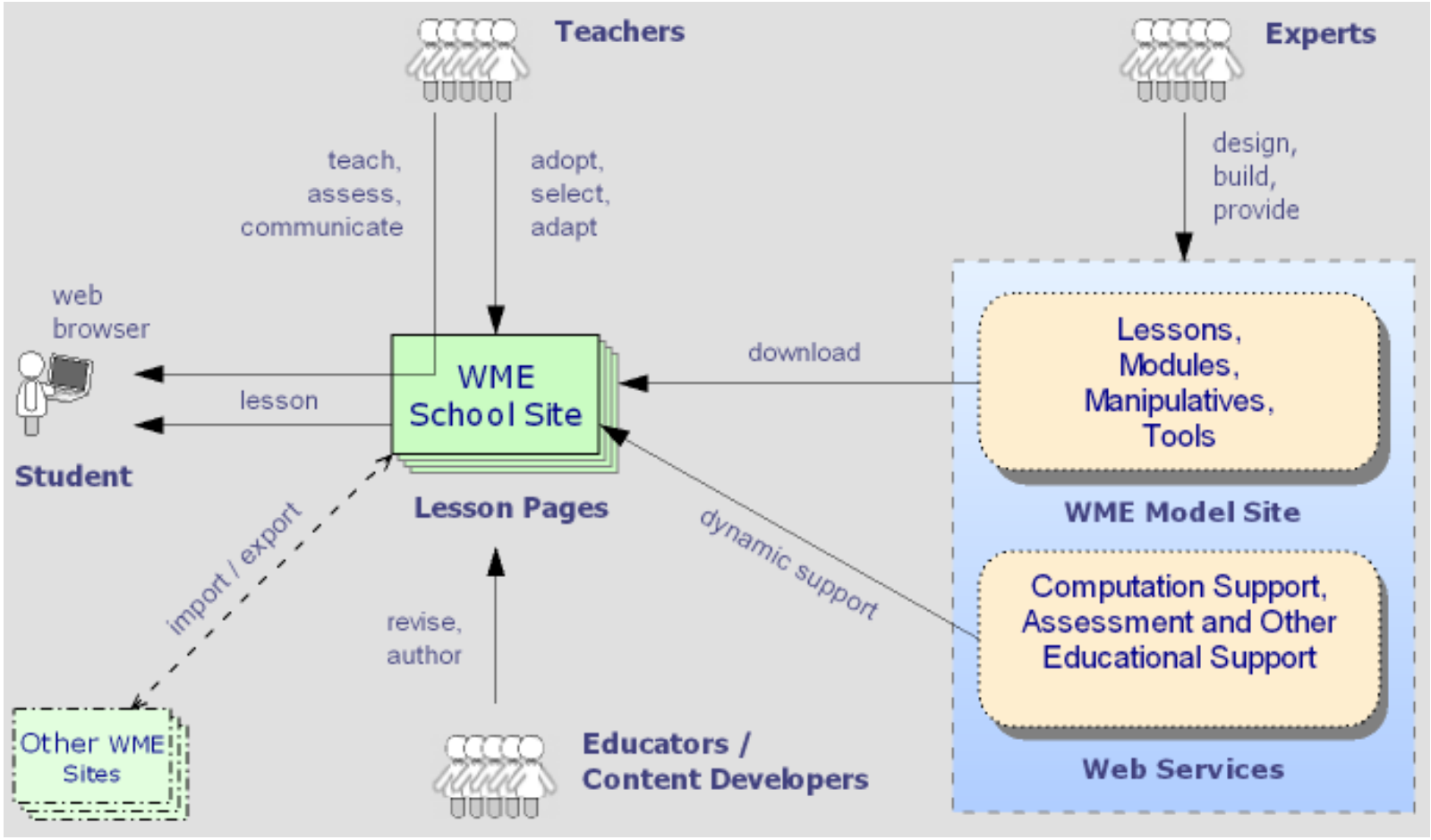
KENT STATE UNIVERSITY

WME and Automatic Mathematical Answer Checking

Paul S. Wang 王士弘
Institute for Computational Mathematics
Kent State University
pwang@cs.kent.edu

KENT STATE UNIVERSITY

The WME Concept



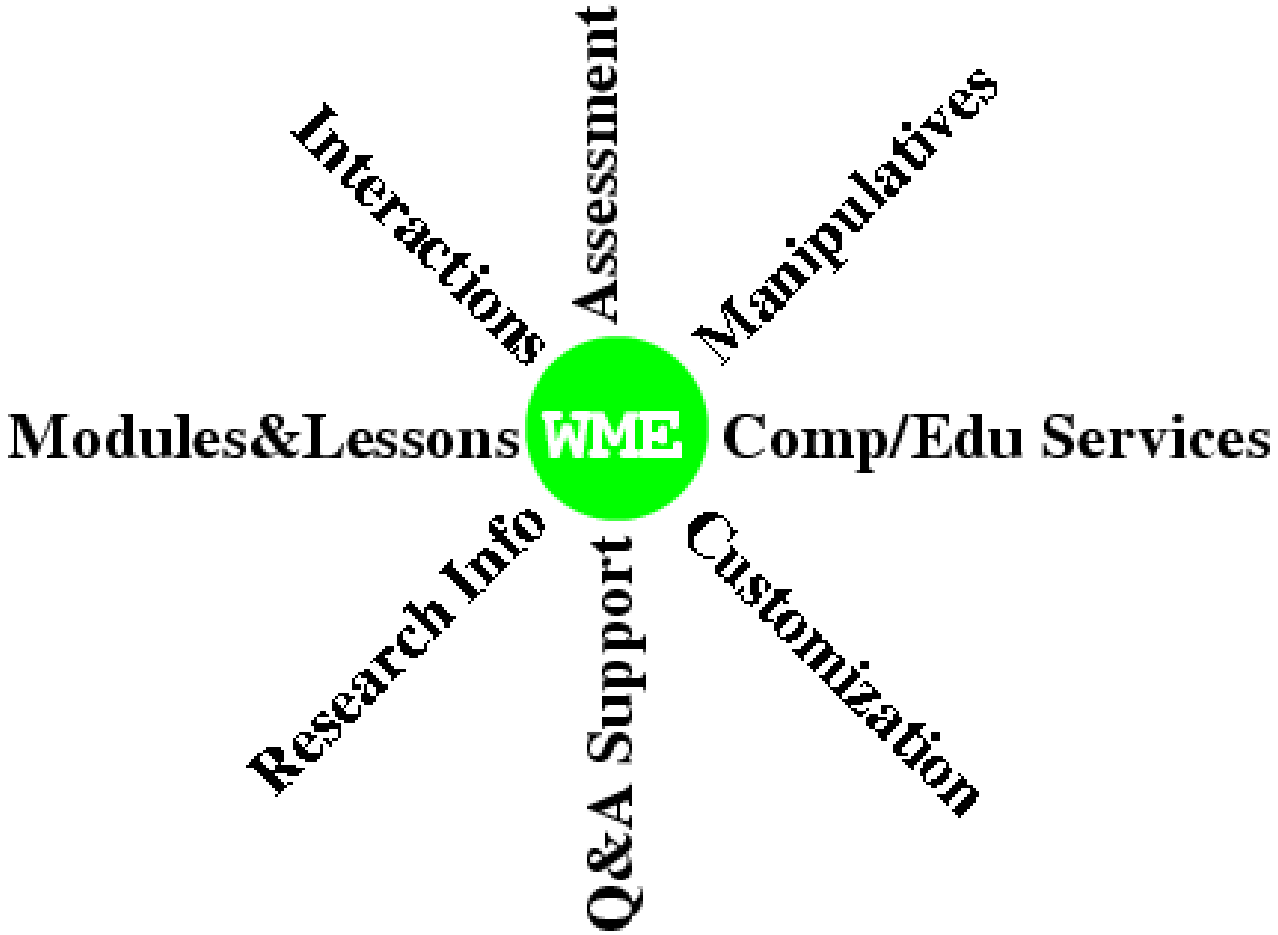
The WME Project Web Site.

