What is Web 2.0

"the network as platform"

- Services, not packaged software, with cost-effective scalability
- Control over unique, hard-to-recreate data sources that get richer as more people use them
- Trusting users as co-developers
- Harnessing collective intelligence
- Leveraging the long tail through customer self-service
- Software above the level of a single device
- Lightweight user interfaces, development models, AND business models

— **Tim O'Reilly**, 2005

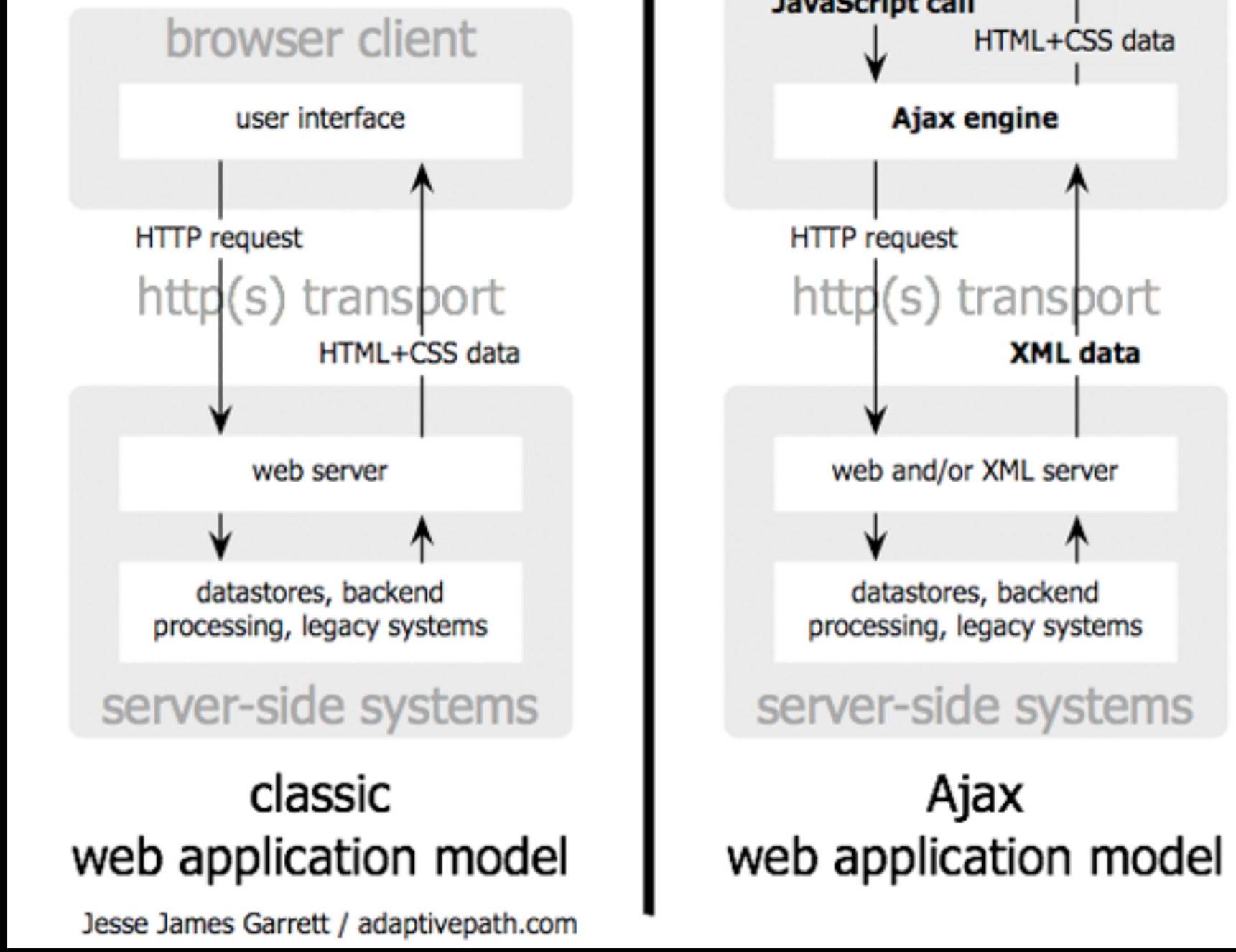
Web Squared

"search as vote"

The collection of technologies used by Google was christened **AJAX**, in a seminal essay by Jesse James Garrett of web design firm Adaptive Path.

