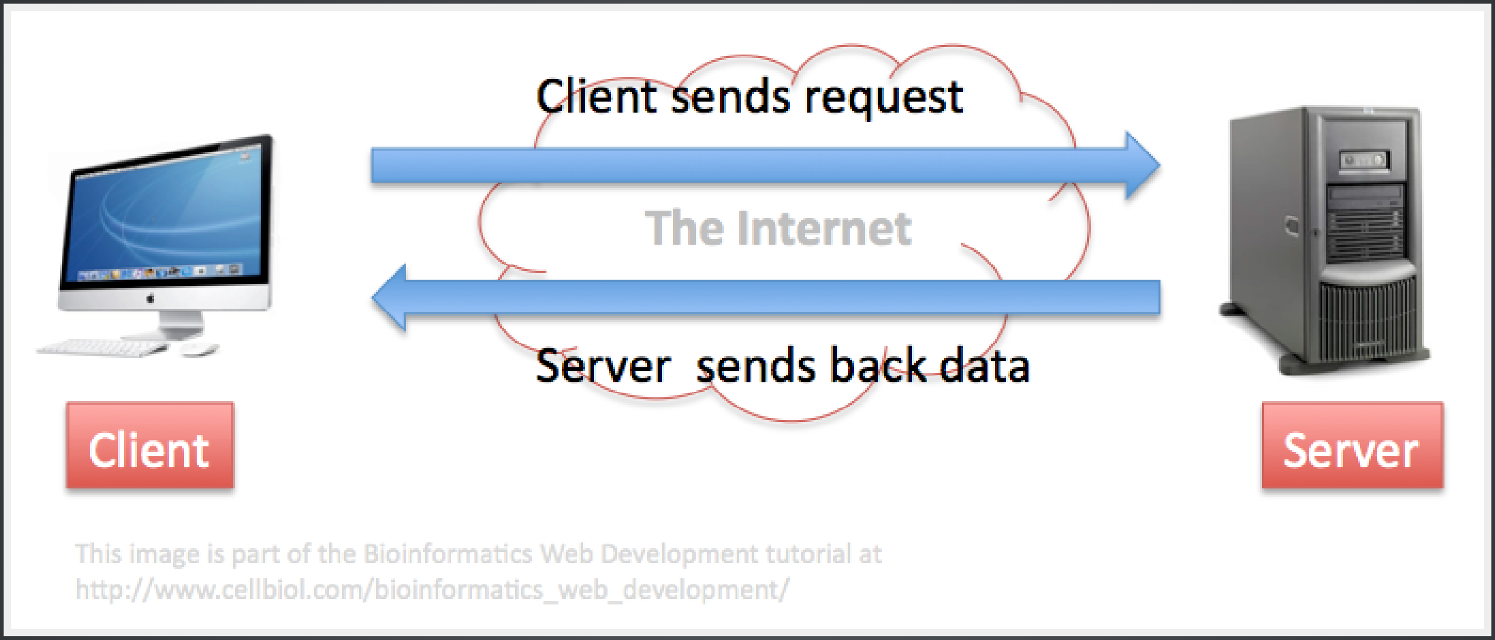
```
document.querySelectorAll("a");
```

```
document.querySelector("a");
```

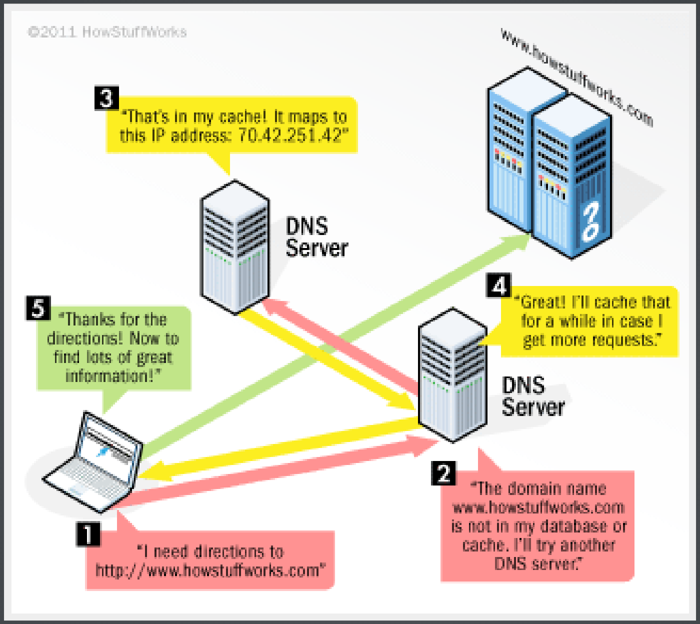
- It will return only the first matching element or null if no elements match.