An Idea Whose Time Has Come

- Symbolic and numerical computation systems, have matured and become *Internet Accessible*.
- Mathematics teachers and students need help especially in the US.
- Availability and standardization of the Web and the Internet have grown and evolved sufficiently.
- Maturing technologies: MathML, ECMAScript, DOM, SVG, XML, CSS, Web Services, ...
- Increasing number of school districts have already deployed Internet/Web in classrooms.
- Web has begun to offer helpful materials for Mathematics teaching/learning.

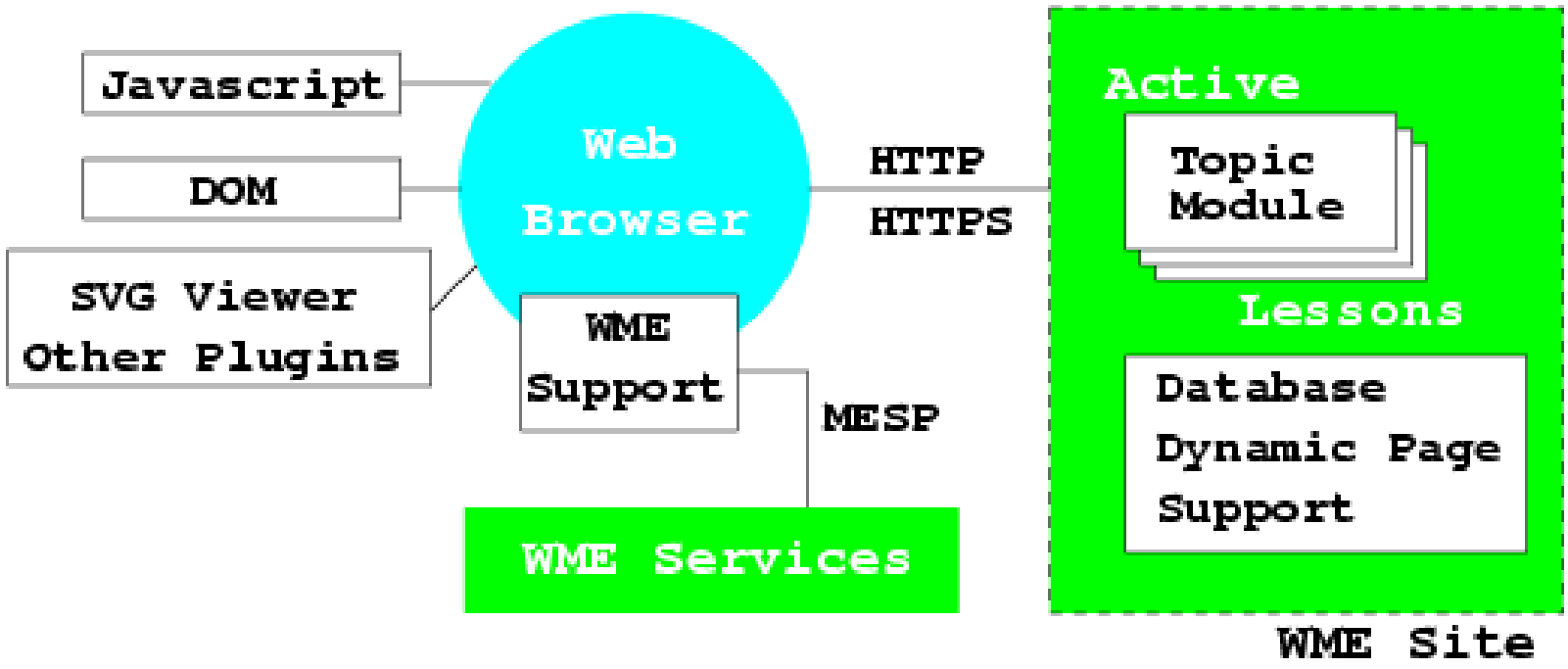
KENT STATE UNIVERSITY

The WME Integration



KENT STATE UNIVERSITY

The WME Architecture

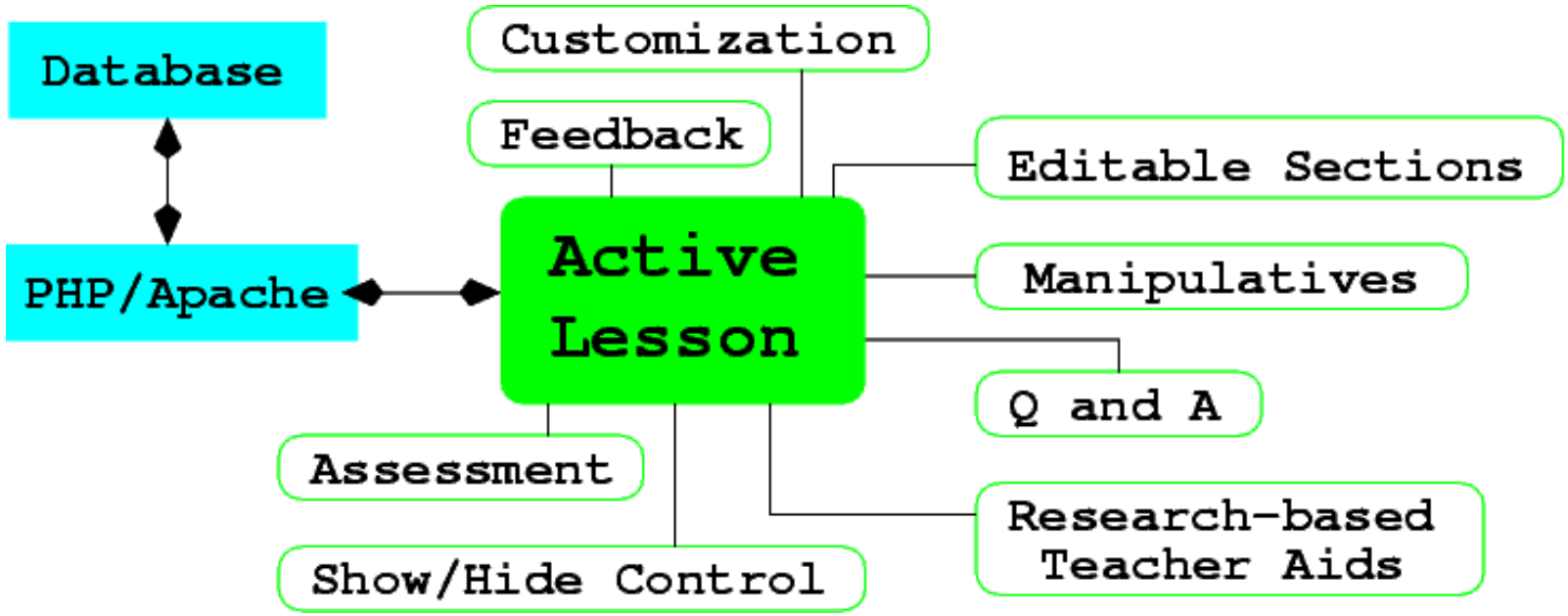


WME Components

- On-Web educational content—*Active Lesson Pages* (ALs) and *Topic Modules* (TMs)
- In-lesson *Manipulatives*—Interoperable, reusable, and user customizable objects.
- Assessment Support—assessment question database, test construction, grading, evaluation, and online tests.
- Client-side Support—regular browsers, javascript, SVG viewer, DOM, browser plug-in.

- Server-side Support—using active pages (PHP) and database (SQLite or MySQL).
- Content-markup Support—MeML and MeML processor, MathGraph.
- WME Services and Tools—GeoSVG, MathEdit, MathPlot, MathGlossary, MathChat, MathBoard, TISM and more.

Kimpton Pilot Project



The Kimpton Site.