"**Ajax** isn't a technology. It's really several technologies, each flourishing in its own right, coming together in powerful new ways.

Ajax incorporates:
- standards-based presentation using **XHTML** and **CSS**;
- dynamic display and interaction using the Document Object Model (DOM);
- data interchange and manipulation using XML and XSLT;
- asynchronous data retrieval using XMLHttpRequest;
- + **JavaScript** binding everything together."



Jesse James Garrett, 2005

(To view this link you'll need the Wayback Machine)

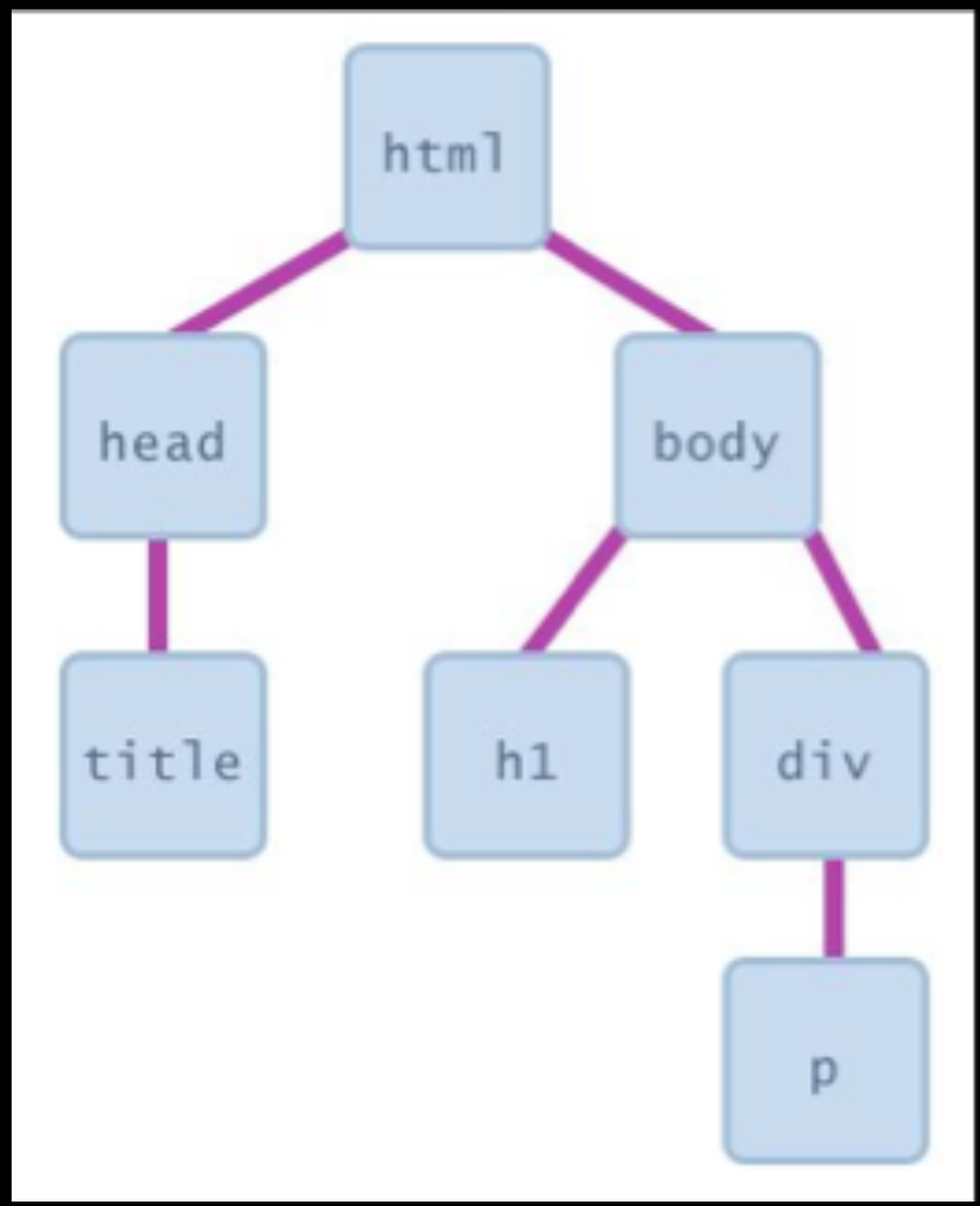
The DOM

The DOM is a tree of node objects corresponding to the HTML elements on a page.