## APIS

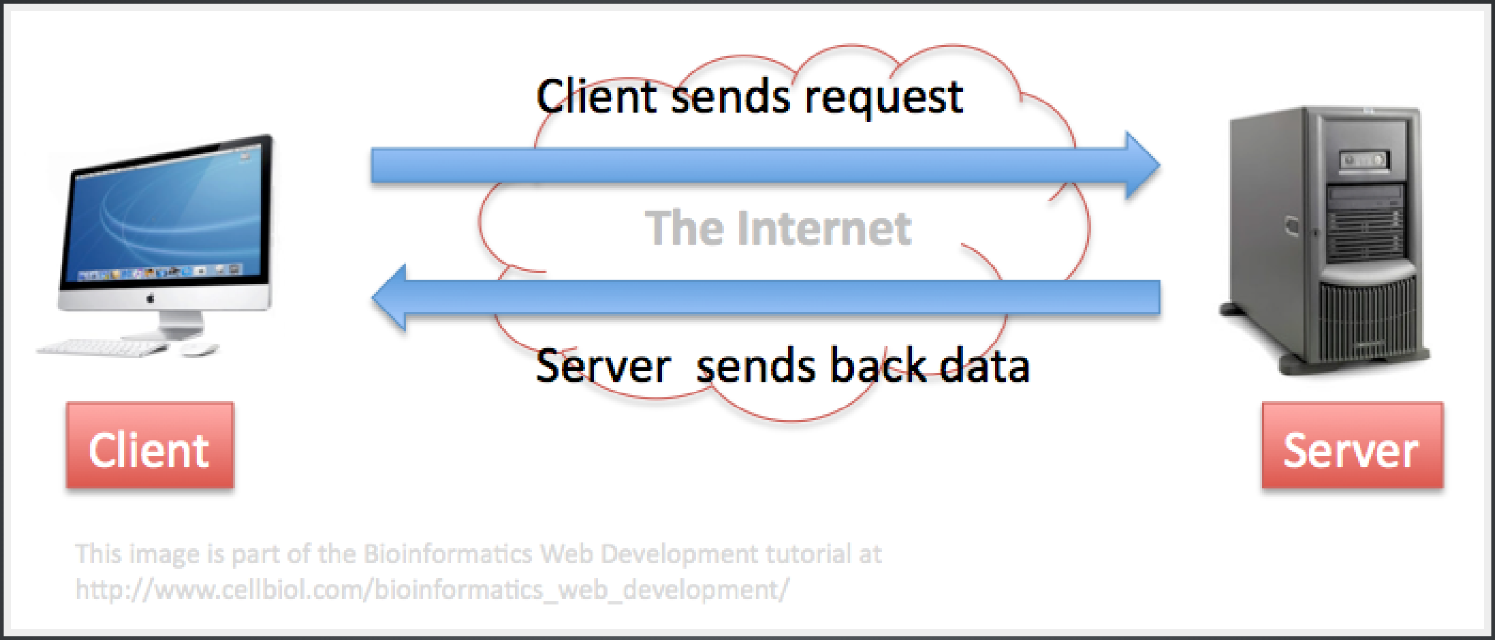
## CLIENT VS. SERVER


Client sends request
The Internet
Server sends back data
Client
Server
This image is part of the Bioinformatics Web Development tutorial at
http://www.cellbiol.com/bioinformatics_web_development/