KENT STATE UNIVERSITY

Manipulatives

Roll Start Over

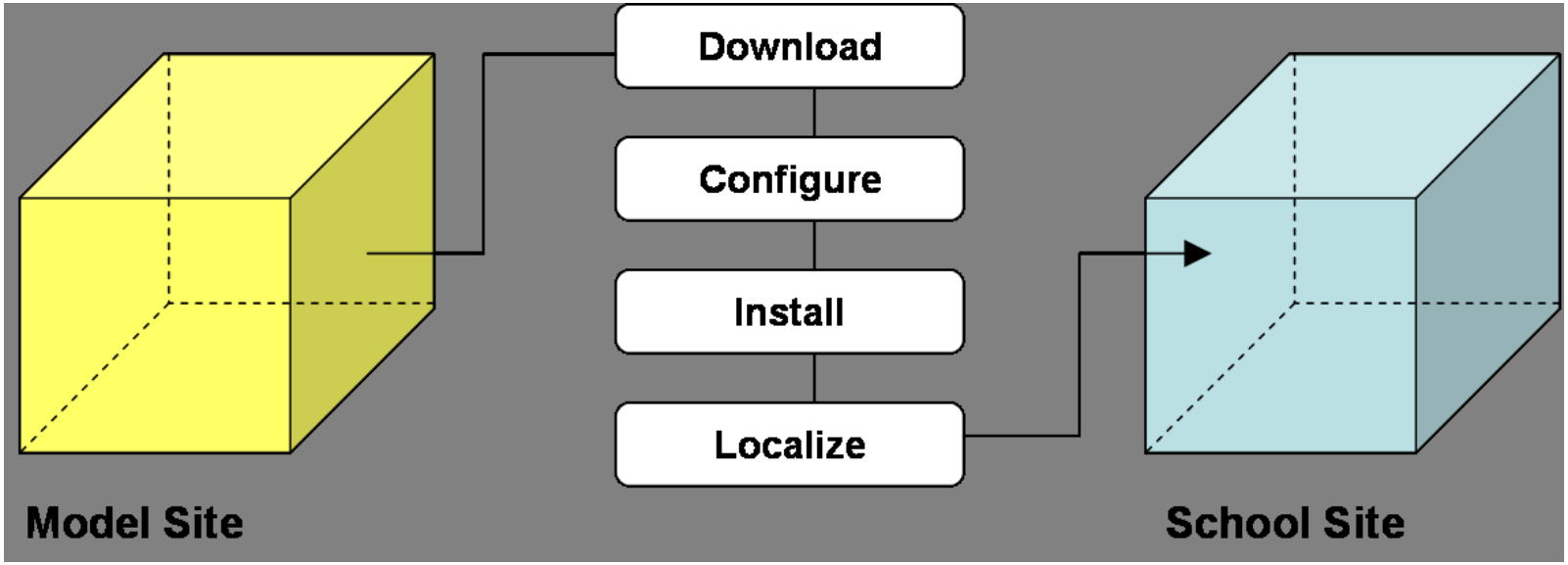
Roll count (the number of rolls you made): 0.

Sum	2	3	4	5	6	7	8	9	10	11	12
Count	0	0	0	0	0	0	0	0	0	0	0

Dice. Equality. Straight Lines.

KENT STATE UNIVERSITY

WME Model Site Download



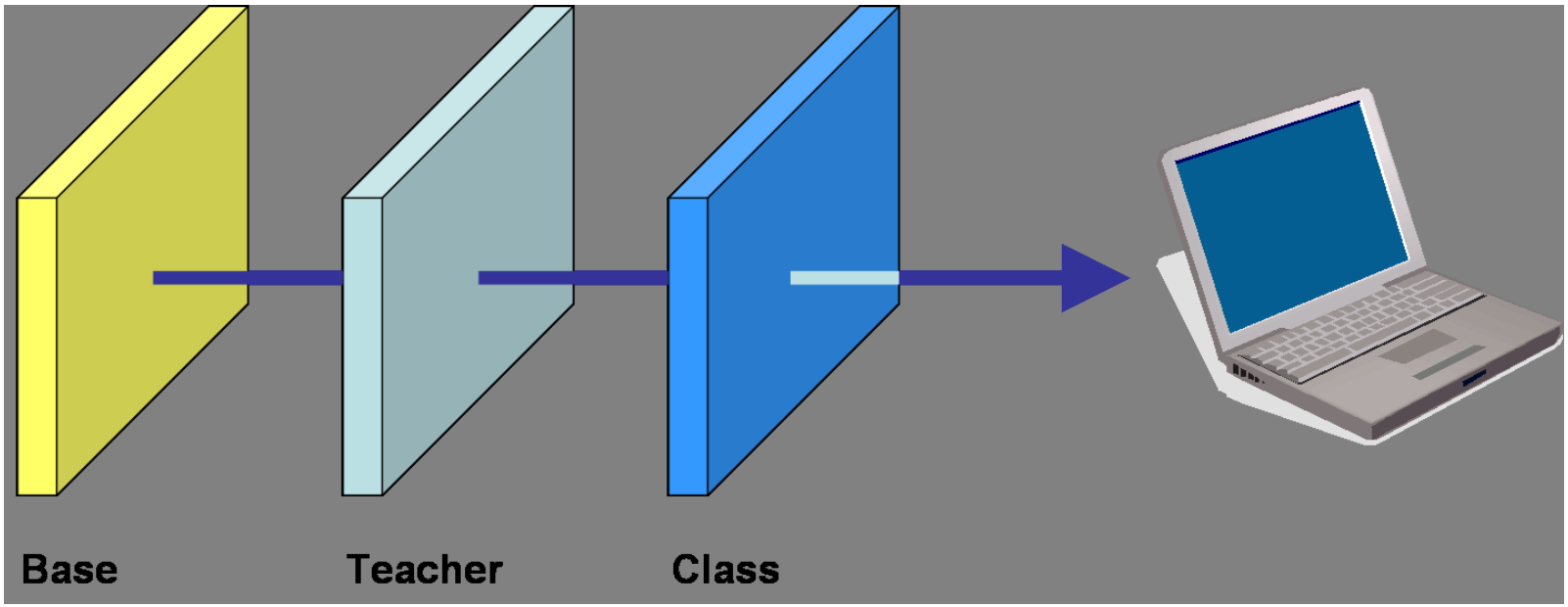
WME Customizations

- For each school—user accounts, grade levels, course listings, course sections.
- For each course—TM and AL selection, student list.
- For each lesson—manipulatives editing: including text, presentation, and functionality, assessment and challenge questions.