- JS code can examine these nodes to see the state of an element (e.g. to get what the user typed in a text box)

- JS code can edit the attributes of these nodes to change the attributes of an element (e.g. to toggle a style or to change the contents of an <h1> tag)

- JS code can add elements to and remove elements from a web page by adding and removing nodes from the DOM



defer

You can add the defer attribute onto the script tag so that the JavaScript doesn't execute until after the DOM is loaded (mdn):

```
<script src="script.js" defer></script>
```

Other old school ways of doing this **(not best practice - don’t do!)**:

- Put the <script> tag at the bottom of the page
- Listen for the “load” event on the window object

You will see tons of examples on the internet that do this. They are out of date. defer is wide supported and better

function scope w/ var

```
function printMessage(message, times) {

  for (var i = 0; i < times; i++) {
    console.log(message);
  }

  console.log('Value of i is ' + i);
}

printMessage('hello', 3);
```

```
3 hello                          theSketch.js:5
  Value of i is 3                theSketch.js:7
>
```

var

The value of "i" is readable outside of the for-loop because variables declared with **var** have function scope.

function scope w/ var

```
var x = 10;

if (x > 0) {
    var y = 10;
}

console.log('Value of y is ' + y);
```

```
Value of y is 10                    theSketch.js:11
```

Variables declared with **var** have function-level scope and do not go out of scope at the end of blocks; only at the end of functions

Therefore you can refer to the same variable after the block has ended (e.g. after the loop or if-statement in which they are declared)

function scope w/ var

```
function willThisWork() {
    var x = 10;
    if (x > 0) {
        var y = 10;
    }
    console.log('y is ' + y);
}

willThisWork();
console.log('y is ' + y);
```

```
y is 10                                          theSketch.js:8
▶ Uncaught ReferenceError:                       theSketch.js:12
y is not defined
    at theSketch.js:12
```

But you can't refer to a variable outside of the function in which it's declared.

scope w/ let

```
function printMessage(message, times) {

    for (let i = 0; i < times; i++) {
        console.log(message);
    }
  console.log('Value of i is ' + i);
}

printMessage('hello', 3);
```

```
3 hello                                          theSketch.js:6
▶ Uncaught ReferenceError: i                     theSketch.js:8
is not defined
    at printMessage (theSketch.js:8)
    at theSketch.js:12
```

let has block-scope so this results in an error

scope w/ const

```
let x = 10;

if (x > 0) {
    const y = 10;
}

console.log(y)
```

```
▸Uncaught ReferenceError: y is not defined    theSketch.js:10
    at theSketch.js:10
```

like, **let** - **const** has block-scope, so accessing the variable outside the block results in an error

const

```
const y = 10;
y = 0; //error
y++; //error
```

```
Uncaught TypeError:                    theSketch.js:5
Assignment to constant variable.
    at theSketch.js:5
```

```
const myList = [1, 2, 3];
myList.push(4); //okay

console.log(myList);
```

```
▶ (4) [1, 2, 3, 4]
```

const declared variables cannot be reassigned.

However, it doesn't provide true **const** correctness, so you can still modify the underlying object

- (In other words, it behaves like Java's final keyword and not C++'s const keyword)

const vs. let

```
let y = 55;
y = 0; // okay
y++;     // okay

let myList = [1,2,3];
myList.push(4);
console.log('y is ' + y);
console.log(myList);
```

```
y is 1                                    theSketch.js:9
▶ (4) [1, 2, 3, 4]                        theSketch.js:10
```

let can be reassigned, which is the difference between **const** **and** **let**

JS Syntax - Variables best practices

- Use **const** whenever possible.

