



WEB DEVELOPMENT

Lesson 2

JavaScript

Style guide: <https://github.com/airbnb/javascript>

JavaScript

- High level
- Dynamic
- Dynamically typed
- Interpreted

JavaScript has nothing to do with Java

ECMAScript — JavaScript standart

- ECMAScript 1 (1997)
- ...
- **ECMAScript 5 (2009) — ES5**
- ...
- **ECMAScript 2015 — ES6**
- ECMAScript 2016
- ECMAScript 2017
- ECMAScript 2018

Data types

- Number
- String
- Boolean
- Null
- Undefined
- Object
- Symbol — ES6
- *function*

Good news

If you know C++, C or Java — JavaScript is similar

```
i = 3;

i = i * 10 + 3 + (i / 10);

while (i >= 0) {
    sum += i*i;    // Comment
    i--;
}

for (i = 0; i < 10; i++) {

}

/* this is a comment */
```

```
if (i < 3) {
    i = foobar(i);
} else {
    i = i * .02;
}
```

Most C operators work:

* / % + - ! >= <= > < && || ?:

```
function foobar(i) { return i;}
```

continue/break/return

Variable scoping

Two scopes: Global and function local

```
var globalVar;
```

```
function() {  
  var localVar;  
  if (globalVar > 0) {  
    var localVar2 = 2;  
  }  
  // localVar2 is valid here  
}
```

All var statements **hoisted** to top of scope:

```
function foo() {  
  var x;  
  x = 2;  
  // Same as:  
  function foo() {  
    x = 2  
    var x;  
  }  
}
```

localVar2 is
hoisted here but
has value undefined

functions can have properties too

```
function plus1(value) {  
    if (plus1.invocations == undefined) {  
        plus1.invocations = 0;  
    }  
    plus1.invocations++;  
    return value + 1;  
}
```

`plus1.invocations` will be the number times function is called

Using prototypes

```
function Rectangle(width, height) {  
    this.width = width;  
    this.height = height;  
}  
Rectangle.prototype.area = function() {  
    return this.width*this.height;  
}  
var r = new Rectangle(26, 14);    // {width: 26, height: 14}  
var v = r.area();                // v == 26*14  
Object.keys(r) == [ 'width', 'height' ] // own properties
```

ECMAScript version 6 extensions

```
class Rectangle extends Shape { // Definition and Inheritance
  constructor(height, width) {
    super(height, width);
    this.height = height;
    this.width = width;
  }
  area() { // Method definition
    return this.width * this.height;
  }
  static countRects() { // Static method
    ...
  }
}

var r = new Rectangle(10,20);
```

Functional Programming

- Imperative:

```
for (var i = 0; i < anArr.length; i++) {  
    newArr[i] = anArr[i]*i;  
}
```
- Functional:

```
newArr = anArr.map(function (val, ind) {  
    return val*ind;  
});
```

Functional Programming - ECMAScript 6

- Imperative:

```
for (var i = 0; i < anArr.length; i++) {  
    newArr[i] = anArr[i]*i;  
}
```

- Functional:

```
newArr = anArr.map((val, ind) => val*ind); // Arrow function
```

JavaScript Object Notation (JSON)

```
var obj = { ps: 'str', pn: 1, pa: [1, 'two', 3, 4], po: { sop: 1}};
```

```
var s = JSON.stringify(obj) =  
      '{"ps":"str","pn":1,"pa":[1,"two",3,4],"po":{"sop":1}}'
```

```
typeof s == 'string'
```

```
JSON.parse(s) // returns object with same properties
```

- JSON is the standard format for sending data to and from a browser

Some JavaScript idioms

- Assign a default value

```
hostname = hostname || "localhost";  
port = port || 80;
```

- Access a possibly undefined object property

```
var prop = obj && obj.propname;
```

Document **O**bject **M**odel

DOM hierarchy

- Rooted at `window.document`
- Follows HTML document structure
 - `window.document.head`
 - `window.document.body`
- DOM objects have tons (~250) of properties,
most private

Accessing DOM Nodes

- Walk DOM hierarchy (not recommended)
 - `element = document.body.firstChild.nextSibling.firstChild;`
- Use DOM lookup method. An example using ids:
 - `element = document.getElementById("div1");`
- Many: `getElementsByClassName()`,
`getElementsByTagName()`, ...
 - `document.body.firstChild.getElementsByTagName()`

More commonly used Node properties

- `textContent`
- `innerHTML`
- `getAttribute()` / `setAttribute()`

DOM and CSS interactions

- `element.className = "active";`
- `element.style.color = "#ff0000";`
- `document.querySelector(".class1")`