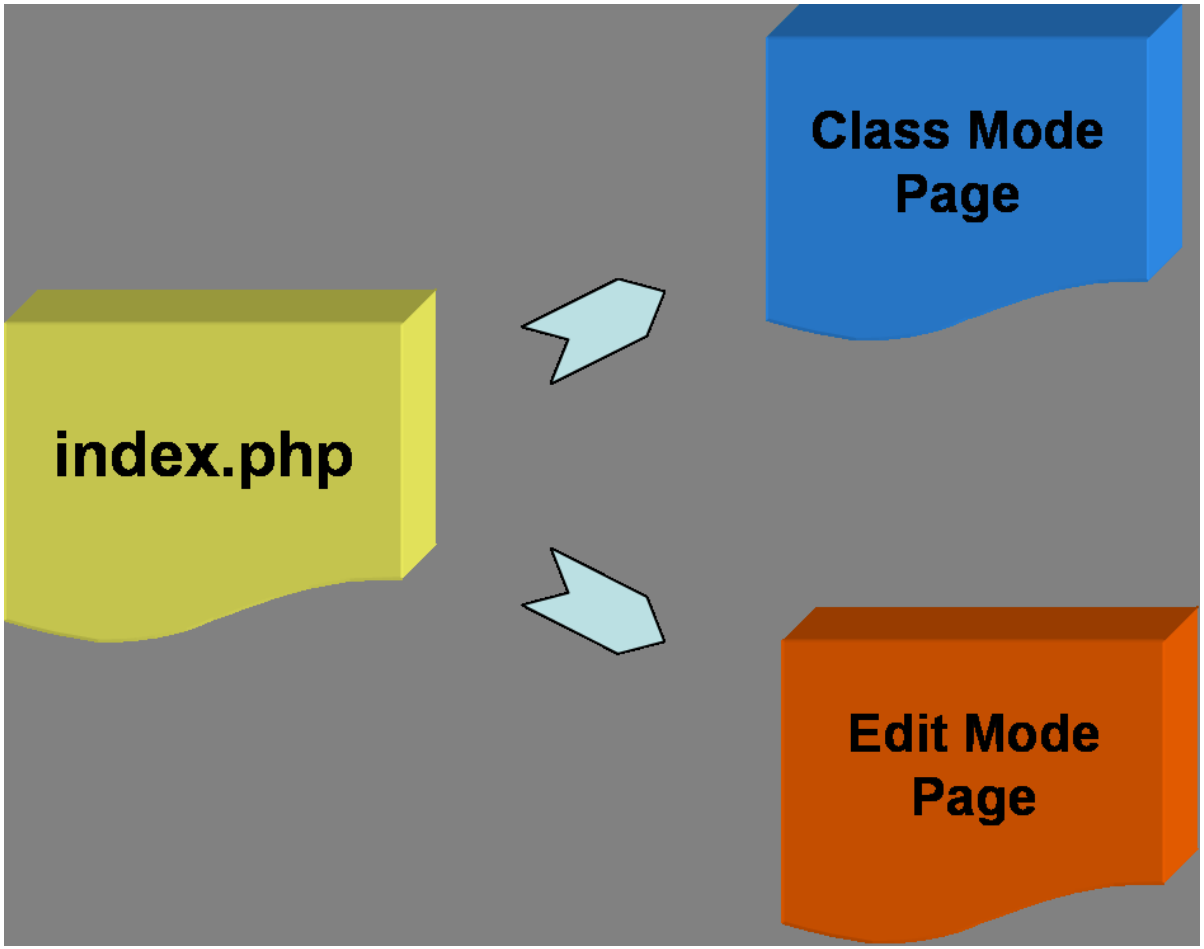
KENT STATE UNIVERSITY

Page Customization Layers

Customizations are per-teacher and per-class.



Dynamic Page Generation

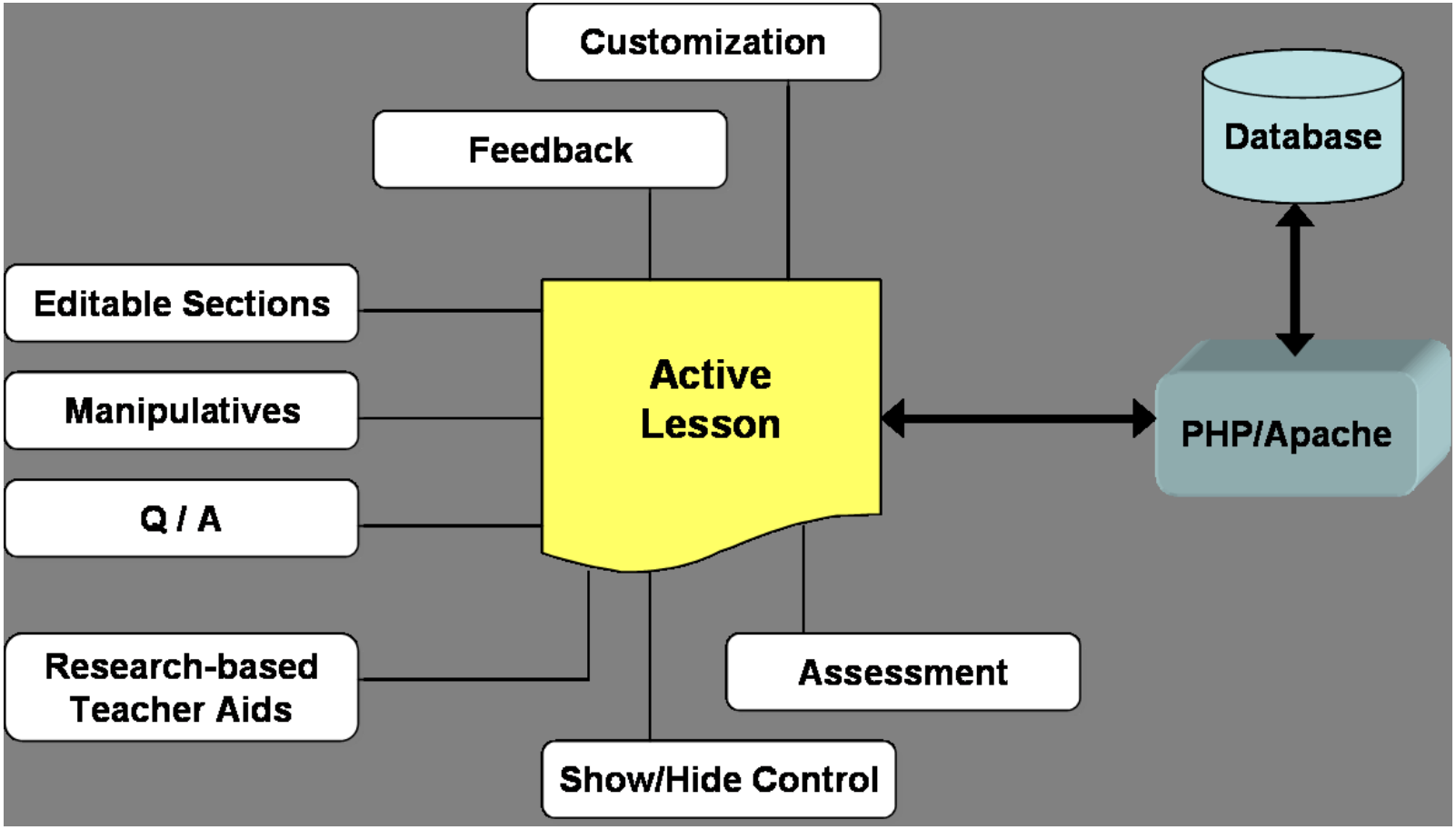


Assessment Help and Automation

- Test authoring, construction, and editing
- Online test taking
- Importing and exporting test questions
- Automatic grading and test data management
- Results evaluation/grading, diagnoses and suggested interventions

KENT STATE UNIVERSITY

Interoperable Modules and Lessons



Answer Checking

- The usefulness and challenges of automatic checking (grading/marking) of mathematical answers have been investigated before.
- *SAGE: a Homework on the Web System* by Brad Lucier, Purdue University, USA, 2005
- *Assessing mathematics automatically using computer algebra and the internet* by C. J. Sangwin, University of Birmingham, UK, 2003

Answer Checking Aspects

- Problem types, answer collection forms.
- Answer input by user.
- Problem authoring/editing, problem generation, problem database.
- Answer grading, common mistakes.
- Feedback to learner.