# LOCATION

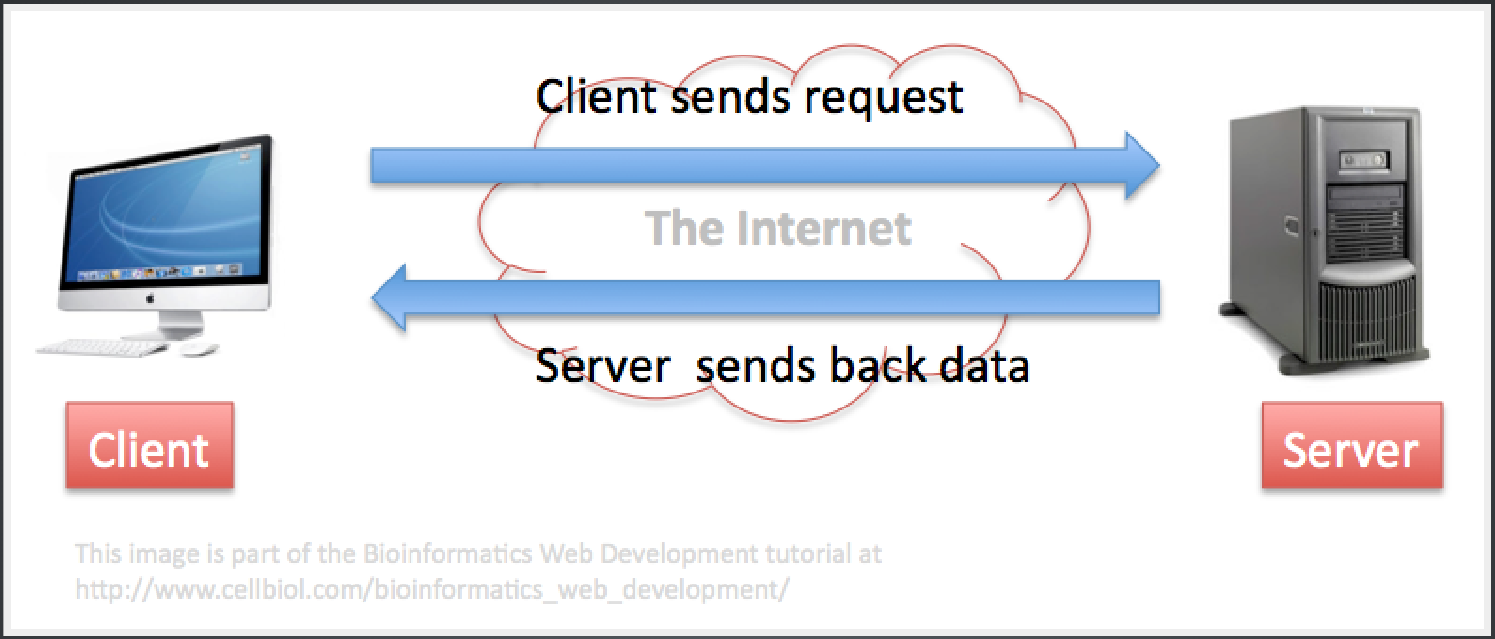

©2011 HowStuffWorks
www.howstuffworks.com
3 "That's in my cache! It maps to this IP address: 70.42.251.42"
DNS Server
4 "Great! I'll cache that for a while in case I get more requests."
5 "Thanks for the directions! Now to find lots of great information!"
DNS Server
2 "The domain name www.howstuffworks.com is not in my database or cache. I'll try another DNS server."
1 "I need directions to http://www.howstuffworks.com"

## TCP (TRANSMISSION CONTROL PROTOCOL)


Client sends request
The Internet
Server sends back data
Client
Server
This image is part of the Bioinformatics Web Development tutorial at
http://www.cellbiol.com/bioinformatics_web_development/

- Listening (Server) vs. Connecting (Client)

## HTTP (HYPERTEXT TRANSFER PROTOCOL)


Client sends request
The Internet
Server sends back data
Client
Server
This image is part of the Bioinformatics Web Development tutorial at
http://www.cellbiol.com/bioinformatics_web_development/

- Network Protocol that allows computers to request documents over the network

# URL

```
 http://eloquentjavascript.net/12_browser.html
|      |                      |               |
protocol       server                path
```

## THINGS WE REQUEST?

- HTML
- CSS
- JavaScript

## Status Codes have standard meanings; here are a few.

| **Code** | **Reason** |
|---|---|
| 200 | OK |
| 301 | Moved Permanently |
| 302 | Moved Temporarily |
| 400 | Bad Request |
| 403 | Forbidden |
| 404 | Not Found |
| 500 | Internal Server Error |

## AJAX AND JAVASCRIPT

AJAX stands for Asynchronous JavaScript and XML. It sends and receives information in a variety of formats. The most commonly used format for communicating with APIs, as we`ve seen, is JSON. AJAX allows us to not only communicate with servers, but it allows us to do this asynchronously, meaning it happens in the background, allow us to update our interfaces and content without refreshing the page.

## jQuery allows us to create quick get and post requests in one step, as opposed to the above multiple steps.

```
// All we need to create a get or post request is use the get or post metho
$.get( 'https://data.cityofnewyork.us/api/views/jb7j-dtam/rows.json?accessT
    // We get the data back from the request in the parameter we pass in th
    console.log(r);
});
```

We need to account for us not receiving data back due to different interruptions/causes:

1. Server timeout
2. Wrong authentication information
3. User loses connection
4. Request URL not found
5. Representational state transfer (REST) is the most common architecture style for passing information to and from these API endpoints.

```javascript
var apiKey = REMOVED;
  var apiURL = "https://api.nasa.gov/planetary/apod?api_key=" + apiKey;
$.ajax({
    url: apiURL,

    // Work with the response
    success: function( response ) {
        console.log( response ); // server response
    },
          error: function(r){
              console.log(r); //server response
          }
});
```

## Practice:

1. Enable user to enter a date
2. Return NASA's Picture of the day for that date