Changing the Node structure

- `element = document.createElement("p");`
- `parent.appendChild(element);`

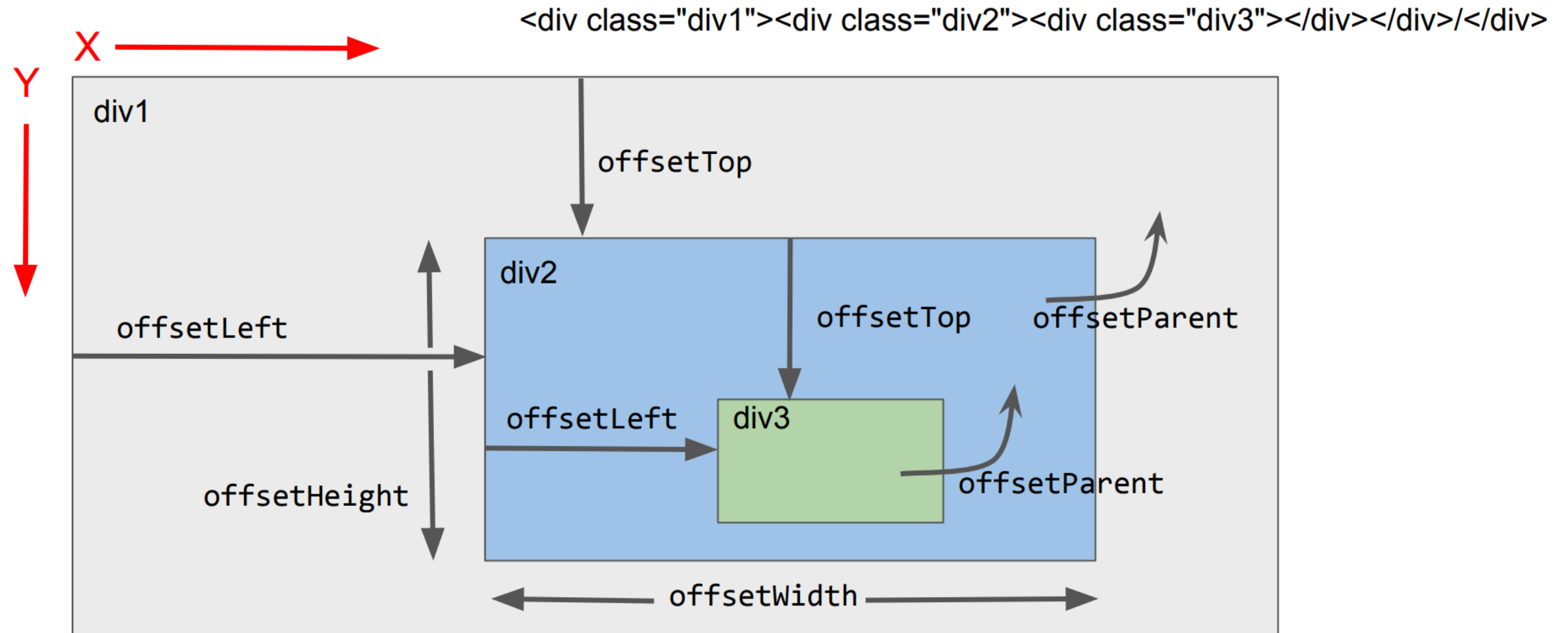
More DOM operations

- `window.location.href = "newPage.html";`
- `console.log("Reached point A");`
- `alert("Wow!"); confirm("OK?");`

DOM's Coordinate System

- The screen origin is at the upper left; y increases as you go down
- The position of an element is determined by the upper-left outside corner of its margin
- Read location with `element.offsetLeft`,
`element.offsetTop`

DOM's Coordinate System



JavaScript and DOM Events

- Mouse-related: mouse movement, button click, enter/leave element
- Keyboard-related: down, up, press
- Focus-related: focus in, focus out (blur)
- Input field changed, Form submitted

Event handling

1. What happened: the event of interest
2. Where it happened: an element of interest.
3. What to do: JavaScript to invoke when the event occurs on the element.

Specifying the JavaScript of an Event

- Option #1: in the HTML:
 - `<div onclick="divClicked();">...</div>`
- Option #2: from Javascript using the DOM:
 - `element.addEventListener("click", mouseClicked);`
 - `element.onclick = mouseClicked;`

moveable box

Timer Event

- `token = setTimeout(myFunc, 5*1000);`
- `token = setInterval(myfunc, 50);`
- `clearInterval(token);`
- Used for animations, automatic page refreshes, etc.

Questions?