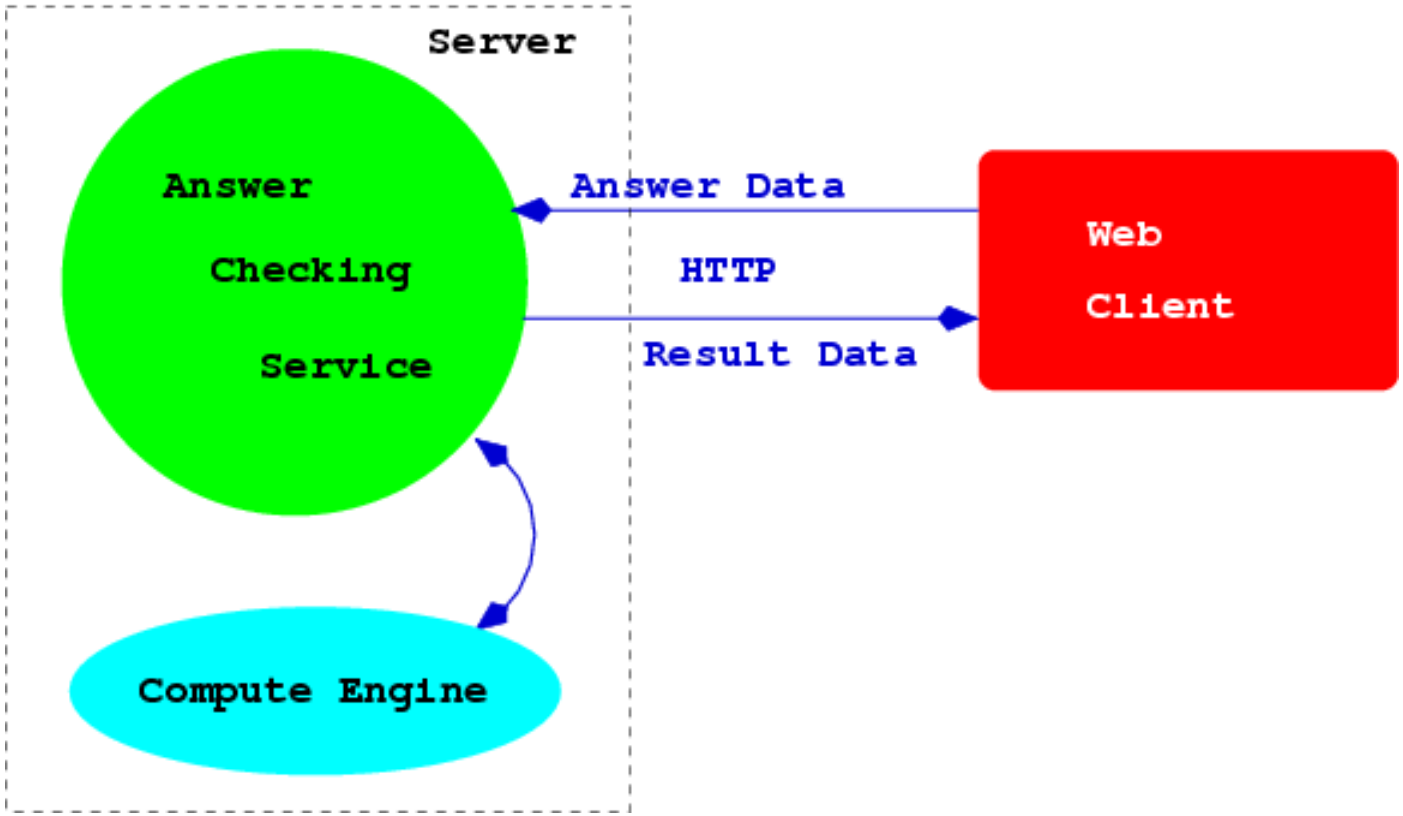
Problem and Answer Types

- True/False and multiple choice problems make answer checking trivial.
- Essay (short and long answer) questions are next to impossible to check automatically.
- What if the answer is a number or a mathematical expression?
- What if the answer is several expressions (such as in factors of a polynomial)
- What if the answer is not unique?
- What if we wish to accept "close" or "almost right" answers?
- What if the problems are automatically generated and the correct answers need to be derived?

Client-Side and Server-Side Answer Checking

- To provide quick feedback to users trying new concepts or skills, client-side answer checking is quick and responsive.
- To support complicate logic, manipulation of mathematical quantities, and to assign grades, server-side answer check is more suitable.
- Answer checking services: support wide variety of problem types, invoked by HTTP `POST/GET` requests and returning `true/false` and perhaps additional results.

Answer Checking Service



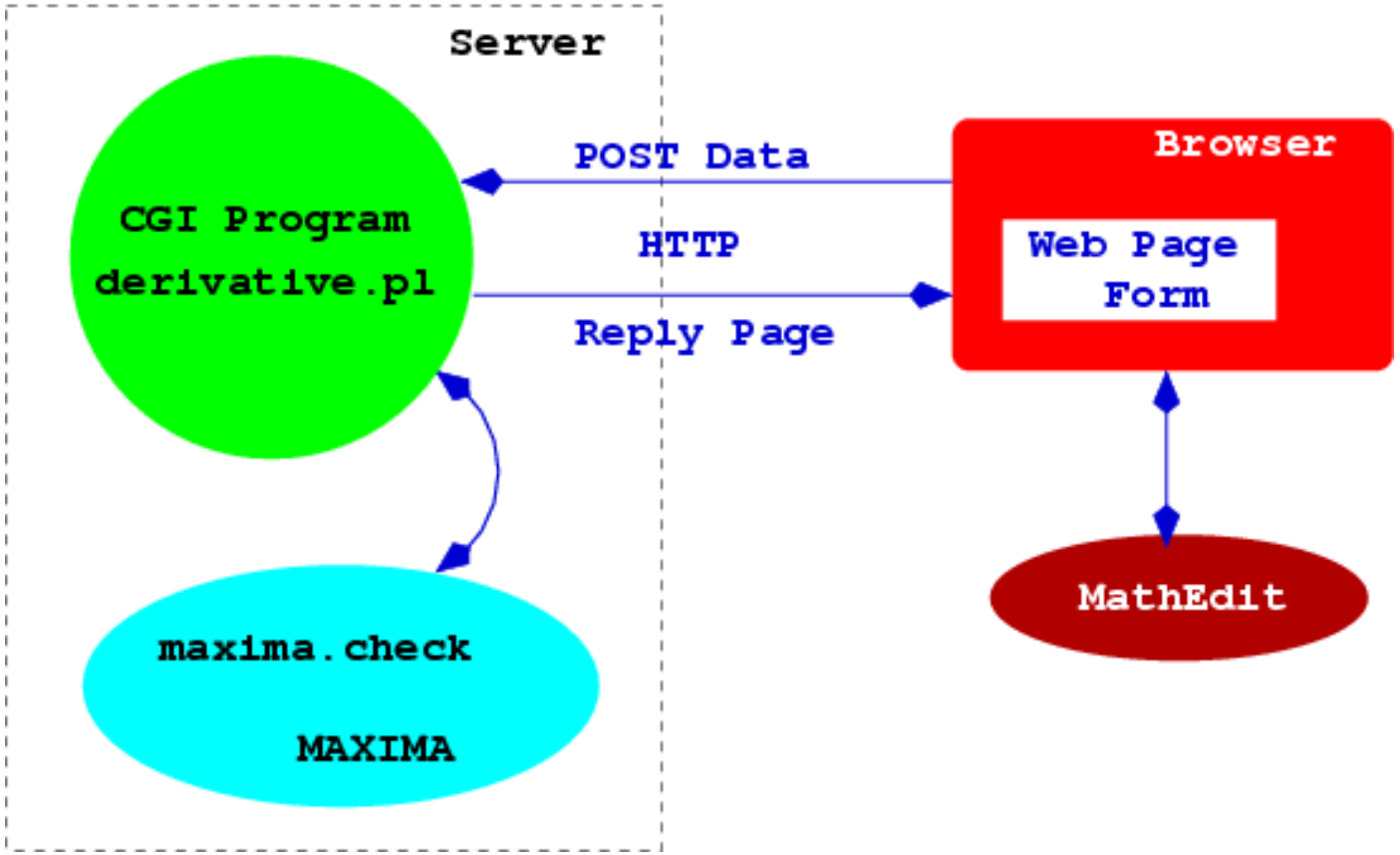
Answer Checking Example

Checking the derivative of a mathematical expression.

- Try this example On IE, On Firefox.
- It involves displaying mathematical formulas (MathML), receiving user entered formula (MathEdit tool), an ad-hoc HTML form to post to a server-side program that checks the answer and provides a response page.
- The user input is displayed using MathML and transmitted to the answer checker in infix notation.
- The correct answer, the derivative, is computed by the checking program rather than given as part of the *answer checking data* posted from the client side.