- If you need a variable to be reassignable, use **let**.

- Don't use **var.**
  You will see a ton of example code on the internet with var since const and let are relatively new.
  However, **const** and **let** are well-supported, so there's no reason not to use them.
  (This is also what the Google and AirBnB JavaScript Style Guides recommend.)

You do not declare the datatype of the variable before using it ("dynamically typed")

JS Variables do not have types, but the values do.

There are six primitive types (mdn):

- **Boolean** : true and false
- **Number** : everything is a double (no integers)
- **String**: in 'single' or "double-quotes"
- **Symbol:** (skipping this today)
- **Null:** null: a value meaning "this has no value"
- **Undefined:** the value of a variable with no value assigned

There are also Object types, including Array, Date, String (the object wrapper for the primitiv
type), etc.

Data Type: Numbers

```
const homework = 0.45;
const midterm = 0.2;
const final = 0.35;
const score = homework * 87 + midterm * 90 + final * 95;
console.log(score); // 90.4
```

- All numbers are floating point real numbers. No integer type.
- Operators are like Java or C++.
- Precedence like Java or C++.
- A few special values: **NaN** (not-a-number), **+Infinity**, **-Infinity**
- There's a Math class: **Math**.floor, **Math**.ceil, etc.

Data Type: Boolean

```
if (username) {
    // username is defined
}
```

Non-boolean values can be used in control statements, which get converted to their "**truthy**" or "**falsy**" value:

- null, undefined, 0, NaN, '', "" evaluate to false
- Everything else evaluates to true

Equality

== (is equal to)

Compares two values to see if they are the same

!= (is not equal to)

Compares two values to see if they are not the same

=== (strict equal to)

Compares two values to check that both the data and value are the same

!== (strict not equal to)

Compares two values to check that both the data and value are not the same

Equality

JavaScript's `==` and `!=` are basically broken: they do an implicit type conversion before the comparison.

```
// false
'' == '0';
// true
'' == 0;
// true
0 == '0';
// false
NaN == NaN;
// true
[''] == '';
// false
false == undefined;
// false
false == null;
// true
null == undefined;
```

Equality

Instead d of fixing `==` and `!=` ,
the ECMAScript standard kept
the existing behavior but added
`===` and `!==`

```
// false
'' === '0';
// false
'' === 0;
// false
0 === '0';
// false -??
NaN === NaN;
 // false
[''] === '';
// false
false === undefined;
 // false
false === null;
// false
null === undefined;
```

Always use `===` and `!==` and don't use `==` or `!=`

Null + Undefined

What's the difference?

null is a value representing the absence of a value, similar to null in Java and nullptr in C++.

undefined is the value given to a variable that has not been a value.

```
let x = null;
let y;
console.log(x);
console.log(y);
```

```
null                    theSketch.js:7
undefined               theSketch.js:8
```

Null + Undefined

What's the difference?

null is a value representing the absence of a value, similar to null in Java and nullptr in C++.

undefined is the value given to a variable that has not been a value.

... however, you can also set a variable's value to undefined bc … javascript.

```
let x = null;
let y = undefined;
console.log(x);
console.log(y);
```

```
null                    theSketch.js:7
undefined               theSketch.js:8
```

Prgrmmng Vocabulary

When you ask function to perform its task, you **call** a function

Some functions need to be provided with information in order to achieve a given task - pieces of information passed to a function are called **parameters**

When you write a function and expect it to provide you with an answer, the response is known as **return value**

Declaring a function

To create a function you give it a name and write statements inside to achieve a task you want it to achieve

function keyword **function name**

```
function buttonClicked() {

    console.log("hello");


}
```

code statement

code block inside curly braces

Calling a function

You call a function by writing it's name somewhere in the code.

```
function buttonClicked() {
    console.log("hello");
}

buttonClicked() // code after
```

Declaring functions with parameters

Sometimes a function needs specific information to perform it's task **(parameters)**
Inside the function the parameters act as variables

parameters

```
function countTotal(itemNumber, price) {
    return itemNumber * price;
}
```

parameters are used like variables inside the function

Calling functions with parameters

When you call a function that has **parameters,** you need to specify the values it should take in. Those values are called **arguments.**