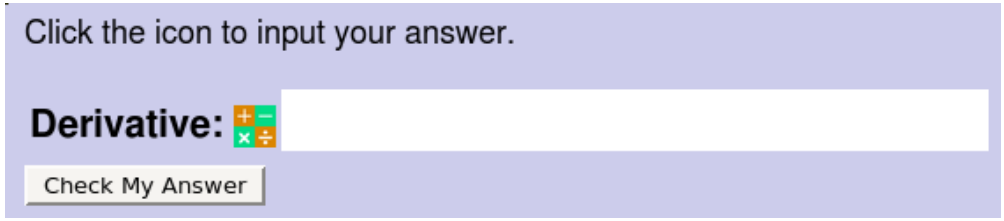
KENT STATE UNIVERSITY

Derivative Checking Service



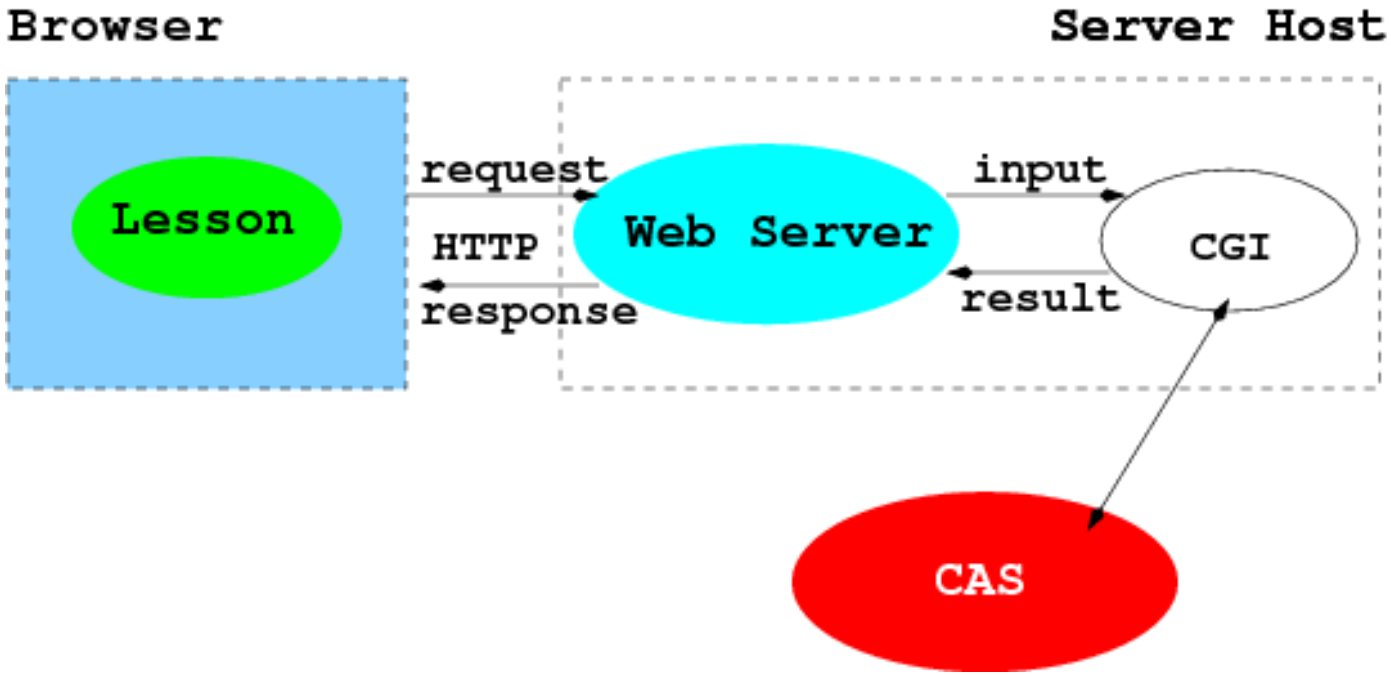
Collecting Answer from User

```
Derivative:
<img onclick="mathedit.newMath('e1', 'der');"
     style="vertical-align: middle"
     src="img/mewhite24.png" />
<div id="der" style="width: 400px">
 </div>
```



KENT STATE UNIVERSITY

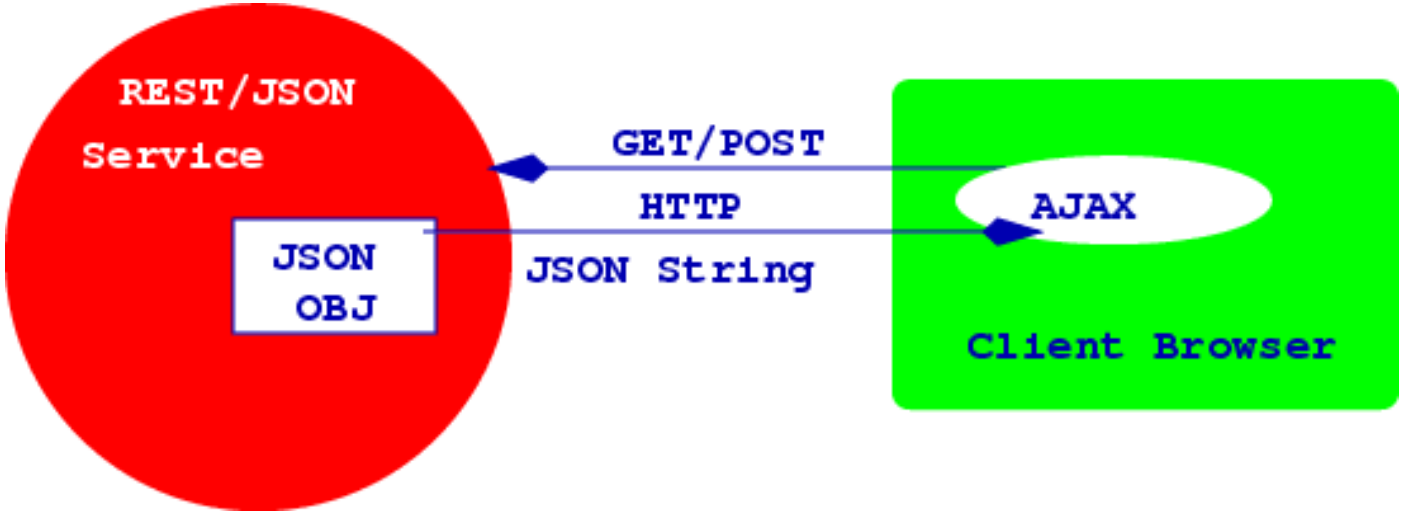
Using CAS for Checking



Decoupling Client and Server

- The above approach tightly couples the client side and server side for any particular answer checking application.
- We can break this tight coupling by client-side AJAX (Asynchronous Javascript And XML) and by establishing an *answer checking protocol* between the client and the answerch checking service.
- We will consider a REST/JSON based answer checking Web service.
- REST = Representational State Transfer
- JSON = Son of Javascript, a data format

Browser Access to JSON Service



Service Returning JSON

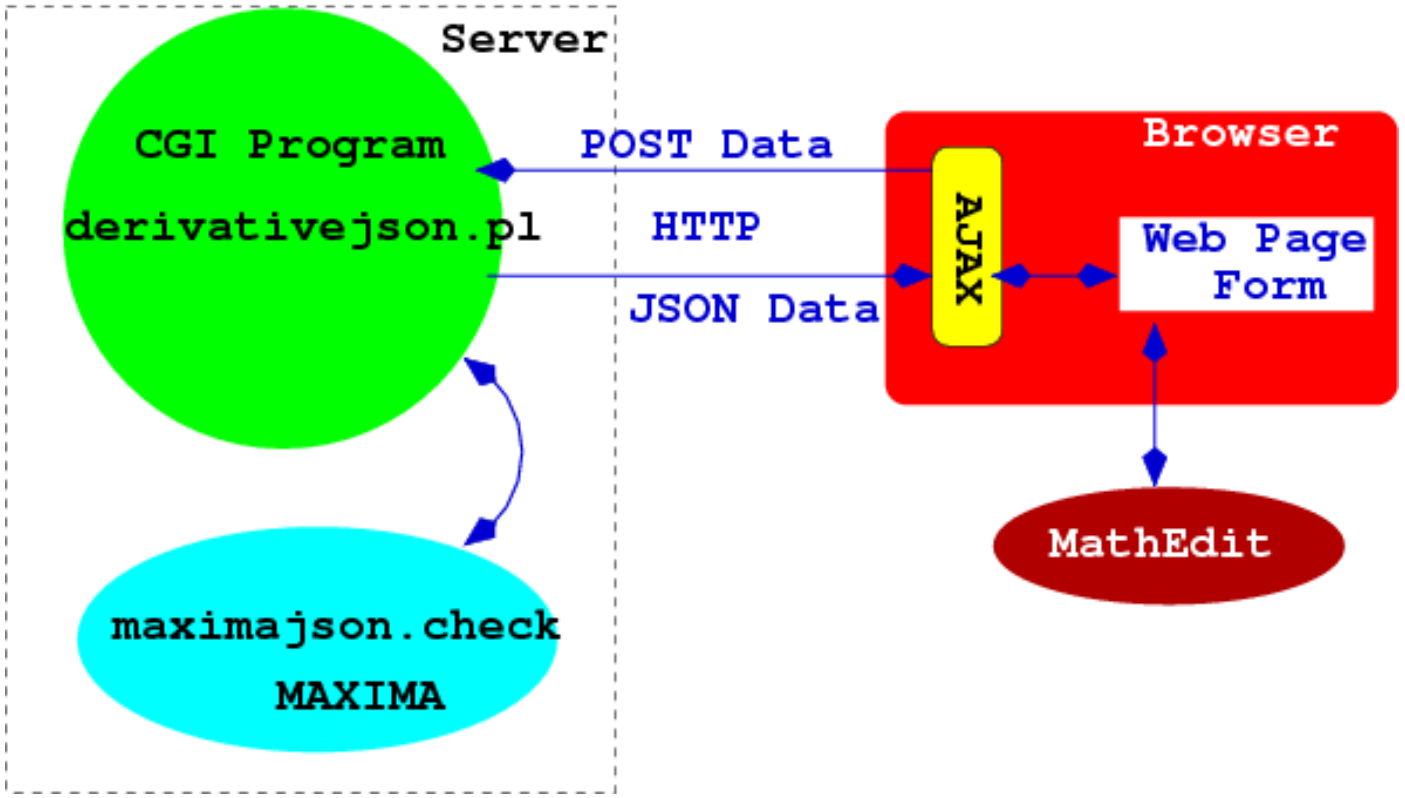
- Made easier by using JSON, a Javascript-defined data representation format.
- The server returns a JSON string which can usually be evaluated directly by Javascript by calling `eval()`.
- The returned MIME type: `application/json`
- The returned data in UTF-8 consists of one or more lines of correct Javascript code. Strict JSON requires returning a *JSON object*.

Client Receiving JSON

- Use an AJAX object to send GET or POST request to the service.
- Use `encodeURIComponent()` to construct query string.
- Service must be on the save server host.
- Because of the async nature of the http request, processing of incoming data is via a *callback function*.
- The results returned are directly accessible in Javascript and can be used to immediately update the host page.

KENT STATE UNIVERSITY

Derivative Checking Service



Derivative Checking Example

Client Side Code for Javascript files

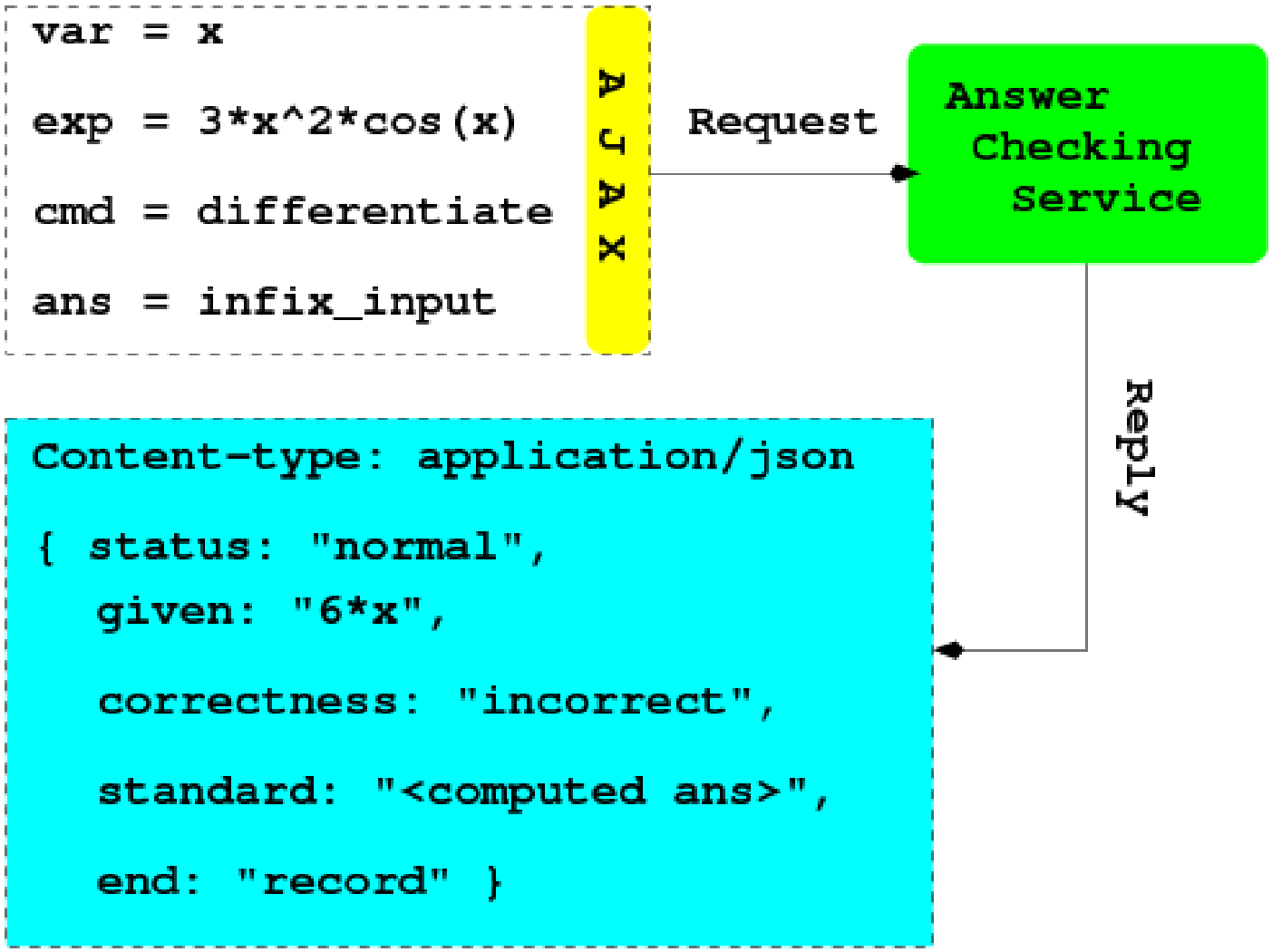
```
<script type="text/javascript" src="ajaxobj.js"></script>
<script type="text/javascript" src="derivativejson.js">
</script> <script type="text/javascript"
                  src="changejson.js"></script>
```

Client Side Code for collecting and checking answer.

```
<p><input type="submit" name="check" value=
    "Check My Answer" onclick="doCheck(theExp)" /></p>
<div style="color: #00c; font-weight: bold"
     id="checkResult"> </div>
```