Arguments are written inside parentheses when you call a function + can be provided as values or as variables

```
function countTotal(itemNumber, price) {
        return itemNumber * price;
}

countTotal(7,15); //will return 105

(itemNumber = 7 * price = 15) countTotal(itemNumber, price);
```

Using “return” in a function

return is used to return a value to the code that called the function

The interpreter leaves the function when return is used and goes back to the statement that called it

```
function countTotal(itemNumber, price) {
  return itemNumber * price;
  // interpreter will skip any code found after return
}
```

Variable Scope

Where you declare a variable affects where it can be used within your code

If it's declared inside a function, it can only be used inside that function

It's known as variable's **scope**

Local + Global Variables

When a variable is created inside a function
It's called **local variable** or **function-level variable**

When a variable is created outside a function
It's called **global variable** can be called anywhere
the in code + will take up more memory

```
var salesTax = .08

function countTotal(itemNumber, price) {
      var theSum = itemNumber * price;
      var theTax = theSum * salesTax;
      var theTotal = theSum + theTax;
      return theTotal;
}

console.log(typeof salesTax);
```

Final

Your final is to create a website using HTML, CSS + JavaScript. It must contain at least three layers of file parenting, clean + heavily commented code, thoughtful + appealing design and a title. Projects should be hosted on your server in it's unique directory + url.

You will create a site concept, site map + wireframes for the different layout of your pages. You must create a webpage that acts as a project proposal slide deck - explains your site + includes relevant details. Create a slide deck for your 5 minute presentation + host it on your class site.

5 minute project proposal presentations **(mandatory)** will be **Monday November 18**. You will have 5 minutes to present + 5 minutes for feed back. Everyone must participate, giving + receiving feedback.You will then begin prototyping your site.

Final

User Testing **(mandatory)**: **Monday December 2**
In class project studio: **Monday December 9**
Final Project Critique **(mandatory)**: **Monday December 16**

Final Projects will be due on a designated date in the days following the in-class critique (Most likely Thursday the 19th at midnight).

More specific technical parameters will be assigned in the coming weeks. For now focus on what you want to build.

Final Project Grade Breakdown - Subject to Change

	EXCEEDED (99)	MET CRITERIA (90)	APPROACHED (80)	DID'T MEET CRITERIA (60)	SCORE
Project concept	Something totally original or new.	Interesting idea or a great take on standards.	Nothing that new but you added nice personal touches.	Little thought or effort put into your concept.	**15**
Design Strategy	Overall aesthetic + UI are exceptional	Overall aesthetic is cohesive, interesting + UI is intuitive.	Looks good, but not overall aesthetically cohesive. Confusing interaction.	Little thought went into the aesthetic or Interaction design.	**20**
Technical Execution	Exceeds requirements.	Meets all requirements.	Most requirements met.	Did not meet requirements.	**30**
Participation	Close attention to your user feedback, enthusiastic + generous feedback as a user + fully attentive to others' work.	Attention to user feedback, feedback as a user + attentive to others' work.	Little attention to classmate's work as presenters, users or testers.	Disrespectful lack of attention to classmate's work as presenters, users or testers.	**15**
Presentation	Engaging oral + visual presentation with insight into your process, ample technical details, user feedback + how you responded.	Good presentation with some insight into your process, technical details, user feedback + how you responded.	Completed presentation with little insight into your process, technical details, user feedback or how you responded.	Unclear presentation with little context as to how or why you made this project.	**10**
On time, commented code w/ sources cited	On time, fully functional, with extensive comments.	On time, fully functional, with comments.	On time, functional, with little comments.	Late, broken code +/or no comments.	**10**

Final Project Prompt

1. Create a hyper text narrative or net art piece
2. Invent a fictional or futuristic product (a machine that records your dreams, a shoe that plays music, etc) + create a website for it. **Inspiration**
3. Create a website for something / someone in your life. Your band, your Mom's ceramics hobby, etc. **Inspiration**
4. Other ideas run past Rebecca.