KENT STATE UNIVERSITY

Checking Data Flow



Javascript Code checkResult

```
var answer;

function doCheck(e)
{ var ex=encodeURIComponent(e);
  var infix=encodeURIComponent(
       mathedit.getInfix("e1"));
  var ct=encodeURIComponent(
       mathedit.getContentMathML("e1"));
  var qr = "var=x&expr=" +ex+ "&answerct="
      +ct+ "&answer=" +infix;
  jsonCheck("http://wme.cs.kent.edu/cgi-bin"
      + "/derivativejson.pl", qr);
}
```

KENT STATE UNIVERSITY

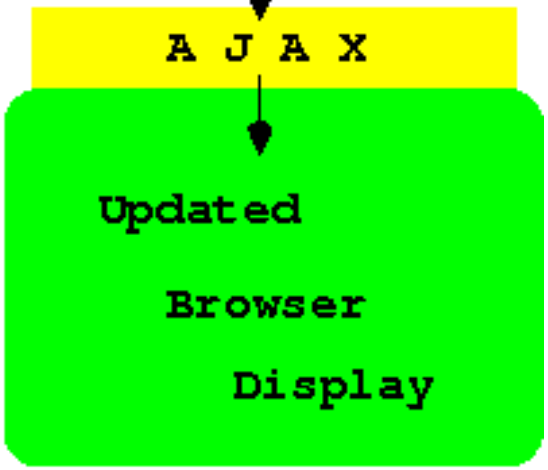
```
function jsonCheck(url, query)
{  var ajaxRequest = new ajaxObject(url);
   ajaxRequest.callback = processResponse;
   // sending the post request
   ajaxRequest.update(query, "POST");
}
```

KENT STATE UNIVERSITY

Browser Display Update

```
Content-type: application/json

{ status: "normal",
  given: "6*x",
  correctness: "incorrect",
  standard: "<computed ans>",
  end: "record" }
```

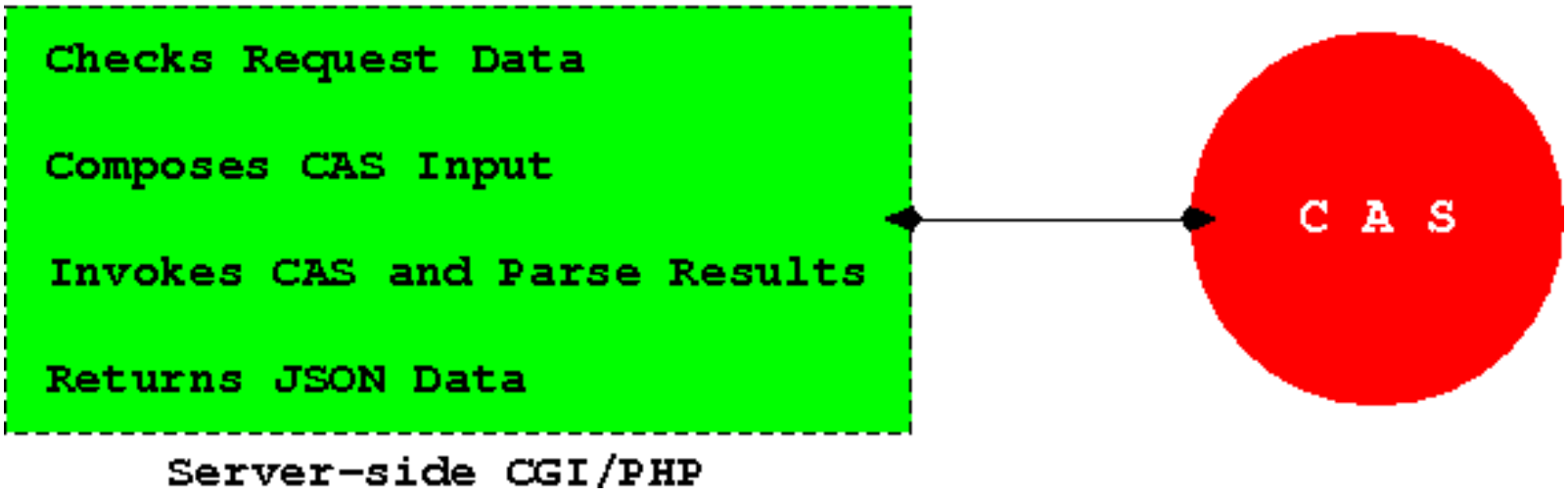


KENT STATE UNIVERSITY

```
function processResponse(responseText)
{ eval("answer=" + responseText + ";");
  var node=document.getElementById(
                          'checkResult');
  if ( answer.correctness=="incorrect" )
  {  node.innerHTML="Your answer is incorrect."+
     "  The correct answer is "+answer.standard;
  }
  else
  {  node.innerHTML = "Congratulations, " +
        "your answer is correct.";
  }
}
```

KENT STATE UNIVERSITY

Interfacing to A CAS



Answer Checking Service Protocol

- Try This Example.
- Use REST for service request
- Use JSON for service reply
- Need to discuss and formulate the request and the reply formats.

Proposed Answer Checking Reply Format

Use JSON as reply format. Only `status` and `correctness` are required. Other values are optional.

- `status:` `normal` or `error`
- `correctness:` `yes`, `no`, `acceptable`, `approx`, or `unknown` (can be an array of such values)
- `yourans_ct:` the mathml content expression of user's answer (or array)
- `yourans_pr:` the mathml presentation expression of user's answer (or array)
- `yourans_infix:` the infix expression of user's answer (or array)
- `ans_ct:` mathml content of standard answer (or array)

- `ans_pr:` mathml presentation of standard answer (or array)
- `ans_infix:` infix notation of standard answer (or array)

KENT STATE UNIVERSITY

Example JSON Data

```
{  status: "normal",
   yourans_ct: '<mathml content>',
   correctness: 'no',
   ans_pr: "<mathml presentation>"
}
```

Proposed Answer Checking Request Format

Use REST format (query strings):

- `var=` the variable(s) involved
- `qans_in=`, `qans_ct=` infix notation, mathml content, of answer(s) to check
- `sans_in=`, `sans_ct=` infix notation, mathml content, of standard answer(s)
- `cmdexp=` command string to be evaluated to produce answer(s)
- `tolerance=` for floating-point answers
- `accept=` a percentage (floating-point number), `PlusC`, or `TimesC`, for acceptable answers

List items are "`,` " comma and space separated.

Infix Notations Used in Answer Checking

The infix notations need to be standardized and please

see this file for a proposal.

Answer Checking Services

- Numerical value checking (possibly with tolerance)
- Expression equality checking
  - rational simplification
  - canonical forms (trig transforms for example)
  - random evaluations points
- Checking for different problem types: integrals (antiderivatives), factoring, polynomial gcd, partial fractions, ...

Simplification VS. Evaluation

- Automatically verifying expression equivalence is *undecidable* in general.
- But in practice, algebraic simplification can often do the job for polynomial and rational expressions.
- Another technique is to test equivalence of expressions at randomly selected points. This is a probabilistic algorithm.
- We can introduce into the Request Format:

```
EvalCheck=true
points=0.0,1.75,-2.5
points=random,6
interval=0.1,9.9
```

- Simple EvalCheck can be done by Javascript on the client side.

Computing Standard Answers

- It is convenient if the correct answer is generated or computed by the answer checking service.
- This can be done by presenting the problem in an *executable* way to the service.
- Examples:

```
derivative(exp,var,n)
factor(polynomial)
gcd(p1, p2)
lcf(p1, p2)
simplify(exp)
```

A Loophole

- A sneaky user may send the problem statement as the answer.
- Any simplifications and transformations may destroy the form of the correct answer or transform the wrong form into the correct form.
- We need a way to specify the nature of the answer. For example we expect the answer given to be an integer, a real, a complex, a monomial, a sum, a product, a power, a polynomial with degree 2, etc.
- This means some problem-specific pattern checking must be performed. A general answer checker may provide some common patterns.
- Some of this can be readily checked on the client side.